UM10912

LPC546xx User manual

Rev. 2.1 — 9 November 2017 User manual

Document information

Info	Content
Keywords	LPC546xx, ARM Cortex-M4, 32-bit microcontroller, LCD, Ethernet AVB, SPIFI, SCT/PWM, USB Host, USB device, CAN FD, I2C, I2S, EEPROM, Flash, EMC, SDRAM controller, DMIC, SDIO interface, SD card interface, LCD controller, eCRP, SHA
Abstract	LPC546xx User Manual

Revision history

Rev	Date	Description
2.1	20171109	LPC546xx User manual.
Modifications:		• Updated Table 260 “Register overview: Input multiplexing (base address 0x4000 5000)”. Fixed cross references of SCT0_INMUX0 to SCT0_INMUX6. • Updated Section 16.3 “Basic configuration”. Added a remark: The maximum frequency for the SCTimer/PWM clock is 100 MHz. Added the remark to Section 7.5.41 “SCTimer/PWM clock select register”. • Updated Table 1 “Ordering information” and Table 2 “Ordering options”. Added the part numbers: LPC54605J256BD100, LPC54605J512BD100, LPC54605J256ET100, LPC54605J512ET100. • Updated Table 232 “Device ID0 register values”. • Added text to Section 15.5.2 “DMA Modes”: Using any specific DMA channel requires initializing the device registers associated with that channel (see Table 300), and setting the channel descriptor. The channel descriptor is shown in Table 303. The channel descriptor is an entry in the channel descriptor table. This table is located somewhere in memory, typically in on-chip SRAM (see Section 15.6.3 “SRAM Base address register”). The DMA transfer is initiated by writing the channels CFG and XFERCFG registers and setting the channels ENABLESET bit. Note that once a DMA operation is initiated, the descriptor entry in the DMA descriptor table may be modified by the DMA controller. Therefore, when a subsequent DMA transaction is initiated, the DMA descriptor entry must be re-initialized. This is not the case with the linked descriptors. The linked list entries, that are not in the main DMA descriptor table, do not need to be re-written, for subsequent DMA requests. These entries are not modified by the controller. • Added Chapter 4 “LPC546xx FRO API ROM routine”. • Added bit 14 to Section 7.5.77 “FRO Control register”.
2.0	20170726	LPC546xx User manual.
Modifications:		• Updated Section 5.5.1.6 “Write block”. Added text: If a block number crosses a sector boundary, the LPC546xx automatically erases the sector prior to the write operation. • Updated Section 5.5.1.15 “Write RAM”: Added text: Write a block of data to the RAM. • Updated remark of Table 1024 “ADC common supply and reference pins”: The supply voltage V_{DD} must be equal to V_{DDA}. • Updated text in Section 50.5.8 “DIGEST register”. • Updated Figure 182 “Serial Wire Debug internal connections”. • Updated Table 46 “I2C / SPI ISP read block response packet”: Length is 0x204.
1.9	20170616	LPC546xx User manual.

Revision history *...continued*

Rev	Date	Description
Modifications:		• Added LPC5462x device. • Updated Table 119 “AHB matrix priority register 0 (AHBMATPRIO, main syscon: offset 0x010) bit description”. Updated all the bits after 5:4. • Updated the description text for Bit 1 ENABLERX in Table 457 “FIFO Configuration register (FIFOCFG - offset 0xE00) bit description”. • Updated the bit description text for SELCC: added- Note that different part number and package variations may provide different capture input pin functions. See Table 374 “Count Control Register (CTCR, offset 0x070) bit description”. • Updated Timer section of Section 1.2 “Features”. • Updated Table 998 “USBD_API_INIT_PARAM class structure”, Table 1003 “USBD_DFU_INIT_PARAM class structure”, Table 1006 “USBD_HW_API class structure”, and Table 1008 “USBD_MSC_INIT_PARAM class structure”. • Updated description for bit 22 of FIFOWR register. See Table 463 “FIFO write data register (FIFOWR - offset 0xE20) bit description”. • Updated Table 252 “Suggested IOCON settings for I2C functions”. Bit 11 I2CFILTER of Type I IOCON register should be 0 or 1 when configuring standard I2C mode. • Updated Table 520 “Register overview: DMIC subsystem (base address 0x4009 0000)”. Updated description of USE2FS register: Use decimate by multiple of 2. • Added Remark to mem_base in Table 998 “USBD_API_INIT_PARAM class structure”: When high_speed_capable is set to 1 (USB1 is used as the device), mem_base must use the USB SRAM region. • Added part number LPC54628J512ET180 to Table 229 “Device ID0 register (DEVICE_ID0, main syscon: offset 0xFF8) bit description”. • Added a remark to Section 6.2.2 “Configure the main clock and system clock” and Section 6.6.4.2 “PLL description”. • Added a Table note to Table 182 “Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description”. • Deleted text: Dual Enhanced images give the ability to run one image from flash, while programming another image located elsewhere in flash from Section 3.3.2.3.2 “Dual Enhanced (DE) images”. • Updated Section 3.3.2.7 “Image qualification”. • Updated Section 49.4 “General description”. • Updated Section 26.7.2.1 “Rate calculations”: added a remark and Table 494 “Settings for 400 KHz clock rate”. • Updated Figure 84 “HWVAD block diagram” and Figure 90 “DMIC FIFO and DMA”.
1.8	20170428	LPC546xx User manual.

Revision history *...continued*

Rev	Date	Description
• Modifications:		• Added Figure 4 “Legacy image boot process flowchart” and Figure 5 “Single Enhanced image and Dual Enhanced image boot process flowchart”. • Updated text in Section 3.3.2.7 “Image qualification”. Was: The length does not include the four bytes that make up the Image Length field, which means that the CRC is not calculated on the four bytes of the Image Length field to: The length does not include the four bytes that make up the CRC value field, which means that the CRC is not calculated on the four bytes of the CRC value field. • Updated Table 230 “Device ID0 register values”. • Added Table 27 “USART ISP Read/Write EEPROM page command”, Table 64 “I2C / SPI ISP EEPROM write page command packet”, Table 65 “I2C / SPI ISP EEPROM write page response packet”, Table 66 “I2C / SPI ISP EEPROM read page command packet”, and Table 67 “I2C / SPI ISP EEPROM read page response packet”. • Updated Table 305 “Register overview: DMA controller (base address 0x4008 2000)”:Fixed offsets from Channel 20 registers to Channel 29 registers. • Added text to Section 25.6.14 “FIFO write data register”: To set-up a slave SPI for receive only, the control bit settings must be pushed into the write FIFO to become active. Therefore, at least one write to the FIFOWR data bits must be done to make the control bits active. • Updated Table 1043 “OTP content”: Changed description text of Function, otpEnableBankWriteLock to Parameter - unsigned read/write lock: Non-zero value disables subsequent access modifications until Reset. • Added a remark to Section 4.3.3 “Flash content protection”: EMC cannot be used when using the SPI ISP feature. • Fixed the address of AHBCLKCTRLSET[0] (0x40000220) in Section 48.4 “Pin description”. • Updated description for Bits 30 and 31 in Table 1031 “ADC Data registers (DAT[0:11], offset [0x020:0x04C]) bit description”: DONE in Bits 30 and 31 descriptions is replaced with "DATA VALID". • Updated Bit 29 LOWPRIO description text in Table 1027 “ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description”: Setting this bit to a 1 will permit any enabled B sequence trigger (including a B sequence software start) to immediately interrupt sequence A and launch a B sequence in its place. The conversion currently in progress will be terminated. Sequence A that was interrupted will automatically resume after Sequence B completes. The conversion of the channel that was terminated is re-sampled and the conversion sequence resumes from that point. 0 High priority. Any B trigger that occurs while an A conversion sequence is active is ignored and lost. • Added Status code 15 to Table 85 “IAP Status codes Summary”. • Updated Status code and Remark in description text. See Table 71 “IAP Copy RAM to flash command”. • Updated Section 14.5.2 “DMA Modes”.
1.7	20170331	LPC546xx User manual.
Modifications:		• Added TFBGA100 and LQFP100 packages.

Revision history *...continued*

Rev	Date	Description
1.6	20170331	LPC546xx User manual.
Modifications:		• Deleted the remark from bits 2 and 3 in Table 108 “EMC system control register (EMCSYSCTRL, main syscon: offset 0x444) bit description” **Remark:** The state of this bit is preserved through a software reset, and only a POR or a BOD event will reset it to its default value. • Updated Table 1006 “Pointer to ISP parameter array”. For byte offset 1: 1: I2C of Flexcomm 1 and 4: SPI of Flexcomm 3. • Updated Table 164 “ISP pin assignments”. In USART mode: FC0_TXDPIO0_30; FCO_RXD PIO0_29. • Updated bit 29, LOWPRIO description text of Table 1022 “ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description”. • Updated Table 963 “I2C / SPI command summary”: Command identifier of Write RAM is 0xB0. • Updated Table 992 “I2C / SPI ISP write RAM command packet”: changed text to: The ISP_CMD_WRITE_RAM (0xB0) command is used to erase more than one sector (from a start sector to an end sector). • Updated Table 993 “I2C / SPI ISP write RAM response packet”: Value of Command 0xB0. • Updated Table 994 “I2C / SPI ISP go command packet”, and Table 995 “I2C / SPI ISP go to RAM response packet”: Value of Command 0xB1.
1.5	20170315	LPC546xx User manual.
Modifications:		• Updated remark: The SDIO function clock to the interface can be up to 50 MHZ. See Section 27.3 “Basic configuration”. Deleted text: Wake-up: Enable interrupts for waking up from deep-sleep and deep power-down modes, enable the interrupts in the STARTER1 register. See Section 4.5.91. • Updated: Changing clock. The cards operate at a maximum of 25 MHz (at maximum of 50 MHz in high-speed mode and speed detection using CMD6). See Section 27.8.2.1 “Initialization”. • Updated Table 522 “Timing requirements”: SDR25 (High Speed mode).
1.4	20170307	LPC546xx User manual.
Modifications:		• Changed title from LPC5460x to LPC546xx.
1.3	20170228	LPC5460x User manual.
Modifications:		• Updated Table 2 “Ordering options” and Table 1 “Ordering information”. Removed references of S parts. • Updated Section 6.1 “Features”. • Updated Table 108 “EMC system control register (EMCSYSCTRL, main syscon: offset 0x444) bit description”. Removed the remark from bits 2 and 3: The state of this bit is preserved through a software reset, and only a POR or a BOD event will reset it to its default value.

Revision history *...continued*

Rev	Date	Description
1.2	20170224	LPC5460x User manual.
Modifications:		• Removed all references to LPC54S60x. • Removed all references to AES-engine and SHA. • Updated Figure 1 “Block diagram” and Figure 2 “Main memory map”. • Updated Table 5 “Connection of interrupt sources to the NVIC”, Table 8 “Interrupt Set-Enable Register 1 register”, and Table 30 “Interrupt priority register 13”. • Updated Table 38 “AHB matrix priority register 0 (AHBMATPRIO, main syscon: offset 0x010) bit description”. • Updated Chapter 6 “LPC5460x Boot process”. • Updated Table 180 “DMA trigger Input mux registers (DMA_ITRIG_INMUX[0:29], offsets [0x0E0:0x154]) bit description”. • Updated Table 1013 “IAP Get ROM API pointer command”. • Updated Chapter 45 “LPC5460x One-Time Programmable (OTP) memory and API”. • Renamed table title of Table 593 “Supported cursor sizes”: was Palette data storage for STN monochrome mode.
1.1	20170206	LPC54S60x/LPC5460x User manual.
Modifications:		• Updated Table 3 “Memory usage and details”: Address range details and description for address range: 0x8000 0000 to 0xDFFF FFFF. Static memory chip select: was 0x9000 0000 - 0x93 FFFF, now, 0x9000 0000 – 0x93FF FFFF. • Updated Figure 4 “Clock generation”. Removed wdt_clk. • Updated Table 68 “System PLL clock source select register (SYSPLLCLKSEL, main syscon: offset 0x290) bit description”: Removed Watchdog Oscillator, wdt_clk. • Updated Section 4.5.77 “FRO Control register”. • Updated Table 113 “SDIO clock in phase and delay control register (SDIOCLKCTRL, main syscon: offset 0x460) bit description”: Bit 7, PHASE_ACTIVE description. • Updated Table 114 “FRO control register (FROCTRL, main syscon: offset 0x500) bit description”: Bit 31: Reserved. • Updated Table 855 “USB1 Data buffer start address (DATABUFSTART, offset 0x00C) bit description”. • Updated Figure 15 “SCT0 input multiplexing”. • Updated Table 185 “GPIO pins available”; Added 180-pin device package. • Updated Table 890 “PTD bit definition”. Fixed NakCnt[3:0] and Cerr[1:0] bit definitions and updated description for MaxPacketLength[10:0] TT_MPS_Len[10:0]. • Added text: PDSLEEPCFG registers can also be used to turn analog peripherals on or off for deep-sleep mode to Section 5.2 “General description”. • Updated Section 5.3.2.1 “Power configuration in active mode”: The clock source for the system clock can be selected from the FRO (default), crystal oscillator, output of the PLL, the system oscillator, 32 kHz oscillator, or the watchdog oscillator (see Figure 4 and related registers).

Revision history *...continued*

Rev	Date	Description
1.1		• Updated Section 6.1 “Features”: Added RSA API calls. • Updated Section 6.3.2.7 “Image qualification”. • Updated description of bit 25:16, XFERCOUNT. See Table 245 “Channel transfer configuration registers bit description”. • Updated Table 578 “Static Memory Read Delay registers (STATICWAITRD[0:3], address 0x4008 120C (STATICWAITRD0), 0x4008 122C (STATICWAITRD1), 0x4008 124C (STATICWAITRD2), 0x4008 126C (STATICWAITRD3)) bit description”. Changed address of STATICWAITRD0 to 0x4008 120C. • Table 580 “Static Memory Write Delay registers (STATICWAITWR[0:3], address 0x4008 1214 (STATICWAITWR0), 0x4008 1234 (STATICWAITWR1), 0x4008 1254 (STATICWAITWR2), 0x4008 1274 (STATICWAITWR3)) bit description” Changed address of STATICWAITWR0 to 0x4008 1214. • Updated Section 33.2 “Features”: Added text: Software support for AVB feature is available from NXP Professional Services. See nxp.com for more details. • Added text: USB on-chip drivers are provided via the USB Stack in SDK and LPCOpen software packages to Section 38.1 “How to read this chapter”. • Updated Table 939 “Power API calls”. • Updated Section 40.4.1 “POWER_SetVoltageForFreq”. and Section 40.4.3 “CLOCK_SetupFROClocking”. • Updated Table 942 “Chip_POWER_EnterPowerMode routine” and Table 943 “Chip_POWER_SetFROHFRate routine”. • Updated Section 40.4.2.2 “Param1: peripherals”. • Added a remark to Section 41.4.5.4 “ISP Write to RAM” and Section 41.4.5.5 “ISP Read Memory”. • Deleted EEPROM from Section 41.4.5.7 “ISP Copy RAM to flash”. • Updated Section 4.6.4.1 “PLL Features”: Removed 6 kHz to 1.5 MHz Watchdog Oscillator.
1.0	20161221	Initial revision. LPC54S60x/LPC5460x User manual.

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

1.1 Introduction

The LPC546xx is a family of ARM Cortex-M4 based microcontrollers for embedded applications featuring a rich peripheral set with very low power consumption and enhanced debug features.

The ARM Cortex-M4 is a 32-bit core that offers system enhancements such as low power consumption, enhanced debug features, and a high level of support block integration. The ARM Cortex-M4 CPU incorporates a 3-stage pipeline, uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals, and includes an internal prefetch unit that supports speculative branching. The ARM Cortex-M4 supports single-cycle digital signal processing and SIMD instructions. A hardware floating-point processor is integrated into the core.

The LPC546xx family includes up to 512 KB of flash, 200 KB of on-chip SRAM, up to 16 kB of EEPROM memory, a quad SPI Flash Interface (SPIFI) for expanding program memory, one high-speed and one full-speed USB host and device controller, Ethernet AVB, LCD controller, Smart Card Interfaces, SD/MMC, CAN FD, an External Memory Controller (EMC), a DMIC subsystem with PDM microphone interface and I^2S, five general-purpose timers, SCTimer/PWM, RTC/alarm timer, Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), ten flexible serial communication peripherals (USART, SPI, I^2S, I^2C interface), Secure Hash Algorithm (SHA), 12-bit 5.0 Msamples/sec ADC, and a temperature sensor.

1.2 Features

- ARM Cortex-M4 CPU:
  - ARM Cortex-M4 processor, running at a frequency of up to 220 MHz.
  - The LPC5460x/61x devices operate at CPU frequencies of up to 180 MHz. The LPC54628 device operates at CPU frequencies of up to 220 MHz.
  - Built-in Memory Protection Unit (MPU) supporting eight regions.
  - Built-in Nested Vectored Interrupt Controller (NVIC).
  - Hardware Floating Point Unit (FPU).
  - Non-maskable Interrupt (NMI) input.
  - Serial Wire Debug (SWD) with six instruction breakpoints, two literal comparators, and four watch points. Includes Serial Wire Output and ETM Trace for enhanced debug capabilities, and a debug timestamp counter.
  - System tick timer.
- On-Chip memory:
  - Up to 512 KB on-chip flash program memory with flash accelerator and 256 byte page erase and write.
  - Up to 200 KB total SRAM consisting of 160 kB contiguous main SRAM and an additional 32 kB SRAM on the I&D buses. 8 kB of SRAM bank intended for USB traffic.

- 16 kB of EEPROM.

- ROM API support:
  - Flash In-Application Programming (IAP) and In-System Programming (ISP).
  - ROM-based USB drivers (HID, CDC, MSC, and DFU). Supports flash updates via USB.
  - Supports booting from valid user code in flash, USART, SPI, and I2C.
  - Legacy, Single, and Dual image boot.
  - OTP API for programming OTP memory.
  - Random Number Generator (RNG) API.
- Clock generation:
  - Crystal oscillator with an operating range of 1 MHz to 25 MHz.
  - 12 MHz internal Free Running Oscillator (FRO). This oscillator provides a selectable 48 MHz or 96 MHz output, and a 12 MHz output (divided down from the selected higher frequency) that can be used as a system clock. The FRO is trimmed to ±1 % accuracy over the entire voltage and temperature range.
  - Independent Watchdog Oscillator (WDOSC) with a frequency range of 6 kHz to 1.5 MHz.
  - 32.768 kHz low-power RTC oscillator.
  - System PLL allows CPU operation up to the maximum CPU rate and can run from the main oscillator, the internal FRO, the watchdog oscillator or the 32 KHz RTC oscillator.
  - Two additional PLLs for USB clock and audio subsystem.
  - Independent clocks for the SPIFI interface, ADC, USBs, and the audio subsystem.
  - Clock output function with divider.
  - Frequency measurement unit for measuring the frequency of on-chip or off-chip clock signal.
- Serial interfaces:
  - Flexcomm Interface contains up to ten serial peripherals. Each Flexcomm Interface can be selected by software to be a USART, SPI, or I^2C interface. Two Flexcomm Interfaces also include an I2S interface. Each Flexcomm Interface includes a FIFO buffer that supports USART, SPI, and I2S if supported by that Flexcomm Interface. A variety of clocking options are available to each Flexcomm Interface and include a shared fractional baud-rate generator.
  - I^2C-bus interfaces support Fast-mode and Fast-mode Plus with data rates of up to 1Mbit/s and with multiple address recognition and monitor mode. Two sets of true I^2C pads also support High Speed Mode (3.4 Mbit/s) as a slave. The slave function is able to wake up the device from Deep-sleep and Power-down modes.
  - Two ISO 7816 Smart Card Interfaces with DMA support.
  - USB0 full-speed host/device controller with on-chip PHY and dedicated DMA controller supporting crystal-less operation in device mode.
  - USB1 high-speed host/device controller with on-chip high-speed PHY.
  - SPIFI with XIP feature uses up to four data lines to access off-chip SPI/DSPI/QSPI flash memory at a much higher rate than standard SPI or SSP interfaces.

- Ethernet MAC with MII/RMII interface with Audio Video Bridging (AVB) support and dedicated DMA controller.
- Two CAN FD modules with dedicated DMA controller.

- Digital peripherals:
  - DMA controller with 30 channels and up to 24 programmable request/trigger sources and able to access all memories and DMA-capable peripherals.
  - LCD Controller supporting both Super-Twisted Nematic (STN) and Thin-Film Transistor (TFT) displays. It has a dedicated DMA controller, selectable display resolution (up to 1024 x 768 pixels), and supports up to 24-bit true-color mode.
  - External Memory Controller (EMC) provides support for asynchronous static memory devices such as RAM, ROM and flash, in addition to dynamic memories such as single data rate SDRAM with an SDRAM clock of up to 100 MHz. EMC bus width (bit) on TFBGA180, TFBGA100, and LQFP100 packages supports up to 8/16 data line wide static memory, in addition to dynamic memories, such as, SDRAM (2 banks only) with an SDRAM clock of up to 100 MHz.
  - Secured digital input/output (SD/MMC) card interface with DMA support.
  - CRC engine block can calculate a CRC on supplied data using one of three standard polynomials with DMA support.
  - Up to 171 General-Purpose Input/Output (GPIO) pins.
  - GPIO registers are located on the AHB for fast access. The DMA supports GPIO ports.
  - Up to eight GPIOs can be selected as pin interrupts (PINT), triggered by rising, falling or both input edges.
  - Two GPIO grouped interrupts (GINT) enable an interrupt based on a logical (AND/OR) combination of input states.
- Analog peripherals:
  - 12-bit, 12-channel, Analog-to-Digital Converter (ADC) supporting 5.0 Msamples/s. The ADC supports two independent conversion sequences.
  - Integrated temperature sensor connected to the ADC.
- Security features:
  - Random number generator can be used to create keys with DMA support.
  - enhanced Code Read Protection (eCRP) to protect user code.
  - OTP memory for ECRP settings and user application specific data.
  - Secure Hash Algorithm (SHA1/SHA2) module with dedicated DMA controller.
- DMIC subsystem includes a dual-channel PDM microphone interface with decimators, filtering, and hardware voice activity detection. The processed output data can be routed directly to an I^2S interface if needed.
- Timers
  - Five 32-bit general purpose timers/counters, with up to 4 capture inputs and 4 compare outputs, PWM mode, and external count input. Specific timer events can be selected to generate DMA requests. The fifth timer does not have external pin connections and may be used for internal timing operations.

  - SCTimer/PWM with 8 input and 10 output functions (including capture and match). Inputs and outputs can be routed to/from external pins and internally to or from selected peripherals. Internally, the SCTimer/PWM supports 10 match/captures, 10 events, and 10 states.
  - 24-bit Multi-Rate Timer (MRT) module with four channels each capable of generating repetitive interrupts at different, programmable frequencies.
  - 32-bit Real-time clock (RTC) with 1 s resolution running in the always-on power domain. A timer in the RTC can be used for wake-up from all low power modes including deep power-down, with 1 ms resolution. The RTC is clocked by the 32.768 kHz oscillator.
  - Multiple-channel multi-rate 24-bit timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
  - Windowed Watchdog Timer (WWDT) with dedicated watchdog oscillator.
  - Ultra-low power Micro-tick Timer, running from the Watchdog oscillator that can be used to wake up the device from low power modes.
  - Repetitive Interrupt Timer (RIT) for debug time-stamping and for general purpose use.
- Power-saving modes and wake-up:
  - Integrated PMU (Power Management Unit) to minimize power consumption.
  - Reduced power modes: sleep, deep-sleep, and deep power-down.
  - Wake-up from deep-sleep via activity on the USART, SPI, and I^2C peripherals when operating as slaves.
  - Wake-up from sleep, deep-sleep, power-down, and deep power-down modes using the RTC alarm.
  - Ultra-low power Micro-tick Timer, running from the Watchdog oscillator that can be used to wake up the device from low power modes.
  - Power-On Reset (POR).
  - Brown-Out Detect (BOD) with separate thresholds for interrupt and forced reset.
- Single power supply 1.71 V to 3.6 V.
- JTAG boundary scan supported.
- 128 bit unique device serial number for identification.
- Operating temperature range −40 °C to +105 °C.
- Available in TFBGA180, TFBGA100, LQFP208, and LQFP100 packages.

1.3 Block diagram

Each Flexcomm Interface includes USART, SPI, and I2C functions. Flexcomm Interface 6 and 7 each also provide an I2S function.

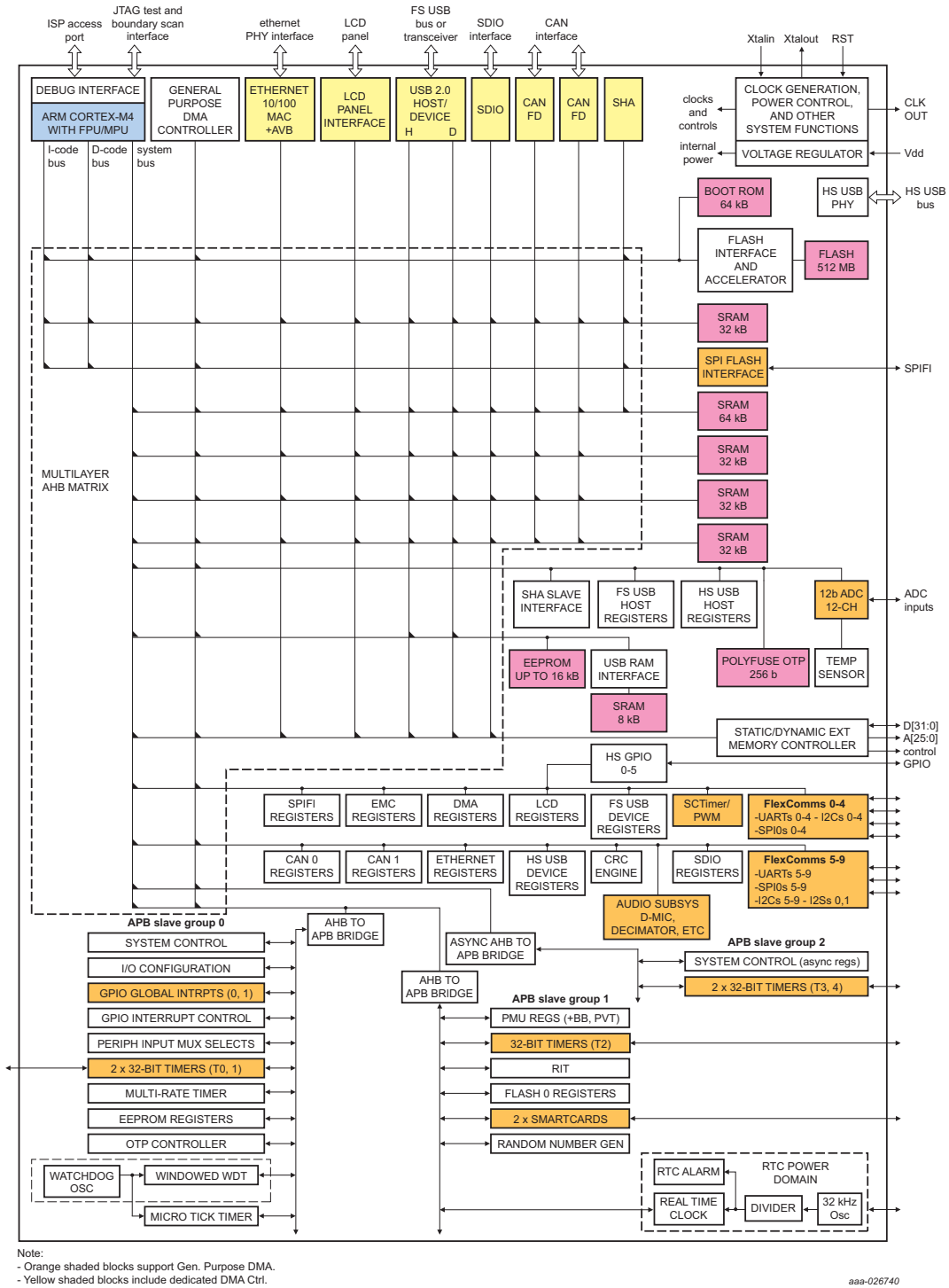


Fig 1. Block diagram

1.4 Architectural overview

The ARM Cortex-M4 includes three AHB-Lite buses, one system bus and the I-code and D-code buses. One bus is dedicated for instruction fetch (I-code), and one bus is dedicated for data access (D-code). The use of two core buses allows for simultaneous operations if concurrent operations target different devices.

A multi-layer AHB matrix connects the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters. More information on the multilayer matrix can be found in Section 2.1.4. Connections in the multilayer matrix are shown in Figure 1. Note that while the AHB bus itself supports word, halfword, and byte accesses, not all AHB peripherals need or provide that support.

APB peripherals are connected to the AHB matrix via two APB buses using separate slave ports from the multilayer AHB matrix. This allows for better performance by reducing collisions between the CPU and the DMA controller, and also for peripherals on the asynchronous bridge to have a fixed clock that does not track the system clock. Note that APB, by definition, does not directly support byte or halfword accesses.

1.5 ARM Cortex-M4 processor

The Cortex-M4 is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The Cortex-M4 offers a Thumb-2 instruction set, low interrupt latency, interruptible/continuable multiple load and store instructions, automatic state save and restore for interrupts, tightly integrated interrupt controller, multiple core buses capable of simultaneous accesses, and a floating point unit.

A 3-stage pipeline is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

Information about Cortex-M4 configuration options can be found in Chapter 52.

1.6 On-chip flash memory system

The LPC546xx contains up to 512 kB of on-chip flash memory. A flash memory accelerator maximizes performance for CPU accesses. This memory may be used for both code and data storage. Programming of the flash memory may be accomplished in several ways. It may be programmed In System via the serial port. The application program may also erase and/or program the flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc.

1.7 On-chip Static RAM

The LPC546xx contains up to 200 kB of on-chip static RAM. 8 kB of SRAM is intended for the use by high-speed USB host/device peripherals. There are 6 banks of SRAM and each bank is placed at a separate slave port. This architecture allows the possibility for CPU and DMA accesses to be separated in such a way that there are few or no delays for the bus masters. It also allows separation of data for different peripherals functions, in

order to improve system performance. For example, LCD DMA can be occurring in one SRAM while Ethernet DMA is occurring in another, all while the CPU is using the Main SRAM for data and/or instruction access.

1.8 On-chip EEPROM

The LPC546xx contains up to 16 kB of on-chip EEPROM memory. The EEPROM is accessible only by the CPU.

1.9 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC54605J256ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54605J512ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54605J256BD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC54605J512BD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC54605J256ET100	TFBGA100	plastic thin fine-pitch ball grid array package; 100 balls; body 9 × 9 × 0.7 mm	SOT926-1
LPC54605J512ET100	TFBGA100	plastic thin fine-pitch ball grid array package; 100 balls; body 9 × 9 × 0.7 mm	SOT926-1
LPC54606J256ET100	TFBGA100	plastic thin fine-pitch ball grid array package; 100 balls; body 9 × 9 × 0.7 mm	SOT926-1
LPC54606J256BD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC54606J256ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54606J512ET100	TFBGA100	plastic thin fine-pitch ball grid array package; 100 balls; body 9 × 9 × 0.7 mm	SOT926-1
LPC54606J512BD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC54606J512BD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC54607J256ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54607J512ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54607J256BD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC54608J512ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54608J512BD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC54616J256ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54616J512ET100	TFBGA100	plastic thin fine-pitch ball grid array package; 100 balls; body 9 × 9 × 0.7 mm	SOT926-1
LPC54616J512BD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC54616J512BD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC54618J512ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3
LPC54618J512BD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC54628J512ET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 ´ 12 ´ 0.8 mm	SOT570-3

1.9.1 Ordering options

Table 2. Ordering options

Type number	Package Name	Frequency/MHz	Flash/kB	SRAM/kB	FS USB	HS USB	Ethernet AVB	Classic CAN	CAN FD	LCD	Flexcomm Interface	EMC data bus width (bit)	GPIO	SHA
LPC54628 devices (HS/FS USB, Ethernet, CAN FD, CAN 2.0, LCD, SHA)														
LPC54628J512ET180	TFBGA180	220	512	200	yes	yes	yes	yes	yes	yes	10	8/16	145	yes
LPC54618 devices (HS/FS USB, Ethernet, CAN FD, CAN 2.0, LCD)														
LPC54618J512ET180	TFBGA180	180	512	200	yes	yes	yes	yes	yes	yes	10	8/16	145	no
LPC54618J512BD208	LQFP208	180	512	200	yes	yes	yes	yes	yes	yes	10	8/16/32	171	no
LPC54616 devices (HS/FS USB, Ethernet, CAN FD, CAN 2.0)														
LPC54616J256ET180	TFBGA180	180	256	136	yes	yes	yes	yes	yes	no	10	8/16	145	no
LPC54616J512BD208	LQFP208	180	512	200	yes	yes	yes	yes	yes	no	10	8/16/32	171	no
LPC54616J512ET100	TFBGA100	180	512	200	yes	yes	yes	yes	yes	no	9	8/16	64	no
LPC54616J512BD100	LQFP100	180	512	200	yes	yes	yes	yes	yes	no	9	8/16	64	no
LPC54608 devices (HS/FS USB, Ethernet, CAN 2.0, LCD)														
LPC54608J512ET180	TFBGA180	180	512	200	yes	yes	yes	yes	no	yes	10	8/16	145	no
LPC54608J512BD208	LQFP208	180	512	200	yes	yes	yes	yes	no	yes	10	8/16/32	171	no
LPC54607 devices (HS/FS USB, LCD)														
LPC54607J256ET180	TFBGA180	180	256	136	yes	yes	no	no	no	yes	10	8/16	145	no
LPC54607J512ET180	TFBGA180	180	512	200	yes	yes	no	no	no	yes	10	8/16	145	no
LPC54607J256BD208	LQFP208	180	256	136	yes	yes	no	no	no	yes	10	8/16/32	171	no
LPC54606 devices (HS/FS USB, Ethernet, CAN 2.0)														
LPC54606J256ET180	TFBGA180	180	256	136	yes	yes	yes	yes	no	no	10	8/16	145	no
LPC54606J512BD208	LQFP208	180	512	200	yes	yes	yes	yes	no	no	10	8/16/32	171	no
LPC54606J256ET100	TFBGA100	180	256	136	yes	yes	yes	yes	no	no	9	8/16	64	no
LPC54606J512ET100	TFBGA100	180	512	200	yes	yes	yes	yes	no	no	9	8/16	64	no
LPC54606J256BD100	LQFP100	180	256	136	yes	yes	yes	yes	no	no	9	8/16	64	no
LPC54606J512BD100	LQFP100	180	512	200	yes	yes	yes	yes	no	no	9	8/16	64	no
LPC54605 devices (HS/FS USB)														
LPC54605J256ET180	TFBGA180	180	256	136	yes	yes	no	no	no	no	10	8/16	145	no
LPC54605J512ET180	TFBGA180	180	512	200	yes	yes	no	no	no	no	10	8/16	145	no
LPC54605J256BD100	LQFP100	180	256	136	yes	yes	no	no	no	no	9	8/16	64	no
LPC54605J512BD100	LQFP100	180	512	200	yes	yes	no	no	no	no	9	8/16	64	no
LPC54605J256ET100	TFBGA100	180	256	136	yes	yes	no	no	no	no	9	8/16	64	no
LPC54605J512ET100	TFBGA100	180	512	200	yes	yes	no	no	no	no	9	8/16	64	no

2.1 General description

The LPC546xx incorporates several distinct memory regions. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset.

The APB peripheral area (detailed in Figure 3) is divided into fixed 4 KB slots to simplify addressing.

The registers incorporated into the CPU, such as NVIC, SysTick, and sleep mode control, are located on the private peripheral bus.

2.1.1 Memory map and peripheral addressing

The ARM Cortex-M4 processor has a single 4 GB address space. The following table shows how this space is used on the LPC546xx.

Table 3. Memory usage and details

Address range	General Use	Address range details and description	
0x0000 0000 to 0x1FFF FFFF	On-chip non-volatile memory	0x0000 0000 - 0x0007 FFFF	Flash memory (512 KB).
	Boot ROM	0x0300 0000 - 0x0300 FFFF	Boot ROM with flash services in a 64 KB space.
	SRAMX	0x0400 0000 - 0x0400 7FFF	I&D SRAM bank (32 KB).
	SPI Flash Interface (SPIFI)	0x1000 0000 - 0x17FF FFFF	SPIFI memory mapped access space (128 MB).
0x2000 0000 to 0x3FFF FFFF	SRAM Banks	0x2000 0000 - 0x2002 7FFF	SRAM banks (160 KB).
	SRAM bit band alias addressing	0x2200 0000 - 0x23FF FFFF	SRAM bit band alias addressing (32 MB)
0x4000 0000 to 0x7FFF FFFF	APB peripherals	0x4000 0000 - 0x4001 FFFF	APB slave group 0 up to 32 peripheral blocks of 4 KB each (128 KB).
		0x4002 0000 - 0x4003 FFFF	APB slave group 1 up to 32 peripheral blocks of 4 KB each (128 KB).
		0x4004 0000 - 0x4005 FFFF	APB asynchronous slave group 2 up to 32 peripheral blocks of 4 KB each (128 KB).
	AHB peripherals	0x4008 0000 - 0x400B FFFF	AHB peripherals (256 KB).
	USB SRAM	0x4010 0000 - 0x4010 1FFC	USB SRAM (8 KB)
	Peripheral bit band alias addressing	0x4200 0000 - 0x43FF FFFF	Peripheral bit band alias addressing (32 MB)

Table 3. Memory usage and details

Address range	General Use	Address range details and description	
0x8000 0000 to 0xDFFF FFFF	Off-chip Memory via the External Memory Controller	Four static memory chip selects:	
		0x8000 0000 - 0x83FF FFFF	Static memory chip select 0 (up to 64 MB)[1]
		0x8800 0000 - 0x8BFF FFFF	Static memory chip select 1 (up to 64 MB)[2]
		0x9000 0000 – 0x93FF FFFF	Static memory chip select 2 (up to 64 MB)
		0x9800 0000 - 0x9BFF FFFF	Static memory chip select 3 (up to 64 MB)
		Four dynamic memory chip selects:	
		0xA000 0000 - 0xA7FF FFFF	Dynamic memory chip select 0 (up to 256MB)
		0xA800 0000 - 0xAFFF FFFF	Dynamic memory chip select 1 (up to 256MB)
		0xB000 0000 - 0xB7FF FFFF	Dynamic memory chip select 2 (up to 256MB)
		0xB800 0000 - 0xBFFF FFFF	Dynamic memory chip select 3 (up to 256MB)
0xE000 0000 to 0xE00F FFFF	Cortex-M4 Private Peripheral Bus	0xE000 0000 - 0xE00F FFFF	Cortex-M4 related functions, includes the NVIC and System Tick Timer.

[1] Can be up to 256 MB, upper address 0x8FFF FFFF, if the address shift mode is enabled. See EMCSYSCTRL register bit 0 (Section 7.5.71).

[2] Can be up to 128 MB, upper address 0x97FF FFFF, if the address shift mode is enabled. See EMCSYSCTRL register bit 0 (Section 7.5.71).

2.1.2 SRAM

The Main SRAM is comprised of up to a total 160 KB of contiguous, on-chip static RAM memory (this is in addition to SRAMX as noted in the next section below, so the total device SRAM can be up to 200 KB). The Main SRAM is further divided to allow for more control of power usage when less SRAM is required: SRAM0 (up to 64 KB), SRAM1 (up to 32 KB), SRAM2 (up to 32 KB), and SRAM3 (up to 32 KB). Each SRAM has a separate clock control and power switch, see Section 7.5.19 “AHB Clock Control register 0” and Section 7.5.84 “Power Configuration register 0”.

An additional on-chip static RAM memory is available that is not contiguous to the main SRAM. This RAM is called SRAMX, and resides on the local buses (I-Code and D-Code) of the Cortex-M4, and on the main bus of the Cortex-M0+. This RAM can be used, for example, as the location for the program stack, common data, or any other use where a separate access away from the Main SRAM has an advantage. SRAMX can be disabled or enabled in the SYSCON block to save power. See Section 7.5.84 “Power Configuration register 0”. Also, an additional 8 KB SRAM is available for USB operations or it can be used as a general purpose SRAM when not used for USB.

Table 4. SRAM configuration

	SRAMX	SRAM0	SRAM1	SRAM2	SRAM3	USB RAM
(total SRAM = up to 200 kB)						
Size	Up to 32 kB	Up to 64 kB	Up to 32 kB	Up to 32 kB	Up to 32 kB	8 kB
Address range	Begins at 0x0400 0000	Begins at 0x2000 0000	Begins at 0x2001 0000	Begins at 0x2001 8000	Begins at 0x2002 0000	If present, begins at 0x4010 0000

2.1.2.1 SRAM usage notes

Although always contiguous on all LPC546xx devices, SRAM0, SRAM1, SRAM2, and SRAM3 are placed on different AHB matrix ports. This allows user programs to potentially obtain better performance by dividing RAM usage among the ports. For example, simultaneous access to SRAM0 by the CPU and SRAM1 by the system DMA controller does not result in any bus stalls for either master.

Generally, data being communicated via peripherals will be accessed by the CPU at some point, even when peripheral data is mainly being transferred via DMA. So, in order to minimizing data read/write stalls, data buffers may be placed in RAMs on different AHB matrix ports. For instance, if DMA is writing to one buffer on a specific AHB matrix port while the CPU is reading data from a buffer on a different AHB matrix port, there is no stall for either the CPU or the DMA. Sequences of data from the same peripheral could be alternated between RAM on each port. This could be helpful if DMA fills or empties a RAM buffer, then signals the CPU before proceeding on to a second buffer. The CPU would then tend to access the data while the DMA is using the other RAM.

2.1.2.2 Bit-band addressing

The ARM Cortex-M4 CPU provides a bit-band addressing feature. This offers efficient bit accesses to selected memory regions.

Bits in the address region 0x2000 0000 to 0x200F FFFF (bits addressed at addresses 0x2200 0000 to 0x23FF FFFF) include the entire main SRAM area (does not include SRAMX which is at a lower address).

Bits in the address region 0x4000 0000 to 0x400F FFFF (bits addressed at addresses 0x4200 0000 to 0x43FF FFFF) include all AHB and APB peripherals. This space does not include the USB RAM and the EEPROM.

Reads from bit-band addresses return the respective bit from the bit-band region. Writes perform an atomic read-modify-write on the respective bit of the bit-band region. For details, see the ARM Cortex-M4 technical reference manual.

To calculate a bit band address:

Bit address = (Byte offset within bit-band space * 32) + (bit number in byte * 4) + bit-band base address

Where:

- Bit addressable RAM base address = 0x2000 0000
- AHB/APB peripheral base address = 0x4000 0000
- RAM bit-band base address = 0x2200 0000

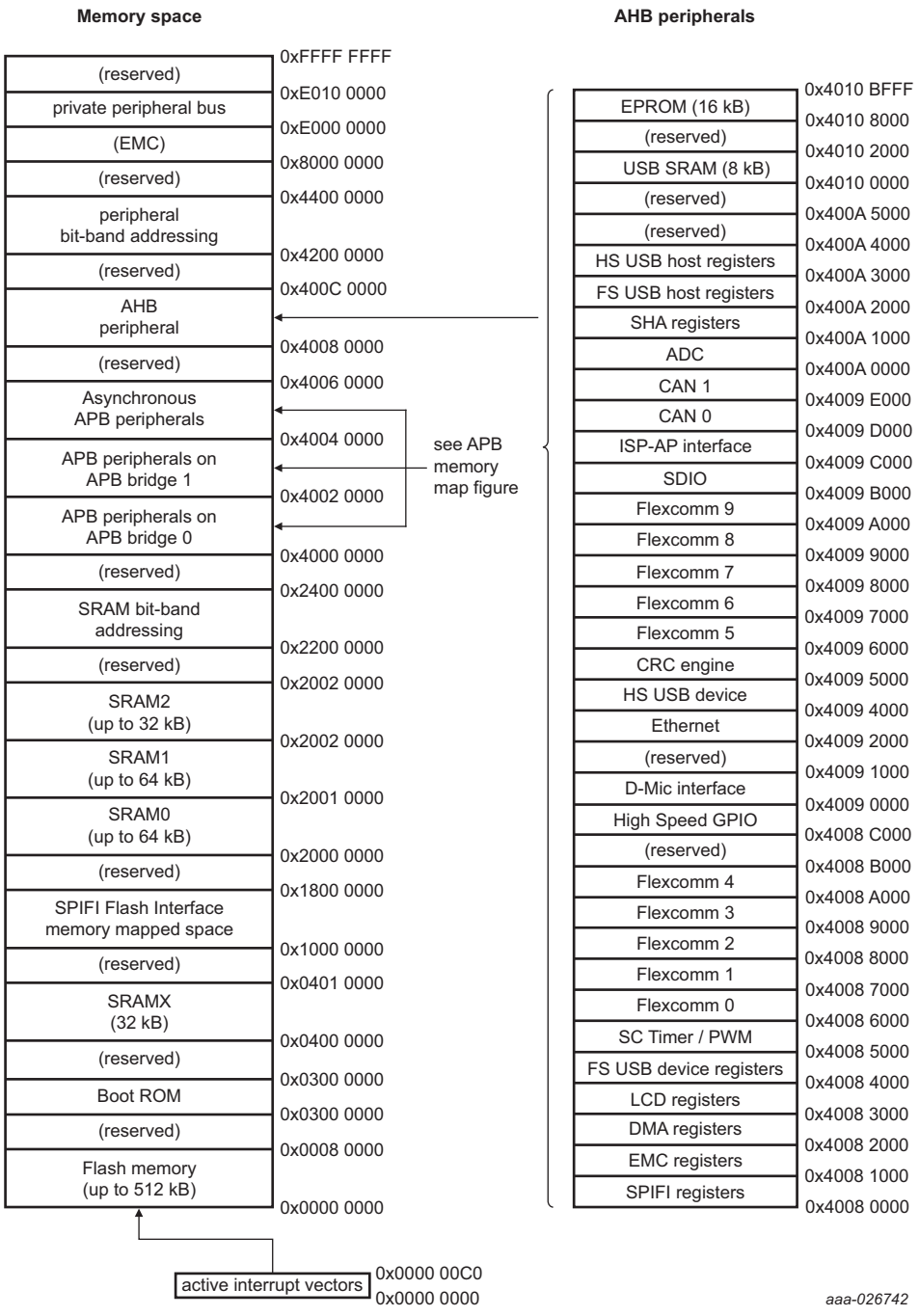
- AHB/APB peripheral bit-band base address = 0x4200 0000
- Byte offset within bit-band space: the offset within the related bit-band space (bit addressable RAM space or AHB/APB peripheral space). For example, the offset of bit addressable RAM address 0x201F A127 = 0x200F A127 - 0x2000 0000 = 0x000F A127

For example, the bit-band address of bit 5 of RAM address 0x201F A127 = ((0x201F A127 - 0x2000 0000) * 32) + 5 * 4) + 0x2200 0000 = 0x1F4 24E0 + 14 + 0x2200 0000 = 0x23F4 24F4.

Remark: Because bit-band operations are implemented as read-modify-write operations, and appear on the AHB bus in that manner, some uses of bit-banding may not work as intended. For example, if a peripheral register contains several write-one-to-clear status flags, attempting to clear one such flag using bit-banding will actually clear all such flags that read as a one in that register.

2.1.3 Memory mapping

The overall memory map is shown in Figure 2 “Main memory map”. Details of APB peripheral mapping are shown in Figure 3 “APB memory map”.



The private peripheral bus includes CPU peripherals such as the NVIC, SysTick, and the core control registers.

Fig 2. Main memory map

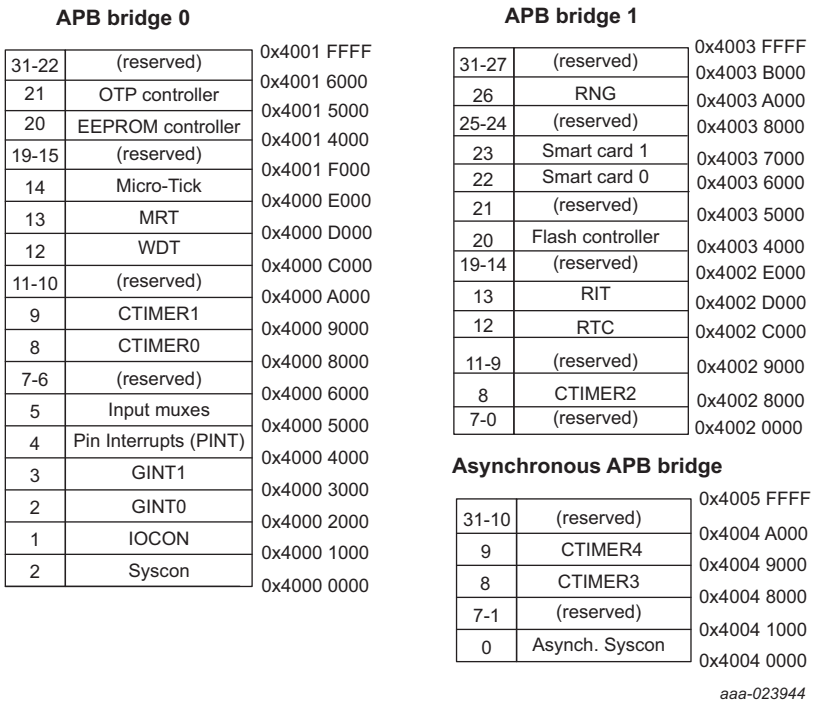


Fig 3. APB memory map

2.1.4 AHB multilayer matrix

The LPC546xx uses a multi-layer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports of the matrix to be accessed simultaneously by different bus masters. Figure 1 shows details of the potential matrix connections.

2.1.5 Memory Protection Unit (MPU)

The Cortex-M4 processor has a memory protection unit (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are critical in many embedded applications.

The MPU register interface is located on the private peripheral bus and is described in detail in Ref. 1 “Cortex-M4 TRM”.

3.1 Features

- 64 KB on-chip boot ROM
- Contains the boot loader with In-System Programming (ISP) facility and the following APIs:
  - In-Application Programming (IAP) of flash memory.
  - ROM-based USB drivers (HID, CDC, MSC, and DFU). Flash updates via USB is supported.
  - Supports booting from valid user code in flash, USART, SPI, and I^2C.
  - Legacy, Single, and Dual image boot.
  - OTP API for programming OTP memory.
  - Random Number Generator (RNG) API.

3.2 Pin description

The part supports ISP using various peripherals.

Table 5. ISP modes

Boot source	ISP-2	ISP-1	ISP-0	Description
Flash, no ISP	1	1	1	ISP is bypassed. The device boots from flash if valid user code is detected.
UART/ I2C / SPI (auto detect)	1	1	0	The first valid probe message on USART, I^2C, or SPI locks in that interface.
USB 0 Mass Storage Device Class (MSC)	1	0	1	Allow programming flash as USB 0 MSC.
Reserved	1	0	0	Reserved
USB 1 MSC	0	1	1	Allow programming flash as USB 1 MSC.
USB 0 Device Firmware Update (DFU)	0	1	0	Allow programming flash as USB 0 DFU.
USB 1 DFU	0	0	1	Allow programming flash as USB 1 DFU.

If the USART/I2C/SPI ISP mode is selected, Flexcomm Interface 0 is used as the USART ISP, Flexcomm Interface 1 is used as the I2C ISP, and Flexcomm Interface 3 is used as the SPI ISP. Table 6 shows the ISP pin assignments and is the default pin assignment used by the ROM code that cannot be changed.

Table 6. ISP pin assignments

ISP pin	Port pin assignment
ISP_0	PIO0_4
ISP_1	PIO0_5
ISP_2	PIO0_6
USART mode	
FC0_TXD	PIO0_30

Table 6. ISP pin assignments *...continued*

ISP pin	Port pin assignment
FC0_RXD	PIO0_29
I2C mode	
FC1_SDA	PIO0_13
FC1_SCL	PIO0_14
SPI mode	
FC3_SCK	PIO0_0
FC3_SSELO	PIO0_1
FC3_MISO	PIO0_2
FC3_MOSI	PIO0_3
USB 0 mode	
USB0_VBUS	PIO0_22
USB0_DP	-
USB0_DM	-
USB 1 mode	
USB1_VBUS	Dedicated pin per package.
USB1_DP	-
USB1_DM	-

3.3 General description

The boot loader controls initial operation after reset and also provides the means to program the flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device, or programming of the flash memory by the application program in a running system.

The boot loader code is executed every time the part is powered ON or reset (see Figure 4). Depending on the settings in ECRP (see Chapter 43), the loader may enter the ISP command handler if any of the ISP pins are held low, or there is no valid image to boot.

The boot loader version can be read by ISP/IAP calls (see Section 5.4.5.13 “ISP Read Boot code version number” or Section 5.6.6 “Read boot code version number”).

Assuming that power supply pins are at their nominal levels when the rising edge on RESET pin is generated, it may take up to 6 ms before the boot pins are sampled and the decision whether to continue with user code or ISP handler is made. If the boot pins are sampled LOW and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If a valid user program is found, then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

See Chapter 5 “LPC546xx ISP and IAP” for ISP and IAP commands.

3.3.1 Boot process flowchart

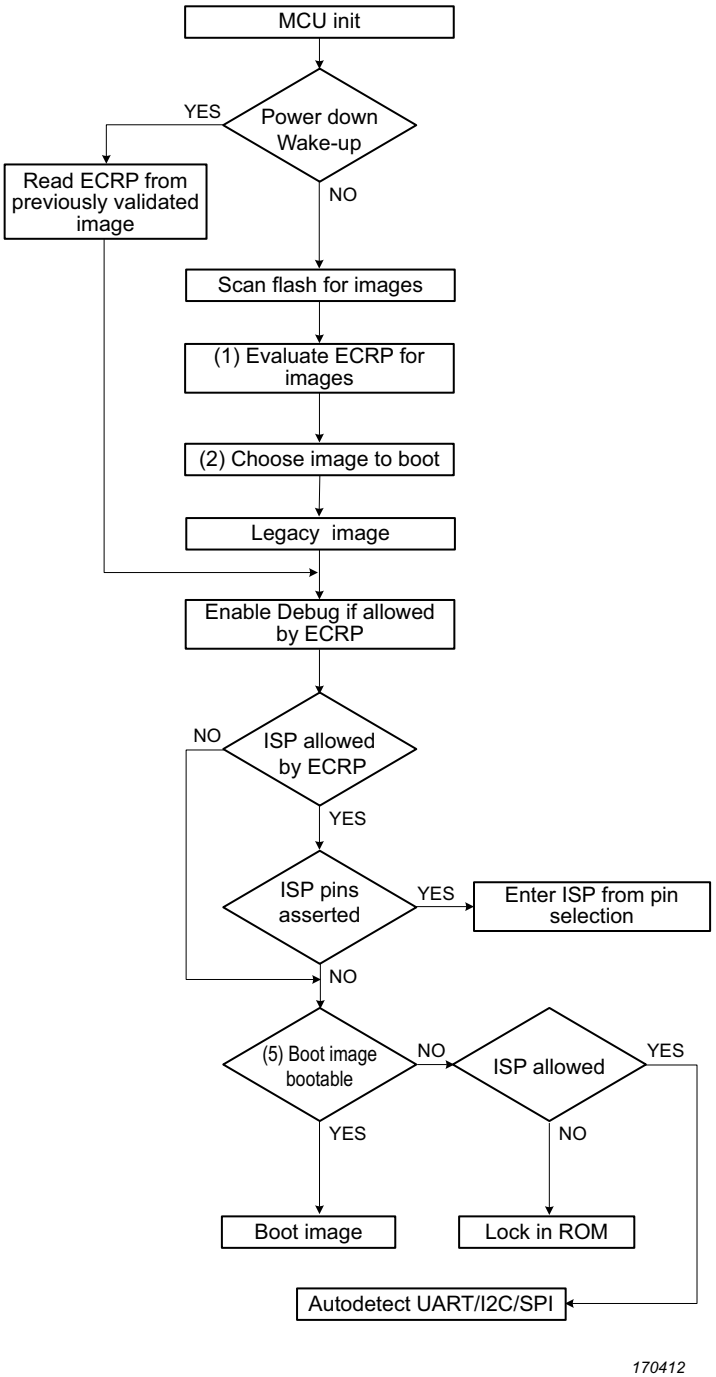


Fig 4. Legacy image boot process flowchart

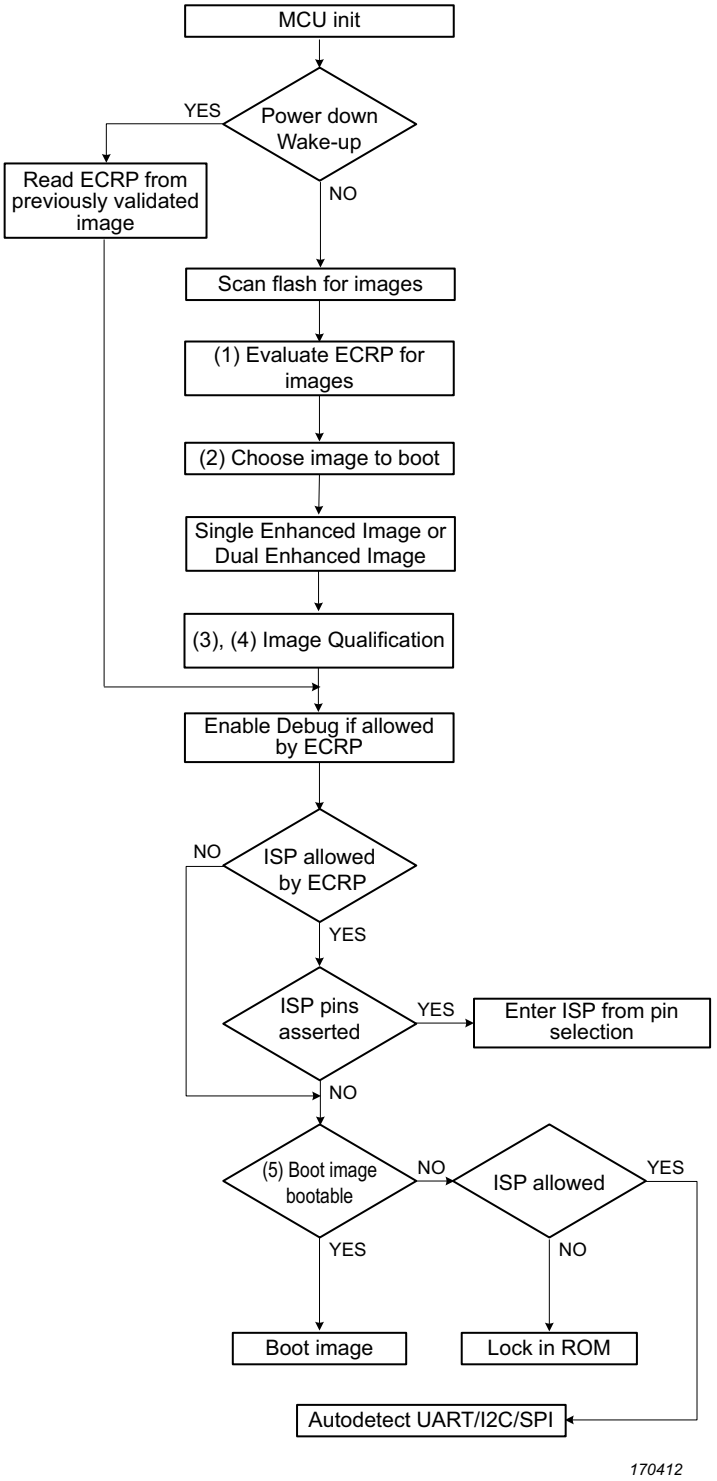


Fig 5. Single Enhanced image and Dual Enhanced image boot process flowchart

1. ECRP is checked using most restrictive combination of OTP and flash feature bits.

2. Image located in sector 0 is always chosen as boot image unless a valid dual_en image is located elsewhere and sector 0 does not contain a valid single_en image. If two dual_en images are present and both are valid (see List item 4), then the image with highest version number is selected for booting. If both versions are same then the image at "sector 0" will be selected for booting. If no boot image is present and if OTP allows ISP, auto-detect UART/I2C/SPI ISP mode will be entered. If OTP does not allow ISP, then the part will lock in ROM. In this case, if SWD is enabled in OTP, the Flash may be programmed via SWD. However, if SWD is also not enabled, the part is permanently disabled.
3. CheckSum and boot_blk (if image is not legacy) are verified.
4. Selected boot image will not be bootable if any of the checks fail
   Vector checksum, *[boot block]. * [] Denotes optional items.
5. If ISP is set to 'Disabled' and the boot image(s) are not bootable (fail boot check), then the part will lock in ROM. This is only recoverable if SWD is enabled: attach the SWD debugger and reprogram the Flash. However, if both ISP and SWD are disabled, and no valid bootable image is present, the part will be permanently disabled. Once programmed, OTP bits are permanent.

3.3.2 Boot flow

3.3.2.1 MCU Init

The first phase of the boot loader is to set configuration values for the peripherals. If the MCU is waking from a low-power state it will by-pass the image scanning and qualification steps and execute the previously validated image; otherwise the full image scan, qualification, and boot process are started.

3.3.2.2 Legacy images

A legacy image is located in sector 0 and does not contain an enhanced image marker or an image header structure. The only requirement for a legacy image is that the checksum of the first 8 32-bit words of the image must add to 0x00000000.

3.3.2.3 Enhanced images

Single Enhanced (SE) and Dual Enhanced (DE) images differ from a legacy image in that they add an enhanced marker value at offset 0x24 in the image and an image header that optionally enables the image CRC capability and adds an image version number. Offset 0x28 in the image must point to the location of a valid image header in the image.

3.3.2.3.1 Single Enhanced (SE) images

The Single Enhanced image can only be located at sector 0. For an image to be Single Enhanced, it needs to have the Single Enhanced image marker value (0xEDDC9494) at offset 0x24. It must also have a valid image header in the image pointed to at offset 0x28. See Table 7 "Image type considerations".

3.3.2.3.2 Dual Enhanced (DE) images

The Dual Enhanced image type is similar to the Single Enhanced image type except that it adds the ability to select and boot 1 of 2 images in flash based on the highest version number in the image header. Up to 2 Dual Enhanced images may be located in flash. If only 1 Dual Enhanced image is located in flash, it may be located at the start of any sector. If 2 dual enhanced images are in flash, one of the images must be at sector 0 and

the other must start at any sector greater than 0. If 2 images exist, the version number is checked on both images. If the version numbers are the same, the image at sector 0 is executed. If the version numbers are not the same, the image with the highest version number is executed. If more than 2 Dual Enhanced images exist, only the first 2 are checked.

The Dual Enhanced image facility may be used to locate a boot loader at sector 0 and an application at another sector. If the application is ever erased and the chip is reset, the image at sector 0 will execute following reset.

For an image to be Dual Enhanced, it needs to have the Dual Enhanced image marker value (0x0FFEB6B6) at offset 0x24. It must also have a valid image header in the image pointed to at offset 0x28. See Table 7 “Image type considerations”.

One of the images must start at sector 0. The other image must be aligned on a sector boundary and linked to run at the sector start address where the image is programmed.

3.3.2.4 When to use legacy, single enhanced or dual enhanced images

Each image type has its own advantages and disadvantages. The table below outlines the advantages and disadvantages of each image type and the best use cases for them.

Table 7. Image type considerations

Image type	Advantage	Disadvantage	Best use
Legacy	Simple to create, works with all tool chains without special processing.	Slightly longer boot time than Single Enhanced images.	When debugging an application.
Single Enhanced	Fastest boot time (without CRC), Supports optional CRC checking, more secure.	More complex to build than legacy images, CRC generation can be complex, may increase boot time if CRC is used.	Production systems that benefit from fastest boot time (no CRC).
Dual Enhanced	Single Enhanced advantages plus up to 2 selectable images based on version, more secure.	More complex to build than legacy images, CRC generation can be complex, may increase boot time.	Systems that require boot image cycling based on version.

3.3.2.5 Image scanning

The boot loader begins scanning for user images by examining the “signature” value located at 0x0000 0024 in Flash sector 0.

- If the value matches 0xEDDC9494 (single enhanced image) then a single enhanced image is considered found.
  - When a single enhanced image is found in sector 0 (the only valid location for a single enhanced image), no other sectors are checked and the single enhanced image is validated for booting.
- If the value matches 0x0FFEB6B6 (dual enhanced image), then a dual enhanced image is considered found.
- Whether or not a dual enhanced image was found at sector 0, the remaining sectors of the Flash are scanned for a dual enhanced image.
  - If more than one image header is found in the remaining sectors of Flash (beyond sector 0) it is an error condition: The ECRP is set to the most restrictive combination of OTP and the ECRP of the images found and then the code locks in

ROM. If this happens, the part may only be recovered if the combined ECRP enables ISP or SWD, in which case the flash can be erased and reprogrammed; otherwise the part is permanently disabled.

- If one dual enhanced image is found it is validated for booting. If the image validation fails: The ECRP is set to the most restrictive combination of OTP and the image ECRP and then the code locks in ROM. If this happens the part may only be recovered if the combined ECRP enables ISP or SWD, in which case the flash can be erased and reprogrammed; otherwise the part is permanently disabled.
- If two dual enhanced images are found (one must be located at sector 0), the image with the highest revision number is selected for qualification first. If the qualification fails, the alternate image is selected for qualification. If both images fail validation: The ECRP is set to the most restrictive combination of OTP and the ECRP of the images found and then the code locks in ROM. If this happens the part may only be recovered if the combined ECRP enables ISP or SWD, in which case the flash can be erased and reprogrammed; otherwise the part is permanently disabled.
- A legacy image is assumed to be at sector zero and executed if all three of the following are true: the vector checksum of the first 8 vectors of sector 0 is 0, the image at sector 0 does not have single enhanced or dual enhanced signatures, and no valid dual enhanced image is found in the remaining Flash sectors.
- If the Flash is erased or no valid image is found, the ECRP will still be read from sector 0 (which will be the ‘least restrictive’ setting for an erased Flash) and combined with the OTP ECRP bits, with OTP taking priority when restrictions are set in OTP. The factory default OTP setting is ‘least restrictive’, with ISP and SWD enabled. If ISP is enabled it will enter ISP mode. If ISP is not enabled it will lock in ROM. If SWD is enabled the flash may be programmed via SWD; otherwise with both ISP and SWD disabled in OTP and no valid image to boot, the part is permanently disabled.

3.3.2.6 Modifications to startup code to enable enhanced boot support

Several modifications need to be made to the startup code to enable enhanced boot support. The value at offset 0x24 in the image must contain an enhanced image marker and the value at offset 0x28 must point to a valid image header in the image. See the box below for an example setup using a Single Enhanced image. Changes are in bold and made to the vector table are of the startup code.

```
; Vector Table Mapped to Address 0 at Reset
          AREA    RESET, DATA, READONLY
          EXPORT  __Vectors
__Vectors  DCD     __initial_sp  ; Top of Stack
          DCD     Reset_Handler ; Reset Handler
          DCD     NMI_Handler
          DCD     HardFault_Handler
          DCD     MemManage_Handler
          DCD     BusFault_Handler
          DCD     UsageFault_Handler
__vector_table_0x1c
          DCD     0           ; Checksum of the first 7 words
          DCD     0
          DCD     0xEDDC9494  ; Enhanced image marker, set to 0x0 for legacy boot
          IMPORT  imageHeader
```

```
DCD     imageHeader ; Pointer to enhanced image header, use 0x0 for legacy
                      boot
```

The image header for the startup code must be located somewhere in non-volatile memory. A simple image header that doesn't perform CRC is shown below. All field sizes are 32-bits.

```
/* Image header */
const IMAGEHEADER_T imageHeader = {
        IMAGE_ENH_BLOCK_MARKER, /* Required marker for image header */
                IMG_NO_CRC,     /* No CRC, makes development easier */
                0x00000000,     /* crc32_len */
                0x00000000,     /* crc32_val */
                0x00000000      /* version */
};
```

3.3.2.7 Image qualification

Qualification of all image types begins by adding the first 8 vectors and comparing the result to 0. If the sum is not 0, then the image is considered invalid. If the sum is 0, the qualification for the legacy image type ends here. For enhanced images (both single and dual), qualification continues by reading the value from the 11th vector (offset 0x0000 0028) and using it as a pointer to a boot block structure. See Table 8.

Table 8. Boot block structure (Image header)

Boot block offset	Description
0x00	Header marker set to 0xFEED A5A5
0x04	Image Type (NORMAL = 0, or NO_CRC = 1)
0x08	Reserved
0x0C	Image length: This length should be actual length – 4 if CRC value field falls within the length.
0x10	CRC value
0x14	Version (only applicable to DUAL_ENH image type.

The pointer is read and the first entry in the structure must match the 0xFEED A5A5 value. Qualification continues by examining the second entry in the boot block structure. If the value is 0, then the fourth entry is used as the length to perform a CRC on. The value is used as the length to perform a CRC ON. This length is set by the user as necessary to meet the application requirements and can be set to a length that allows calculating the CRC on the entire image or on some subset of the image. The CRC calculation begins at offset 0x0 from the beginning of the image sector and continues up to the number of bytes specified by the length. The length does not include the four bytes that make up the CRC value field, which means that the CRC is not calculated on the four bytes of the CRC value field. The result is then compared to the fifth entry in the structure and the image is considered valid if a match exists, otherwise the image is considered invalid. If the second entry = 1, then no CRC is performed and the image is considered valid.

3.3.2.8 Vectors

The vector block contains the standard ARM vector table with the following components.

Table 9. Vector table components

Vector offset	Description
0x0 - 0x18	Vectors 1 - 7 (standard ARM vectors).
0x1C	Vector checksum (set by tool vendors).
0x20	ECRP
0x24	Image Type (not applicable to legacy image) SINGLE_EN = 0xEDDC9494 DUAL_EN = 0x0FFEB6B6
0x28	Boot Block Pointer (not applicable to legacy image). Pointer to beginning of Boot bock.

For legacy images, Image Type and Boot block components should be set to either 0xFFFF FFFF or 0x0. Otherwise they should contain valid signature and pointer values. Offset 0x20 should be set to 0xFFFF FFFF or 0x0 (if “Allow 0 in ECRP” OTP bit is set) or contain a valid ECRP value.

See Table 8 “Boot block structure (Image header)”.

3.3.2.9 Enhanced Code Read Protection (ECRP) calculation

After the image qualification is complete, ECRP is calculated by reading the 9th vector (offset 0x0000 0020) from the boot image and combining it with the value read from OTP (see “Chapter Chapter 43 “LPC546xx Enhanced Code Read Protection (ECRP)”). Any restriction enabled in OTP will override settings read from the flash location. Sector 0 offset 0x0000 0020 will be read in the event that no valid image is qualified.

3.3.2.10 ISP entry

The boot loader continues by checking the ECRP settings. If ISP is allowed and the watchdog was not the source of the reset, the ISP pins are then sampled and ISP mode is entered if any pins are asserted (see Table 5 “ISP modes”).

3.3.2.11 Image boot

If a valid boot image was qualified, the boot loader then transfers control to the reset vector of the boot image. In the event that no image is qualified, the boot loader (if allowed by ECRP) automatically enters the ISP mode by selecting USART/ I2C/ SPI as the mode (see Chapter 5 “LPC546xx ISP and IAP”).

4.1 How to read this chapter

The ROM-based FRO high frequency output API call is available on all parts.

4.2 Features

- Selects FRO high frequency output to 48 MHz or 96 MHz.

4.3 General description

Control of FRO output frequency can be configured through a call to the ROM.

The set_fro_frequency API call must be used to select the required FRO high frequency output to 48 MHz or 96 MHz. This is performed by executing a function, which is pointed by a pointer within the ROM Driver Table (address location is: 0x3007933).

Remark: Disable all interrupts before making calls to the FRO API. The interrupts can be re-enabled after the FRO API call is completed.

4.4 API description

The FRO_HF API provides a function to configure the FRO high frequency output to 48 MHz or 96 MHz. The API can be called in the application code through a simple API call. An example is provided with the SDK software package on nxp.com

The following function prototypes are used:

Table 10. FRO API call

Function prototype	API description	Reference
`void set_fro_frequency(uint32_t iFreq);`	Setup the FRO high frequency output for either 48 MHz or 96 MHz. Updates the correct trim value and settings for high frequency FRO operation.	4.4.1

4.4.1 set_fro_frequency

This routine sets up the FRO high frequency output. The selected operating frequency must be either 48 MHz or 96 MHz, which are the two potential high frequency outputs of the FRO. The requested frequency is set up and the appropriate factory trim value will be used. Use the procedure in Section 7.5.77 “FRO Control register” to select the required higher frequency.

Table 11 shows the set_fro_frequency.

Table 11. set_fro_frequency

Routine	set_fro_frequency
Prototype	void set_fro_frequency(uint32_t iFreq);
Input parameter	**Param0 —** Required frequency (in Hz).
Return	None.
Description	Setup the FRO high frequency output for the selected frequency, either 48 MHz or 96 MHz.

4.4.1.1 Param0: frequency

The frequency is the required high frequency output clock for the FRO, 48 MHz or 96 MHz.

5.1 How to read this chapter

All LPC546xx devices include ROM-based services for programming and reading the flash memory in addition to other functions. In-System Programming works on an unprogrammed or previously programmed device using one from a selection of hardware interfaces. In-Application Programming allows application software to do the same kinds of operations.

See specific device data sheets for different flash configurations.

Remark: In addition to the ISP and IAP commands, the flash configuration register (FLASHCFG) can be accessed in the SYSCON block to configure flash memory access times, see Section 7.5.64.

5.2 Features

- In-System Programming: In-System programming (ISP) is programming or re-programming the on-chip flash memory, using the boot loader software through the USART, I2C, SPI, or USB. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.
- Small size (256 byte) page erase programming.

5.3 General description

5.3.1 Boot loader

For the boot loader operation and boot pin, see Chapter 3 “LPC546xx Boot process”.

The boot loader version can be read by ISP/IAP calls (see Section 5.4.5.13 or Section 5.6.6).

5.3.2 In-System Programming (ISP) and In-Application Programming (IAP)

Flash programming, and other related functions, are supported in several different ways:

- For details of USART In-System Programming, see Section 5.4.
- For details of I2C and SPI In-System Programming, see Section 5.5.
- For details of In-Application Programming, see Section 5.6.
- For details of USB programming, see Chapter 41 “LPC546xx USB ROM API”.

The boot loader version can be read by ISP/IAP calls (see Section 5.4.5.13 or Section 5.6.6).

Remark: When using the SPI/ISP feature, the External Memory Controller interface cannot be used because the EMC_D0 and EMC_D1 pins are multiplexed with SPIISP pins.

5.3.3 Flash content protection

Flash content may be protected via ECRP, see Chapter 43 “LPC546xx Enhanced Code Read Protection (ECRP)”.

Remark: EMC cannot be used when using the SPI ISP feature.

5.3.4 Memory map after any reset

The boot ROM is located in the memory region starting from the address 0x0300 0000. Both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in Section 5.3.8.

5.3.5 Flash error correction

The LPC546xx is equipped with the Error Correction Code (ECC) capable flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user’s code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible flash. Consequently, flash bytes from 0x0000 0000 to 0x0000 000F are protected by the first ECC byte, flash bytes from 0x0000 0010 to 0x0000 001F are protected by the second ECC byte, etc.

Whenever the CPU requests a read from user’s flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user’s flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of flash memory is erased, the corresponding ECC bytes are also erased. Once an ECC byte is written, it cannot be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 16 bytes (or multiples of 16), aligned as described above.

5.3.6 Criteria for valid user code

The boot ROM supports several types of boot images, that is, user code. The types are:

- Legacy
- Single enhanced
- Dual enhanced

All types of boot images (Legacy and Enhanced) are qualified by examining the first 8 exception vectors located between offset 0x0000 00000 and 0x0000 00020. The sum of the first 8 vectors must result in 0. This is accomplished by placing the 2's compliment of the first 7 vectors in the 8th vector location. Qualification ends at this step for Basic images.

Enhanced images (Single and Dual) are identified by placing 0xEDDC9494 or 0xFFEB6B6 (respectively) in the 10th vector location (offset 0x0000 00024). Additional qualification is performed by examining the 11th vector location (offset 0x0000 0028), which should contain a pointer to an additional enhanced boot block structure. Qualification of the Enhanced boot block structure consists of reading the first location and comparing against the value 0xFEEDA5A5. See Chapter 3 "LPC546xx Boot process" for details. If a valid image cannot be located, the ROM will enter ISP mode and auto-detect activity on the I2C / SPI or USART interfaces. The auto-detect looks for activity on the USART, I2C, and SPI interfaces and selects the appropriate interface once a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes. USART, I2C and SPI ISP communications are described in Section 5.4 and Section 5.5.

5.3.7 Flash partitions

Some IAP and ISP commands operate on sectors and specify sector numbers. In addition, a page erase command is available. The size of a sector is 32 KB and the size of a page is 256 Byte. One sector contains 128 pages. Sector 0 and page 0 are located at address 0x0000 0000.

Table 12. Flash sectors and pages

Sector number	Sector size	Page numbers	Address range	Total flash (including this sector)
0	32 KB	0 - 127	0x0000 0000 - 0x0000 7FFF	32 KB
1	32 KB	128 - 255	0x0000 8000 - 0x0000 FFFF	64 KB
2	32 KB	256 - 383	0x0001 0000 - 0x0001 7FFF	96 KB
3	32 KB	384 - 511	0x0001 8000 - 0x0001 FFFF	128 KB
4	32 KB	512 - 639	0x0002 0000 - 0x0002 7FFF	160 KB
5	32 KB	640 - 767	0x0002 8000 - 0x0002 FFFF	192 KB
6	32 KB	768 - 895	0x0003 0000 - 0x0003 7FFF	224 KB
7	32 KB	896 - 1023	0x0003 8000 - 0x0003 FFFF	256 KB
8	32 KB	1024 - 1151	0x0004 0000 - 0x0004 7FFF	288 KB
9	32 KB	1152 - 1279	0x0004 8000 - 0x0004 FFFF	320 KB
10	32 KB	1280 - 1407	0x0005 0000 - 0x0005 7FFF	352 KB
11	32 KB	1408 - 1535	0x0005 8000 - 0x0005 FFFF	384 KB
12	32 KB	1536 - 1663	0x0006 0000 - 0x0006 7FFF	416 KB
13	32 KB	1664 - 1791	0x0006 8000 - 0x0006 FFFF	448 KB
14	32 KB	1792 - 1919	0x0007 0000 - 0x0007 7FFF	480 KB
15	32 KB	1920 - 2047	0x0007 8000 - 0x0007 FFFF	512 KB

5.3.8 ISP interrupt and SRAM use

5.3.8.1 Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing, the interrupt vectors from the user flash area are active. Before making any IAP call, either disable the interrupts or ensure that the user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM. The IAP code does not use or disable interrupts.

5.3.8.2 RAM used by ISP command handler

Memory for the ISP commands is allocated dynamically at the end of SRAM0.

5.3.8.3 RAM used by IAP command handler

Flash programming commands use the user stack space and may use up to 128 bytes growing downward.

5.4 USART In-System Programming

All USART ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in plain binary format.

5.4.1 USART ISP initialization

Once the USART ISP mode is entered, the auto-baud routine needs to synchronize with the host via the serial port (USART).

The host should send a '?' (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this, the host should send back the same string ("Synchronized<CR><LF>").

The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. The response is required for backward compatibility of the boot loader code and is ignored. "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the clock frequency should be greater than or equal to 10 MHz. In USART ISP mode, the part is clocked by the FRO 12 MHz and the crystal frequency is ignored.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 5.4.5 "USART ISP commands".

5.4.2 USART ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

5.4.3 USART ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Data only for Read commands).

For error codes, see Section 5.5.1.19 "ISP Error codes".

5.4.4 USART ISP data format

The data stream is in plain binary format.

5.4.5 USART ISP commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 13. USART ISP command summary

ISP Command	Usage	Section
Unlock	U <Unlock Code>	5.4.5.1
Set Baud Rate	B <Baud Rate> <stop bit>	5.4.5.2
Echo	A <setting>	5.4.5.3
Write to RAM	W <start address> <number of bytes>	5.4.5.4
Read Memory	R <address> <number of bytes>	5.4.5.5
Prepare sectors for write operation	P <start sector number> <end sector number>	5.4.5.6
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	5.4.5.7
Go	G <address> <Mode>	5.4.5.8
Erase sector(s)	E <start sector number> <end sector number>	5.4.5.9
Erase page(s)	X <start page number> <end page number>	5.4.5.10
Blank check sector(s)	I <start sector number> <end sector number>	5.4.5.11
Read Part ID	J	5.4.5.12
Read Boot code version	K	5.4.5.13
Compare	M <address1> <address2> <number of bytes>	5.4.5.14
ReadUID	N	5.4.5.15
Read/Write EEPROM page	O	5.4.5.16
Read CRC checksum	S <address> <number of bytes>	5.4.5.17
ISP Boot image	T	5.4.5.19
Read flash signature	Z	5.4.5.18

5.4.5.1 ISP Unlock

Table 14. USART ISP Unlock command

Command	U
Input	Unlock code: 23130_{10}
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command unlocks the flash Write, Erase, and Go commands.
Example	"U 23130<CR><LF>" unlocks the flash Write/Erase & Go commands.

5.4.5.2 ISP Set Baud Rate

Table 15. USART ISP Set Baud Rate command

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command changes the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

5.4.5.3 Echo

Table 16. USART ISP Echo command

Command	A
Input	Setting: ON = 1 | OFF = 0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

5.4.5.4 ISP Write to RAM

The host should send the plain binary code after receiving the CMD_SUCCESS return code. This ISP command handler responds with “OK<CR><LF>” when the transfer has finished.

Remark: ISP Write to RAM terminates the write with a zero byte. If 10 bytes are written (W 536936448 10), the 11th byte (W 536936458) will be set to zero.

Table 17. USART ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. This command is blocked when code read protection levels 2 or 3 are enabled. Writing
Example	"W 536871424 4<CR><LF>" writes 4 bytes of data to address 0x2000 0200.

5.4.5.5 ISP Read Memory

Reads the plain binary code of the data stream, followed by the CMD_SUCCESS return code.

Remark: First word (0x0) of flash memory cannot be read. Start address for flash read must be 0x4 and onwards.

Table 18. USART ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (plain binary)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or flash memory. This command is blocked when code read protection is enabled. This command will return raw binary data read from the given address.
Example	"R 536871424 4<CR><LF>" read 4 bytes of data from address 0x2000 0200.

5.4.5.6 ISP Prepare sectors for write operation

This command makes flash write/erase operation a two-step process.

Table 19. USART ISP Prepare sectors for write operation command

Command	P
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. To prepare a single sector use the same start and end sector numbers.
Example	”P 0 5<CR><LF>" prepares the flash sector 0 to 5.

5.4.5.7 ISP Copy RAM to flash

When writing to the flash, the following limitations apply:

1. The smallest amount of data that can be written to flash by the copy RAM to flash command is 256 byte (equal to one page).
2. One page consists of 16 flash words (lines), and the smallest amount that can be modified per flash write is one flash word (one line). This limitation exists because ECC is applied during the flash write operation, see Section 5.3.5.
3. To avoid write disturbance (a mechanism intrinsic to flash memories), an erase should be performed after 16 consecutive writes inside the same page. Note that the erase operation then erases the entire sector.

Remark: Once a page has been written to 16 times, it is still possible to write to other pages within the same sector without performing a sector erase (assuming that those pages have been erased previously).

Table 20. USART ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. This command is blocked when code read protection is enabled. Also see Section 5.3.5 for the number of bytes that can be written.
Example	"C 536903680 196608 512 <CR><LF>" This example copies 512 bytes from the RAM starting at address 0x2000 8000 to flash starting at address 0x0003 0000.

5.4.5.8 ISP Go

Table 21. USART ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode) | A (Execute program in ARM mode).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or flash memory and can return to the ISP command handler by restoring ‘LR’ to ‘PC’. This code, executed from RAM, should not affect the last 1024 bytes of the main RAM to return to ISP. This command is blocked when code read protection is enabled.
Example	"G 595 T<CR><LF>" branches to address 0x0000 0253.

5.4.5.9 Erase sectors

Table 22. USART ISP Erase sector command

Command	E
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip flash memory. This command only allows erasure of all user sectors when the code read protection is enabled. **Remark:** This device does not have a specific Mass Erase command. When the range passed to the erase command includes the entire flash, it is processed as a special Mass Erase case. A Mass Erase case only checks if the CRP_MASS_ERASE_DISABLE bit enables Mass Erase, and if so, it proceeds to erase the entire flash. Otherwise it returns an error. The ECRP Sector Protection bits are ignored for the Mass Erase case.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

5.4.5.10 ISP Erase pages

Table 23. USART ISP Erase page command

Command	X
Input	**Start Page Number** **End Page Number:** Should be greater than or equal to start page number.
Return Code	CMD_SUCCESS | BUSY | INVALID_PAGE | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more page(s) of on-chip flash memory.
Example	"X 2 3<CR><LF>" erases the flash pages 2 and 3.

5.4.5.11 ISP Blank check sectors

Table 24. USART ISP Blank check sector command

Command	I
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check one or more sectors of on-chip flash memory. When CRP is enabled, the blank check command returns 0 for the offset and value of sectors which are not blank. Blank sectors are correctly reported irrespective of the CRP setting.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

5.4.5.12 ISP Read Part Identification number

Table 25. USART ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number.
Description	This command is used to read the part identification number.

See Table 232 “Device ID0 register values” for LPC546xxdevice identification numbers.

5.4.5.13 ISP Read Boot code version number

Table 26. USART ISP Read Boot Code version number command

Command	K
Input	None
Return Code	CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>.
Description	This command is used to read the boot code version number.

5.4.5.14 ISP Compare

Table 27. USART ISP Compare command

Command	M
Input	**Address1 (DST):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Number of Bytes:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) | COMPARE_ERROR | (Followed by the offset of first mismatch) | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to compare the memory contents at two locations. Compare result may not be correct when source or destination address contains any of the first 512 bytes starting from address zero. First 512 bytes are re-mapped to boot ROM
Example	"M 8192 33587200 4<CR><LF>" compares 4 bytes from the RAM address 0x0200 8000 to the 4 bytes from the flash address 0x2000.

5.4.5.15 ISP ReadUID

Table 28. USART ReadUID command

Command	N
Input	None
Return Code	CMD_SUCCESS followed by four 32-bit words of a unique serial number in ASCII format. The word sent at the lowest address is sent first.
Description	This command is used to read the unique ID.

5.4.5.16 Read/Write EEPROM page

Table 29. USART ISP Read/Write EEPROM page command

Command	O
Input	**EEPROM Page Read/Write (0 - EEPROM page Read, 1 - EEPROM page Write)** **EEPROM Page Number:** Should be greater than or equal to start page number.
Return Code	CMD_SUCCESS | PARAM_ERROR |
Description	This command is used to read or write to a page.
Example	Example "O 0 5<CR><LF>" reads the EEPROM page 5. The read data will be raw binary of size 128 bytes. Example “O 1 8<CR><LF>” initiates a write to EEPROM page 8. When the system is ready to receive the data, it responds with 0<CR><LF> [CMD_SUCCESS] and then the host should send raw binary data of size 128 bytes. The EEPROM programming starts on receiving the 128th byte. After the programming is completed successfully, the ISP command responds with 0<CR><LF> [CMD_SUCCESS] error code.

5.4.5.17 ISP Read CRC checksum

Get the CRC checksum of a block of RAM or flash. CMD_SUCCESS followed by 8 bytes of CRC checksum in decimal format.

The checksum is calculated as follows:

CRC-32 polynomial: x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

Seed Value: 0xFFFF FFFF

Table 30. USART ISP Read CRC checksum command

Command	S
Input	**Address:** The data are read from this address for CRC checksum calculation. This address must be on a word boundary. **Number of Bytes:** Number of bytes to be calculated for the CRC checksum; must be a multiple of 4.
Return Code	CMD_SUCCESS followed by data in decimal format | ADDR_ERROR (address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read the CRC checksum of a block of RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"S 33587200 4<CR><LF>" reads the CRC checksum for 4 bytes of data from address 0x0200 8000. If checksum value is 0xCBF43926, then the host will receive: "3421780262 <CR><LF>"

5.4.5.18 ISP Read flash signature

Get the signature for the entire flash memory using an internal flash signature generator. Signature generation is described in Chapter 42 “LPC546xx Flash signature generator”. CMD_SUCCESS followed by the 128-bit flash signature represented in decimal format.

Table 31. USART ISP Read flash signature command

Command	Z
Input	none
Return Code	CMD_SUCCESS followed by data in decimal format | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read the signature of the entire flash memory. This command is blocked when code read protection is enabled.
Example	"Z<CR><LF>" returns the signature for the entire flash memory. If signature value is 0x3BD7, then the host will receive: "15319 <CR><LF>"

5.4.5.19 ISP Boot image

Table 32. USART Boot image command

Command	T
Input	none
Return Code	Will not return any status on Success ERR_ISP_USER_CODE_CHECKSUM ERR_ISP_NO_VALID_IMAGE
Description	This command boots the image available in flash following the boot flow.
Example	"T<CR><LF>” boots the image in flash.

5.5 I²C / SPI In-System Programming

To use I²C or SPI as the ISP communication channel, the host sends a probe frame. See Section 5.5.1 “I²C / SPI commands”. The LPC546xx responds to the probe command with the appropriate response packet (see “I2C / SPI Probe command”). The host should resend the probe command if a response is not received within 5 milliseconds. The host should continue sending probe commands until the target responds. After successfully responding to the probe command, any SPI / I2C command can be sent.

Remark: If the probe response is not received correctly, the interface speed may need to be reduced.

The commands can be executed without the unlock command. The unlock command is required to be executed once per ISP session. The unlock command is explained in Section 5.4.5 “USART ISP commands”.

5.5.1 I²C / SPI commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

I2C/SPI ISP supports subblocks. A subblock is an amount of data that is smaller than a flash block, two or more subblocks will make up flash block. The purpose of this is to allow the host processor to break up data into smaller portions for transmission.

Table 33. I2C / SPI command summary

ISP Command	Command identifier	Section
Get version	0xA1	5.5.1.1
Soft reset	0xA2	5.5.1.2
Boot image	0xA3	5.5.1.3
Check image	0xA4	5.5.1.4
Host interface probe	0xA5	5.5.1.5
Write block	0xA6	5.5.1.6
Read block	0xA7	5.5.1.7
Erase sector	0xA8	5.5.1.8
Erase page	0xA9	5.5.1.9
Write page	0xAA	5.5.1.10
Read page	0xAB	5.5.1.11
Write subblock	0xB0	5.5.1.12
Read subblock	0xB1	5.5.1.13
Bulk erase	0xAE	5.5.1.14
Write RAM	0xB0	5.5.1.15

Table 33. I2C / SPI command summary *...continued*

ISP Command	Command identifier	Section
Go	0xB1	5.5.1.16
EEPROM Write page	0xBF	5.5.1.17
EEPROM Read page	0xBE	5.5.1.18

5.5.1.1 Get version

The ISP_CMD_GET_VERSION command is used to get the version information of the ISP.

Table 34. I2C / SPI ISP Get version command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA1	Command identifier.

Table 35. I2C / SPI ISP Get version response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA1	Command identifier
Length	0x2	2	0x0002	length of the response packet
Major revision	0x4	1	Version	Bit 7 [always 1] Bits [6:0] Major Number (0x13; 19dec)
Minor revision	0x5	1	Version	Minor version number (Current Minor version 0x00)

5.5.1.2 Soft reset

The ISP_CMD_RESET command executes a soft reset.

Table 36. I2C / SPI ISP Soft reset command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA2	Command identifier.

There is no response packet.

5.5.1.3 Boot image

The ISP_CMD_BOOT command is used to boot the application image.
Before boot, the clock is set to 12MHz.

Table 37. I2C / SPI ISP boot image command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA3	Command identifier.

If a valid image is located in the flash, the boot command does not send any response. If no valid bootable image is found, the boot command sends an error response. See Table 38.

Table 38. I2C / SPI ISP boot image response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA3	Command identifier
Length	0x2	2	0x0004	on error this field is set to 4.
ErrorCode	0x4	4	Version	0x000001D: No valid images found 0x0000015: Images present has CRC errors

5.5.1.4 Check image

The ISP_CMD_CHECK_IMAGE command is used to check if the flash has a valid application image.

Steps:

1. Image header is verified.
2. Image CRC32 is computed and verified against the CRC in the header. If CRC32 matches, the command returns 0, else it returns the computed CRC32.

Table 39. I2C / SPI ISP check image command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA4	Command identifier.
Reserved	0x1	3	0	Reserved
imgBaseAddr	0x4	4		Base address of an enhanced image

Table 40. I2C / SPI ISP check image response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA4	Command identifier.
Length	0x2	2	0x0008	Length of the response packet.
CRC32	0x4	4	CRC32	0 – If CRC value present in image header matches the computed value. Computed CRC32 – if the CRC does not match.

5.5.1.5 Host interface probe

The ISP_CMD_PROBE command sends the host interface information. The host should send this command repeatedly until a proper response is received.

Note: Only the ifType field is used. The rest are ignored.

Table 41. I2C / SPI ISP probe command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA5	Command identifier.
ifType	0x1	1		Host interface type and port. 1 – I2C0 port 4 – SPI0 port
Reserved	0x2	5	0x00	Reserved bytes must be 0x00.
checksum	0x7	1		XOR of all the 7 bytes above.

Table 42. I2C / SPI ISP probe response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA5	Command identifier.
Length	0x2	2	0x0000	Length of the response packet.

5.5.1.6 Write block

The ISP_CMD_WRITE_BLOCK command writes a block to flash. A flash block is 512 bytes. If a block number crosses a sector boundary, the LPC546xx automatically erases the sector prior to the write operation.

Note: The initial value of the flash memory and the memory value after an erase operation is all 1s.

Table 43. I2C / SPI ISP write block command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA6	Command identifier.
crcCheck	0x1	1		0 – Do CRC check for this packet. 1 - Ignore CRC field for this packet.
blockNum	0x2	2		Flash block number.
Data	0x2	512		Data to be programmed in flash.
checksum	0x204	4		CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1.

Table 44. I2C / SPI ISP write block response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA6	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to 0.
errorCode	0x4	4		Error code. 0x0001001: Invalid parameters.

5.5.1.7 Read block

The ISP_CMD_READ_BLOCK command reads a block of flash. A flash block is 512 bytes.

Table 45. I2C / SPI ISP read block command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA7	Command identifier.
Reserved	0x1	1	0x00	Should be zero.
blockNum	0x2	2		Flash block to read.

Table 46. I2C / SPI ISP read block response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA7	Command identifier.

Table 46. I2C / SPI ISP read block response packet *…continued*

Field	Offset	Size (bytes)	Value	Description
Length	0x2	2	0x0204 0x0004	Length of the response packet.
Data	0x4	512		Flash block content.
checksum	0x204	4	Crc32	CRC32 of the packet excluding this field.

Table 47. I2C / SPI ISP read block error response

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA7	Command identifier.
Length	0x2	2	0x0004	On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled

5.5.1.8 Erase sector

The ISP_CMD_SECTOR_ERASE command erases a flash sector. A flash sector is 32 Kb.

Note: The initial value of the flash memory and the memory value after an erase operation is all 1s.

Table 48. I2C / SPI ISP erase sector command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA8	Command identifier.
Reserved	0x1	1	0x00	Should be zero.
blockNum	0x2	2		Flash sector number to be erased.

Table 49. I2C / SPI ISP erase sector block response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA8	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to 0. On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled 0x0001001: Invalid parameters

5.5.1.9 Erase page

The ISP_CMD_PAGE_ERASE (0xA9) command erases a flash. Each flash page is 256 bytes.

Notes:

- The initial value of the flash memory and the memory value after an erase operation is all 1s.

Table 50. I2C / SPI ISP erase page command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xA9	Command identifier.
Reserved	0x1	1	0x00	Should be zero.
pageNum	0x2	2		Flash page.

Table 51. I2C / SPI ISP erase page block response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xA9	Command identifier.
Length	0x2	2	0x0006	Length of the response packet.
errorCode	0x4	4		0x00000013: Read protection enabled 0x0001001: Invalid parameters

5.5.1.10 Write page

The ISP_CMD_PAGE_WRITE command writes to flash page. Each flash page is 256 bytes.

The pageNum stores a page number for flash.

Table 52. I2C / SPI ISP write page command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xAA	Command identifier.
crcCheck	0x1	1		0 - Do CRC check for this packet. 1 - Ignore CRC field for this packet
pageNum	0x2	2		Flash page.
Data	0x4	256		Data to be programed in flash.
checksum	0x104	4		CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1.

Table 53. I2C / SPI ISP write page response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xAA	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to 0. On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled. 0x0001001: Invalid parameters.

5.5.1.11 Read page

The ISP_CMD_PAGE_READ command reads a flash page. Each flash page is 256 bytes.

The pageNum stores a page number for flash.

Table 54. I2C / SPI ISP read page command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xAB	Command identifier.
Reserved	0x1	1	0x00	Should be zero.
pageNum	0x2	2		Flash page.

Table 55. I2C / SPI ISP read page response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xAB	Command identifier.
Length	0x2	2	0x0104 0x0004	On Success this field is set to: page size + 4. On error this field is set to 4.
Data	0x4	256		Flash page content
Checksum	0x104	4	Crc32	CRC32 of the packet excluding this field.

5.5.1.12 Write subblock

The ISP_CMD_WRITE_SUBBLOCK command writes a flash block but transfers small chunks of data. The flash block size is 512 bytes but some host processors cannot send or receive more than 256 bytes in a single transaction. This command supports block splitting into multiple sub-blocks. All sub-blocks must be transferred before the block is committed to flash.

Note:

- Host must send the sub-blocks in sequential order.
- If a different command is sent in between the sub-block commands, the collection buffer is reset.
- If the block number falls on a sector boundary, the sector will be erased (32 KB) before programming the block.

Table 56. I2C / SPI ISP write subblock command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xAC	Command identifier.
subBlock	0x1	1		Bit 0: If set CRC check is not done for this packet Bits [5:1]: Specifies the sub-block number Bits [7:6]: Specifies the sub-block size 00 – 32 bytes 01 – 64 bytes 10 – 128 bytes 11 – 256 bytes
blockNum	0x2	2		Flash block number
Data	0x4	Subblock size		Data to be programed into flash
Checksum	Sub-block size + 4	4		CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1

Table 57. I2C / SPI ISP write subblock response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xAC	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to 0. On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled 0x0001001: Invalid parameters

5.5.1.13 Read subblock

The ISP_CMD_READ_SUBBLOCK command reads a flash block. The flash block size is 512 bytes but some host processor cannot send/receive more than 256 bytes in a single transaction. This command allows a host processor to read data blocks in a host specified size.

Table 58. I2C / SPI ISP read subblock command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xAD	Command identifier.
subBlock	0x1	1		Bit 0: If set CRC check is not done for this packet Bits [5:1]: Specifies the sub-block number Bits [7:6]: Specifies the sub-block size 00 – 32 bytes 01 – 64 bytes 10 – 128 bytes 11 – 256 bytes
blockNum	0x2	2		Flash block number

Table 59. I2C / SPI ISP read subblock response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xAD	Command identifier.
Length	0x2	2	0xyyyy 0x0004	On success this field is set to: Sub-block size + 4. On error this field is set to 4.
Data	0x4	Subblock size		Flash content.
Checksum	Subblock size + 4	4	Crc32	CRC32 of the packet excluding this field.

5.5.1.14 Bulk erase

The ISP_CMD_BULK_ERASE (0xAE) command is used to erase more than one sector (from a start sector to an end sector).

Table 60. I2C / SPI ISP bulk erase command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xAE	Command identifier.
Reserved	0x1	1	0x00	Should be zero.
startSec	0x2	1		Start sector number.
endSec	0x3	1		End sector number.

Table 61. I2C / SPI ISP bulk erase response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xAE	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to: Sub-block size + 4. On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled 0x0001001: Invalid parameters

5.5.1.15 Write RAM

The ISP_CMD_WRITE_RAM (0xB0) command is used to write a block of data to the RAM.

Table 62. I2C / SPI ISP write RAM command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xB0	Command identifier.
crcEnable	0x1	1		0 – CRC check not enabled 1 – CRC check enabled
Size	0x2	2		Number of bytes in the data field (must be a multiple of 4). The maximum size is 512 bytes.
Address	0x4	4		Destination RAM address to which the data be written.
CRC	0x8	4		Calculated CRC32 value for the data field (only for data field).
Data	0xC	Size		Data to be written to RAM, the size of the data is provided by the Size field.

Table 63. I2C / SPI ISP write RAM response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xB0	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to 0. On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled 0x0001001: Invalid parameters 0x000000D: ISP_ADDR_ERROR 0x000000E: ISP_ADDR_NOT_MAPPED 0x0000015: ISP_USER_CODE_CHECKSUM

5.5.1.16 Go

The ISP_CMD_GOTO (0xB1) command is used to make the execution jump to a specific address.

Note: This function will not return a status when the execution is successful.

Table 64. I2C / SPI ISP go command packet

Field‘	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xB1	Command identifier.
Mode	0x1	1	0x00	0 – Thumb mode. (No other modes are supported).
Reserved	0x2	2		Reserved (must be 0).
Address	0x4	4		Address from where execution starts.

Table 65. I2C / SPI ISP go to RAM response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xB1	Command identifier.
Length	0x2	2	0x0000 0x0004	On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled. 0x0001001: Invalid parameters.

5.5.1.17 EEPROM Write page

The ISP_CMD_PAGE_WRITE command writes to EEPROM page. Each EEPROM page is 128 bytes.

The pageNum stores a page number for EEPROM.

Table 66. I2C / SPI ISP EEPROM write page command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xBF	Command identifier.
crcCheck	0x1	1		0 – Do CRC check for this packet. 1 - Ignore CRC field for this packet
pageNum	0x2	2		EEPROM page.
Data	0x4	128		Data to be programmed into EEPROM.
checksum	0x84	4		CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1.

Table 67. I2C / SPI ISP EEPROM write page response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xBF	Command identifier.
Length	0x2	2	0x0000 0x0004	On Success this field is set to 0. On error this field is set to 4.
errorCode	0x4	4		0x00000013: Read protection enabled. 0x0001001: Invalid parameters.

5.5.1.18 EEPROM Read page

The ISP_CMD_PAGE_READ command reads EEPROM page. Each EEPROM page is 128 bytes.

The pageNum stores a page number for EEPROM.

Table 68. I2C / SPI ISP EEPROM read page command packet

Field	Offset	Size (bytes)	Value	Description
Command	0x0	1	0xBE	Command identifier.
Reserved	0x1	1	0x00	Should be zero.
pageNum	0x2	2		EEPROM page.

Table 69. I2C / SPI ISP EEPROM read page response packet

Field	Offset	Size (bytes)	Value	Description
SOP	0x0	1	0x55	Start of packet identifier.
Command	0x1	1	0xBE	Command identifier.
Length	0x2	2	0x0104 0x0004	On Success this field is set to: page size + 4. On error this field is set to 4.
Data	0x4	128		EEPROM page content
Checksum	0x84	4	Crc32	CRC32 of the packet excluding this field.

5.5.1.19 ISP Error codes

Table 70. USART/I2C/SPI ISP error codes

Return Code	Error code	Description
0x0	ERR_ISP_CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
0x1	ERR_ISP_INVALID_COMMAND	Invalid command.
0x2	ERR_ISP_SRC_ADDR_ERROR	Source address is not on word boundary.
0x3	ERR_ISP_DST_ADDR_ERROR	Destination address is not on a correct boundary.
0x4	ERR_ISP_SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken into consideration where applicable.
0x5	ERR_ISP_DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken into consideration where applicable.
0x6	ERR_ISP_COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
0x7	ERR_ISP_INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
0x8	ERR_ISP_SECTOR_NOT_BLANK	Sector is not blank.
0x9	ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
0xA	ERR_ISP_COMPARE_ERROR	Source and destination data not equal.
0xB	ERR_ISP_BUSY	Flash programming hardware interface is busy.
0xC	ERR_ISP_PARAM_ERROR	Insufficient number of parameters or invalid parameter.
0xD	ERR_ISP_ADDR_ERROR	Address is not on word boundary.
0xE	ERR_ISP_ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken into consideration where applicable.
0xF	ERR_ISP_CMD_LOCKED	Command is locked.
0x10	ERR_ISP_INVALID_CODE	Unlock code is invalid.
0x11	ERR_ISP_INVALID_BAUD_RATE	Invalid baud rate setting.
0x12	ERR_ISP_INVALID_STOP_BIT	Invalid stop bit setting.
0x13	ERR_ISP_CODE_READ_PROTECTION_ENABLED	Code read protection enabled.
0x14	-	Reserved.
0x15	ERR_ISP_USER_CODE_CHECKSUM	Valid enhanced image present with invalid CRC.
0x16	-	Reserved.
0x17	ERR_ISP_FRO_NO_POWER	FRO not turned on in the PDRUNCFG register.
0x18	ERR_ISP_FLASH_NO_POWER	Flash not turned on in the PDRUNCFG register.
0x19	-	Reserved.
0x1A	-	Reserved.
0x1B	ERR_ISP_FLASH_NO_CLOCK	Flash clock disabled in the AHBCLKCTRL register.
0x1C	ERR_ISP_REINVOKE_ISP_CONFIG	Reinvoke ISP not successful.
0x1D	ERR_ISP_NO_VALID_IMAGE	Image does not pass validation checks.

5.6 In-Application Programming

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. The result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case the number of results are more than number of parameters. Parameter passing is illustrated in the Figure 6.

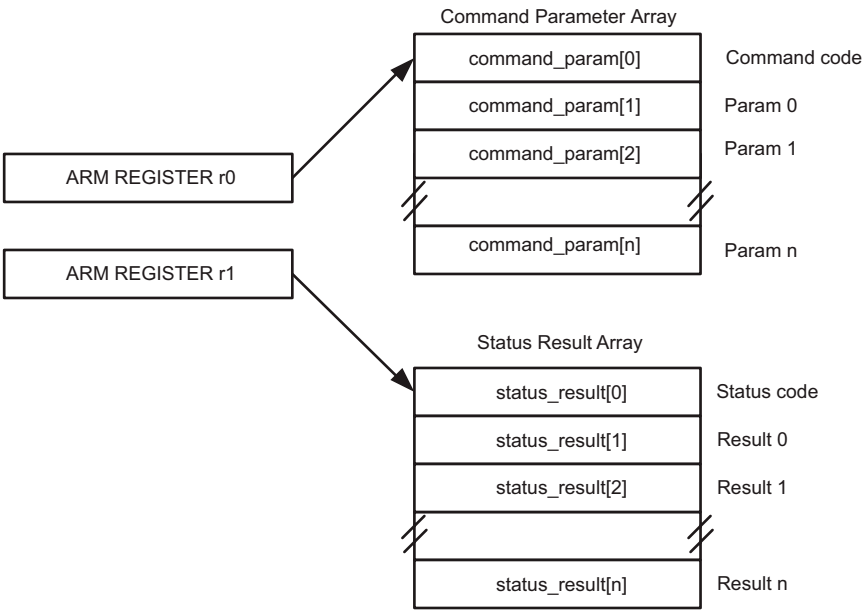


Fig 6. IAP parameter passing

The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to flash" command. The maximum number of results is 5, returned by the "ReadUID" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at location 0x0300 0204 and is in ARM Thumb code, therefore called as 0x0300 0205 by the Cortex-M4 to insure Thumb operation.

The IAP function could be called in the following way using C:

Define the IAP location entry point. Since the least significant bit of the IAP location is set there will be a change to Thumb instruction set if called by the Cortex-M4.

```
#define IAP_LOCATION 0x03000204
```

Define data structure or pointers to pass IAP command table and result table to the IAP function:

```
unsigned int command_param[5];
unsigned int status_result[5];

unsigned int * command_param;
```

```
unsigned int * status_result;
command_param = (unsigned int *) 0x...
status_result =(unsigned int *) 0x...
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting the function pointer:

```
#define IAP_LOCATION 0x03000205
iap_entry=(IAP) IAP_LOCATION;
```

To call the IAP use the following statement.

```
iap_entry (command_param,status_result);
```

Up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively (see the *ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05)*. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not be use this space if IAP flash programming is permitted in the application.

Table 71. IAP command summary

IAP Command	Command code (decimal)	Section
Prepare sector(s) for write operation	50	5.6.1
Copy RAM to flash	51	5.6.2
Erase sector(s)	52	5.6.3
Blank check sector(s)	53	5.6.4
Read part ID	54	5.6.5
Read boot code version	55	5.6.6
Compare	56	5.6.7
Reinvoke ISP	57	5.6.8
Read unique ID number	58	5.6.9
Erase page(s)	59	5.6.10
Extended flash signature read	73	5.6.11
Read EEPROM Page	80	5.6.12
Write EEPROM Page	81	5.6.13

5.6.1 Prepare sector(s) for write operation

Flash write/erase operation a two-step process, starting with this command.

Table 72. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Input	**Command code: 50 (decimal)** **Param0:** Start sector number **Param1:** End sector number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to flash" or "Erase sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase sector(s)" command causes relevant sectors to be protected again. To prepare a single sector use the same start and end sector numbers.

5.6.2 Copy RAM to flash

See Section 5.4.5.7 for limitations on the write-to-flash process.

Table 73. IAP Copy RAM to flash command

Command	Copy RAM to flash
Input	**Command code: 51 (decimal)** **Param0(DST):** Destination flash address where data bytes are to be written. This address must be on a 256 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **Param3:** System clock frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY| USER_CODE_CHECKSUM
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare sector for write operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. Also see Section 5.3.5 for the number of bytes that can be written. **Remark:** All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is being programmed. At the reserved ARM interrupt vector location (0x0000 001C), place the 2's complement of the sum of the first 7 vectors. This causes the checksum of all of the first 8 vectors together to be 0.

5.6.3 Erase sector(s)

Table 74. IAP Erase sector(s) command

Command	Erase sector(s)
Input	**Command code: 52 (decimal)** **Param0:** Start sector number **Param1:** End sector number (must be greater than or equal to start sector number). **Param2:** System clock frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip flash memory. To erase a single sector use the same start and end sector numbers. **Remark:** All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is erased. **Remark:** This device does not have a specific 'Mass Erase' command. When the range passed to the erase command includes the entire flash, it is processed as a special 'Mass Erase' case. A Mass Erase case only checks if the CRP_MASS_ERASE_DISABLE bit enables Mass Erase, and if so it proceeds to erase the entire flash. Otherwise it returns an error. The ECRP Sector Protection bits are ignored for the Mass Erase case.

5.6.4 Blank check sector(s)

Table 75. IAP Blank check sector(s) command

Command	Blank check sector(s)
Input	**Command code: 53 (decimal)** **Param0:** Start sector number **Param1:** End sector number (must be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	**Result0:** Offset of the first non blank word location if the status code is SECTOR_NOT_BLANK. **Result1:** Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip flash memory. To blank check a single sector use the same start and end sector numbers.

5.6.5 Read part identification number

Table 76. IAP Read part identification command

Command	Read part identification number
Input	**Command code: 54 (decimal)** **Parameters:** None
Status code	CMD_SUCCESS
Result	**Result0:** Part Identification Number.
Description	This command is used to read the part identification number.

5.6.6 Read boot code version number

Table 77. IAP Read Boot code version number command

Command	Read boot code version number
Input	**Command code: 55 (decimal)** **Parameters:** None
Status code	CMD_SUCCESS
Result	**Result0:** Two 32-bit words. Word 0 is the major version, word 1 is the minor version.
Description	This command is used to read the boot code version number.

5.6.7 Compare

Table 78. IAP Compare command

Command	Compare
Input	**Command code: 56 (decimal)** **Param0(DST):** Starting flash or RAM address of data bytes to be compared; should be a word boundary. **Param1(SRC):** Starting flash or RAM address of data bytes to be compared; should be a word boundary. **Param2:** Number of bytes to be compared; should be a multiple of 4.
Status code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the status code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations.

5.6.8 Reinvoke ISP

Table 79. Reinvoke ISP

Command	Compare
Input	**Command code: 57 (decimal)** **Param0:** Pointer to ISP parameter structure. See Table 80.
Status code	None.
Result	ERR_ISP_REINVOKE_ISP_CONFIG
Description	This command is used to invoke the boot loader in ISP mode. It maps boot vectors and configures the peripherals for ISP. This command may be used when a valid user program is present in the internal flash memory and the ISP entry pin are not accessible to force the ISP mode. **Remark:** The error response is returned if IAP is disabled, or if there is an invalid ISP type selection. When there is no error the call does not return, so there can be no status code.

Table 80. Pointer to ISP parameter array

Byte offset	Description
0	Reserved.
1	ISP type selection: 0: Auto-detect 1: I2C of Flexcomm 1 4: SPI of Flexcomm 3 6: UART of Flexcomm 0 8: USB0 Others: reserved
2 - 6	Reserved, must be 0.
7	Check value: XOR of the 7 preceding bytes.

5.6.9 Read unique ID number

Table 81. IAP ReadUID command

Command	Compare
Input	**Command code: 58 (decimal)**
Status code	CMD_SUCCESS
Result	**Result0:** The first 32-bit word (least significant word). **Result1:** The second 32-bit word. **Result2:** The third 32-bit word. **Result3:** The fourth 32-bit word (most significant word).
Description	This command is used to read the unique ID.

5.6.10 Erase page(s)

Table 82. IAP Erase page command

Command	Erase page
Input	**Command code: 59 (decimal)** **Param0:** Start flash address. **Param1:** End flash address (should be greater than or equal to start address) **Param2:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_PAGE
Result	None
Description	This command is used to erase a page or multiple pages of on-chip flash memory. To erase a single page use the same start and end address. Start and end addresses must be in the same flash sector. **Remark:** All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is erased.

5.6.11 Extended flash signature read

Table 83. IAP Extended flash signature read command

Command	Extended Read signature
Input	**Command code: 73 (decimal)** **Param0:** Start flash address. **Param1:** End flash address. **Param2:** Number of wait states (settings are the same as the FLASHTIM field of the FLASHCFG register. see Section 7.5.64). **Param3:** 0 (Must pass value 0).
Status code	CMD_SUCCESS
Result	**Result0:** Word 0 of 128-bit signature (bits 31 to 0). **Result1:** Word 1 of 128-bit signature (bits 63 to 32). **Result2:** Word 2 of 128-bit signature (bits 95 to 64). **Result3:** Word 3 of 128-bit signature (bits 127 to 96).
Description	This command calculates the signature value for one or more pages of on-chip flash memory.

5.6.12 Read EEPROM Page

Table 84. IAP Read EEPROM page command

Command	Read EEPROM page
Input	**Command code: 80 (decimal)** **Param0:** EEPROM page number. **Param1:** Memory address to store the value read from EEPROM. **Param2:** Current core clock frequency in kHz. [Value 0xFFFFFFFF will retain the timing and clock settings for EEPROM].
Status code	CMD_SUCCESS | INVALID_SECTOR | SRC_ADDR_NOT_MAPPED |
Result	None
Description	This command reads given page of EEPROM into the memory provided.

5.6.13 Write EEPROM Page

Table 85. IAP Write EEPROM page command

Command	Write EEPROM page
Input	**Command code: 81 (decimal)** **Param0:** EEPROM page number. **Param1:** Memory address holding data to be stored on to EEPROM page. **Param2:** Current core clock frequency in kHz. [Value 0xFFFFFFFF will retain the timing and clock settings for EEPROM].
Status code	CMD_SUCCESS | INVALID_SECTOR | SRC_ADDR_NOT_MAPPED
Result	None
Description	This command writes given data in the provided memory to a page of EEPROM.

5.6.14 Get ROM API pointer

Table 86. IAP Get ROM API pointer command

Command	Get ROM API pointer
Input	**Command code: 89 (decimal)** **Param0:** Driver index. 0: USB driver 2: power driver 14: OTP driver 0xFF: base address of the API structure
Status code	CMD_SUCCESS | ERR_FAILED
Result	**Result0:** Pointer to requested driver
Description	Returns a point to a specific API driver.

5.6.15 IAP Status Codes

Table 87. IAP Status codes Summary

Status code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken into consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken into consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	Flash programming hardware interface is busy.
12	-	-
13	-	-
14	-	-
15	USER_CODE_CHECKSUM	Valid image present with invalid CRC.

6.1 How to read this chapter

Available interrupt sources may vary with specific LPC546xx device types.

6.2 Features

- Nested Vectored Interrupt Controller that is an integral part of each CPU.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC of the Cortex-M4 supports:
  - 54 vectored interrupt slots.
  - 8 programmable interrupt priority levels with hardware priority level masking.
  - Vector table offset register VTOR.
  - Software interrupt generation.
- Support for NMI from any interrupt (see Section 7.5.3).

6.3 General description

The tight coupling to the NVIC to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

6.3.1 Interrupt sources

Table 88 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. The interrupt number does not imply any interrupt priority.

See Ref. 1 “Cortex-M4 TRM” for detailed descriptions of the NVIC and the NVIC registers.

Table 88. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
0	WDT, BOD	Windowed watchdog timer, Brownout detect	WARNINT - watchdog warning interrupt BODINTVAL - BOD interrupt level
1	DMA	DMA controller	Interrupt A and interrupt B, error interrupt
2	GINT0	GPIO group 0	Enabled pin interrupts
3	GINT1	GPIO group 1	Enabled pin interrupts
4	PIN_INT0	Pin interrupt 0 or pattern match engine slice 0	PSTAT - pin interrupt status
5	PIN_INT1	Pin interrupt 1or pattern match engine slice 1	PSTAT - pin interrupt status
6	PIN_INT2	Pin interrupt 2 or pattern match engine slice 2	PSTAT - pin interrupt status

Table 88. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
7	PIN_INT3	Pin interrupt 3 or pattern match engine slice 3	PSTAT - pin interrupt status.
8	UTICK	Micro-tick Timer	INTR.
9	MRT	Multi-rate timer	Global MRT interrupts: GFLAG0, 1, 2, 3.
10	CTIMER0	Standard counter/timer CTIMER0	Match and Capture interrupts.
11	CTIMER1	Standard counter/timer CTIMER1	Match and Capture interrupts.
12	SCTimer/PWM0	SCTimer/PWM0	EVFLAG SCT event.
13	CTIMER3	Standard counter/timer CTIMER3	Match and Capture interrupts.
14	Flexcomm0	Flexcomm Interface 0 (USART, SPI, I2C)	See Table 433, Table 455, Table 481.
15	Flexcomm1	Flexcomm Interface 1 (USART, SPI, I2C)	Same as Flexcomm0.
16	Flexcomm2	Flexcomm Interface 2 (USART, SPI, I2C)	Same as Flexcomm0.
17	Flexcomm3	Flexcomm Interface 3 (USART, SPI, I2C)	Same as Flexcomm0.
18	Flexcomm4	Flexcomm Interface 4 (USART, SPI, I2C)	Same as Flexcomm0.
19	Flexcomm5	Flexcomm Interface 5 (USART, SPI, I2C)	Same as Flexcomm0.
20	Flexcomm6	Flexcomm Interface 6 (USART, SPI, I2C, I2S)	Same as Flexcomm0, plus I2S.
21	Flexcomm7	Flexcomm Interface 7 (USART, SPI, I2C, I2S)	Same as Flexcomm0, plus I2S. (Table 509)
22	ADC0_SEQA	ADC0 sequence A completion.	See Table 1040.
23	ADC0_SEQB	ADC0 sequence B completion.	See Table 1040.
24	ADC0_THCMP	ADC0 threshold compare and error.	See Table 1040.
25	DMIC	Digital microphone and audio subsystem	See Chapter 29.
26	HWVAD	Hardware Voice Activity Detection	See Chapter 29.
27	USB0_NEEDCLK	USB0 Activity Interrupt	USB0_NEEDCLK, see Chapter 37.
28	USB0	USB0 host and device	See Table 901.
29	RTC	RTC alarm and wake-up interrupts	See Table 389.
30	Reserved	-	-
31	Reserved	-	-
32	PIN_INT4	Pin interrupt 4 or pattern match engine slice 4 int	PSTAT - pin interrupt status.
33	PIN_INT5	Pin interrupt 5 or pattern match engine slice 5 int	PSTAT - pin interrupt status.
34	PIN_INT6	Pin interrupt 6 or pattern match engine slice 6 int	PSTAT - pin interrupt status.
35	PIN_INT7	Pin interrupt 7 or pattern match engine slice 7 int	PSTAT - pin interrupt status.
36	CTIMER2	Standard counter/timer CTIMER2	Match and Capture interrupts.
37	CTIMER4	Standard counter/timer CTIMER4	Match and Capture interrupts.
38	RIT	Repetitive Interrupt Timer	RITINT; masked compare interrupt.
39	SPIFI	SPI flash interface	SPIFI interrupt.
40	Flexcomm8	Flexcomm Interface 8 (USART, SPI, I2C)	Same as Flexcomm0.
41	Flexcomm9	Flexcomm Interface 9 (USART, SPI, I2C)	Same as Flexcomm0.
42	SDIO	SD/MMC interrupt	

Table 88. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
43	CAN0_IRQ0	CAN0 interrupt 0	CAN0 interrupt 0.
44	CAN0_IRQ1	CAN0 interrupt 1	CAN0 interrupt 1.
45	CAN1_IRQ0	CAN1 interrupt 1	CAN1 interrupt 0.
46	CAN1_IRQ1	CAN1 interrupt 1	CAN1 interrupt 1.
47	USB1_IRQ	USB1 interrupt	
48	USB1_NEEDCLK	USB1 activity	
49	ETHERNET_IRQ	Ethernet	
50	ETHERNET_PMT_IRQ	Ethernet power management interrupt	
51	ETHERNET_MACLP_IRQ	Ethernet MAC interrupt	
52	EEPROM_IRQ	EEPROM interrupt	
53	LCD_IRQ	LCD interrupt	
54	SHA_IRQ	SHA interrupt	
55	SMARTCARD0_IRQ	Smart card 0 interrupt	
56	SMARTCARD1_IRQ	Smart card 1 interrupt	

6.4 Register description

The NVIC registers are located on the ARM private peripheral bus.

Table 89. Register overview: NVIC (base address 0xE000 E000)

Name	Access	Offset	Description	Reset value	Section
ISER0	R/W	0x100	Interrupt set enable register 0. This register allows enabling interrupts and reading back the interrupt enables for peripheral functions.	0	6.4.1
ISER1	R/W	0x104	Interrupt set enable register 1. See ISER0 description.	0	6.4.2
ICER0	R/W	0x180	Interrupt clear enable register 0. This register allows disabling interrupts and reading back the interrupt enables for peripheral functions.	0	6.4.3
ICER1	R/W	0x184	Interrupt clear enable register 1. See ISER0 description.	0	6.4.4
ISPR0	R/W	0x200	Interrupt set pending register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for peripheral functions.	0	6.4.5
ISPR1	R/W	0x204	Interrupt set pending register 1. See ISPR0 description.	0	6.4.6
ICPR0	R/W	0x280	Interrupt clear pending register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for peripheral functions.	0	6.4.7
ICPR1	R/W	0x284	Interrupt clear pending register 1. See ICPR0 description.	0	6.4.8
IABR0	RO	0x300	Interrupt active bit register 0. This register allows reading the current interrupt active state for specific peripheral functions.	0	6.4.9
IABR1	RO	0x304	Interrupt active bit register 1. See IABR0 description.	0	6.4.10
IPR0	R/W	0x400	Interrupt priority register 0. This register contains the 3-bit priority fields for interrupts 0 to 3.	0	6.4.11
IPR1	R/W	0x404	Interrupt priority register 1. This register contains the 3-bit priority fields for interrupts 4 to 7.	0	6.4.12
IPR2	R/W	0x408	Interrupt priority register2. This register contains the 3-bit priority fields for interrupts 8 to 11.	0	6.4.13
IPR3	R/W	0x40C	Interrupt priority register 3. This register contains the 3-bit priority fields for interrupts 12 to 15.	0	6.4.14
IPR4	R/W	0x410	Interrupt priority register 4. This register contains the 3-bit priority fields for interrupts 16 to 19.	0	6.4.15
IPR5	R/W	0x414	Interrupt priority register 5. This register contains the 3-bit priority fields for interrupts 20 to 23.	0	6.4.16
IPR6	R/W	0x418	Interrupt priority register 6. This register contains the 3-bit priority fields for interrupts 24 to 27.	0	6.4.17
IPR7	R/W	0x41C	Interrupt priority register 7. This register contains the 3-bit priority fields for interrupts 28 to 31.	0	6.4.18
IPR8	R/W	0x420	Interrupt priority register 8. This register contains the 3-bit priority fields for interrupts 32 to 35.	0	6.4.19
IPR9	R/W	0x424	Interrupt priority register 9. This register contains the 3-bit priority fields for interrupts 36 to 39.	0	6.4.20
IPR10	R/W	0x428	Interrupt priority register 10. This register contains the 3-bit priority fields for interrupts 40 to 43.	0	6.4.20
IPR11	R/W	0x42C	Interrupt priority register 11. This register contains the 3-bit priority fields for interrupts 44 to 47.	0	6.4.20

Table 89. Register overview: NVIC (base address 0xE000 E000) *...continued*

Name	Access	Offset	Description	Reset value	Section
IPR12	R/W	0x430	Interrupt priority register 12. This register contains the 3-bit priority fields for interrupts 48 to 51.	0	6.4.20
IPR13	R/W	0x434	Interrupt priority register13. This register contains the 3-bit priority fields for interrupts 52 to 55.	0	6.4.20
IPR14	R/W	0x438	Interrupt priority register14. This register contains the 3-bit priority fields for interrupt up to 56.	0	6.4.20
STIR	WO	0xF00	Software trigger interrupt register, allows software to generate interrupts.	-	6.4.26

6.4.1 Interrupt Set-Enable Register 0

The ISER0 register allows enabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are enabled via the ISER1 register (Section 6.4.2). Disabling interrupts is done through the ICER0 and ICER1 registers (Section 6.4.3 and Section 6.4.4).

Table 90. Interrupt Set-Enable Register 0

Bit	Name	Value	Function
0	ISE_WDTBOD	[1]	Watchdog Timer, BOD interrupt enable.
1	ISE_DMA	[1]	DMA interrupt enable.
2	ISE_GINT0	[1]	GPIO group 0 interrupt enable.
3	ISE_GINT1	[1]	GPIO group 1 interrupt enable.
4	ISE_PINT0	[1]	Pin interrupt / pattern match engine slice 0 interrupt enable.
5	ISE_PINT1	[1]	Pin interrupt / pattern match engine slice 1 interrupt enable.
6	ISE_PINT2	[1]	Pin interrupt / pattern match engine slice 2 interrupt enable.
7	ISE_PINT3	[1]	Pin interrupt / pattern match engine slice 3 interrupt enable.
8	ISE_UTICK	[1]	Micro-Tick Timer interrupt enable.
9	ISE_MRT	[1]	Multi-Rate Timer interrupt enable.
10	ISE_CTIMER0	[1]	Standard counter/timer CTIMER0 interrupt enable.
11	ISE_CTIMER1	[1]	Standard counter/timer CTIMER1 interrupt enable.
12	ISE_SCT0	[1]	SCT0 interrupt enable.
13	ISE_CTIMER3	[1]	Standard counter/timer CTIMER3 interrupt enable.
14	ISE_FC0	[1]	Flexcomm Interface 0 interrupt enable.
15	ISE_FC1	[1]	Flexcomm Interface 1 interrupt enable.
16	ISE_FC2	[1]	Flexcomm Interface 2 interrupt enable.
17	ISE_FC3	[1]	Flexcomm Interface 3 interrupt enable.
18	ISE_FC4	[1]	Flexcomm Interface 4 interrupt enable.
19	ISE_FC5	[1]	Flexcomm Interface 5 interrupt enable.
20	ISE_FC6	[1]	Flexcomm Interface 6 interrupt enable.
21	ISE_FC7	[1]	Flexcomm Interface 7 interrupt enable.
22	ISE_ADC0SEQA	[1]	ADC0 sequence A interrupt enable.
23	ISE_ADC0SEQB	[1]	ADC0 sequence B interrupt enable.
24	ISE_ADC0THOV	[1]	ADC0 threshold and error interrupt enable.
25	ISE_DMIC	[1]	Digital microphone subsystem interrupt enable.
26	ISE_HWVAD	[1]	Hardware voice activity detect interrupt enable.
27	ISE_USB0_NEEDCLK	[1]	USB activity interrupt enable.
28	ISE_USB0	[1]	USB device interrupt enable.
29	ISE_RTC	[1]	Real Time Clock (RTC) interrupt enable.
31: 30	-	-	Reserved. Read value is undefined, only zero should be written.

[1] Write: writing 0 has no effect, writing 1 enables the interrupt.
Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

6.4.2 Interrupt Set-Enable Register 1

The ISER1 register allows enabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER0 and ICER1 registers (Section 6.4.3 and Section 6.4.4).

Table 91. Interrupt Set-Enable Register 1 register

Bit	Name	Value	Function
0	ISE_PINT4	[1]	Pin interrupt / pattern match engine slice 4 interrupt enable.
1	ISE_PINT5	[1]	Pin interrupt / pattern match engine slice 5 interrupt enable.
2	ISE_PINT6	[1]	Pin interrupt / pattern match engine slice 6 interrupt enable.
3	ISE_PINT7	[1]	Pin interrupt / pattern match engine slice 7 interrupt enable.
4	ISE_CTIMER2	[1]	Standard counter/timer CTIMER2 interrupt enable.
5	ISE_CTIMER4	[1]	Standard counter/timer CTIMER4 interrupt enable.
6	ISE_RIT	[1]	Repetitive interrupt timer enable.
7	ISE_SPIFI	[1]	SPI flash interface interrupt enable.
8	ISE_FC8	[1]	Flexcomm Interface 8 interrupt enable.
9	ISE_FC9	[1]	Flexcomm Interface 9 interrupt enable.
10	ISE_SDIO	[1]	SD/MMC interrupt enable.
11	ISE_CAN0_INT0	[1]	CAN0 interrupt 0 enable.
12	ISE_CAN0_INT1	[1]	CAN0 interrupt 1 enable.
13	ISE_CAN1_INT0	[1]	CAN1 interrupt 0 enable.
14	ISE_CAN1_INT1	[1]	CAN1 interrupt 1enable.
15	ISE_USB1	[1]	USB1 device interrupt enable.
16	ISE_USB1_NEEDCLK	[1]	USB1 Activity Interrupt enable.
17	ISE_ETH	[1]	Ethernet interrupt enable.
18	ISE_ETH_PMT	[1]	Ethernet power management interrupt enable.
19	ISE_ETH_MACLP	[1]	Ethernet MAC interrupt enable.
20	ISE_EEPROM	[1]	EEPROM interrupt enable.
21	ISE_LCD	[1]	LCD interrupt enable.
22	ISE_SHA	[1]	SHA interrupt enable.
23	ISE_SC0_INT	[1]	Smart card 0 interrupt enable.
24	ISE_SC1_INT	[1]	Smart card 1 interrupt enable.
31:25	-	-	Reserved. Read value is undefined, only zero should be written.

[1] Write: writing 0 has no effect, writing 1 enables the interrupt.
Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

6.4.3 Interrupt clear-enable register 0

The ICER0 register allows disabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are disabled via the ICER1 register (Section 6.4.4). Enabling interrupts is done through the ISER0 and ISER1 registers (Section 6.4.1 and Section 6.4.2).

Table 92. Interrupt clear-enable register 0

Bit	Name	Function
31:0	ICE_...	Peripheral interrupt disables. Bit numbers match ISER0 registers (Table 90). Unused bits are reserved. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

6.4.4 Interrupt clear-enable register 1

The ICER1 register allows disabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER0 and ISER1 registers (Section 6.4.1 and Section 6.4.2).

Table 93. Interrupt clear-enable register 1

Bit	Name	Function
31:0	ICE_...	Peripheral interrupt disables. Bit numbers match ISER1 registers (Table 91). Unused bits are reserved. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

6.4.5 Interrupt set-pending register 0

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state set via the ISPR1 register (Section 6.4.6). Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 6.4.7 and Section 6.4.8).

Table 94. Interrupt set-pending register 0

Bit	Name	Function
31:0	ISP_...	Peripheral interrupt pending set. Bit numbers match ISER0 registers (Table 90). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

6.4.6 Interrupt set-pending register 1

The ISPR1 register allows setting the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 6.4.7 and Section 6.4.8).

Table 95. Interrupt set-pending register 1

Bit	Name	Function
31:0	ISP_...	Peripheral interrupt pending set. Bit numbers match ISER1 registers (Table 91). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

6.4.7 Interrupt clear-pending register 0

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state cleared via the ICPR1 register (Section 6.4.8). Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 6.4.5 and Section 6.4.6).

Table 96. Interrupt clear-pending register 0

Bit	Name	Function
31:0	ICP_...	Peripheral interrupt pending clear. Bit numbers match ISER0 registers (Table 90). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

6.4.8 Interrupt clear-pending register 1

The ICPR1 register allows clearing the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 6.4.5 and Section 6.4.6).

Table 97. Interrupt Clear-Pending Register 1

Bit	Name	Function
31:0	ICP_...	Peripheral interrupt pending clear. Bit numbers match ISER1 registers (Table 91). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

6.4.9 Interrupt active bit register 0

The IABR0 register is a read-only register that allows reading the active state of the first 32 peripheral interrupts. Bits in IABR are set while the corresponding interrupt service routines are in progress. Additional interrupts can have their active state read via the IABR1 register (Section 6.4.10).

Table 98. Interrupt active bit register 0

Bit	Name	Function
31:0	IAB_...	Peripheral interrupt active. Bit numbers match ISER0 registers (Table 90). Unused bits are reserved. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.

6.4.10 Interrupt active bit register 1

The IABR1 register is a read-only register that allows reading the active state of the second group of peripheral interrupts. Bits in IABR are set while the corresponding interrupt service routines are in progress.

Table 99. Interrupt active bit register 1

Bit	Name	Function
31:0	IAB_...	Peripheral interrupt active. Bit numbers match ISER1 registers (Table 91). Unused bits are reserved. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.

6.4.11 Interrupt priority register 0

The IPR0 register controls the priority of the first 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 100. Interrupt priority register 0

Bit	Name	Function
4:0	-	Unused
7:5	IP_WDTBOD	Watchdog Timer and BOD interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_DMA	DMA interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_GINT0	GPIO Group 0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_GINT1	GPIO Group 1 interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.12 Interrupt priority register 1

The IPR1 register controls the priority of the second group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 101. Interrupt priority register 1

Bit	Name	Function
4:0	-	Unused
7:5	IP_PINT0	Pin interrupt / pattern match engine slice 0 priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_PINT1	Pin interrupt / pattern match engine slice 1 priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_PINT2	Pin interrupt / pattern match engine slice 2 priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_PINT3	Pin interrupt / pattern match engine slice 3 priority. 0 = highest priority. 7 = lowest priority.

6.4.13 Interrupt priority register 2

The IPR2 register controls the priority of the third group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 102. Interrupt priority register 2

Bit	Name	Function
4:0	-	Unused
7:5	IP_UTICK	Micro-Tick Timer interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_MRT	Multi-Rate Timer interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_CTIMER0	Standard counter/timer CTIMER0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_CTIMER1	Standard counter/timer CTIMER1 interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.14 Interrupt priority register 3

The IPR3 register controls the priority of the fourth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 103. Interrupt priority register 3

Bit	Name	Function
4:0	-	Unused
7:5	IP_SCT0	SCT0 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_CTIMER3	Standard counter/timer CTIMER3 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_FC0	Flexcomm Interface 0 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_FC1	Flexcomm Interface 1 interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.15 Interrupt priority register 4

The IPR4 register controls the priority of the fifth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 104. Interrupt priority register 4

Bit	Name	Function
4:0	-	Unused
7:5	IP_FC2	Flexcomm Interface 2 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_FC3	Flexcomm Interface 3 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_FC4	Flexcomm Interface 4 interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_FC5	Flexcomm Interface 5 interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.16 Interrupt priority register 5

The IPR5 register controls the priority of the sixth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 105. Interrupt priority register 5

Bit	Name	Function
4:0	-	Unused
7:5	IP_FC6	Flexcomm Interface 6 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_FC7	Flexcomm Interface 7 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_ADC0SEQA	ADC 0 sequence A interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_ADC0SEQB	ADC 0 sequence B interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.17 Interrupt priority register 6

The IPR6 register controls the priority of the seventh group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 106. Interrupt priority register 6

Bit	Name	Function
4:0	-	Unused
7:5	IP_ADC0THOV	ADC 0 threshold and error interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_DMIC	Digital microphone subsystem interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_HWVAD	Hardware voice activity detect interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_USB0ACT	USB Activity interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.18 Interrupt priority register 7

The IPR7 register controls the priority of the eighth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 107. Interrupt priority register 7

Bit	Name	Function
4:0	-	Unused
7:5	IP_USB	USB interrupt enable. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_RTC	Real Time clock (RTC) interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	-	Reserved
28:24	-	Unused
31:29	-	Reserved

6.4.19 Interrupt priority register 8

The IPR8 register controls the priority of the ninth and last group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 108. Interrupt priority register 8

Bit	Name	Function
4:0	-	Unused
7:5	IP_PINT4	Pin interrupt / pattern match engine slice 4 priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_PINT5	Pin interrupt / pattern match engine slice 5 priority 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_PINT6	Pin interrupt / pattern match engine slice 6 priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_PINT7	Pin interrupt / pattern match engine slice 7 priority. 0 = highest priority. 7 = lowest priority.

6.4.20 Interrupt priority register 9

The IPR9 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 109. Interrupt priority register 9

Bit	Name	Function
4:0	-	Unused
7:5	IP_CTIMER2	Standard counter/timer CTIMER2 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_CTIMER4	Standard counter/timer CTIMER4 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_RIT	Repetitive interrupt timer interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_SPIFI	SPI flash interface interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.21 Interrupt priority register 10

The IPR10 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 110. Interrupt priority register 10

Bit	Name	Function
4:0	-	Unused
7:5	IP_FC8	Flexcomm Interface 8 interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_FC9	Flexcomm Interface 9 interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_SDIO	SDIO interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_CAN0_IRQ0	CAN0 interface interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.22 Interrupt priority register 11

The IPR11 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 111. Interrupt priority register 11

Bit	Name	Function
4:0	-	Unused
7:5	IP_CAN0_IRQ1	CAN0 interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_CAN1_IRQ0	CAN1 interface interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_CAN1_IRQ1	CAN1 interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_USB1_IRQ	USB1 interface interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.23 Interrupt priority register 12

The IPR12 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 112. Interrupt priority register 12

Bit	Name	Function
4:0	-	Unused
7:5	IP_USB1_NEEDCLK	High speed USB interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_ETHERNET_IRQ	Ethernet interface interrupt priority. 0 = highest priority. 7 = lowest priority.
20:16	-	Unused
23:21	IP_ETHERNET_PMT_IRQ	Ethernet power management interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_ETHERNET_MACLP_IRQ	Ethernet MAC interface interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.24 Interrupt priority register 13

The IPR13 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 113. Interrupt priority register 13

Bit	Name	Function
4:0	-	Unused
7:5	IP_EEPROM_IRQ	EEPROM interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused
15:13	IP_LCD_IRQ	LCD interface interrupt priority. 0 = highest priority. 7 = lowest priority.
23:21	IP_SHA_IRQ	SHA interface interrupt priority. 0 = highest priority. 7 = lowest priority.
28:24	-	Unused
31:29	IP_SMARCARD0_IRQ	Smart card0interface interrupt priority. 0 = highest priority. 7 = lowest priority.

6.4.25 Interrupt priority register 14

The IPR14 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 7 priorities, where 0 is the highest priority.

Table 114. Interrupt priority register 14

Bit	Name	Function
4:0	-	Unused.
7:5	IP_SMARCARD1_IRQ	Smart card 1 interface interrupt priority. 0 = highest priority. 7 = lowest priority.
12:8	-	Unused.
15:13	-	Reserved.
20:16	-	Unused.
23:21	-	Reserved.
28:24	-	Unused.
31:29	-	Reserved.

6.4.26 Software trigger interrupt register

The STIR register provides an alternate way for software to generate an interrupt, in addition to using the ISPR registers. This mechanism can only be used to generate peripheral interrupts, not system exceptions.

By default, only privileged software can write to the STIR register. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the CCR register.

The interrupt number to be programmed in this register is listed in Table 88.

Table 115. Software trigger interrupt register (STIR)

Bit	Symbol	Description
8:0	INTID	Writing a value to this field generates an interrupt for the specified the interrupt number.
31:9	-	Reserved. Read value is undefined, only zero should be written.

7.1 Features

- System and bus configuration.
- Clock select and control.
- PLL configuration.
- Reset control.
- Wake-up control.
- BOD configuration.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- Uses a selection of on-chip clocks as reference clock.
- Device ID register.

7.2 Basic configuration

Configure the SYSCON block as follows:

- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the FRO 12 MHz (fro_12m).
- Target and reference clocks for the frequency measurement function are selected in the input mux block. See Table 265.
- The SYSCON block controls use of CLKOUT pins, which must also be configured through IOCON. See Section 7.3. $\overline{\text{RESET}}$ is a dedicated pin.

  Remark: For USB high-speed host and device operation, the external crystal oscillator and PLL must be configured to a minimum of 60 MHz (CPU clock).

7.2.1 Set up the System PLL

The System PLL creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz or 48 MHz clock (fro_hf) is not appropriate, use the PLL to boost the input frequency. The system PLL can be set up by calling an API supplied by NXP Semiconductors. Also see Section 7.6.4 “System PLL functional description” and Section 7.5.81 “PLL registers”.

7.2.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock can be selected from the sources listed in step 1 below.

The main clock, after being optionally divided by the CPU Clock Divider, is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

1. Select the main clock. The following options are available:

- FRO 12 MHz output (fro_12m) from internal oscillator (default).
- FRO high speed output (fro_hf), 96 or 48 MHz from internal oscillator. See Section 7.5.77 “FRO Control register”.
- CLKIN (this is the internal clock that comes from external crystal oscillator through dedicated pins).
- Watchdog oscillator.
- The output of the system PLL.
- The RTC 32 kHz oscillator.

Remark: Use the PLL to boost the input frequency if a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz or 48 MHz clock (fro_hf) is not appropriate. The system PLL must be set up by calling an API (POWER_SetVoltageForFreq API in SDK software package) to deliver the amount of power needed for the CPU operating frequency. See Chapter 9 “LPC546xx Power profiles/Power control API”.

Section 7.5.28 “Main clock source select register A” and Section 7.5.29 “Main clock source select register B”.

2. Select the divider value for the system clock.

   Section 7.5.50 “System clock divider register”

3. Select the memories and peripherals that are operating in the application and therefore must have an active clock. The core is always clocked.

   Section 7.5.19 “AHB Clock Control register 0” and Section 7.5.20 “AHB Clock Control register 1”.

7.2.3 Measure the frequency of a clock signal

The frequency of any on-chip or off-chip clock signal can be measured accurately with a selectable reference clock. For example, the frequency measurement function can be used to accurately determine the frequency of the watchdog oscillator which varies over a wide range depending on process and temperature.

The clock frequency to be measured and the reference clock are selected in the input mux block. See Section 11.6.5 “Frequency measure function reference clock select register” and Section 11.6.6 “Frequency measure function target clock select register”.

Details on the accuracy and measurement process are described in Section 7.6.7 “Frequency measure function”.

To start a frequency measurement cycle and read the result, see Table 187.

7.3 Pin description

Table 116. SYSCON pin description

Function	Package Type	Type	Pin	Description	Reference
CLKOUT	BGA180	O	PIO0_16	Clock output.	Chapter 10
		O	PIO0_26		Chapter 10
		O	PIO1_27		Chapter 10
		O	PIO2_29		Chapter 10
		O	PIO3_12		Chapter 10
		O	PIO3_20		Chapter 10

7.4 General description

7.4.1 Clock generation

The system control block facilitates the clock generation. Many clocking variations are possible. Figure 7 and Figure 8 give an overview of potential clock options. The LPC5460x/61x devices operate at CPU frequencies of up to 180 MHz. The LPC54628 device operates at CPU frequencies of up to 220 MHz.

Remark: The indicated clock multiplexers shown in Figure 7 are synchronized. In order to operate, the currently selected clock must be running, and the clock to be switched to must also be running. This is so that the multiplexer can gracefully switch between the two clocks without glitches. Other clock multiplexers are not synchronized. The output divider can be stopped and restarted gracefully during switching if a glitch-free output is needed.

The low-power watchdog oscillator provides a selectable frequency in the range of 6 kHz to 1.5 MHz. The accuracy of this clock is limited to +/- 40% over temperature, voltage, and silicon processing variations. To determine the actual watchdog oscillator output, use the frequency measure block. See Section 7.2.3.

The part contains one system PLL that can be configured to use a number of clock inputs and produce an output clock in the range of 1.2 MHz up to the maximum chip frequency, and can be used to run most on-chip functions. The output of the PLL can be monitored through the CLKOUT pin.

The part also contains audio PLL and USB PLL.

Table 117. Clocking diagram signal name descriptions

Name	Description
32k_clk	The 32 kHz output of the RTC oscillator. The 32 kHz clock must be enabled in the RTCOSCCTRL register (see Section 7.5.80).
audio_pll_clk	The output of the Audio PLL. The Audio PLL and its source selection are shown in Figure 7 “Clock generation”.
clk_in	This is the internal clock that comes from external crystal oscillator through dedicated pins.
fcn_fclk	The function clock of Flexcomm Interfaces. See Figure 8 “Clock generation (continued)”.
frg_clk	The output of the Fractional Rate Generator. The FRG and its source selection are shown in Figure 7.
fro_12m	The 12 MHz output of the currently selected on-chip FRO oscillator. See Section 7.5.77.
fro_hf	The currently selected FRO high speed output. This may be either 96 MHz or 48 MHz. See Section 7.5.77.
lcdclkin	The clock input to the LCD display controller. See Figure 8 “Clock generation (continued)”.
main_clk	The main clock used by the CPU and AHB bus, and potentially many others. The main clock and its source selection are shown in Figure 7.
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the IOCON block.
“none”	A tied-off source that should be selected to save power when the output of the related multiplexer is not used.
pll_clk	The output of the System PLL. The System PLL and its source selection are shown in Figure 7 “Clock generation”.
usb_pll_clk	The output of the USB PLL. The USB PLL and its source selection are shown in Figure 7 “Clock generation”.
wdt_clk	The output of the watchdog oscillator, which has a selectable target frequency (see Section 7.5.79). It must also be enabled in the PDRINCFG0 register (see Section 7.5.84).
xtalin	Input of the main oscillator. If used, this is connected to an external crystal and load capacitor.
xtalout	Output of the main oscillator. If used, this is connected to an external crystal and load capacitor.

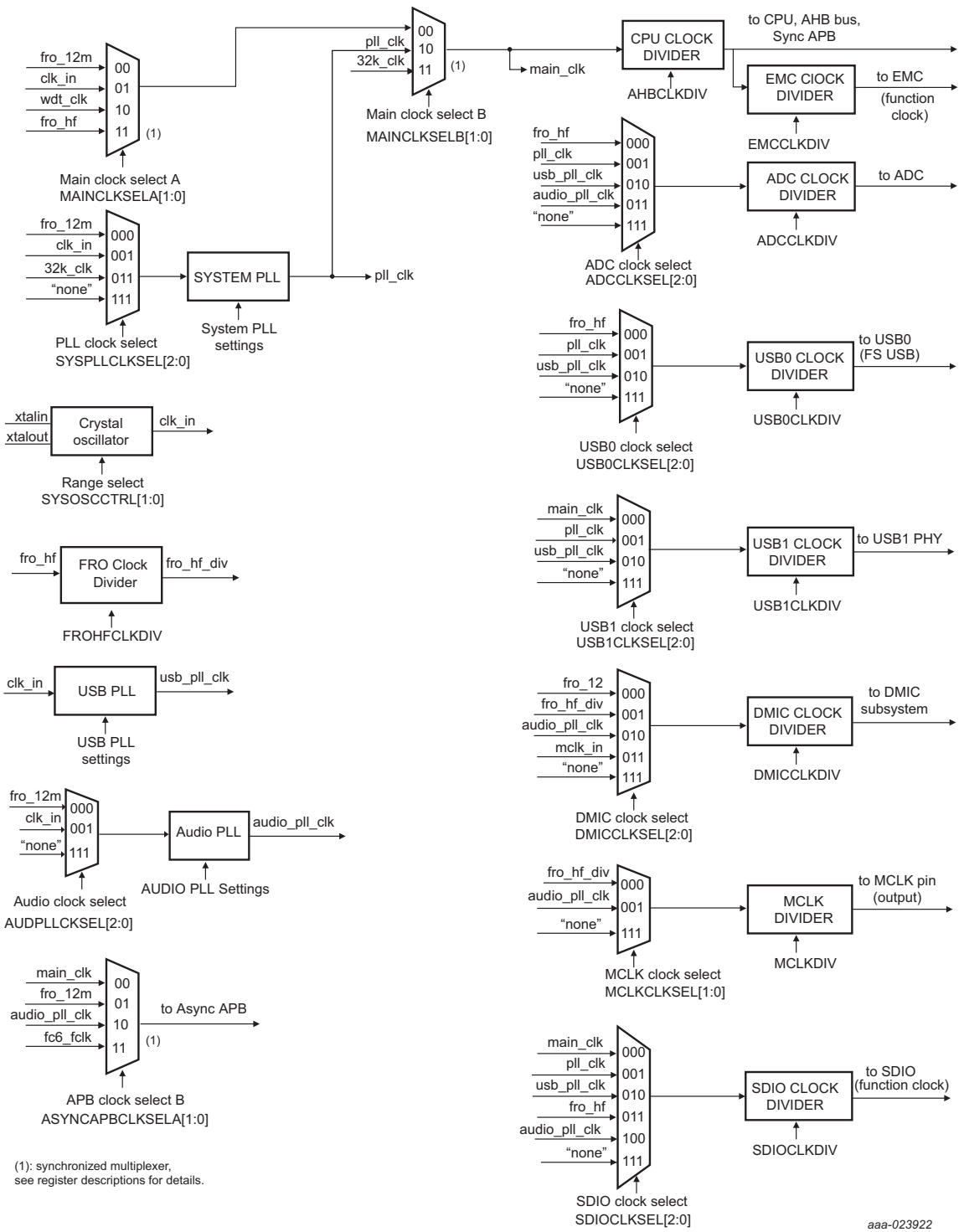


Fig 7. Clock generation

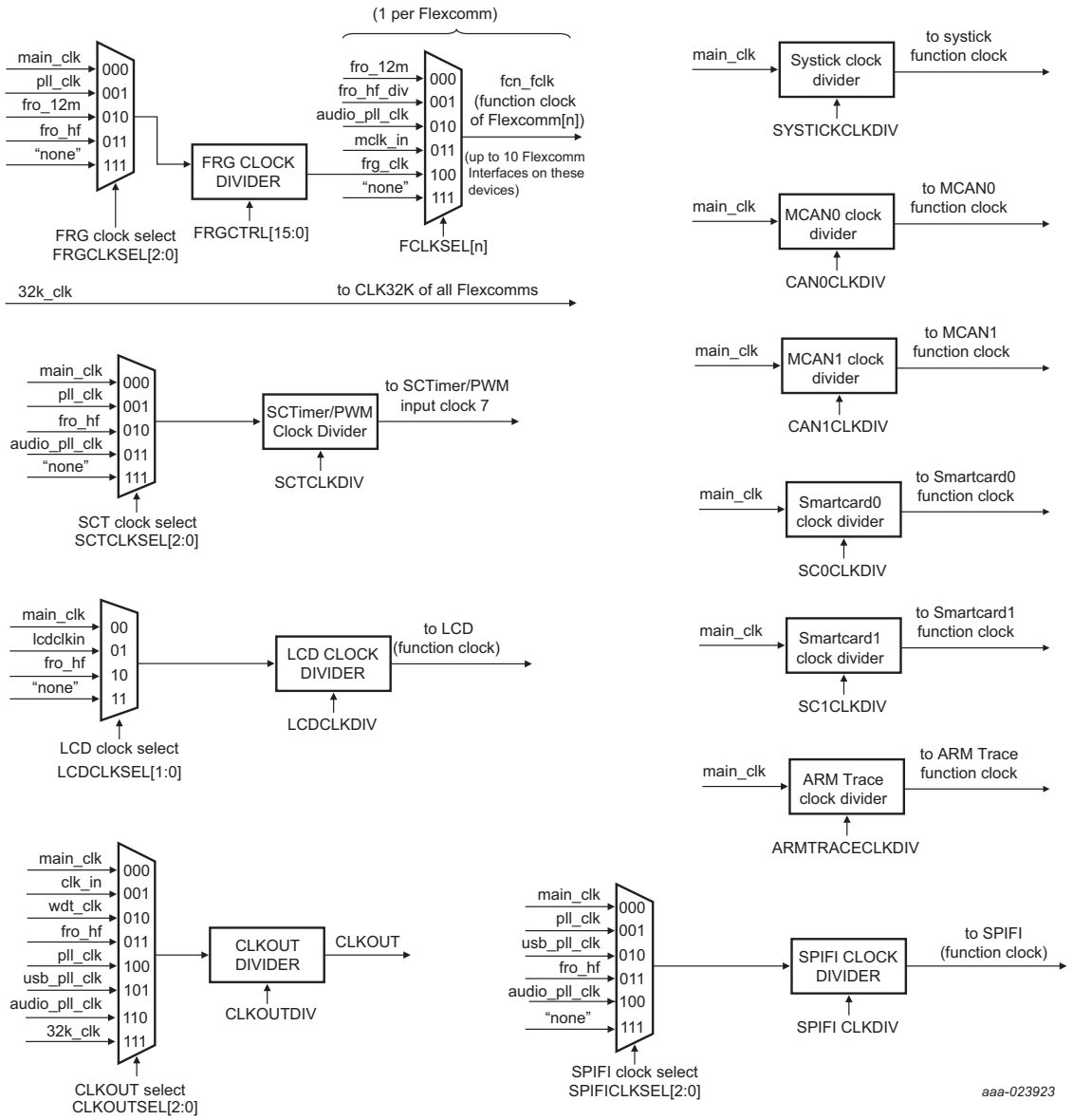


Fig 8. Clock generation (continued)

7.5 Register description

All system control block registers reside on word address boundaries. Details of the registers appear in the description of each function. System configuration functions are divided into 3 groups:

- Main system configuration at base address 0x4000 0000 (see Table 118).
- Asynchronous system configuration at base address 0x4004 0000 (see Table 119).
- Other system registers at base address 0x4002 0000 (see Table 120).

All address offsets not shown in the tables are reserved and should not be written to.

Remark: The reset value column shows the reset value seen when the boot loader executes and the flash contains valid user code. During code development, a different value may be seen if a debugger is used to halt execution prior to boot completion.

Table 118. Register overview: Main system configuration (base address 0x4000 0000)

Name	Access	Offset	Description	Reset value [1]	Section
AHBMATPRIO	R/W	0x010	AHB multilayer matrix priority control.	0x0	7.5.1
SYSTCKCAL	R/W	0x040	System tick counter calibration.	0x0	7.5.2
NMISRC	R/W	0x048	NMI source select.	0x0	7.5.3
ASYNCAPBCTRL	R/W	0x04C	Asynchronous APB control.	0x1	7.5.4
PIOPORCAP0	RO	0x0C0	POR captured value of port 0.	Note [2]	7.5.5
PIOPORCAP1	RO	0x0C4	POR captured value of port 1.	Note [2]	7.5.6
PIORESCAP0	RO	0x0D0	Reset captured value of port 0.	Note [3]	7.5.7
PIORESCAP1	RO	0x0D4	Reset captured value of port 1.	Note [3]	7.5.8
PRESETCTRL0	R/W	0x100	Peripheral reset control 0.	0x0	7.5.9
PRESETCTRL1	R/W	0x104	Peripheral reset control 1.	0x0	7.5.10
PRESETCTRL2	R/W	0x108	Peripheral reset control 2.	0x0	7.5.11
PRESETCTRLSET0	WO	0x120	Set bits in PRESETCTRL0.	-	7.5.12
PRESETCTRLSET1	WO	0x124	Set bits in PRESETCTRL1.	-	7.5.13
PRESETCTRLSET2	WO	0x128	Set bits in PRESETCTRL2.	-	7.5.14
PRESETCTRLCLR0	WO	0x140	Clear bits in PRESETCTRL0.	-	7.5.15
PRESETCTRLCLR1	WO	0x144	Clear bits in PRESETCTRL1.	-	7.5.16
PRESETCTRLCLR2	WO	0x148	Clear bits in PRESETCTRL2.	-	7.5.17
SYSRSTSTAT	R/W	0x1F0	System reset status register.	Note [4]	7.5.18
AHBCLKCTRL0	R/W	0x200	AHB Clock control 0.	0x18B	7.5.19
AHBCLKCTRL1	R/W	0x204	AHB Clock control 1.	0x0	7.5.20
AHBCLKCTRL2	R/W	0x208	AHB Clock control 2.	0x0	7.5.21
AHBCLKCTRLSET0	WO	0x220	Set bits in AHBCLKCTRL0.	-	7.5.22
AHBCLKCTRLSET1	WO	0x224	Set bits in AHBCLKCTRL1.	-	7.5.23
AHBCLKCTRLSET2	WO	0x228	Set bits in AHBCLKCTRL2.	-	7.5.24
AHBCLKCTRLCLR0	WO	0x240	Clear bits in AHBCLKCTRL0.	-	7.5.25
AHBCLKCTRLCLR1	WO	0x244	Clear bits in AHBCLKCTRL1.	-	7.5.26
AHBCLKCTRLCLR2	WO	0x248	Clear bits in AHBCLKCTRL2.	-	7.5.27

Table 118. Register overview: Main system configuration (base address 0x4000 0000) *...continued*

Name	Access	Offset	Description	Reset value [1]	Section
MAINCLKSELA	R/W	0x280	Main clock source select A.	0x0	7.5.28
MAINCLKSELB	R/W	0x284	Main clock source select B.	0x0	7.5.29
CLKOUTSELA	R/W	0x288	CLKOUT clock source select .	0x7	7.5.30
SYSPLLCLKSEL	R/W	0x290	PLL clock source select.	0x0	7.5.31
AUDPLLCLKSEL	R/W	0x298	Audio PLL clock source select .	0x0	7.5.32
SPIFICLKSEL	R/W	0x2A0	SPIFI clock source select.	0x7	7.5.33
ADCCLKSEL	R/W	0x2A4	ADC clock source select.	0x7	7.5.34
USB0CLKSEL	R/W	0x2A8	USB0 clock source select.	0x7	7.5.35
USB1CLKSEL	R/W	0x2AC	USB1 clock source select.	0x7	7.5.36
FCLKSEL0	R/W	0x2B0	Flexcomm Interface 0 clock source select.	0x7	7.5.37
FCLKSEL1	R/W	0x2B4	Flexcomm Interface 1 clock source select.	0x7	7.5.37
FCLKSEL2	R/W	0x2B8	Flexcomm Interface 2 clock source select.	0x7	7.5.37
FCLKSEL3	R/W	0x2BC	Flexcomm Interface 3 clock source select.	0x7	7.5.37
FCLKSEL4	R/W	0x2C0	Flexcomm Interface 4 clock source select.	0x7	7.5.37
FCLKSEL5	R/W	0x2C4	Flexcomm Interface 5 clock source select.	0x7	7.5.37
FCLKSEL6	R/W	0x2C8	Flexcomm Interface 6 clock source select.	0x7	7.5.37
FCLKSEL7	R/W	0x2CC	Flexcomm Interface 7 clock source select.	0x7	7.5.37
FCLKSEL8	R/W	0x2D0	Flexcomm Interface 8 clock source select.	0x7	7.5.37
FCLKSEL9	R/W	0x2D4	Flexcomm Interface 9 clock source select.	0x7	7.5.37
MCLKCLKSEL	R/W	0x2E0	MCLK clock source select.	0x7	7.5.38
FRGCLKSEL	R/W	0x2E8	Fractional Rate Generator clock source select.	0x7	7.5.39
DMICCLKSEL	R/W	0x2EC	Digital microphone (DMIC) subsystem clock select.	0x7	7.5.40
SCTCLKSEL	R/W	0x2F0	SCTimer/PWM clock source select.	0x7	7.5.41
LCDCLKSEL	R/W	0x2F4	LCD clock source select.	0x7	7.5.42
SDIOCLKSEL	R/W	0x2F8	SDIO clock source select.	0x7	7.5.43
SYSTICKCLKDIV	R/W	0x300	SYSTICK clock divider.	0x4000 0000	7.5.44
ARMTRCLKDIV	R/W	0x304	ARM Trace clock divider.	0x4000 0000	7.5.45
CAN0CLKDIV	R/W	0x 308	MCAN0 clock divider.	0x4000 0000	7.5.46
CAN1CLKDIV	R/W	0x30C	MCAN1 clock divider.	0x4000 0000	7.5.47
SC0CLKDIV	R/W	0x310	Smartcard0 clock divider.	0x4000 0000	7.5.48
SC1CLKDIV	R/W	0x314	Smartcard1 clock divider.	0x4000 0000	7.5.49
AHBCLKDIV	R/W	0x380	System clock divider.	0x0	7.5.50
CLKOUTDIV	R/W	0x384	CLKOUT clock divider.	0x4000 0000	7.5.51
FROHFDIV	R/W	0x388	FROHF clock divider.	0x0	7.5.52
SPIFICLKDIV	R/W	0x390	SPIFI clock divider.	0x4000 0000	7.5.53
ADCCLKDIV	R/W	0x394	ADC clock divider.	0x4000 0000	7.5.54
USB0CLKDIV	R/W	0x398	USB0 clock divider.	0x4000 0000	7.5.55
USB1CLKDIV	R/W	0x39C	USB1 clock divider.	0x4000 0000	7.5.56
FRGCTRL	R/W	0x3A0	Fractional rate divider.	0xFF	7.5.57
DMICCLKDIV	R/W	0x3A8	DMIC clock divider.	0x4000 0000	7.5.58

Table 118. Register overview: Main system configuration (base address 0x4000 0000) *...continued*

Name	Access	Offset	Description	Reset value [1]	Section
MCLKDIV	R/W	0x3AC	I2S MCLK clock divider.	0x4000 0000	7.5.59
LCDCLKDIV	R/W	0x3B0	LCD clock divider.	0x4000 0000	7.5.60
SCTCLKDIV	R/W	0x3B4	SCT/PWM clock divider.	0x4000 0000	7.5.61
EMCCLKDIV	R/W	0x3B8	EMC clock divider.	0x0	7.5.62
SDIOCLKDIV	R/W	0x3BC	SDIO clock divider.	0x4000 0000	7.5.63
FLASHCFG	R/W	0x400	Flash wait states configuration.	0x001A	7.5.64
USB0CLKCTRL	R/W	0x40C	USB0 clock control.	0x0	7.5.65
USB0CLKSTAT	R/W	0x410	USB0 clock status.	0x0	7.5.66
FREQMECTRL	R/W	0x418	Frequency measure register.	0x0	7.5.67
MCLKIO	R/W	0x420	MCLK input/output control.	0x0	7.5.68
USB1CLKCTRL	R/W	0x424	USB1 clock control.	0x10	7.5.69
USB1CLKSTAT	R/W	0x428	USB1 clock status.	0x0	7.5.70
EMCSYSCTRL	R/W	0x444	EMC system control.	0x1	7.5.71
EMCDLYCTRL	R/W	0x448	EMC clock delay control.	0x210	7.5.72
EMCDLYCAL	R/W	0x44C	EMC delay chain calibration control .	0x1F00	7.5.73
ETHPHYSEL	R/W	0x450	Ethernet PHY selection.	0x0	7.5.74
ETHSBDCTRL	R/W	0x454	Ethernet SBD flow control.	0x0	7.5.75
SDIOCLKCTRL	R/W	0x460	SDIO CCLKIN phase and delay control.	0x0	7.5.76
FROCTRL	R/W	0x500	FRO oscillator control.	-	7.5.77
SYSOSCCTRL	R/W	0x504	System oscillator control.	0x0	7.5.78
WDTOSCCTRL	R/W	0x508	Watchdog oscillator control.	0xA0	7.5.79
RTCOSCCTRL	R/W	0x50C	RTC oscillator 32 kHz output control.	0x1	7.5.80
USBPLLCTRL	R/W	0x51C	USB PLL control.	0x0	7.5.81.2.1
USBPLLSTAT	R/W	0x520	USB PLL status.	0x0	7.5.81.2.2
SYSPLLCTRL	R/W	0x580	System PLL control.	0x0	7.5.81.1.1
SYSPLLSTAT	RO	0x584	PLL status.	0x0	7.5.81.1.5
SYSPLLNDEC	R/W	0x588	PLL N divider.	0x0	7.5.81.1.2
SYSPLLPDEC	R/W	0x58C	PLL P divider.	0x0	7.5.81.1.3
SYSPLLMDEC	R/W	0x590	System PLL M divider.	0x0	7.5.81.1.4
AUDPLLCTRL	R/W	0x5A0	Audio PLL control.	0x0	7.5.81.3.1
AUDPLLSTAT	R/W	0x5A4	Audio PLL status.	0x0	7.5.81.3.2
AUDPLLNDEC	R/W	0x5A8	Audio PLL N divider.	0x0	7.5.81.3.3
AUDPLLPDEC	R/W	0x5AC	Audio PLL P divider.	0x0	7.5.81.3.4
AUDPLLMDEC	R/W	0x5B0	Audio PLL M divider.	0x0	7.5.81.3.5
AUDPLLFRAC	R/W	0x5B4	Audio PLL fractional divider control.	0x0	7.5.81.3.6
PDSLEEPCFG0	R/W	0x600	Sleep configuration register 0.	0x16F80740	7.5.82
PDSLEEPCFG1	R/W	0x604	Sleep configuration register 1.	0x2000008F	7.5.83
PDRUNCFG0	R/W	0x610	Power configuration register 0.	0x16F80740	7.5.84
PDRUNCFG1	R/W	0x614	Power configuration register 1.	0x2000008F	7.5.85
PDRUNCFGSET0	WO	0x620	Set bits in PDRUNCFG0.	-	7.5.86

Table 118. Register overview: Main system configuration (base address 0x4000 0000) *...continued*

Name	Access	Offset	Description	Reset value [1]	Section
PDRUNCFGSET1	WO	0x624	Set bits in PDRUNCFG1.	-	7.5.86
PDRUNCFGCLR0	WO	0x630	Clear bits in PDRUNCFG0.	-	7.5.88
PDRUNCFGCLR1	WO	0x634	Clear bits in PDRUNCFG1.	-	7.5.89
STARTER0	R/W	0x680	Start logic 0 wake-up enable register.	0x0	7.5.90
STARTER1	R/W	0x684	Start logic 1 wake-up enable register.	0x0	7.5.91
STARTERSET0	WO	0x6A0	Set bits in STARTER0.	-	7.5.92
STARTERSET1	WO	0x6A4	Set bits in STARTER1.	-	7.5.93
STARTERCLR0	WO	0x6C0	Clear bits in STARTER0.	-	7.5.94
STARTERCLR1	WO	0x6C4	Clear bits in STARTER1.	-	7.5.95
HWWAKE	R/W	0x780	Configures special cases of hardware wake-up.	0x0	7.5.96
AUTOCGOR	R/W	0xE04	Auto clock-gate override.	0x0	7.5.97
JTAGIDCODE	RO	0xFF4	JTAG ID code.	see table	7.5.98
DEVICE_ID0	RO	0xFF8	Part ID.	Note [5]	7.5.99
DEVICE_ID1	RO	0xFFC	Boot ROM and die revision.	Note [5]	7.5.100

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

[2] Determined by the voltage levels on device pins upon power-on reset.

[3] Determined by the voltage levels on device pins when a reset other than power-on reset occurs.

[4] Depends on the source of the most recent reset.

[5] Part dependent.

Table 119. Register overview: Asynchronous system configuration (base address 0x4004 0000)

Name	Access	Offset	Description	Reset value [1]	Section
ASYNCPRESETCTRL	R/W	0x000	Async peripheral reset control	0x0	7.5.101
ASYNCPRESETCTRLSET	WO	0x004	Set bits in ASYNCPRESETCTRL	-	7.5.102
ASYNCPRESETCTRLCLR	WO	0x008	Clear bits in ASYNCPRESETCTRL	-	7.5.103
ASYNCAPBCLKCTRL	R/W	0x010	Async peripheral clock control	0x0	7.5.104
ASYNCAPBCLKCTRLSET	WO	0x014	Set bits in ASYNCAPBCLKCTRL	-	7.5.105
ASYNCAPBCLKCTRLCLR	WO	0x018	Clear bits in ASYNCAPBCLKCTRL	-	7.5.106
ASYNCAPBCLKSELA	R/W	0x020	Async APB clock source select A	0x0	7.5.107

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

Table 120. Register overview: Other system configuration (base address 0x4002 0000)

Name	Access	Offset	Description	Reset value [1]	Section
BODCTRL	R/W	0x44	Brown-Out Detect control	0x0	7.5.108

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

7.5.1 AHB matrix priority register

The Multilayer AHB Matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. Care should be taken if the value in this register is changed, improper settings can seriously degrade performance.

Priority values are 3 = highest, 0 = lowest. When the priority is the same, the master with the lower master number is given priority. An example setting could put the Cortex-M4 D-code bus as the highest priority, followed by the I-Code bus. All other masters could share a lower priority.

Table 121. AHB matrix priority register 0 (AHBMATPRIO, main syscon: offset 0x010) bit description

Bit	Symbol	Description	Reset value
1:0	PRI_ICODE	I-Code bus priority. Should typically be lower than PRI_DCODE for best operation.	0
3:2	PRI_DCODE	D-Code bus priority.	0
5:4	PRI_SYS	System bus priority.	0
7:6	PRI_DMA	DMA controller priority.	0
9:8	PRI_ETH	Ethernet DMA priority.	0
11:10	PRI_LCD	LCD DMA priority.	0
13:12	PRI_USB0	USB0 DMA priority.	0
15:14	PRI_USB1	USB1 DMA priority.	0
17:16	PRI_SDIO	SDIO priority.	0
19:18	PRI_MCAN1	MCAN1 priority.	0
21:20	PRI_MCAN2	MCAN2 priority.	0
25:24	PRI_SHA	SHA priority.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.2 System tick counter calibration register

This register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of each CPU. See Chapter 22.

Table 122. System tick timer calibration register (SYSTCKCAL, main syscon: offset 0x040) bit description

Bit	Symbol	Description	Reset value
23:0	CAL	System tick timer calibration value.	0
24	SKEW	Initial value for the Systick timer.	
25	NOREF	Initial value for the Systick timer.	
31:26	-	Reserved.	-

7.5.3 NMI source selection register

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt.o For a list of all peripheral interrupts and their IRQ numbers see Table 88. For a description of the NMI functionality, see Ref. 1 “Cortex-M4 TRM”.

Remark: In order to change the interrupt source for the NMI, the NMI source must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the IRQN bits and re-enable the NMI source by setting NMIEN.

Table 123. NMI source selection register (NMISRC, main syscon: offset 0x048) bit description

Bit	Symbol	Description	Reset value
5:0	IRQM4	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the Cortex-M4, if enabled by NMIENM4.	0
29:6	-	Reserved. Read value is undefined, only zero should be written.	-
31	NMIENM4	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQM4.	0

Remark: If the NMISRC register is used to select an interrupt as the source of Non-Maskable interrupts, and the selected interrupt is enabled, one interrupt request can result in both a Non-Maskable and a normal interrupt. This can be avoided by disabling the normal interrupt in the NVIC.

7.5.4 Asynchronous APB Control register

ASYNCAPBCTRL contains a global enable bit for the asynchronous APB bridge and subsystem, allowing connection to the associated peripherals.

Table 124. Asynchronous APB Control register (ASYNCAPBCTRL, main syscon: offset 0x04C) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		Enables the asynchronous APB bridge and subsystem.	1
		0	Disabled. Asynchronous APB bridge is disabled.	
		1	Enabled. Asynchronous APB bridge is enabled.	
31:1	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.5 POR captured value of port 0

The PIOPORCAP0 register captures the state of GPIO port 0 at power-on-reset. Each bit represents the power-on reset state of one GPIO pin. This register is a read-only register.

Table 125. POR captured PIO status register 0 (PIOPORCAP0, main syscon: offset 0x0C0) bit description

Bit	Symbol	Description	Reset value
31:0	PIOPORCAP	State of PIO0_31 through PIO0_0 at power-on reset.	Depends on external circuitry

7.5.6 POR captured value of port 1

The PIOPORCAP1 register captures the state of GPIO port 1 at power-on-reset. Each bit represents the power-on reset state of one GPIO pin. This register is a read-only register.

Table 126. POR captured PIO status register 1 (PIOPORCAP1, main syscon: offset 0x0C4) bit description

Bit	Symbol	Description	Reset value
31:0	PIOPORCAP	State of PIO1_31 through PIO1_0 at power-on reset.	Depends on external circuitry

7.5.7 Reset captured value of port 0

The PIORESCAP0 register captures the state of GPIO port 0 when a reset other than a power-on reset occurs. Each bit represents the reset state of one GPIO pin. This register is a read-only register.

Table 127. Reset captured PIO status register 0 (PIORESCAP0, main syscon: offset 0x0D0) bit description

Bit	Symbol	Description	Reset value
31:0	PIORESCAP	State of PIO0_31 through PIO0_0 for resets other than POR.	Depends on external circuitry

7.5.8 Reset captured value of port 1

The PIORESCAP0 register captures the state of GPIO port 1 when a reset other than a power-on reset occurs. Each bit represents the reset state of one GPIO pin. This register is a read-only register.

Table 128. Reset captured PIO status register 1 (PIORESCAP1, main syscon: offset 0x0D4) bit description

Bit	Symbol	Description	Reset value
31:0	PIORESCAP	State of PIO1_31 through PIO1_0 for resets other than POR.	Depends on external circuitry

7.5.9 Peripheral reset control register 0

The PRESETCTRL0 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 129. Peripheral reset control register 0 (PRESETCTRL0, main syscon: offset 0x100) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved. Read value is undefined, only zero should be written.	0
7	FLASH_RST	Flash controller reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
8	FMC_RST	Flash accelerator reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
9	EEPROM_RST	EEPROM reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
10	SPIFI_RST	SPIFI reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	
11	MUX_RST	Input mux reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
12	-	Reserved. Read value is undefined, only zero should be written.	0
13	IOCON_RST	IOCON reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
14	GPIO0_RST	GPIO0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
15	GPIO1_RST	GPIO1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
16	GPIO2_RST	GPIO2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	
17	GPIO3_RST	GPIO3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	
18	PINT_RST	Pin interrupt (PINT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0

Table 129. Peripheral reset control register 0 (PRESETCTRL0, main syscon: offset 0x100) bit description

Bit	Symbol	Description	Reset value
19	GINT_RST	Grouped interrupt (GINT0 and GINT1) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
20	DMA_RST	DMA reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
21	CRC_RST	CRC generator reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
22	WWDT_RST	Watchdog timer reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
26:23	-	Reserved. Read value is undefined, only zero should be written.	0
27	ADC0_RST	ADC0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
31:28	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.10 Peripheral reset control register 1

The PRESETCTRL1 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 130. Peripheral reset control register 1 (PRESETCTRL1, main syscon: offset 0x104) bit description

Bit	Symbol	Description	Reset value
0	MRT_RST	Multi-rate timer (MRT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
1	-	Reserved. Read value is undefined, only zero should be written.	-
2	SCT0_RST	State configurable timer 0 (SCT0) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
6:3	-	Reserved. Read value is undefined, only zero should be written.	-
7	MCAN0_RST	0 = Clear reset to this function. 1 = Assert reset to this function.	0
8	MCAN1_RST	0 = Clear reset to this function. 1 = Assert reset to this function.	0
9	-	Reserved. Read value is undefined, only zero should be written.	-
10	UTICK_RST	Micro-tick Timer reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
11	FC0_RST	Flexcomm Interface 0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
12	FC1_RST	Flexcomm Interface 1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
13	FC2_RST	Flexcomm Interface 2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
14	FC3_RST	Flexcomm Interface 3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
15	FC4_RST	Flexcomm Interface 4 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0

Table 130. Peripheral reset control register 1 (PRESETCTRL1, main syscon: offset 0x104) bit description

Bit	Symbol	Description	Reset value
16	FC5_RST	Flexcomm Interface 5 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
17	FC6_RST	Flexcomm Interface 6 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
18	FC7_RST	Flexcomm Interface 7 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
19	DMIC_RST	Digital microphone interface reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
21:20	-	Reserved. Read value is undefined, only zero should be written.	0
22	CTIMER2_RST	CTIMER2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function	0
24:23	-	Reserved. Read value is undefined, only zero should be written.	0
25	USB0D_RST	USB0 device reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
26	CTIMER0_RST	CTIMER0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
27	CTIMER1_RST	CTIMER1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
31:28	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.11 Peripheral reset control register 2

The PRESETCTRL2 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 131. Peripheral reset control register 2 (PRESETCTRL2, main syscon: offset 0x108) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
2	LCD_RST	LCD reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
3	SDIO_RST	SDIO reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
4	USB1H_RST	USB1 Host reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
5	USB1D_RST	USB1 Device reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
6	USB1RAM_RST	USB1 RAM reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
7	EMC_RESET	EMC reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
8	ETH_RST	Ethernet reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0

Table 131. Peripheral reset control register 2 (PRESETCTRL2, main syscon: offset 0x108) bit description

Bit	Symbol	Description	Reset value
9	GPIO4_RST	GPIO4 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
10	GPIO5_RST	GPIO5 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
11	-	Reserved.	-
12	OTP_RST	OTP reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
13	RNG_RST	RNG reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
14	FC8_RST	Flexcomm Interface 8 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
15	FC9_RST	Flexcomm Interface 9 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
16	USB0HMR_RST	USB0 HOST master reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
17	USB0HSL_RST	USB0 HOST slave reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
18	SHA_RST	SHA reset control.	0
19	SC0_RST	Smart card 0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
20	SC1_RST	Smart card 1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
31-21	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.12 Peripheral reset control set register 0

Writing a 1 to a bit position in PRESETCTRLSET0 sets the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 129.

Table 132. Peripheral reset control set register 0 (PRESETCTRLSET0, main syscon: offset 0x120) bit description

Bit	Symbol	Description	Reset value
31:0	RST_SET0	Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL0 are reserved and only zeroes should be written to them.	-

7.5.13 Peripheral reset control set register 1

Writing a 1 to a bit position in PRESETCTRLSET1 sets the corresponding position in PRESETCTRL1. This is a write-only register. For bit assignments, see Table 130.

Table 133. Peripheral reset control set register 1 (PRESETCTRLSET1, main syscon: offset 0x124) bit description

Bit	Symbol	Description	Reset value
31:0	RST_SET1	Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL1 are reserved and only zeroes should be written to them.	-

7.5.14 Peripheral reset control set register 2

Writing a 1 to a bit position in PRESETCTRLSET2 sets the corresponding position in PRESETCTRL2. This is a write-only register. For bit assignments, see Table 131.

Table 134. Peripheral reset control set register 2 (PRESETCTRLSET2, main syscon: offset 0x128) bit description

Bit	Symbol	Description	Reset value
31:0	RST_SET2	Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL2 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL2 are reserved and only zeroes should be written to them.	-

7.5.15 Peripheral reset control clear register 0

Writing a 1 to a bit position in PRESETCTRLCLR0 clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 129.

Table 135. Peripheral reset control clear register 0 (PRESETCTRLCLR0, main syscon: offset 0x140) bit description

Bit	Symbol	Description	Reset value
31:0	RST_CLR0	Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL0 are reserved and only zeroes should be written to them.	-

7.5.16 Peripheral reset control clear register 1

Writing a 1 to a bit position in PRESETCTRLCLR1 clears the corresponding position in PRESETCTRL1. This is a write-only register. For bit assignments, see Table 130.

Table 136. Peripheral reset control clear register 1 (PRESETCTRLCLR1, main syscon: offset 0x144) bit description

Bit	Symbol	Description	Reset value
31:0	RST_CLR1	Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL1 are reserved and only zeroes should be written to them.	-

7.5.17 Peripheral reset control clear register 2

Writing a 1 to a bit position in PRESETCTRLCLR2 clears the corresponding position in PRESETCTRL2. This is a write-only register. For bit assignments, see Table 131.

Table 137. Peripheral reset control clear register 2 (PRESETCTRLCLR2, main syscon: offset 0x148) bit description

Bit	Symbol	Description	Reset value
31:0	RST_CLR2	Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL2 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL2 are reserved and only zeroes should be written to them.	-

7.5.18 System reset status register

The SYSRSTSTAT register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external $\overline{\text{RESET}}$ pin - remains asserted after the POR signal is negated, then its bit is set to detected. Write a one to clear the reset.

Table 138. System reset status register (SYSRSTSTAT, main syscon: offset 0x01F0) bit description

Bit	Symbol	Value	Description
0	POR		POR reset status. Assertion of the POR signal sets this bit, and clears all of the other bits in this register. But if another Reset signal (e.g., External Reset) remains asserted after the POR signal is negated, then its bit is set. This bit is not affected by any of the other sources of Reset.
		0	No POR detected.
		1	POR detected. Writing a one clears this flag.
1	EXTRST		Status of the external $\overline{\text{RESET}}$ pin. External reset status. Assertion of the external RESET signal sets this bit.
		0	No reset event detected.
		1	Reset detected. This bit is cleared by software writing a one to this bit, and by POR.
2	WDT		Status of the Watchdog reset. This bit is set when the Watchdog Timer times out and the WDTRESET bit in the Watchdog Mode Register is 1.
		0	No WDT reset detected.
		1	WDT reset detected. Writing a one clears this flag. This bit is cleared by software writing a one to this bit, and by POR.

Table 138. System reset status register (SYSRSTSTAT, main syscon: offset 0x01F0) bit description *…continued*

Bit	Symbol	Value	Description
3	BOD		Status of the Brown-out detect reset. This bit is set when the VDD voltage reaches a level below the BOD reset trip level. If the VDD voltage dips from the normal operating range to below the BOD reset trip level and recovers, the BOD bit will be set to 1. If the VDD voltage dips from the normal operating range to below the BOD reset trip level and continues to decline to the level at which POR is asserted, the BOD bit is cleared. If the VDD voltage rises continuously from the POR assertion level to a level above the BOD reset trip level, the BOD bit will be set to 1. This bit is cleared by software writing a one to this bit, and by POR. Note: Only in the case where a reset occurs and the POR = 0, the BODR bit indicates if the VDD voltage was below the BOD reset trip level.
		0	No BOD reset detected.
		1	BOD reset detected. Writing a one clears this flag.
4	SYSRST		Status of the software system reset. This bit is set if the processor has been reset due to a system reset request. Setting the SYSRESETREQ bit in the Cortex-M4 AIRCR register causes a chip reset. This bit is cleared by software writing a one to this bit, and by POR.
		0	No system reset detected.
		1	System reset detected. Writing a one clears this flag.
31:5	-	-	Reserved.

7.5.19 AHB Clock Control register 0

The AHBCLKCTRL0 register enables the clocks to individual system and peripheral blocks. The system clock (bit 0) provides the clock for the AHB, the APB bridge, the CPU, the SYSCON block, and the PMU. This clock cannot be disabled.

Remark: It is strongly recommended that changes to the AHBCLKCTRL registers be accomplished by using the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Regarding bits 3, 4, and 5, see Section 2.1.2 for details of SRAM configuration.

Table 139. AHB Clock Control register 0 (AHBCLKCTRL0, main syscon: offset 0x200) bit description

Bit	Symbol	Description	Reset value after boot
0	-	Reserved. This read-only bit cannot be cleared.	1
1	ROM	Enables the clock for the Boot ROM. 0 = Disable; 1 = Enable.	1
2	-	Reserved.	-
3	SRAM1	Enables the clock for SRAM1. 0 = Disable; 1 = Enable.	0
4	SRAM2	Enables the clock for SRAM2. 0 = Disable; 1 = Enable.	0
5	SRAM3	Enables the clock for SRAM3. 0 = Disable; 1 = Enable.	0
6	-	Reserved.	-
7	FLASH	Enables the clock for the flash controller. 0 = Disable; 1 = Enable. This clock is needed for flash programming, not for flash read.	1
8	FMC	Enables the clock for the Flash accelerator. 0 = Disable; 1 = Enable. This clock is needed if the flash is being read.	1
9	EEPROM	Enables the clock for EEPROM. 0 = Disable; 1 = Enable.	-
10	SPIFI	Enables the clock for the SPIFI. 0 = Disable; 1 = Enable.	0
11	INPUTMUX	Enables the clock for the input muxes. 0 = Disable; 1 = Enable.	0
12	-	Reserved.	0
13	IOCON	Enables the clock for the IOCON block. 0 = Disable; 1 = Enable.	0
14	GPIO0	Enables the clock for the GPIO0 port registers. 0 = Disable; 1 = Enable.	0
15	GPIO1	Enables the clock for the GPIO1 port registers. 0 = Disable; 1 = Enable.	0
16	GPIO2	Enables the clock for the GPIO2 port registers. 0 = Disable; 1 = Enable.	0
17	GPIO3	Enables the clock for the GPIO3 port registers. 0 = Disable; 1 = Enable.	0
18	PINT	Enables the clock for the pin interrupt block.0 = Disable; 1 = Enable.	0
19	GINT	Enables the clock for the grouped pin interrupt block. 0 = Disable; 1 = Enable.	0
20	DMA	Enables the clock for the DMA controller. 0 = Disable; 1 = Enable.	0
21	CRC	Enables the clock for the CRC engine. 0 = Disable; 1 = Enable.	0
22	WWDT	Enables the clock for the Watchdog Timer. 0 = Disable; 1 = Enable.	0
23	RTC	Enables the bus clock for the RTC. 0 = Disable; 1 = Enable.	0
26:24	-	Reserved.	-
27	ADC0	Enables the clock for the ADC0 register interface. 0 = Disable; 1 = Enable.	0
31:28	-	Reserved.	-

7.5.20 AHB Clock Control register 1

The AHBCLKCTRL1 register enables the clocks to individual peripheral blocks.

Table 140. AHB Clock Control register 1 (AHBCLKCTRL1, main syscon: offset 0x204) bit description

Bit	Symbol	Description	Reset value
0	MRT	Enables the clock for the Multi-Rate Timer. 0 = Disable; 1 = Enable.	0
1	RIT	Enables the clock for the Repetitive Interrupt Timer. 0 = Disable; 1 = Enable.	0
2	SCT0	Enables the clock for SCT0. 0 = Disable; 1 = Enable.	0
6:3	-	Reserved.	-
7	MCAN0	Enables the clock for MCAN0. 0 = Disable; 1 = Enable.	0
8	MCAN1	Enables the clock for MCAN1. 0 = Disable; 1 = Enable.	0
9	-	Reserved.	-
10	UTICK	Enables the clock for the Micro-tick Timer. 0 = Disable; 1 = Enable.	0
11	FLEXCOMM0	Enables the clock for Flexcomm Interface 0. 0 = Disable; 1 = Enable.	0
12	FLEXCOMM1	Enables the clock for Flexcomm Interface 1. 0 = Disable; 1 = Enable.	0
13	FLEXCOMM2	Enables the clock for Flexcomm Interface 2. 0 = Disable; 1 = Enable.	0
14	FLEXCOMM3	Enables the clock for Flexcomm Interface 3. 0 = Disable; 1 = Enable.	0
15	FLEXCOMM4	Enables the clock for Flexcomm Interface 4. 0 = Disable; 1 = Enable.	0
16	FLEXCOMM5	Enables the clock for Flexcomm Interface 5. 0 = Disable; 1 = Enable.	0
17	FLEXCOMM6	Enables the clock for Flexcomm Interface 6. 0 = Disable; 1 = Enable.	0
18	FLEXCOMM7	Enables the clock for Flexcomm Interface 7. 0 = Disable; 1 = Enable.	0
19	DMIC	Enables the clock for the digital microphone interface. 0 = Disable; 1 = Enable.	0
21:20	-	Reserved.	0
22	CTIMER2	Enables the clock for CTIMER 2. 0 = Disable; 1 = Enable.	0
24:23	-	Reserved.	-
25	USB0D	Enables the clock for the USB0 device interface. 0 = Disable; 1 = Enable.	0
26	CTIMER0	Enables the clock for timer CTIMER0. 0 = Disable; 1 = Enable.	0
27	CTIMER1	Enables the clock for timer CTIMER1. 0 = Disable; 1 = Enable.	0
31:28	-	Reserved.	-

7.5.21 AHB Clock Control register 2

The AHBCLKCTRL2 register enables the clocks to individual peripheral blocks.

Table 141. AHB Clock Control register 2 (AHBCLKCTRL2, main syscon: offset 0x208) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
2	LCD	Enables the clock for the LCD interface. 0 = Disable; 1 = Enable.	0
3	SDIO	Enables the clock for the SDIO interface. 0 = Disable; 1 = Enable.	0
4	USB1H	Enables the clock for the USB1 host interface. 0 = Disable; 1 = Enable.	0
5	USB1D	Enables the clock for the USB1 device interface. 0 = Disable; 1 = Enable.	0
6	USB1RAM	Enables the clock for the USB1 RAM interface. 0 = Disable; 1 = Enable.	0
7	EMC	Enables the clock for the EMC interface. 0 = Disable; 1 = Enable.	0
8	ETH	Enables the clock for the ethernet interface. 0 = Disable; 1 = Enable.	0
9	GPIO4	Enables the clock for the GPIO4 interface. 0 = Disable; 1 = Enable.	0
10	GPIO5	Enables the clock for the GPIO5 interface. 0 = Disable; 1 = Enable.	0
11	-	Reserved.	-
12	OTP	Enables the clock for the OTP interface. 0 = Disable; 1 = Enable.	0
13	RNG	Enables the clock for the RNG interface. 0 = Disable; 1 = Enable.	0
14	FLEXCOMM8	Enables the clock for the Flexcomm Interface 8. 0 = Disable; 1 = Enable.	0
15	FLEXCOMM9	Enables the clock for the Flexcomm Interface 9. 0 = Disable; 1 = Enable.	0
16	USB0HMR	Enables the clock for the USB host master interface. 0 = Disable; 1 = Enable.	0
17	USB0HSL	Enables the clock for the USB host slave interface. 0 = Disable; 1 = Enable.	0
18	SHA	Enables the clock for the SHA interface. 0 = Disable; 1 = Enable.	0
19	SC0	Enables the clock for the Smart card0 interface. 0 = Disable; 1 = Enable.	0
20	SC1	Enables the clock for the Smart card1 interface. 0 = Disable; 1 = Enable.	0
31:21	-	Reserved.	-

7.5.22 AHB clock control set register 0

Writing a 1 to a bit position in AHBCLKCTRLSET0 sets the corresponding position in AHBCLKCTRL0. This is a write-only register. For bit assignments, see Table 139.

Table 142. Clock control set register 0 (AHBCLKCTRLSET0, main syscon: offset 0x220) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_SET0	Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL0 are reserved and only zeroes should be written to them.	-

7.5.23 AHB clock control set register 1

Writing a 1 to a bit position in AHBCLKCTRLSET1 sets the corresponding position in AHBCLKCTRL1. This is a write-only register. For bit assignments, see Table 140.

Table 143. Clock control set register 1 (AHBCLKCTRLSET1, main syscon: offset 0x224) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_SET1	Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL1 are reserved and only zeroes should be written to them.	-

7.5.24 AHB clock control set register 2

Writing a 1 to a bit position in AHBCLKCTRLSET2 sets the corresponding position in AHBCLKCTRL2. This is a write-only register. For bit assignments, see Table 140.

Table 144. Clock control set register 2 (AHBCLKCTRLSET2, main syscon: offset 0x228) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_SET2	Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL2 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL2 are reserved and only zeroes should be written to them.	-

7.5.25 AHB clock control clear register 0

Writing a 1 to a bit position in AHBCLKCTRLCLR0 clears the corresponding position in AHBCLKCTRL0. This is a write-only register. For bit assignments, see Table 139.

Table 145. Clock control clear register 0 (AHBCLKCTRLCLR0, main syscon: offset 0x240) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_CLR0	Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL0 are reserved and only zeroes should be written to them.	-

7.5.26 AHB clock control clear register 1

Writing a 1 to a bit position in AHBCLKCTRLCLR1 clears the corresponding position in AHBCLKCTRL1. This is a write-only register. For bit assignments, see Table 140.

Table 146. Clock control clear register 1 (AHBCLKCTRLCLR1, main syscon: offset 0x244) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_CLR1	Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL1 are reserved and only zeroes should be written to them.	-

7.5.27 AHB clock control clear register 2

Writing a 1 to a bit position in AHBCLKCTRLCLR2 clears the corresponding position in AHBCLKCTRL2. This is a write-only register. For bit assignments, see Table 140.

Table 147. Clock control clear register 1 (AHBCLKCTRLCLR2, main syscon: offset 0x248) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_CLR2	Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL2 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL2 are reserved and only zeroes should be written to them.	-

7.5.28 Main clock source select register A

This register selects one of the internal oscillators, FRO, system oscillator, or watchdog oscillator. The oscillator selected is then one of the inputs to the main clock source select register B (see Table 149), which selects the clock source for the main clock. All clocks to the core, memories, and peripherals on the synchronous APB bus are derived from the main clock.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 148. Main clock source select register A (MAINCLKSELA, main syscon: offset 0x280) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for main clock source selector A.	0
		0x0	FRO 12 MHz (fro_12m).	
		0x1	CLKIN (clk_in).	
		0x2	Watchdog oscillator (wdt_clk).	
		0x3	FRO 96 or 48 MHz (fro_hf).	
31:2	-	-	Reserved	-

7.5.29 Main clock source select register B

This register selects the clock source for the main clock. All clocks to the core, memories, and peripherals are derived from the main clock.

One input to this register is the main clock source select register A (see Table 148), which selects one of the three internal oscillators, FRO, system oscillator, or watchdog oscillator.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 149. Main clock source select register B (MAINCLKSELB, main syscon: offset 0x284) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for main clock source selector B. Selects the clock source for the main clock.	0
		0x0	MAINCLKSELA. Use the clock source selected in MAINCLKSELA register.	
		0x1	Reserved setting.	
		0x2	System PLL output (pll_clk).	
		0x3	RTC oscillator 32 kHz output (32k_clk).	
31:2	-	-	Reserved.	-

7.5.30 CLKOUT clock source select register

This register selects the clock source for the CLKOUT pin.

Table 150. CLKOUT clock source select register (CLKOUTSELA, main syscon: offset 0x288) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CLKOUT clock source selection.	0x7
		0x0	Main clock (main_clk).	
		0x1	CLKIN (clk_in).	
		0x2	Watchdog oscillator (wdt_clk).	
		0x3	FRO 96 or 48 MHz (fro_hf).	
		0x4	System PLL output (pll_clk).	
		0x5	USB PLL clock (usb_pll_clk).	
		0x6	Audio PLL clock (audio_pll_clk).	
		0x7	RTC oscillator 32 KHz ouput.	
31:3	-	-	Reserved.	-

7.5.31 System PLL clock source select register

This register selects the clock source for the system PLL.

Table 151. System PLL clock source select register (SYSPLLCLKSEL, main syscon: offset 0x290) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		System PLL clock source selection.	0
		0x0	FRO 12 MHz (fro_12m).	
		0x1	CLKIN (clk_in).	
		0x3	RTC oscillator 32 KHz ouput.	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved setting.	
31:3	-	-	Reserved.	-

7.5.32 Audio PLL clock select register

This register selects the clock source for the system PLL.

Table 152. Audio PLL clock select register (AUDPLLCLKSEL, main syscon: offset 0x298) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Audio PLL clock source selection.	0x0
		0x00	FRO 12 MHz (fro_12m).	
		0x01	CLKIN (clk_in).	
		0x07	None, this may be selected in order to reduce power when no output is needed.	
		Others	Reserved settings.	
31:3	-	-	Reserved.	-

7.5.33 SPIFI clock select register

This register configures the peripheral clock for the SPI Flash Interface.

Table 153. SPIFI clock select register (SPIFICLKSEL, main syscon: offset 0x2A0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		System PLL clock source selection.	0x7
		0x0	Main clock (main_clk).	
		0x1	System PLL output (pll_clk).	
		0x2	USB PLL clock (usb_pll_clk).	
		0x3	FRO 96 or 48 MHz (fro_hf).	
		0x4	Audio PLL clock (audio_pll_clk).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved.	-

7.5.34 ADC clock source select register

This register selects a clock source for the 12-bit ADCs that is to the system clock. To use a clock other than the Main clock, select the asynchronous clock mode in the ADC control register.

Table 154. ADC clock source select (ADCCLKSEL, main syscon: offset 0x2A4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		ADC clock source selection.	0x7
		0x0	FRO 96 or 48 MHz (fro_hf).	
		0x1	System PLL output (pll_clk).	
		0x2	USB PLL clock (usb_pll_clk).	
		0x3	Audio PLL clock (audio_pll_clk).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-		Reserved.	-

7.5.35 USB0 clock source select register

This register selects a clock source for the USB0 full-speed controller.

Table 155. USB0 clock source select register (USB0CLKSEL, main syscon: offset 0x2A8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		USB0 device clock source selection.	0x7
		0x0	FRO 96 or 48 MHz (fro_hf).	
		0x1	System PLL output (pll_clk).	
		0x2	USB PLL clock (usb_pll_clk).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved.	-

7.5.36 USB1 clock source select register

This register selects a clock source for the USB1 high-speed controller.

Table 156. USB1 clock source select register (USB1CLKSEL, main syscon: offset 0x2A8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		USB1 PHY clock source selection.	0x7
		0x0	Main clock.	
		0x1	System PLL output (pll_clk).	
		0x2	USB PLL clock (usb_pll_clk).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved.	-

7.5.37 Flexcomm Interface clock source select registers

These registers select the clock source for each Flexcomm Interface serial peripheral. Each Flexcomm Interface has its own clock source selection.

Table 157. Flexcomm Interface clock source select registers (FCLKSEL0-9, main syscon: offsets 0x2B0 through 2D4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface clock source selection. One per Flexcomm Interface.	0x7
		0x0	FRO 12 MHz (fro_12m).	
		0x1	FRO 96 or 48 MHz divided (fro_hf_div).	
		0x2	Audio PLL clock (audio_pll_clk).	
		0x3	MCK (output of the MCLK clock select multiplexer).	
		0x4	FRG clock (output of the fractional rate generator).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved.	-

7.5.38 MCLK clock source select register

This register selects a clock to provide to the MCLK output function. In a system using I^2S and/or digital microphone, this should be related to the clock used by those functions.

Table 158. MCLK clock source select register (MCLKCLKSEL, main syscon: offset 0x2E0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		MCLK source select. This may be used by Flexcomms that support I2S, and/or by the digital microphone subsystem.	0x7
		0x0	FRO 96 or 48 MHz divided (fro_hf_div).	
		0x1	Audio PLL clock (audio_pll_clk).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.39 FRG clock source select register

This register selects a clock source for the Fractional Rate Generator.

Table 159. FRG clock source select register (FRGCLKSEL, main syscon: offset 0x2E8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Rate Generator clock source select.	0x7
		0x0	Main clock (main_clk).	
		0x1	System PLL output (pll_clk).	
		0x2	FRO 12 MHz (fro_12m).	
		0x3	FRO 96 or 48 MHz (fro_hf).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.40 DMIC clock source select register

This register selects a clock to provide to the digital microphone/audio subsystem.

Table 160. DMIC clock source select register (DMICCLKSEL, main syscon: offset 0x2EC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		DMIC (audio subsystem) clock source select.	0x7
		0x0	FRO 12 MHz (fro_12m).	
		0x1	FRO 96 or 48 MHz divided (fro_hf_div).	
		0x2	Audio PLL clock (audio_pll_clk).	
		0x3	MCLK pin input, when selected in IOCON	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.41 SCTimer/PWM clock select register

This register configures the SCTimer/PWM.

Remark: The maximum frequency for the SCTimer/PWM clock is 100 MHz.

Table 161. SCTimer/PWM clock select (SCTCLKSEL, main syscon: offset 0x2F0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL	-	SCT clock source select	0x7
		0x0	Main clock (main_clk).	
		0x1	System PLL output (pll_clk).	
		0x2	FRO 96 or 48 MHz (fro_hf).	
		0x3	Audio PLL clock (audio_pll_clk).	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.42 LCD clock source select register

This register selects a clock to provide to the graphics LCD controller.

Table 162. LCD clock source select register (LCDCLKSEL, main syscon: offset 0x2F4) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL	-	LCD clock source select.	0x3
		0x0	Main clock (main_clk).	
		0x1	LCD external clock input (LCD_CLKIN).	
		0x2	FRO 96 or 48 MHz (fro_hf).	
		0x3	None, this may be selected to reduce power when no output is needed.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.43 SDIO clock source select register

This register selects a clock to provide to the SDIO peripheral.

Table 163. SDIO clock source select register (SDIOCLKSEL, main syscon: offset 0x2F8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL	-	SDIO clock source select.	0x7
		0x0	Main clock (main_clk).	
		0x1	System PLL output (pll_clk).	
		0x2	USB PLL clock (usb_pll_clk).	
		0x3	FRO 96 or 48 MHz (fro_hf).	
		0x4	Audio PLL clock (audio_pll_clk).	
		0x7	None, this may be selected to reduce power when no output is needed.	
		others	Reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.44 SYSTICK clock divider register

This register configures the SYSTICK peripheral clock.

Table 164. SYSTICK clock divider (SYSTICKCLKDIV, main syscon: offset 0x300) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.45 ARM trace clock divider register

This register configures the ARM trace clock.

Table 165. ARM trace clock divider (ARMTRACECLKDIV, main syscon: offset 0x304) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.46 CAN0 clock divider register

This register configures the clock divider of the MCAN0 controller.

Table 166. Can0 clock divider (CAN0CLKDIV, main syscon: offset 0x308) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.47 CAN1 clock divider register

This register configures the clock divider of the MCAN1 controller.

Table 167. CAN1 clock divider (CAN1CLKDIV, main syscon: offset 0x30C) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.48 Smart card 0 clock divider register

This register configures the smart card0 divider clock.

Table 168. Smart card 0 (SC0CLKDIV, main syscon: offset 0x310) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.49 Smart card 1 clock divider register

This register configures the smartcard1 divider clock.

Table 169. Smart card 1(SC1CLKDIV, main syscon: offset 0x314) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.50 System clock divider register

This register controls how the main clock is divided to provide the system clock to the CPU, AHB bus, and memories.

Table 170. System clock divider register (AHBCLKDIV, main syscon: offset 0x380) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
30:29	-	Reserved. Read value is undefined, only zero should be written.	-
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.51 CLKOUT clock divider register

This register determines the divider value for the clock signal on the CLKOUT pin.

Table 171. CLKOUT clock divider register (CLKOUTDIV, main syscon: offset 0x384) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.52 FRO_HF clock divider register

This register can be programmed to divide the fro_hf clock.

Table 172. FRO_HF clock divider register (FROHFCLKDIV, main syscon: offset 0x388) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.53 SPIFI clock divider register

This register determines the divider value for the SPIFI clock.

Table 173. SPIFI clock divider register (SPIFICLKDIV, main syscon: offset 0x390) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.54 ADC clock source divider register

This register divides the clock to the ADC.

Table 174. ADC clock source divider (ADCCLKDIV, main syscon: offset 0x394) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.55 USB0 clock divider register

This register determines the divider value for the USB0 function clock.

Table 175. USB clock divider register (USB0CLKDIV, main syscon: offset 0x398) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.56 USB1 clock divider register

This register determines the divider value for the USB1 function clock.

Table 176. USB clock divider register (USB1CLKDIV, main syscon: offset 0x39C) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.57 Fractional baud rate generator register

All Flexcomm Interfaces have, as one of their possible clock sources, a common clock (see Figure 7), that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator.

Remark: When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48 MHz.

The output rate is:

Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV)

The clock used by the fractional rate generator is selected via the FRGSEL register (see Section 7.5.39).

Remark: In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

See also Section 25.3.1 “Configure the Flexcomm Interface clock and USART baud rate” and Section 25.7.2 “Clocking and baud rates”.

Table 177. Fractional baud rate generator register (FRGCTRL, main syscon: offset 0x3A0) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0
31:16	-	Reserved	-

7.5.58 Digital microphone interface clock divider register

This register determines the divider value for the digital microphone interface and subsystem.

Table 178. Digital microphone interface clock divider register (DMICCLKDIV, main syscon: offset 0x3A8) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider’s clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.59 MCLK clock divider register

This register determines the divider value for the I2SMCLK output, if used by the application.

Table 179. MCLK clock divider register (MCLKDIV, main syscon: offset 0x3AC) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider’s clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.60 LCD clock divider register

This register determines the divider value for the LCD clock.

Table 180. LCD clock divider register LCDCLKDIV, main syscon: offset 0x3B0) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.61 SCTimer/PWM clock divider register

This register determines the divider value for the SCTimer/PWM input7.

Table 181. SCT clock divider register SCTCLKDIV, main syscon: offset 0x3B4) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.62 EMC clock divider register

This register determines the divider value for the EMC functional clock.

Table 182. EMC clock divider register EMCCLKDIV, main syscon: offset 0x3B8) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.63 SDIO clock divider register

This register determines the divider value for the SDIO functional clock.

Table 183. SDIO clock divider register SDIOCLKDIV, main syscon: offset 0x3BC) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0
28:8	-	Reserved. Read value is undefined, only zero should be written.	-
29	RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.	0
30	HALT	Halts the divider counter. The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.	1
31	REQFLAG	Divider status flag. Set when a change is made to the divider value, cleared when the change is complete.	0

7.5.64 Flash configuration register

Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FLASHCFG register. It is recommended to use the Chip_POWER_SetVoltage API (see Section 9.4.2) to configure device operation in order to achieve lower power operation. However, flash timing can also be set up by user software as shown in Table 184.

Enabling buffering, acceleration, and prefetch will substantially improve performance. Buffering saves power by allowing previously accessed information to be reused without a flash read. Acceleration saves power by reducing CPU stalls. Prefetch typically has a small power cost due to some flash reads being performed that ultimately are not needed.Additional information about the flash accelerator may be found in Section 7.6.3.

Remark: Improper setting of this register may result in incorrect operation of the flash memory. Do not change the flash access time when using the power API library in SDK software platform.

Table 184. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
1:0	FETCHCFG		Instruction fetch configuration. This field determines how flash accelerator buffers are used for instruction fetches.	0x2
		0x0	Instruction fetches from flash are not buffered. Every fetch request from the CPU results in a read of the flash memory. This setting may use significantly more power than when buffering is enabled.	
		0x1	One buffer is used for all instruction fetches.	
		0x2	All buffers may be used for instruction fetches.	
		0x3	Reserved setting, do not use.	
3:2	DATACFG		Data read configuration. This field determines how flash accelerator buffers are used for data accesses.	0x2
		0x0	Data accesses from flash are not buffered. Every data access from the CPU results in a read of the flash memory.	
		0x1	One buffer is used for all data accesses.	
		0x2	All buffers may be used for data accesses.	
		0x3	Reserved setting, do not use.	
4	ACCEL		Acceleration enable.	1
		0	Flash acceleration is disabled. Every flash read (including those fulfilled from a buffer) takes FLASHTIM + 1 system clocks. This allows more determinism at a cost of performance.	
		1	Flash acceleration is enabled. Performance is enhanced, dependent on other FLASHCFG settings.	
5	PREFEN		Prefetch enable.	0
		0	No instruction prefetch is performed.	
		1	If the FETCHCFG field is not 0, the next flash line following the current execution address is automatically prefetched if it is not already buffered.	

Table 184. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
6	PREFOVR		Prefetch override. This bit only applies when PREFEN = 1 and a buffered instruction is completing for which the next flash line is not already buffered or being prefetched.	0x0
		0	Any previously initiated prefetch will be completed.	
		1	Any previously initiated prefetch will be aborted, and the next flash line following the current execution address will be prefetched if not already buffered.	
11:7	-	-	Reserved.	0x1A
15:12	FLASHTIM		Flash memory access time. The number of system clocks used for flash accesses is equal to FLASHTIM +1.	0x0[1]
		0x0	1 system clock flash access time (for system clock rates up to 12 MHz).	
		0x1	2 system clocks flash access time (for system clock rates up to 24 MHz).	
		0x2	3 system clocks flash access time (for system clock rates up to 36 MHz).	
		0x3	4 system clocks flash access time (for system clock rates up to 60 MHz).	
		0x4	5 system clocks flash access time (for system clock rates up to 96 MHz).	
		0x5	6 system clocks flash access time (for system clock rates up to 120 MHz).	
		0x6	7 system clocks flash access time (for system clock rates up to 144 MHz).	
		0x7	8 system clocks flash access time (for system clock rates up to 168 MHz and for system clock rates 180 MHz < CCLK ≤ 220 MHz).	
		0x8	9 system clocks flash access time (for system clock rates up to 180 MHz).	
31:16	-	-	Reserved.	-

[1] Use the PLL to boost the input frequency if a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz or 48 MHz clock (fro_hf) is not appropriate. Before connecting the output of the system PLL to the main clock, the user must call the POWER_SetVoltageForFreq API in SDK software package to deliver the amount of power needed for the CPU operating frequency. The API call sets the FLASHCFG register. See Chapter 9 “LPC546xx Power profiles/Power control API”. At 220 MHz the system clock/access time can be lower when compared to 180 MHz because the power library optimizes the on-chip voltage regulator.

7.5.65 USB0 clock control register

This register configures the clock for USB0.

Table 185. USB0 clock control register (USB0CLKCTRL, main syscon: offset 0x40C) bit description

Bit	Symbol	Value	Description	Reset value
0	AP_FS_DEV_CLK		USB0 Device USB0_NEEDCLK signal control.	0
		0	Under hardware control.	
		1	Forced high.	
1	POL_FS_DEV_CLK		USB0 Device USB0_NEEDCLK polarity for triggering the USB0 wake-up interrupt.	0
		0	Falling edge of device USB0_NEEDCLK triggers wake-up.	
		1	Rising edge of device USB0_NEEDCLK triggers wake-up.	
2	AP_FS_HOST_CLK		USB0 Host USB0_NEEDCLK signal control.	0
		0	Under hardware control.	
		1	Forced high.	

Table 185. USB0 clock control register (USB0CLKCTRL, main syscon: offset 0x40C) bit description

Bit	Symbol	Value	Description	Reset value
3	POL_FS_HOST_CLK		USB0 Host USB0_NEEDCLK polarity for triggering the USB0 wake-up interrupt.	0
		0	Falling edge of device USB0_NEEDCLK triggers wake-up.	
		1	Rising edge of device USB0_NEEDCLK triggers wake-up.	
4	PU_DISABLE		Internal pull-up disable control.	0
31:5	-	-	Reserved.	-

7.5.66 USB0 clock status register

This register is read-only and returns the status of the USB0_NEEDCLK signal. For details of how to use the USB0_NEEDCLK signal for waking up the part from deep-sleep mode, see Section 37.7.6.

Table 186. USB0 clock status register (USB0CLKSTAT, main syscon: offset 0x410) bit description

Bit	Symbol	Value	Description	Reset value
0	DEV_NEED_CLKST		USB0 Device USB0_NEEDCLK signal status.	0
		0	Low.	
		1	High.	
1	HOST_NEED_CLKST		USB0 Host USB0_NEEDCLK signal status.	0
		0	Low.	
		1	High.	
31:2	-	-	Reserved.	-

7.5.67 Frequency measure function control register

This register starts the frequency measurement function and stores the result in the CAPVAL field. The target frequency can be calculated as follows with the frequencies given in MHz:

$F_{target} = (CAPVAL - 2) \times F_{reference}/2^{14}$

Select the reference and target frequencies using the FREQMEAS_REF and FREQMEAS_TARGET before starting a frequency measurement by setting the PROG bit in FREQMECTRL.

Table 187. Frequency measure function control register (FREQMECTRL, main syscon: offset 0x418) bit description

Bit	Symbol	Description	Reset value
13:0	CAPVAL	Stores the capture result which is used to calculate the frequency of the target clock. This field is read-only.	0
30:14	-	Reserved. Read value is undefined, only zero should be written.	-
31	PROG	Set this bit to one to initiate a frequency measurement cycle. Hardware clears this bit when the measurement cycle has completed and there is valid capture data in the CAPVAL field (bits 13:0).	0

Also see:

- Section 7.2.3 “Measure the frequency of a clock signal”

- Section 7.6.7 “Frequency measure function”
- Frequency reference clock select register (FREQMEAS_REF) - Section 11.6.5
- Frequency target clock select register (FREQMEAS_TARGET) - Section 11.6.6

7.5.68 MCLK input/output control register

This register selects the direction of the pin associated with MCLK when MCLK is the elected function on that pin.

Table 188. MCLK input/output control register (MCLKIO, main syscon: offset 0x420) bit description

Bit	Symbol	Value	Description	Reset value
0	DIR		MCLK direction control.	0
		0	The MCLK function is an input.	
		1	The MCLK function is an output.	
31:1	-	-	Reserved, only zero should be written.	-

7.5.69 USB1 clock control register

This register configures the clock for USB1.

Table 189. USB1 clock control register (USB1CLKCTRL, main syscon: offset 0x424) bit description

Bit	Symbol	Value	Description	Reset value
0	AP_FS_DEV_CLK		USB1 Device need_clock signal control.	0
		0	Under hardware control.	
		1	Forced high.	
1	POL_FS_DEV_CLK		USB1 Device need_clock polarity for triggering the USB1 wake-up interrupt.	0
		0	Falling edge of device need_clock triggers wake-up.	
		1	Rising edge of device need_clock triggers wake-up.	
2	AP_FS_HOST_CLK		USB1 Host need_clock signal control.	0
		0	Under hardware control.	
		1	Forced high.	
3	POL_FS_HOST_CLK		USB1 Host need_clock polarity for triggering the USB1 wake-up interrupt.	0
		0	Falling edge of device need_clock triggers wake-up.	
		1	Rising edge of device need_clock triggers wake-up.	
4	HS_DEV_WAKEUP_N		External user wake-up signal for device mode; asserting this signal (active low) will result in exiting the low power mode; input to asynchronous control logic.	1
		0	Forces PHY to wake-up.	
		1	Normal PHY behavior.	
31:5	-	-	Reserved.	-

7.5.70 USB1 clock status register

This register is read-only and returns the status of the USB1_NEEDCLK signal.

Table 190. USB clock status register (USB1CLKSTAT, main syscon: offset 0x428) bit description

Bit	Symbol	Value	Description	Reset value
0	DEV_NEED_CLKST		USB1 Device USB1_NEEDCLK signal status.	-
		0	Low.	
		1	High.	
1	HOST_NEED_CLKST		USB1 Device host USB1_NEEDCLK signal status.	-
		0	Low.	
		1	High.	
31:2	-	-	Reserved.	-

7.5.71 EMC system control register

This register contains bits that can be used to control and configure the external memory controller.

Table 191. EMC system control register (EMCSYSCTRL, main syscon: offset 0x444) bit description

Bit	Symbol	Value	Description	Reset value
0	EMCSC		EMC Shift Control. Controls how addresses are output on the EMC address pins for static memories.	1
		0	Static memory addresses are shifted to match the data bus width. For example, when accessing a 32-bit wide data bus, the address is shifted right two places so that bit 2 is the LSB. In this mode, address bit 0 for this device is connected to address bit 0 of the memory device, simplifying memory connections. This also makes a larger memory address range possible, because additional upper address bits can appear on the higher address pins due to the shift.	
		1	Static memory addresses are always output as byte addresses regardless of the data bus width. For example, when word data is accessed on a 32-bit bus, address bits 1 and 0 will always be 0. In this mode, one or both lower address bits may not be connected to memories that are part of a bus that is wider than 8 bits.	
1	EMCRD		EMC Reset Disable. External Memory Controller Reset Disable. **Remark:** The state of this bit is preserved through a software reset, and only a POR or a BOD event will reset it to its default value.	0
		0	Both EMC resets are asserted when any type of chip reset event occurs. In this mode, all registers and functions of the EMC are initialized upon any reset condition.	
		1	Many portions of the EMC are only reset by a power-on or brown-out event, in order to allow the EMC to retain its state through a warm reset (external reset or watchdog reset). If the EMC is configured correctly, auto-refresh can be maintained through a warm reset.	
2	EMCBC		External Memory Controller burst control.	0
		0	Burst enabled.	
		1	Burst disabled. This mode can be used to prevent multiple sequential accesses to memory mapped I/O devices connected to EMC static memory chip selects.	

Table 191. EMC system control register (EMCSYSCTRL, main syscon: offset 0x444) bit description

Bit	Symbol	Value	Description	Reset value
3	EMCFBCLKINSEL		External Memory Controller clock select.	0
		0	Using internal loop back from EMC_CLK output	
		1	Using external EMC_FBCK input.	
31:4	-	-	Reserved.	-

7.5.72 EMC clock delay control register

The EMCDLYCTRL register controls on-chip programmable delays that can be used to fine tune timing to external SDRAM memories. See the LPC546xx data sheet for details on clock delay times.

Table 192. EMC clock delay control register (EMCDYCTRL, main syscon: offset 0x448) bit description

Bit	Symbol	Value	Description	Reset value
4:0	CMD_DELAY		Programmable delay value for EMC outputs in command delayed mode. See Chapter 33. The delay amount is approximately (CMDDLY+1) * 250 picoseconds. This field applies only when the command delayed read strategy is selected in the EMCDynamicReadConfig register. In this mode, all control outputs from the EMC are delayed, but the output clock is not. Delaying the control outputs changes dynamic characteristics defined in the device data sheet.	0x10
7:5	-		Reserved. Read value is undefined, only zero should be written.	0
12:8	FBCLK_DELAY		Programmable delay value for the feedback clock that controls input data sampling. See Chapter 33. The delay amount is approximately (FBCLKDLY+1) * 250 picoseconds.	0x02
31:13	-		Reserved. Read value is undefined, only zero should be written.	-

Figure 9 shows the connections of the programmable delays.

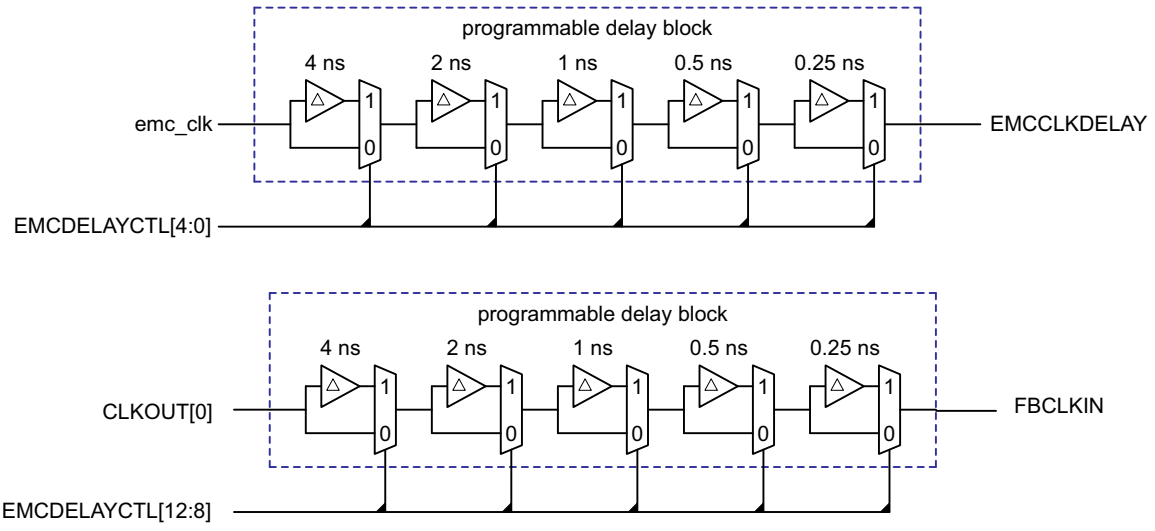


Fig 9. EMC programmable delays

7.5.73 EMC delay chain calibration control register

The EMCCAL register allows calibration of the EMC programmable delays by providing a real-time representation of the value of those delays. Delay settings that are in use in the application can be calibrated to compensate for intrinsic differences between devices, and for changes in ambient conditions.

Table 193. EMC delay chain calibration control register (EMCCAL, main syscon: offset 0x44C) bit description

Bit	Symbol	Description	Reset value
7:0	CALVALUE	Returns the count of the approximately 50 MHz ring oscillator that occur during 32 clocks of the FRO 12 MHz. This represents the composite effect of processing variation, internal regulator supply voltage, and ambient temperature.	0
13:8	-	Reserved. Read value is undefined, only zero should be written.	
14	START	Start control bit for the EMC calibration counter. Writing a 1 to this bit begins the measurement process. This bit is cleared automatically when the measurement is complete.	0
15	DONE	Measurement completion flag. this bit is set when a calibration measurement is completed. This bit is cleared automatically when the START bit is set.	0
31:16	-	Reserved.	-

Figure 10 shows the delay calibration circuit.

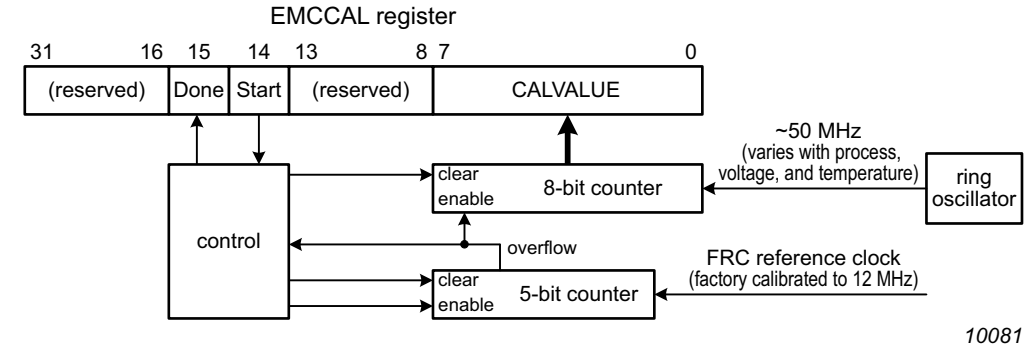


Fig 10. EMC delay calibration

7.5.74 Ethernet PHY selection register

This register selects the PHY interface. This signal is sampled only during the ethernet peripheral reset assertion and after which it is ignored.

Table 194. Ethernet PHY Selection register (ETHPHYSEL, main syscon: offset 0x450) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-		Reserved	
2	PHY_SEL		PHY interface select.	0
		0	Select MII PHY Interface.	
		1	Select RMII PHY Interface.	
31:3	-	-	Reserved	-

7.5.75 Ethernet SBD flow control register

This register is controls sideband flow control for each channel.

Table 195. Ethernet SBD flow control register (ETHSBDCTRL, main syscon: offset 0x454) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SBD_CTRL		Sideband Flow Control. When set high, instructs the MAC to transmit Pause frames in full-duplex mode. In half-duplex mode, the MAC enables the backpressure function until this signal is made low again.	0
		0x1	Controls channel 0.	
		0x2	Controls channel 1.	
31:2	-	-	Reserved	-

7.5.76 SDIO clock in phase and delay control register

This register delays the cclk_in_sample and cclk_in_drc wrt ccclk_in. Both phase and delay shifts are sequential, so if activated together it is cumulative.

Table 196. SDIO clock in phase and delay control register (SDIOCLKCTRL, main syscon: offset 0x460) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CCLK_DRV_PHASE		Programmable delay value by which cclk_in_drv is phase-shifted with regard to cclk_in. **Remark:** Bit 7, PHASE_ACTIVE must be set to 1.	0
		0	0 degree shift	
		1	90 degree shift	
		2	180 degree shift	
		3	270 degree shift	
3:2	CCLK_SAMPLE_ PHASE		Programmable delay value by which cclk_in_sample is delayed with regard to cclk_in. **Remark:** Bit 7, PHASE_ACTIVE must be set to 1.	0
		0	0 degree shift	
		1	90 degree shift	
		2	180 degree shift	
		3	270 degree shift	
6:4	-	-	Reserved	-
7	PHASE_ACTIVE		Enables the delays CCLK_DRV_PHASE and CCLK_SAMPLE_PHASE. These are accomplished by dividing sdio_clk by 2, before feeding into cclk_in, cclk_in_sample, and cclk_in_drv.	0
		0	Bypassed	
		1	Activates phase shift logic. When active, the clock divider is active and phase delays are enabled.	
15:8	-	-	Reserved	-

Table 196. SDIO clock in phase and delay control register (SDIOCLKCTRL, main syscon: offset 0x460) bit description

Bit	Symbol	Value	Description	Reset value
20:16	CCLK_DRV_DELAY	-	Programmable delay value by which cclk_in_drv is delayed with regard to cclk_in. The delay amount is approximately (CCLK_DRV_DELAY+1) * 250 picoseconds. Delaying the clock output changes dynamic characteristics defined in the device data sheet. **Remark:** Bit 23, CCLK_DRV_DELAY_ACTIVE must be set to 1.	0
22:21	-	-	Reserved	-
23	CCLK_DRV_DELAY_ACTIVE		Enables drive delay, as controlled by the CCLK_DRV_DELAY field.	
		0	Disabled	
		1	Enabled	
28:24	CCLK_SAMPLE_DELAY		Programmable delay value by which cclk_in_sample is delayed with regard to cclk_in. The delay amount is approximately (CCLK_SAMPLE_DELAY+1) * 250 picoseconds. Delaying the clock output changes dynamic characteristics defined in the device data sheet. **Remark:** Bit 31, CCLK_SAMPLE_DELAY_ACTIVE must be set to 1.	0
30:29	-	-	Reserved	-
31	CCLK_SAMPLE_DELAY_ACTIVE		Enables sample delay, as controlled by the CCLK_SAMPLE_DELAY field.	
		0	Disabled	
		1	Enabled	

7.5.77 FRO Control register

This register is used to select the on-chip FRO oscillator for the higher frequency clock, as well as configuration for the automatic USB rate adjustment mode. The trim value is factory-preset for the 96 MHz oscillator and written by the boot code on start-up.

Use the set_fro_frequency API ROM call to select 48 MHz or 96 MHz with trim. See Section 4.4.1 “set_fro_frequency”.

Table 197. FRO control register (FROCTRL, main syscon: offset 0x500) bit description

Bit	Symbol	Value	Description	Reset value
13:0	-	-	Reserved, only 0 should be written.	NA
14	SEL	-	fro_hf_output frequency status bit. If read as 0, fro_hf is 48 MHz. If read as 1, fro_hf is 96 MHz.	1
15	-	-	Reserved, only 0 should be written.	NA

Table 197. FRO control register (FROCTRL, main syscon: offset 0x500) bit description

Bit	Symbol	Value	Description	Reset value
23:16	FREQTRIM	-	Frequency trim. Boot code configures this to a device-specific factory trim value for the 96 MHz FRO. If USBCLKADJ = 1, this field is read-only and provides the value resulting from USB rate adjustment. See the USBMODCFG flag regarding reading this field. Application code may adjust this field when USBCLKADJ = 0. A single step of FREQTRIM is roughly equivalent to 0.1% of the selected FRO frequency.	see description
24	USBCLKADJ		USB clock adjust mode.	0
		0	Normal operation.	
		1	Automatic USB rate adjustment mode. If the USB FS device peripheral is enabled and connected to a USB host, it provides clock adjustment information to the FRO based on SOF packets. USB rate adjustment requires a number of cycles to take place. the USBMODCHG bit (see below) indicates when initial adjustment is complete, and when later adjustments are in progress. **Remark:** software must not alter TRIM and FREQTRIM while USBCLKADJ = 1. **Remark:** see USBCLKADJ usage notes below this table.	
25	USBMODCHG	-	USB Mode value Change flag. When 1, indicates that the USB trim is currently being updated (or is still starting up) and software should wait to read FREQTRIM. Update occurs at most once per millisecond.	0
29:26	-	-	Reserved, only 0 should be written.	NA
30	HSPDCLK		High speed clock enable. Allows disabling the high-speed FRO output if it is not needed.	0x0
		0	The high-speed FRO output is disabled.	
		1	The selected high-speed FRO output (48 MHz or 96 MHz) is enabled.	
31	-	-	Reserved, only 0 should be written.	0

Notes on using USBCLKADJ

When turning on USBCLKADJ, the current FREQTRIM value will be used as the starting value. From then on, the adjusted value will be used as long as enabled (whether USB is active or not).

If USBCLKADJ is turned off, the application may take one of the following actions:

- Read the register to pick up the adjusted FREQTRIM and then write back with the USBADJ cleared. The FRO will continue to use the adjusted value.
- If software saved the original factory trimmed value of FREQTRIM, it can be written back as above.

7.5.78 System oscillator control register

This register controls the main system oscillator.

Table 198. System oscillator control register (SYSOSCCTRL, main syscon: offset 0x504) bit description

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved, only zero should be written.	-
1	FREQRANGE		Determines frequency range for system oscillator.	0
		0	Low frequency. 1 MHz - 20 MHz frequency range.	
		1	High frequency. 15 MHz - 50 MHz frequency range.	
31:2	-	-	Reserved, only zero should be written.	-

7.5.79 Watchdog oscillator control register

This register controls the frequency of the watchdog oscillator, in the range of 6 kHz to 1.5 MHz. This oscillator is connected to the watchdog timer and the Micro-tick Timer. The low-power nature of this oscillator limits its accuracy to +/- 40% over temperature, voltage, and silicon processing variations. The actual frequency may be measured using the frequency measure block. See Section 7.2.3.

Table 199. Watchdog oscillator control register (WDTOSCCTRL, main syscon: offset 0x508) bit description

Bit	Symbol	Description	Reset value
4:0	DIVSEL	Divider select. Selects the value of the divider that adjusts the output of the oscillator. 0x00 = divide by 2 0x01 = divide by 4 0x02 = divide by 6 … 0x1E = divide by 62 0x1F = divide by 64	0
9:5	FREQSEL	Frequency select. Selects the frequency of the oscillator. 0x00 = invalid setting when watchdog oscillator is running 0x01 = 0.4 MHz 0x02 = 0.6 MHz 0x03 = 0.75 MHz 0x04 = 0.9 MHz 0x05 = 1.0 MHz 0x06 = 1.2 MHz 0x07 = 1.3 MHz 0x08 = 1.4 MHz 0x09 = 1.5 MHz 0x0A = 1.6 MHz 0x0B = 1.7 MHz 0x0C = 1.8 MHz 0x0D = 1.9 MHz 0x0E = 2.0 MHz 0x0F = 2.05 MHz 0x10 = 2.1 MHz 0x11 = 2.2 MHz 0x12 = 2.25 MHz 0x13 = 2.3 MHz 0x14 = 2.4 MHz 0x15 = 2.45 MHz 0x16 = 2.5 MHz 0x17 = 2.6 MHz 0x18 = 2.65 MHz 0x19 = 2.7 MHz 0x1A = 2.8 MHz 0x1B = 2.85 MHz 0x1C = 2.9 MHz 0x1D = 2.95 MHz 0x1E = 3.0 MHz 0x1F = 3.05 MHz	0xA
31:6	-	Reserved.	-

7.5.80 RTC oscillator control register

This register enables the 32 kHz output of the RTC oscillator (32k_clk). This clock can be used to create the main clock when the PLL input or output is selected as the clock source to the main clock.

Table 200. RTC oscillator control register (RTCOSCCTRL, main syscon: offset 0x50C) bit description

Bit	Symbol	Value	Description	Reset value
0	EN		RTC 32 kHz clock enable.	1
		0	Disabled. RTC clock off.	
		1	Enabled. RTC clock on.	
31:1	-	-	Reserved.	0

7.5.81 PLL registers

The PLL provides a wide range of frequencies and can potentially be used for many on-chip functions. This device has three PLLs. System PLL, USB PLL, and Audio PLL. See Section 7.6.4 “System PLL functional description” for additional details of System PLL operation.

7.5.81.1 System PLL

7.5.81.1.1 System PLL control register

The SYSPLLCTRL register provides control over basic selections of PLL modes and operating details.

Table 201. System PLL control register (SYSPLLCTRL, main syscon: offset 0x580 bit description

Bit	Symbol	Value	Description	Reset value
3:0	SELR	-	Bandwidth select R value.	
9:4	SELI	-	Bandwidth select I value.	
14:10	SELP	-	Bandwidth select P value.	
15	BYPASS		PLL bypass control.	0
		0	Bypass disabled. PLL CCO is sent to the PLL post-dividers.	
		1	Bypass enabled. PLL input clock is sent directly to the PLL output (default).	
16	-		Reserved. Read value is undefined, only zero should be written.	0
17	UPLIMOFF		Disable upper frequency limiter.	0
		0	Normal mode. Only zero should be written.	
		1	Upper frequency limiter disabled.	
18	-		Reserved.	0
19	DIRECTI		PLL direct input enable.	0
		0	Disabled. The PLL input divider (N divider) output is used to drive the PLL CCO.	
		1	Enabled. The PLL input divider (N divider) is bypassed. the PLL input clock is used directly to drive the PLL CCO input.	

Table 201. System PLL control register (SYSPLLCTRL, main syscon: offset 0x580 bit description *…continued*

Bit	Symbol	Value	Description	Reset value
20	DIRECTO		PLL direct output enable.	0
		0	Disabled. The PLL output divider (P divider) is used to create the PLL output.	
		1	Enabled. The PLL output divider (P divider) is bypassed, the PLL CCO output is used as the PLL output.	
31:21	-	-	Reserved. Read value is undefined, only 0 should be written.	-

The values for SELP, SELI, and SELR depend on the value for M as expressed by the following pseudo-code:

```
if (M < 60) then
        SELP = (M>>1) + 1
     else
        SELP = 31;
if (M > 16384) then
        SELI = 1
     else if (M > 8192) then
        SELI = 2
     else if (M > 2048) then
        SELI = 4
     else if (M >= 501) then
        SELI = 8
     else if (M >=60) then
        SELI = 4*(1024/(M+9))
     else
        SELI = (M & 0x3C) + 4; /* & denotes bitwise AND */
     SELR = 0;
```

7.5.81.1.2 System PLL N-divider register

Remark: The PLL N-divider register does not use the direct binary representation of N divide value directly. Instead, it uses an encoded version NDEC.

Remark: While the PLL output is in use, do not change the NDEC value. Changing the NDEC value changes the FCCO frequency and can cause the system to fail.

- The valid range for N is 1 to 2^8. This value is encoded into a 10-bit NDEC value. The relationship can be expressed through the following pseudo-code:

```
N_max=0x00000100, x=0x00000080;
     switch (N) {
           case 0: x = 0xFFFFFFFF;
           case 1: x = 0x00000302;
           case 2: x = 0x00000202;

           default: for (i = N; i <= N_max; i++)
                          x = (((x ^ (x>>2) ^ (x>>3) ^ (x>>4)) & 1) << 7) |
    ((x>>1) & 0x7F); }
      NENC[9:0] = x;
```

Table 202. System PLL N-divider register (SYSPLLNDEC, main syscon: offset 0x588) bit description

Bit	Symbol	Description	Reset value
9:0	NDEC	Decoded N-divider coefficient value.	0
10	NREQ	NDEC reload request. When a 1 is written to this bit, the NDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the NDEC value is changed.	0
31:11	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.81.1.3 System PLL P-divider register

Remark: The PLL P-divider register does not use the direct binary representation of P divide value directly. Instead, it uses an encoded version PDEC.

Remark: While the PLL output is in use, do not change the PDEC value. Changing the PDEC value changes the PLL output frequency and can cause the system to fail.

- The valid range for P is from 1 to 2^5. This value is encoded into a 7-bit PDEC value. The relationship can be expressed through the following pseudo-code:

```
P_max=0x20, x=0x10;
     switch (P) {
           case 0: x = 0xFFFFFFFF;
           case 1: x = 0x00000062;
           case 2: x = 0x00000042;

           default: for (i = P; i <= P_max; i++)
                          x = (((x ^ (x>>2)) & 1) << 4) | ((x>>1) & 0xF); }
     PDEC[6:0] = x;
```

Table 203. System PLL P-divider register (SYSPLLPDEC, main syscon: offset 0x58C) bit description

Bit	Symbol	Description	Reset value
6:0	PDEC	Decoded P-divider coefficient value.	0
7	PREQ	PDEC reload request. When a 1 is written to this bit, the PDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the PDEC value is changed.	0
31:8	-	Reserved. Read value is undefined, only 0 should be written.	-

7.5.81.1.4 System PLL M divider register

Remark: MDEC does not use the direct binary representations of M directly. Instead, it uses an encoded version of M. The valid range for M is 1 to 2^15. This value is encoded into a 17-bit MDEC value.

The relationship between M and MDEC is expressed via the following pseudo-code.

```
M_max=0x00008000, x=0x00004000;
     switch (M) {
           case 0: x = 0xFFFFFFFF;
           case 1: x = 0x00018003;
           case 2: x = 0x00010003;

           default: for (i = M; i <= M_max; i++)
                          x = (((x ^ (x>>1)) & 1) << 14) | ((x>>1) & 0x3FFF); }
     MDEC[16:0] = x;
```

Table 204. System PLL M divider register (SYSPLLMDEC, main syscon: offset 0x590) bit description

Bit	Symbol	Description	Reset value
16:0	MDEC	Decoded M-divider coefficient value.	0
17	MREQ	MDEC reload request. When a 1 is written to this bit, the MDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the MDEN_SYS_PLL bit in the PDRUNCFG register if the MDEC value is changed.	0
31:18	-	Reserved. Read value is undefined, only 0 should be written.	-

7.5.81.1.5 System PLL status register

The read-only SYSPLLSTAT register provides the PLL lock status

Remark: The lock status does not reliably indicate the PLL status for the following configuration: fractional enabled or low input clock frequencies, such as, 32 kHz. In this case, see the PLL lock times listed in the specific device data sheet to obtain appropriate wait times for the PLL to lock.

Table 205. System PLL status register (SYSPLLSTAT, main syscon: offset 0x584) bit description

Bit	Symbol	Description	Reset value
0	LOCK	PLL lock indicator.	0
31:1	-	Reserved.	-

7.5.81.2 USB PLL

The following registers apply to the USB PLL.

7.5.81.2.1 USB PLL control register

The USBPLLCTRL register provides control of the USB PLL.

Table 206. USB PLL control register (USBPLLCTRL, main syscon: offset 0x51C bit description

Bit	Symbol	Value	Description	Reset value
7:0	MSEL		PLL feedback Divider value. Supplies the value "M" in the PLL frequency calculations. This value is encoded as follows: 0000 000 (0x0) = divide by 1 0000 001 (0x1) = divide by 2 0000 010 (0x2) = divide by 3 0000 011 (0x3) = divide by 4 .. .. 1111 1111 (0xFF) = divide by 256	
9:8	PSEL		PLL Post Divider value. Supplies the value "P" in the PLL frequency calculations. This value is encoded as follows: 00 (0x0) = divide by 1 01 (0x1) = divide by 2 10 (0x2) = divide by 4 11 (0x3) = divide by 8	
11:10	NSEL		PLL Pre Divider value. Supplies the value "N" in the PLL frequency calculations. This value is encoded as follows: 00 (0x0) = divide by 1 01 (0x1) = divide by 2 10 (0x2) = divide by 3 11 (0x3) = divide by 4	
12	DIRECT		Direct CCO clock output control.	0
		0	Clock signal goes through post divider.	
		1	Clock signal goes directly to output(s).	
13	BYPASS		Input clock bypass control.	0
		0	CCO clock is sent to post dividers.	
		1	PLL input clock is sent to post dividers.	
14	FBSEL		Feedback divider input clock control.	0
		0	Feedback divider clocked by CCO clock.	
		1	Feedback divider clocked by PLL output clock.	
31:15	-		Reserved. Read value is undefined, only 0 should be written.	-

7.5.81.2.2 USB PLL status register

The read-only USBPLLSTAT register provides the USBPLL lock status.

Remark: The lock status does not reliably indicate the USBPLL status for the following configuration: fractional enabled or low input clock frequencies, such as, 32 kHz. In this case, see the USBPLL lock times listed in the specific device data sheet to obtain appropriate wait times for the USBPLL to lock.

Table 207. USB PLL status register (USBPLLSTAT, main syscon: offset 0x520 bit description

Bit	Symbol	Description	Reset value
0	LOCK	USBPLL lock indicator.	0
31:1	-	Reserved.	-

7.5.81.3 Audio PLL

The following registers apply to the Audio PLL.

7.5.81.3.1 Audio PLL control register

The AUDPLLCTRL register provides control over basic selections of PLL modes and operating details.

Table 208. Audio PLL control register (AUDPLLCTRL, main syscon: offset 0x5A0) bit description

Bit	Symbol	Value	Description	Reset value
3:0	SELR		Bandwidth select R value.	
9:4	SELI		Bandwidth select I value.	
14:10	SELP		Bandwidth select P value.	
15	BYPASS		PLL bypass control.	0
		0	Bypass disabled. PLL CCO is sent to the PLL post-dividers.	
		1	Bypass enabled. PLL input clock is directly to the PLL output (default).	
16	-		Reserved. Read value is undefined, only zero should be written.	
17	UPLIMOFF		Disable upper frequency limiter.	0
		0	Normal mode.	
		1	Upper frequency limiter disabled.	
18	-		Reserved. Only zero should be written.	0
19	DIRECTI		PLL direct input enable.	0
		0	Disabled. The PLL input divider (N divider) output is used to drive the PLL CCO.	
		1	Enabled. The PLL input divider (N divider) is bypassed. the PLL input clock is used directly to drive the PLL CCO input.	
20	DIRECTO		PLL direct output enable.	0
		0	Disabled. The PLL output divider (P divider) is used to create the PLL output.	
		1	Enabled. The PLL output divider (P divider) is bypassed, the PLL CCO output is used as the PLL output.	
31:21	-		Reserved. Read value is undefined, only zero should be written.	-

7.5.81.3.2 Audio PLL status control register

The read-only AUDPLLSTAT register provides the Audio PLL lock status.

Remark: The lock status does not reliably indicate the Audio PLL status for the following configuration: fractional enabled or low input clock frequencies, such as, 32 kHz. In this case, see the Audio PLL lock times listed in the specific device data sheet to obtain appropriate wait times for the Audio PLL to lock.

Table 209. Audio PLL status control register (AUDPLLSTAT, main syscon: offset 0x5A4) bit description

Bit	Symbol	Description	Reset value
0	LOCK	PLL lock indicator.	0
31:1	-	Reserved.	-

7.5.81.3.3 Audio PLL N-divider register

Remark: The PLL N-divider register does not use the direct binary representation of N divide value directly. Instead, it uses an encoded version NDEC.

Remark: While the PLL output is in use, do not change the NDEC value. Changing the NDEC value changes the FCCO frequency and can cause the system to fail.

- The valid range for N is 1 to 2^8. This value is encoded into a 10-bit NDEC value. The relationship can be expressed through the following pseudo-code:

```
N_max=0x00000100, x=0x00000080;
     switch (N) {
          case 0: x = 0xFFFFFFFF;
          case 1: x = 0x00000302;
          case 2: x = 0x00000202;

          default: for (i = N; i <= N_max; i++)
                        x = (((x ^ (x>>2) ^ (x>>3) ^ (x>>4)) & 1) << 7) |
    ((x>>1) & 0x7F); }
     NENC[9:0] = x;
```

Table 210. Audio PLL N divider register (AUDPLLNDEC, main syscon: offset 0x5A8) bit description

Bit	Symbol	Description	Reset value
9:0	NDEC	Decoded N-divider coefficient value.	0
10	NREQ	NDEC reload request. When a 1 is written to this bit, the NDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the NDEC value is changed.	0
31:11	-	Reserved. Read value is undefined, only 0 should be written.	-

7.5.81.3.4 Audio PLL P divider register

Remark: The PLL P-divider register does not use the direct binary representation of P divide value directly. Instead, it uses an encoded version PDEC.

Remark: While the PLL output is in use, do not change the PDEC value. Changing the PDEC value changes the PLL output frequency and can cause the system to fail.

- The valid range for P is from 1 to 2^5. This value is encoded into a 7-bit PDEC value. The relationship can be expressed through the following pseudo-code:

```
P_max=0x20, x=0x10;
     switch (P) {
          case 0: x = 0xFFFFFFFF;
          case 1: x = 0x00000062;
          case 2: x = 0x00000042;

          default: for (i = P; i <= P_max; i++)
                        x = (((x ^ (x>>2)) & 1) << 4) | ((x>>1) & 0xF); }
     PDEC[6:0] = x;
```

Table 211. Audio PLL P divider register (AUDPLLPDEC, main syscon: offset 0x5AC) bit description

Bit	Symbol	Description	Reset value
6:0	PDEC	Decoded P-divider coefficient value.	0
7	PREQ	PDEC reload request. When a 1 is written to this bit, the PDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the PDEC value is changed.	0
31:8	-	Reserved. Read value is undefined, only 0 should be written.	-

7.5.81.3.5 Audio PLL M divider register

Remark: MDEC does not use the direct binary representations of M directly. Instead, it uses an encoded version of M. The valid range for M is 1 to 2^15. This value is encoded into a 17-bit MDEC value.

The relationship between M and MDEC is expressed using the following pseudo-code.

```
M_max=0x00008000, x=0x00004000;
     switch (M) {
           case 0: x = 0xFFFFFFFF;
           case 1: x = 0x00018003;
           case 2: x = 0x00010003;

           default: for (i = M; i <= M_max; i++)
                          x = (((x ^ (x>>1)) & 1) << 14) | ((x>>1) & 0x3FFF); }
     MDEC[16:0] = x;
```

Table 212. Audio PLL M divider register (AUDPLLMDEC, main syscon: offset 0x5B0) bit description

Bit	Symbol	Description	Reset value
16:0	MDEC	Decoded M-divider coefficient value.	0
17	MREQ	MDEC reload request. When a 1 is written to this bit, the MDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the MDEN_SYS_PLL bit in the PDRUNCFG register if the MDEC value is changed.	0
31:18	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.81.3.6 Audio PLL fractional divider control register

The Audio PLL includes an additional fractional divider. The SEL_EXT bit in the AUDPLLFRAC control register determines whether the fractional divider is used (SEL_EXT=0) or bypassed (SEL_EXT=1).

When the fractional divider is active, the sigma-delta modulator block generates divider values M and M+1 in the correct proportion so that an average division ratio of M+K/L is realized where $0 \leq K \leq L$ and M, K, and L are integer values. M Is determined by the integer part of the AUDPLLFRAC register (CTRL[21:15]) and K is determined by the fractional part of the AUDPLLFRAC register (CTRL[14:0]). Consecutive M and M+1 values are then further encoded into appropriate MDEC values before being presented as input to the M-divider.

Table 213. Audio PLL fractional divider control register (AUDPLLFRAC, main syscon: offset 0x5B4) bit description

Bit	Symbol	Value	Description	Reset value
21:0	CTRL		PLL fractional divider control word	0
22	REQ		Writing 1 to REQ signal loads CTRL value into fractional wrapper modulator. This bit must be pulled low and high again to reload a new CTRL value.	0
23	SEL_EXT		Select fractional divider.	0
		0	Enable fractional divider.	
		1	MDEC enabled. Fractional divider bypassed.	
31:24	-		Reserved. Read value is undefined, only zero should be written.	-

7.5.82 Sleep configuration register 0

The PDSLEEPCFG0 register controls the power to various analog blocks while the CPU is in a reduced power mode. Entering reduced power modes is typically accomplished by calling the Power API. See Section 9.4.2 “Chip_POWER_EnterPowerMode”. It is also possible to configure the PDSLEEPCFG0 and PDSLEEPCFG1 directly, then execute a WFI instruction to enter reduced power modes.

Table 214. Sleep configuration register (PDSLEEPCFG0, main syscon: offset 0x600) bit description

Bit	Symbol	Description	Reset value
31:0	-	See bit descriptions in the PDRUNCFG0 register.	0x03F8 0540

7.5.83 Sleep configuration register 1

See description of PDSLEEPCFG0.

Table 215. Sleep configuration register (PDSLEEPCFG1, main syscon: offset 0x604) bit description

Bit	Symbol	Description	Reset value
31:0	-	See bit descriptions in the PDRUNCFG1 register.	0x1000 0000

7.5.84 Power Configuration register 0

The PDRUNCFG0 register controls the power to various analog blocks.

Remark: For safety, this register should not be written. Changing the contents of PDRUNCFG0 should be accomplished by writing to PDRUNCFGSET0 and/or PDRUNCFGCLR0. This prevents inadvertent changes to unintended bits. **Reserved bits must not be changed by user software.**

Regarding bits 13 through 16, see Section 2.1.2 for details of SRAM configuration.

Table 216. Power Configuration register (PDRUNCFG0, main syscon: offset 0x610) bit description

Bit	Symbol	Value	Description	Reset value
3:0	-		Reserved.	-
4	PDEN_FRO		FRO oscillator.	0
		0	Powered.	
		1	Powered down.	
5	-		Reserved.	-
6	PDEN_TS		Temp sensor. Also, enable/disable bit 9.	1
		0	Powered.	
		1	Powered down.	
7	PDEN_BOD_RST		Brown-out Detect reset.	0
		0	Powered.	
		1	Powered down.	
8	PDEN_BOD_INTR		Brown-out Detect interrupt.	1
		0	Powered.	
		1	Powered down.	
9	PDEN_VD2_ANA		Analog supply for System Oscillator (also enable/disable bit 3 in PDRUNCFG1 register), Temperature Sensor (also, enable/disable bit 6), ADC (also, enable/disable bits 10, 19, and 23).	1
		0	Powered.	
		1	Powered down.	
10	PDEN_ADC0		ADC power. Also, enable/disable bits 9, 19, and 23.	1
		0	Powered.	
		1	Powered down.	
12:11	-		Reserved.	-
13	PDEN_SRAMX		PDEN_SRAMX controls SRAMX (also enable/disable bit 27).	0
		0	Powered.	
		1	Powered down.	
14	PDEN_SRAM0		PDEN_SRAM0 controls SRAM0 (also enable/disable bit 27).	0
		0	Powered.	
		1	Powered down.	
15	PDEN_SRAM1_2_3		PDEN_SRAM1_2_3 controls SRAM1, SRAM2, and SRAM3 (also enable/disable bit 27).	0
		0	Powered.	
		1	Powered down.	

Table 216. Power Configuration register (PDRUNCFG0, main syscon: offset 0x610) bit description

Bit	Symbol	Value	Description	Reset value
16	PDEN_USB_RAM		PDEN_USB_SRAM controls USB_RAM (also enable/disable bit 27).	0
		0	Powered.	
		1	Powered down.	
17	PDEN_ROM		ROM (also enable/disable bit 27).	0
		0	Powered.	
		1	Powered down.	
18	-		Reserved.	-
19	PDEN_VDDA		VDDA to the ADC, must be enabled for the ADC to work (also enable/disable bit 9, 10, and 23).	1
		0	Powered.	
		1	Powered down.	
20	PDEN_WDT_OSC		Watchdog oscillator.	1
		0	Powered.	
		1	Powered down.	
21	PDEN_USB0_PHY		USB0 PHY power (also enable/disable bit 28).	1
		0	Powered.	
		1	Powered down.	
22	PDEN_SYS_PLL		System PLL power (also enable/disable bit 26).	1
		0	Powered.	
		1	Powered down.	
23	PDEN_VREFP		VREFP to the ADC must be enabled for the ADC to work (also enable/disable bit 9, 10, and 19).	1
		0	Powered.	
		1	Powered down.	
25:24	-		Reserved.	-
26	PDEN_VD3		Power control for all PLLs. Also, see bit 22 in PDRUNCFG0, bits 1 and 2 in PDRUNCFG1 register.	1
		0	Powered.	
		1	Powered down.	
27	PDEN_VD4		Power control for all SRAMs and ROM. Also, see bits 13 to 17 in PDRUNCFG0.	0
		0	Powered.	
		1	Powered down.	
28	PDEN_VD5		Power control both USB0 PHY and USB1 PHY. Also, see bit 21 in PDRUNCFG0 register and bit 0 in PDRUNCFG1 register.	1
		0	Powered.	
		1	Powered down.	
29	PDEN_VD6		Power control for EEPROM. Also, see bit 5 in PDRUNCFG1 register.	0
		0	Powered.	
		1	Powered down.	
31:30	-		Reserved.	-

7.5.85 Power Configuration register 1

The PDRUNCFG1 register controls the power to various analog blocks.

Remark: For safety, changes to this register should generally be accomplished by writing to PDRUNCFGSET1 and/or PDRUNCFGCLR1. This avoids the possibility of an interrupt changing the value of PDRUNCFG1 after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. **Reserved bits must not be changed by user software.**

Table 217. Power Configuration register (PDRUNCFG1, main syscon: offset 0x614) bit description

Bit	Symbol	Value	Description	Reset value
0	PDEN_USB1_PHY		USB1 high speed PHY (also, enable/disable bit 28 in PDRUNCFG0 register).	0
		0	Powered.	
		1	Powered-down.	
1	PDEN_USB1_PLL		USB PLL power (also, enable/disable bit 26 in PDRUNCFG0 register).	0
		0	Powered.	
		1	Powered-down.	
2	PDEN_AUD_PLL		Audio PLL power and fractional divider (also, enable/disable bit 26 in PDRUNCFG0 register).	0
		0	Powered.	
		1	Powered-down.	
3	PDEN_SYSOSC		System Oscillator Power (also, enable/disable bit 9 in PDRUNCFG0 register).	0
		0	Powered.	
		1	Powered-down.	
4	-	-	Reserved.	-
5	PDEN_EEPROM		EEPROM power (also, enable/disable bit 29 in PDRUNCFG0 register).	0
		0	Powered.	
		1	Powered-down.	
6	-		Reserved.	
7	PDEN_RNG		Random Number Generator Power.	1
		0	Powered.	
		1	Powered-down.	
31:8	-		Reserved.	-

7.5.86 Power configuration set register 0

Writing a 1 to a bit position in PDRUNCFGSET0 sets the corresponding position in PDRUNCFG0. This is a write-only register. For bit assignments, see Table 216.

Table 218. Power configuration set registers (PDRUNCFGSET0 main syscon: offset 0x620) bit description

Bit	Symbol	Description	Reset value
31:0	PD_SET	Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG0 are reserved and only zeroes should be written to them.	-

7.5.87 Power configuration set register 1

Writing a 1 to a bit position in PDRUNCFGSET1 sets the corresponding position in PDRUNCFG1. This is a write-only register. For bit assignments, see Table 216.

Table 219. Power configuration set registers (PDRUNCFGSET1 main syscon: offset 0x624) bit description

Bit	Symbol	Description	Reset value
31:0	PD_SET	Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG1 register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG1 are reserved and only zeroes should be written to them.	-

7.5.88 Power configuration clear register 0

Writing a 1 to a bit position in PDRUNCFGCLR0 clears the corresponding position in PDRUNCFG0. This is a write-only register. For bit assignments, see Table 216.

Table 220. Power configuration clear registers (PDRUNCFGCLR0, main syscon: offset 0x630) bit description

Bit	Symbol	Description	Reset value
31:0	PD_CLR	Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG0 are reserved and only zeroes should be written to them.	-

7.5.89 Power configuration clear register 1

Writing a 1 to a bit position in PDRUNCFGCLR1 clears the corresponding position in PDRUNCFG1. This is a write-only register. For bit assignments, see Table 216.

Table 221. Power configuration clear registers (PDRUNCFGCLR1, main syscon: offset 0x634) bit description

Bit	Symbol	Description	Reset value
31:0	PD_CLR	Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG1 register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG1 are reserved and only zeroes should be written to them.	-

7.5.90 Start enable register 0

The STARTER0 register enables an interrupt for wake-up from deep-sleep mode.

GPIO Pin interrupts, GPIO group interrupts, and selected peripherals such as USB0, USB1, DMIC, SPI, I2C, USART, WWDT, RTC, Micro-tick Timer, and BOD can be left running to allow wake-up from deep-sleep mode.

The pattern match feature of the pin interrupt requires a clock in order to operate, and will not wake up the device from reduced power modes beyond Sleep mode.

Remark: It is recommended that changes to the STARTER registers be accomplished by using the related STARTERSET and STARTERCLR registers. This avoids any unintentional setting or clearing of other bits.

Remark: Also enable the corresponding interrupts in the NVIC. See Table 90 “Interrupt Set-Enable Register 0”.

Table 222. Start enable register 0 (STARTER0, main syscon: offset 0x680) bit description

Bit	Symbol	Description	Reset value
0	WDT, BOD	WWDT and BOD interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
1	DMA	DMA wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
2	GINT0	Group interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	-
3	GINT1	Group interrupt 1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
4	PIN_INT0	GPIO pin interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
5	PIN_INT1	GPIO pin interrupt 1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
6	PIN_INT2	GPIO pin interrupt 2 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
7	PIN_INT3	GPIO pin interrupt 3 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
8	UTICK	Micro-tick Timer wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
13:9	-	Reserved. Read value is undefined, only zero should be written.	-
14	FLEXCOMM0	Flexcomm Interface 0 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
15	FLEXCOMM1	Flexcomm Interface 1 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
16	FLEXCOMM2	Flexcomm Interface 2 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
17	FLEXCOMM3	Flexcomm Interface 3 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
18	FLEXCOMM4	Flexcomm Interface 4 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
19	FLEXCOMM5	Flexcomm Interface 5 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
20	FLEXCOMM6	Flexcomm Interface 6 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
21	FLEXCOMM7	Flexcomm Interface 7 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
24:22	-	Reserved. Read value is undefined, only zero should be written.	-
25	DMIC	Digital microphone interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
26	HWVAD	Hardware voice activity detect interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
27	USB0_NEEDCLK	USB activity interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0

Table 222. Start enable register 0 (STARTER0, main syscon: offset 0x680) bit description *…continued*

Bit	Symbol	Description	Reset value
28	USB0	USB function interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
29	RTC	RTC interrupt alarm and wake-up timer. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
31:30	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.91 Start enable register 1

The STARTER1 register selects additional interrupts that allow waking up the part from deep-sleep mode.

The pattern match feature of the pin interrupt requires a clock in order to operate, and will not wake up the device from reduced power modes beyond Sleep mode.

Remark: It is recommended that changes to the STARTER registers be accomplished by using the related STARTERSET and STARTERCLR registers. This avoids any unintentional setting or clearing of other bits.

Remark: Also enable the corresponding interrupts in the NVIC. See Table 91 “Interrupt Set-Enable Register 1 register”.

Table 223. Start enable register 1 (STARTER1, main syscon: offset 0x684) bit description

Bit	Symbol	Description	Reset value
0	PINT4	GPIO pin interrupt 4 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
1	PINT5	GPIO pin interrupt 5 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
2	PINT6	GPIO pin interrupt 6 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
3	PINT7	GPIO pin interrupt 7 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
7:4	-	Reserved. Read value is undefined, only zero should be written.	-
8	FLEXCOMM8	Flexcomm Interface 8 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function.	0x0
9	FLEXCOMM9	Flexcomm Interface 9 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function.	0x0
14:10	-	Reserved. Read value is undefined, only zero should be written.	-
15	USB1	USB 1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
16	USB1 activity	USB 1 activity wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. USB 1 clock must be enabled.	0x0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.92 Start enable set register 0

Writing a 1 to a bit position in STARTERSET0 sets the corresponding position in STARTER0. This is a write-only register. For bit assignments, see Table 222.

Table 224. Start enable set register 0 (STARTERSET0, main syscon: offset 0x6A0) bit description

Bit	Symbol	Description	Reset value
31:0	START_SET0	Writing ones to this register sets the corresponding bit or bits in the STARTER0 register, if they are implemented. Bits that do not correspond to defined bits in STARTER0 are reserved and only zeroes should be written to them.	-

7.5.93 Start enable set register 1

Writing a 1 to a bit position in STARTERSET1 sets the corresponding position in STARTER1. This is a write-only register. For bit assignments, see Table 223.

Table 225. Start enable set register 1 (STARTERSET1, main syscon: offset 0x6A4) bit description

Bit	Symbol	Description	Reset value
31:0	START_SET1	Writing ones to this register sets the corresponding bit or bits in the STARTER1 register, if they are implemented. Bits that do not correspond to defined bits in STARTER1 are reserved and only zeroes should be written to them.	-

7.5.94 Start enable clear register 0

Writing a 1 to a bit position in STARTERCLR0 clears the corresponding position in STARTER0. This is a write-only register. For bit assignments, see Table 222.

Table 226. Start enable clear register 0 (STARTERCLR0, main syscon: offset 0x6C0) bit description

Bit	Symbol	Description	Reset value
31:0	START_CLR0	Writing ones to this register clears the corresponding bit or bits in the STARTER0 register, if they are implemented. Bits that do not correspond to defined bits in STARTER0 are reserved and only zeroes should be written to them.	-

7.5.95 Start enable clear register 1

Writing a 1 to a bit position in STARTERCLR1 clears the corresponding position in STARTER1. This is a write-only register. For bit assignments, see Table 223.

Table 227. Start enable clear register 1 (STARTERCLR1, main syscon: offset 0x6C4) bit description

Bit	Symbol	Description	Reset value
31:0	START_CLR1	Writing ones to this register clears the corresponding bit or bits in the STARTER1 register, if they are implemented. Bits that do not correspond to defined bits in STARTER1 are reserved and only zeroes should be written to them.	-

7.5.96 Hardware Wake-up control register

The primary of the hardware Wake-up control register is to provide the possibility for some peripherals to have DMA service during deep-sleep mode without waking up entire device.

These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts, and can

Table 228. Hardware Wake-up control register (HWWAKE, main syscon: offset 0x780) bit description

Bit	Symbol	Description	Reset value
0	FORCEWAKE	Force peripheral clocking to stay on during deep-sleep mode. When 1, clocking to peripherals is prevented from being shut down when the CPU enters deep-sleep mode. This is intended to allow a coprocessor to continue operating while the main CPU(s) are shut down.	0
1	FCWAKE	Wake for Flexcomm Interfaces. When 1, any Flexcomm Interface FIFO reaching the level specified by its own TXLVL will cause peripheral clocking to wake up temporarily while the related status is asserted.	0
2	WAKEDMIC	Wake for Digital Microphone. When 1, the digital microphone input FIFO reaching the level specified by TRIGLVL of either channel will cause peripheral clocking to wake up temporarily while the related status is asserted.	0
3	WAKEDMA	Wake for DMA. When 1, DMA being busy will cause peripheral clocking to remain running until DMA completes. This is generally used in conjunction with bit 1 and/or 2 in order to prevent peripheral clocking from being shut down as soon as the cause of wake-up is cleared, but before DMA has completed its related activity.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.97 Auto Clock-Gate Override Register

This register allows selectively disabling automatic clock gating for device SRAMs. By default, automatic clock gating turns off clocks to each internal SRAM after 16 bus clocks with no activity. This saves power when the SRAMs are not used for a period of time. When turned off due to automatic clock gating, there is a 1 clock delay for the next access to an SRAM. Automatic clock gating may be disabled for time-critical code, which may typically give a 1 or 2% speed improvement.

Table 229. Auto Clock-Gate Override Register (AUTOCGOR, main syscon: offset 0xE04) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only zero should be written.	-
1	RAM0X	When 1, automatic clock gating for RAMX and RAM0 are turned off.	0x0
2	RAM1	When 1, automatic clock gating for RAM1 is turned off.	0x0
3	RAM2	When 1, automatic clock gating for RAM2 is turned off.	0x0
4	RAM3	When 1, automatic clock gating for RAM3 is turned off.	0x0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

7.5.98 JTAG ID code register

This register contains the JTAG ID code.

Table 230. JTAG ID code register (JTAGIDCODE, main syscon: offset 0xFF4) bit description

Bit	Symbol	Description	Value
31:0	JTAGID	JTAG ID code.	0x10CAA02B

7.5.99 Device ID0 register

This register contains the part ID. The part ID can also be obtained using the ISP or IAP ReadPartID commands. See Table 25 and Table 76.

Table 231. Device ID0 register (DEVICE_ID0, main syscon: offset 0xFF8) bit description

Bit	Symbol	Description	Reset value
31:0	PARTID	Part ID	part-dependent

Table 232. Device ID0 register values

Part number	Part ID
LPC54605J256ET180	0x7F954605
LPC54605J512ET180	0xFFF54605
LPC54605J256BD100	0x7F954605
LPC54605J512BD100	0xFFF54605
LPC54605J256ET100	0x7F954605
LPC54605J512ET100	0xFFF54605
LPC54606J256ET100	0x7F954606
LPC54606J512ET100	0xFFF54606
LPC54606J256BD100	0x7F954606
LPC54606J512BD100	0xFFF54606
LPC54606J256ET180	0x7F954606
LPC54606J512BD208	0xFFF54606
LPC54607J256ET180	0x7F954607
LPC54607J512ET180	0xFFF54607
LPC54607J256BD208	0x7F954607
LPC54608J512ET180	0xFFF54608
LPC54608J512BD208	0xFFF54608
LPC54616J512ET100	0xFFF54616
LPC54616J512BD100	0xFFF54616
LPC54616J256ET180	0x7F954616
LPC54616J512BD208	0xFFF54616
LPC54618J512ET180	0xFFF54618
LPC54618J512BD208	0xFFF54618
LPC54628J512ET180	0xFFF54628

7.5.100 Device ID1 register

This register contains the boot ROM and die revisions.

Table 233. Device ID1 register (DEVICE_ID1, main syscon: offset 0xFFC) bit description

Bit	Symbol	Description	Value
31:0	REVID	Revision	0x08410CAA

7.5.101 Asynchronous peripheral reset control register

The ASYNCPRESETCTRL register allows software to reset specific peripherals attached to the async APB bridge. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the ASYNCPRESETCTRL registers be accomplished by using the related ASYNCPRESETCTRLSET and ASYNCPRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 234. Asynchronous peripheral reset control register (ASYNCPRESETCTRL, async syscon: offset 0x000) bit description

Bit	Symbol	Description	Reset value
12:0	-	Reserved.	-
13	CTIMER3	Standard counter/timer CTIMER3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
14	CTIMER4	Standard counter/timer CTIMER4 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0
31:15	-	Reserved.	-

7.5.102 Asynchronous peripheral reset control set register

Writing a 1 to a bit position in ASYNCPRESETCTRLSET sets the corresponding position in ASYNCPRESETCTRL. This is a write-only register. For bit assignments, see Table 234.

Table 235. Asynchronous peripheral reset control set register (ASYNCPRESETCTRLSET, async syscon: offset 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	ARST_SET	Writing ones to this register sets the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them.	-

7.5.103 Asynchronous peripheral reset control clear register

Writing a 1 to a bit position in ASYNCPRESETCTRLCLR clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 234.

Table 236. Asynchronous peripheral reset control clear register (ASYNCPRESETCTRLCLR, async syscon: offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	ARST_CLR	Writing ones to this register clears the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them.	-

7.5.104 Asynchronous APB clock control register

This register controls how the clock selected for the asynchronous APB peripherals is divided to provide the clock to the asynchronous peripherals. The clock will be stopped if the DIV field is set to zero.

Remark: It is recommended that changes to the ASYNCAPBCLKCTRL registers be accomplished by using the related ASYNCAPBCLKCTRLSET and ASYNCAPBCLKCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 237. Asynchronous APB clock control register (ASYNCAPBCLKCTRL, async syscon: offset 0x010) bit description

Bit	Symbol	Description	Reset value
12:0	-	Reserved	-
13	CTIMER3	Controls the clock for CTIMER3. 0 = Disable; 1 = Enable.	0
14	CTIMER4	Controls the clock for CTIMER4. 0 = Disable; 1 = Enable.	0
31:15	-	Reserved	-

7.5.105 Asynchronous APB clock control set register

Writing a 1 to a bit position in ASYNCAPBCLKCTRLSET sets the corresponding position in ASYNCAPBCLKCTRL. This is a write-only register. For bit assignments, see Table 234.

Table 238. Asynchronous APB clock control set register (ASYNCAPBCLKCTRLSET, async syscon: offset 0x014) bit description

Bit	Symbol	Description	Reset value
31:0	ACLK_SET	Writing ones to this register sets the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them.	-

7.5.106 Asynchronous APB clock control clear register

Writing a 1 to a bit position in ASYNCAPBCLKCTRLCLR clears the corresponding position in ASYNCAPBCLKCTRL. This is a write-only register. For bit assignments, see Table 234.

Table 239. Asynchronous APB clock control clear register (ASYNCAPBCLKCTRLCLR, async syscon: offset 0x018) bit description

Bit	Symbol	Description	Reset value
31:0	ACLK_CLR	Writing ones to this register clears the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCAPBCLKCTRL are reserved and only zeroes should be written to them.	-

7.5.107 Asynchronous clock source select register A

This register selects a potential clock for the asynchronous APB peripherals from among several clock sources.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 240. Asynchronous clock source select register A (ASYNCAPBCLKSELA, async syscon: offset 0x020) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for asynchronous clock source selector A.	0
		0x0	Main clock.	
		0x1	FRO 12 MHz.	
		0x2	audio_pll_clk.	
		0x3	i2c_clk_fc6.	
31:2	-	-	Reserved.	-

7.5.108 BOD control register

The BOD control register selects four separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in Table 241 are typical values. More details can be found in specific device data sheets.

Both the BOD interrupt and the BOD reset can wake-up the chip from sleep and deep-sleep modes if enabled. See Chapter 9 “LPC546xx Power profiles/Power control API”.

Table 241. BOD control register (BODCTRL, other system registers: offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
1:0	BODRSTLEV		BOD reset level.	0
		0x0	Level 0.	
		0x1	Level 1.	
		0x2	Level 2.	
		0x3	Level 3.	
2	BODRSTENA		BOD reset enable.	0
		0	Disable reset function.	
		1	Enable reset function.	
4:3	BODINTLEV		BOD interrupt level	0
		0x0	Level 0.	
		0x1	Level 1.	
		0x2	Level 2.	
		0x3	Level 3.	
5	BODINTENA		BOD interrupt enable.	0
		0	Disable interrupt function.	
		1	Enable interrupt function.	
6	BODRSTSTAT		BOD reset status. When 1, a BOD reset has occurred. Cleared by writing 1 to this bit.	0
7	BODINTSTAT		BOD interrupt status. When 1, a BOD interrupt has occurred. Cleared by writing 1 to this bit.	0
31:8	-	-	Reserved.	-

7.6 Functional description

7.6.1 Reset

Reset has the following sources:

- The $\overline{\text{RESET}}$ pin.
- Watchdog reset.
- Power-On Reset (POR).
- Brown Out Detect (BOD).
- ARM software reset.
- ISP-AP debug reset.

Assertion of the POR or the BOD reset, once the operating voltage attains a usable level, starts the FRO. After the FRO-start-up time (maximum of 6 μs on power-up), the FRO provides a stable clock output. The reset remains asserted until the external Reset is released, the oscillator is running, and the flash controller has completed its initialization.

On the assertion of any reset source (ARM software reset, POR, BOD reset, External reset, and Watchdog reset), the following processes are initiated:

1. The FRO is enabled or starts up if not running.
2. The flash wake-up starts. This takes approximately 250 μs or less.
3. The boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash.

When the internal Reset is removed, the processor begins executing at address 0, which is initially the Reset vector mapped from the boot block. At that point, all of the processor and peripheral registers have been initialized to predetermined values.

7.6.2 Brown-out detection

The part includes up to four levels for monitoring the voltage on the V_{DD} pin. If this voltage falls below one of the selected levels, the BOD asserts an interrupt signal to the NVIC or issues a reset, depending on the value of the BODRSTENA bit in the BOD control register (Table 241).

The interrupt signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC (see Table 90) in order to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register.

If the BOD interrupt is enabled in the STARTER0 register and in the NVIC, the BOD interrupt can wake up the chip from reduced power modes, not including deep power-down.

If the BOD reset is enabled, the forced BOD reset can wake up the chip from reduced power modes, not including deep power-down.

7.6.3 Flash accelerator functional description

The flash accelerator block allows maximization of the performance of the CPU when it is running code from flash memory, while also saving power. The flash accelerator also provides speed and power improvements for data accesses to the flash memory.

A description of the flash accelerator configuration register may be found in Section 7.5.64.

The flash accelerator is divided into several functional blocks:

- AHB matrix interface, accessible by all bus masters that have a connection to the matrix slave port used for flash memory.
- An array of eight 128-bit buffers
- Flash accelerator control logic, including address compare and flash control
- A flash memory interface

Figure 11 shows a simplified diagram of the flash accelerator blocks and data paths.

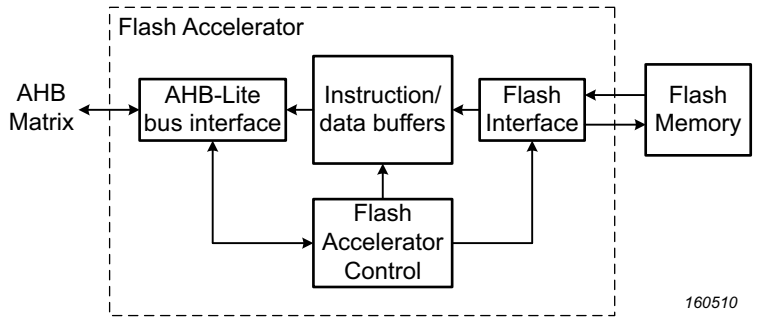


Fig 11. Simplified block diagram of the flash accelerator

In the following descriptions, the term “fetch” applies to an explicit flash read request from the CPU. “Prefetch” is used to denote a flash read of instructions beyond the current processor fetch address.

7.6.3.1 Flash memory bank

Flash programming operations are not controlled by the flash accelerator, but are handled as a separate function. The boot code includes flash programming functions that may be called as part of the application program, as well as loaders that may be used to accomplish initial flash programming.

7.6.3.2 Flash programming constraints

Since the flash memory does not allow accesses during programming and erase operations, it is necessary for the flash accelerator to force the CPU to wait if a memory access to a flash address is requested while the flash memory is busy with a programming operation. Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the flash memory. Application code, especially interrupts, can continue to run from other memories during flash erase/write operations.

In order to preclude the possibility of stale data being read from the flash memory, the flash accelerator buffers are automatically invalidated at the beginning of any flash programming or erase operation. Any subsequent read from a flash address will cause a new fetch to be initiated after the flash operation has completed.

7.6.4 System PLL functional description

The System PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the CPU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (example, the 12 MHz FRO) can be divided down to 1 MHz, then multiplied up to any other integer MHz (For example, 13, 14, 15, etc.).

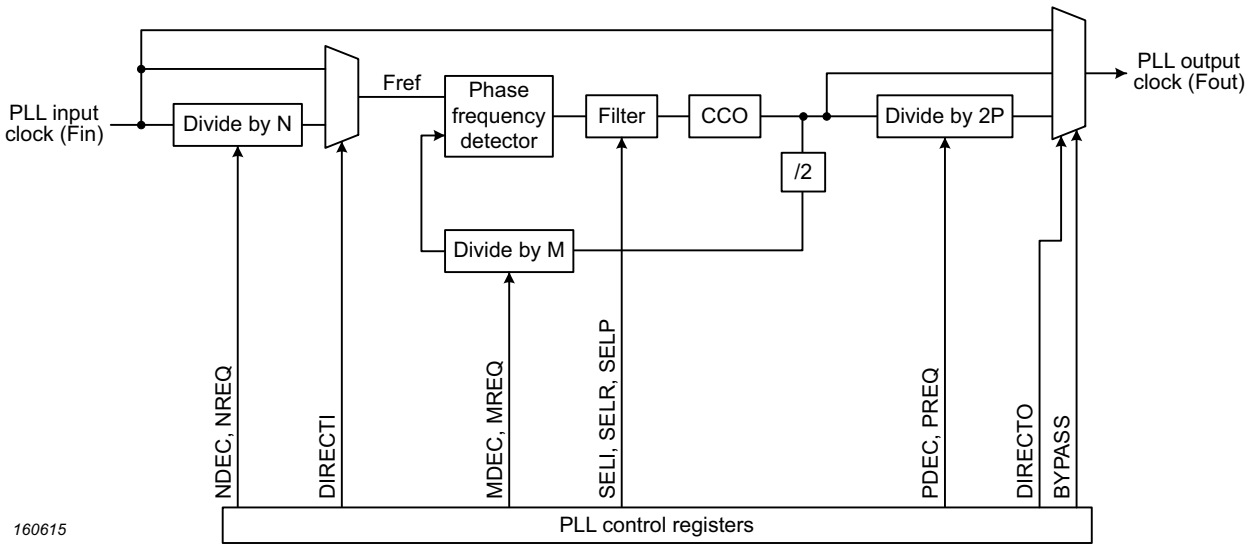


Fig 12. System PLL block diagram showing typical operation

7.6.4.1 PLL Features

- In fractional mode, Fref is between 2 MHz and 4 MHz.
- Input frequency: in normal mode, can include:
  - 6 kHz to 1.5 MHz Watchdog Oscillator
  - 32.768 kHz RTC clock
  - 12 MHz FRO
  - 1 MHz to 25 MHz from the external crystal oscillator
- CCO frequency: 275 MHz to 550 MHz.
- Output clock range: 4.3 MHz to 550 MHz. Note that the upper frequency limit of the PLL exceeds the upper frequency limit of this device.
- Programmable dividers:
  - Pre-divider. Divide by N, where N = 1 to 256
  - Feedback-divider. Divide by M or 2 x M (where M = 1 to 32,768)
  - Post-divider. Divide by 1 or 2 x P, where P = 1 to 32

- Lock detector.
- Power-down mode.
- Fractional divider mode.

7.6.4.2 PLL description

A number of sources may be used as an input to the PLL, see Figure 7. In addition, a block diagram of the PLL is shown in Figure 12. The PLL input, in the range of 6 kHz to 25 MHz, may initially be divided down by a value "N", which may be in the range of 1 to 256. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 1 through 32,768. The resulting frequency must be in the range of 275 MHz to 550 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

The PLL output may further be divided by a value "2P" if desired, where P is value in the range of 1 to 32.

All of the dividers that are part of the PLL use an encoded value, not the binary divide value.

There are additional dividers in the clocking system to bring the PLL output frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description.

For PLL register descriptions, see Section 7.5.81.

Remark: Use the PLL to boost the input frequency if a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz or 48 MHz clock (fro_hf) is not appropriate. The system PLL must be set up by calling an API (POWER_SetVoltageForFreq API in SDK software package) to deliver the amount of power needed for the CPU operating frequency. See Chapter 9 “LPC546xx Power profiles/Power control API”.

7.6.4.2.1 Lock detector

The lock detector measures the phase difference between the rising edges of the input and feedback clocks. Only when this difference is smaller than the so called “lock criterion” for more than seven consecutive input clock periods, the lock output switches from low to high. A single too large phase difference immediately resets the counter and causes the lock signal to drop (if it was high). Requiring seven phase measurements in a row to be below a certain figure ensures that the lock detector will not indicate lock until both the phase and frequency of the input and feedback clocks are very well aligned. This effectively prevents false lock indications, and thus ensures a glitch free lock signal.

The PLL lock indicator is not dependable when Fref is below 100 kHz or above 20 MHz. Instead, software should use a 6 ms time interval to insure the PLL will be stable.

7.6.4.2.2 Power-down

To reduce the power consumption when the PLL clock is not needed, a PLL Power-down mode has been incorporated. This mode is enabled by setting the PDEN_SYS_PLL bit to one in the power configuration register PDRUNCFG (Section 7.5.84). In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low to indicate that the PLL is not in lock.

When the PLL Power-down mode is terminated by setting the PDEN_SYS_PLL bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock. While in this state, new divider values may be entered, which will be used when the PLL Power-down state is exited by clearing PDEN_SYS_PLL.

7.6.4.3 Operating modes

The PLL includes several main operating modes, and a power-down mode. These are summarized in Table 242 and detailed in the following sections.

Table 242. PLL operating mode summary

Mode	PDEN_SYS_PLL bit in PDRUNCFG	Bits in SYSPLLCTRL:		
		BYPASS	UPLIMOFF	BANDSEL
Normal	0	0	0	1
Power-down	1	x [1]	x	x

[1] Use 1 if the PLL output is used even though the PLL is not altering the frequency.

7.6.4.3.1 Normal modes

Typical operation of the PLL includes an optional pre-divide of the PLL input, followed by a frequency multiplication, and finally an optional post-divide to produce the PLL output.

Notations used in the frequency equations:

- Fin = the input to the PLL.
- Fout = the output of the PLL.
- Fref = the PLL reference frequency, the input to the phase frequency detector.
- N = optional pre-divider value.
- M = feedback divider value, which represents the multiplier for the PLL.
- P = optional post-divider value. An additional divide-by-2 is included in the post-divider path.

A block diagram of the PLL as used in normal modes is shown in Figure 7.

In all variations of normal mode, the following requirements must be met:

- 275 MHz ≤ Fcco ≤ 550 MHz.
- 4 kHz ≤ Fin / N ≤ 25 MHz.

Normal mode with optional pre-divide

In the equations, use N = 1 when the pre-divider is not used:

The extra divide by 2 **is** in the feedback divider path.

Fout = Fcco = 2 x M x Fin / N

Normal mode with post-divide and optional pre-divide

In the equations, use N = 1 when the pre-divider is not used:

Fout = Fcco / (2 x P) = M x Fin / (N x P)

7.6.4.3.2 PLL power-down mode

If the PLL is not used, or if it there are cases where it is turned off in a running application, power can be saved by putting the PLL in power-down mode. Before this is done, the CPU and any peripherals that are not meant to stopped as well must be running from some other clock source.

7.6.4.4 PLL related registers

The PLL is controlled by registers described elsewhere in this chapter (see Section 7.5.81), and summarized below.

Table 243. Summary of PLL related registers

Register	Description	Section
SYSPLLCTRL	PLL control.	7.5.81.1.1
SYSPLLSTAT	PLL status.	7.5.81.1.5
SYSPLLNDEC	PLL N divider.	7.5.81.1.2
SYSPLLPDEC	PLL P divider.	7.5.81.1.3

7.6.4.5 PLL usage

As previously noted, the PLL divider settings used in the PLL registers are not simple binary values, they are encoded as shown in the PLL register descriptions. The divider values and their encoding can be found by calculation using the information in this document. Also, a PLL setting calculator can be found on the NXP website. The latter two possibilities are recommended in order to avoid PLL setup issues.

7.6.4.5.1 Procedure for determining PLL settings

In general, PLL configuration values may be found as follows:

1. Identify a desired PLL output frequency. This may depend on a specific interface frequency needed or be based on expected CPU performance requirements, and may be limited by system power availability.
2. Determine which clock source to use as the PLL input. This can be influenced by power or accuracy required, or by the potential to obtain the desired PLL output frequency.
3. Identify PLL settings to obtain the desired output from the selected input. The Fcco frequency must be either the actual desired output frequency, or the desired output frequency times 2 x P, where P is from 2 to 32. The Fcco frequency must also be a multiple of the PLL reference frequency, which is either the PLL input, or the PLL input divided by N, where N is from 2 to 256.

4. There may be several ways to obtain the same PLL output frequency. PLL power depends on Fcco (a lower frequency uses less power) and the divider used. Bypassing the input and/or output divider saves power.
5. Check that the selected settings meet all of the PLL requirements:
   - Fin is in the range of 32 kHz to 25 MHz.
   - Fcco is in the range of 275 MHz to 550 MHz.
   - Fout is in the range of 4.3 MHz to 550 MHz.
   - The pre-divider is either bypassed, or N is in the range of 2 to 256.
   - The post-divider is either bypassed, or P is in the range of 2 to 32.
   - M is in the range of 3 to 32,768.

   Also note that PLL startup time becomes longer as Fref drops below 500 kHz. At 500 kHz and above, startup time is up to 500 microseconds. Below 500 kHz, startup time can be estimated as 200 / Fref, or up to 6.1 milliseconds for Fref = 32 kHz. PLL accuracy and jitter is better with higher values of Fref.

7.6.4.5.2 PLL setup sequence

The following sequence should be followed to initialize and connect the PLL:

1. Make sure that the PLL output is disconnected from any downstream functions. If the PLL was previously being used to clock the CPU, and the CPU Clock Divider is being used, it may be set to speed up operation while the PLL is disconnected.
2. Select a PLL input clock source. See Section 7.5.31 “System PLL clock source select register”.
3. Set up the PLL dividers and mode settings. See Section 7.5.81 “PLL registers”.
4. Wait for the PLL output to stabilize. The value of the PLl lock may not be stable when the PLL reference frequency (FREF, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 µs when Fref is 500 kHz or greater and 200 / Fref seconds when FREF is less than 500 kHz.
5. If the PLL is used to clock the CPU, change the CPU Clock Divider setting for operation with the PLL, if needed. This must be done before connecting the PLL.
6. Connect the PLL to whichever downstream function is will be used with. The structure of the clock dividers may be seen on the right of Figure 7 “Clock generation”.

7.6.5 USB PLL functional description

The USB PLL is mainly intended to clock the USB. When used as the USB clock source, the input frequency is multiplied up to a multiple of 48 MHz. The PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the CPU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (example, the 12 MHz FRO) can be divided down to 1 MHz, then multiplied up to any other integer MHz (For example, 13, 14, 15, etc.). A precise external oscillator must be used to correctly operate USB1 in high speed mode.

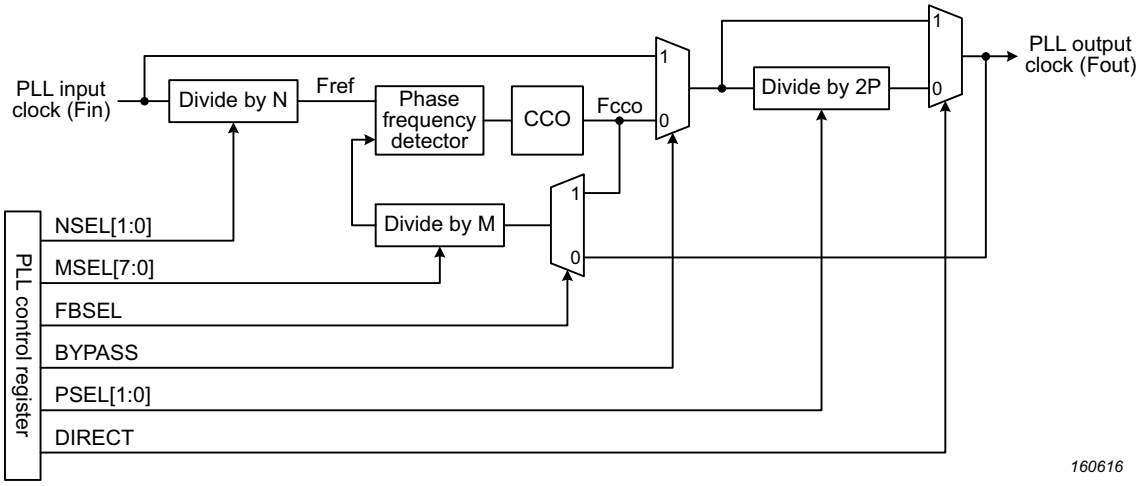


Fig 13. USB PLL block diagram showing typical operation

7.6.5.1 USB PLL Features

- Input frequency ranges from 1 MHz up to 25 MHz.
- CCO frequency: 156 MHz to 320 MHz.
- Output clock range: 9.75 MHz to 320 MHz. Note that the upper frequency limit of the PLL exceeds the upper frequency limit of this device.
- Programmable dividers:
  - Pre-divider. Divide by N, where N = 1 to 4
  - Feedback-divider. Divide by M (where M = 1 to 256)
  - Post-divider. Divide by 1 or 2 x P, where P = 1 to 8
- Lock detector.
- Power-down mode.

7.6.5.2 USB PLL description

A number of sources may be used as an input to the PLL, see Figure 7. In addition, a block diagram of the PLL is shown in Figure 13. The PLL input, in the range of 1 MHz to 25 MHz, may initially be divided down by a value "N", which may be in the range of 1 to 256. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 1 through 32,768. The resulting frequency must be in the range of 156 MHz to 320 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

The PLL output may further be divided by a value "2P" if desired, where P is value in the range of 1 to 32.

For PLL register descriptions, see Section 7.5.81.

7.6.6 Audio PLL functional description

The Audio PLL is mainly designed to generate audio clock. The Audio PLL is identical to the System PLL except that it supports an additional fractional divider to obtain more PLL frequencies with higher accuracy for audio applications. The PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the CPU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (example, the 12 MHz FRO) can be divided down to 1 MHz, then multiplied up to any other integer MHz (For example, 13, 14, 15, etc.).

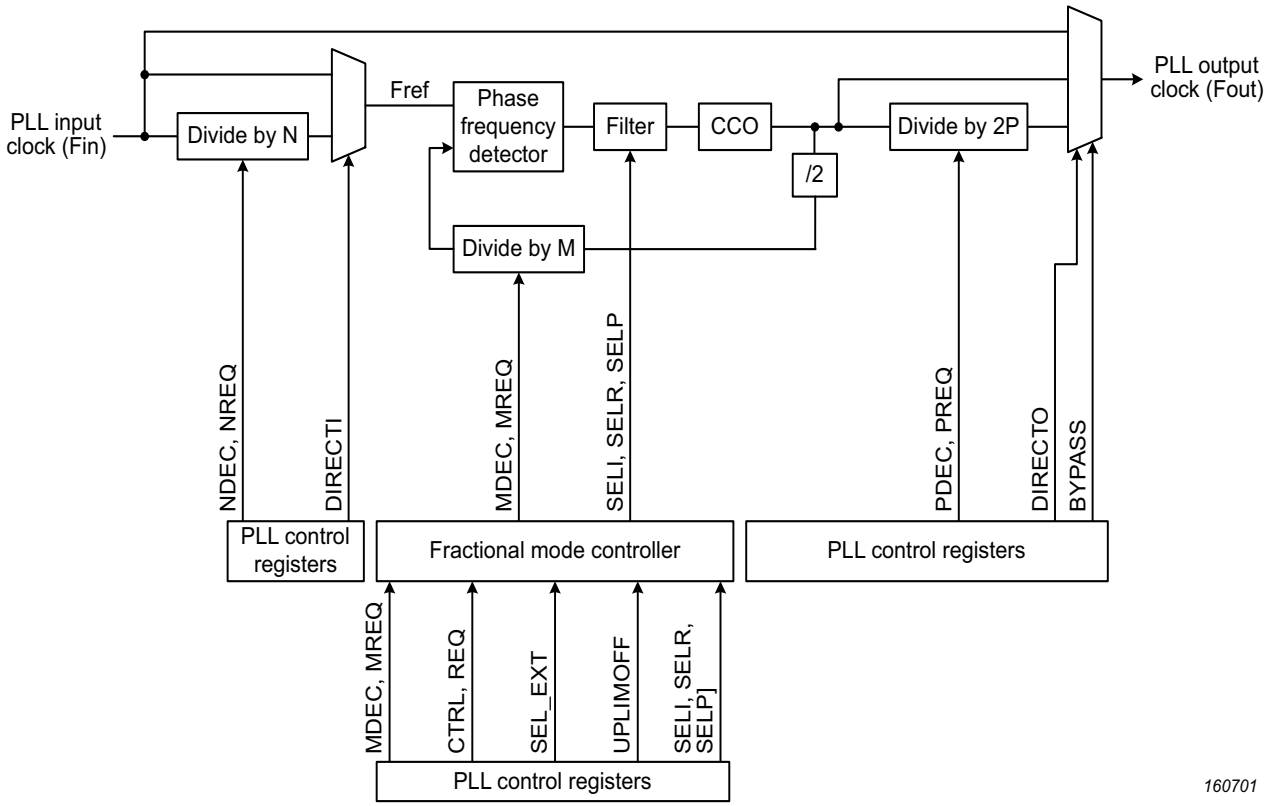


Fig 14. Audio PLL block diagram showing typical operation

7.6.6.1 Audio PLL Features

- In fractional mode, Fref is between 2 MHz and 4 MHz.
- Input frequency: in normal mode, can include:
  - 12 MHz FRO
  - 1 MHz to 25 MHz from the external crystal oscillator
- CCO frequency: 275 MHz to 550 MHz.
- Output clock range: 4.3 MHz to 550 MHz. Note that the upper frequency limit of the PLL exceeds the upper frequency limit of this device.
- Programmable dividers:
  - Pre-divider. Divide by N, where N = 1 to 256

- Feedback-divider. Divide by M or 2 x M (where M = 1 to 32,768)
- Post-divider. Divide by 1 or 2 x P, where P = 1 to 32

- Lock detector.
- Power-down mode.
- Fractional divider mode.

7.6.6.2 Audio PLL description

A number of sources may be used as an input to the PLL, see Figure 4. In addition, a block diagram of the PLL is shown in Figure 14. The PLL input, in the range of 1 MHz to 25 MHz, may initially be divided down by a value "N", which may be in the range of 1 to 256. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 1 through 32,768. The resulting frequency must be in the range of 275 MHz to 550 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

The PLL output may further be divided by a value "2P" if desired, where P is value in the range of 1 to 32. All of the dividers that are part of the PLL use an encoded value, not the binary divide value. There are additional dividers in the clocking system to bring the PLL output frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description.

For PLL register descriptions, see Section 7.5.81.

7.6.6.3 Lock detector

The lock detector measures the phase difference between the rising edges of the input and feedback clocks. Only when this difference is smaller than the so called “lock criterion” for more than seven consecutive input clock periods, the lock output switches from low to high. A single too large phase difference immediately resets the counter and causes the lock signal to drop (if it was high). Requiring seven phase measurements in a row to be below a certain figure ensures that the lock detector will not indicate lock until both the phase and frequency of the input and feedback clocks are very well aligned. This effectively prevents false lock indications, and thus ensures a glitch free lock signal.

The PLL lock indicator is not dependable when Fref is below 100 kHz or above 20 MHz. Instead, software should use a 6 ms time interval to insure the PLL will be stable. In fractional mode, the PLL will generally not lock, software should use a 6 ms time interval to insure the PLL will be stable.

7.6.6.4 Power-down

To reduce the power consumption when the PLL clock is not needed, a PLL Power-down mode has been incorporated. This mode is enabled by setting the PDEN_SYS_PLL bit to one in the power configuration register PDRUNCFG0 (Section 7.5.84). In this mode, the

internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low to indicate that the PLL is not in lock.

When the PLL Power-down mode is terminated by setting the PDEN_SYS_PLL bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock. While in this state, new divider values may be entered, which will be used when the PLL Power-down state is exited by clearing PDEN_SYS_PLL.

7.6.7 Frequency measure function

The Frequency Measure circuit is based on two 14-bit counters, one clocked by the reference clock and one by the target clock. Synchronization between the clocks is performed at the start and end of each count sequence.

A measurement cycle is initiated by software setting a control/status bit in the FREQMECTRL register (Table 187). The software can then poll this same measurement-in-progress bit which will be cleared by hardware when the measurement operation is completed.

The measurement cycle terminates when the reference counter rolls-over. At that point the state of the target counter is loaded into a capture field in the FREQMEAS register, and the measure-in-progress bit is cleared. Software can read this capture value and apply to it a specific calculation which will return the precise frequency of the target clock in MHz.

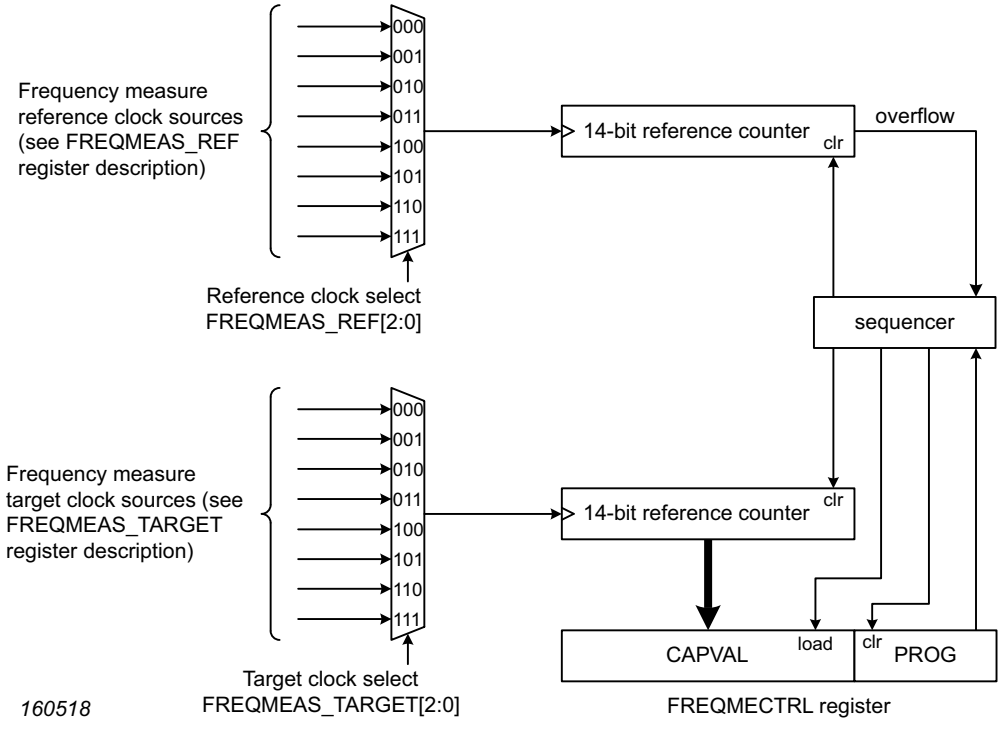


Fig 15. Frequency measure block diagram

Figure 15 shows a block diagram of the frequency measure function. Also see:

- Section 7.2.3 “Measure the frequency of a clock signal”
- Frequency measure control register (FREQMECTRL)- Section 7.5.67
- Frequency reference clock select register (FREQMEAS_REF) - Section 11.6.5
- Frequency target clock select register (FREQMEAS_TARGET) - Section 11.6.6

7.6.7.1 Accuracy

The frequency measurement function can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) to a high degree of accuracy using another on-chip clock of known frequency as a reference.

The following constraints apply:

- The frequency of the reference clock must be (somewhat) greater that the frequency of the target clock.
- The system clock used to access the frequency measure function register must also be greater than the frequency of the target clock.

The frequency measurement function circuit is able to measure the target frequency with an error of less than 0.1%, provided the reference frequency is precisely known.

Uncertainty in the reference clock (for example the ±1% accuracy of the FRO) will add to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

There can also be a modest loss of accuracy if the reference frequency exceeds the target frequency by a very large margin (25x or more). Accuracy is not a simple function of the magnitude of the frequency difference, however. Nearly identical frequency combinations, still with a spread of about 43x, result in errors of less than 0.05%.

If the target and reference clocks are different by more than a factor of approximately 500, then the accuracy decreases to ±4%.

8.1 Introduction

This chapter provides an overview of power related information about LPC546xx devices. These devices include a variety of power switches, and clock switches to allow fine tuning power usage to match requirements at different performance levels and reduced power modes.

To turn analog components on or off in active and sleep modes, use the PDRUNCFG0 register (see Table 216). In deep-sleep mode, the power profile API controls which analog peripherals remain powered up (see Section 9.4.2 “Chip_POWER_EnterPowerMode”). PDSLEEPCFG registers can also be used to turn analog peripherals on or off for deep-sleep mode.

8.2 General description

Power to the part is supplied via two power domains. The main power domain is powered by VDD and supplies power to the core, peripheral, memories, inputs and outputs via an on-chip regulator.

A second, always-on power domain is also powered by Vdd, and includes the RTC and wake-up timer. This domain always has power as long as sufficient voltage is supplied to Vdd.

Power usage is controlled by settings in register within the SYSCON block, regulator settings controlled via a Power API, and the operating mode of a CPU. The following modes are supported in order from highest to lowest power consumption:

1. Active mode:

   The part is in active mode after a Power-On Reset (POR) and when it is fully powered and operational after booting.

2. Sleep mode:

   Sleep mode saves a significant amount of power by stopping CPU execution without affecting peripherals or requiring significant wake-up time. The sleep mode affects the relevant CPU only. The clock to the core is shut off. Peripherals and memories are active and operational.

3. Deep-sleep mode:

   Deep-sleep mode is configurable and can potentially turn off nearly all on-chip power consumption other than the on-chip power supply, with the cost of a longer wake-up time. The deep-sleep mode affects the entire system, the clock to all CPUs is shut down and, if not configured, the peripherals receive no internal clocks. All SRAM and registers maintain their internal states.

   Through the power profiles API, selected peripherals such as USB, DMIC, SPI, I2C, USART, WWDT, RTC, Micro-tick Timer, and BOD can be left running in deep-sleep mode.

4. Deep power-down mode:

   Deep power-down mode shuts down virtually all on-chip power consumption, but requires a significantly longer wake-up time. For maximal power savings, the entire system (CPUs and all peripherals) is shut down except for the PMU and the RTC. On wake-up, the part reboots.

Table 244. Peripheral configuration in reduced power modes

Peripheral	Reduced power mode		
	Sleep	Deep-sleep	Deep power-down
FRO	Software configured	Software configured	Off
Flash	Software configured	Standby	Off
BOD	Software configured	Software configured	Off
PLL	Software configured	Off	Off
Watchdog osc and WWDT	Software configured	Software configured	Off
Micro-tick Timer	Software configured	Software configured	Off
DMA	Active	Configurable some for operations, see Section 8.3.4	Off
USART	Software configured	Off; but can create a wake-up interrupt in synchronous slave mode or 32 kHz clock mode	Off
SPI	Software configured	Off; but can create a wake-up interrupt in slave mode	Off
I2C	Software configured	Off; but can create a wake-up interrupt in slave mode	Off
USB0	Software configured	Software configured	Off
USB1	Software configured	Software configured	Off
Ethernet	Software configured	Off	Off
DMIC	Software configured	Software configured	Off
Other digital peripherals	Software configured	Off	Off
RTC oscillator	Software configured	Software configured	Software configured

8.2.1 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, you must configure the wake-up source. Each reduced power mode supports its own wake-up sources and needs to be configured accordingly as shown in Table 245.

Table 245. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Sleep	Any interrupt	Enable interrupt in NVIC.
	HWWAKE	Certain Flexcomm Interface and DMIC subsystem activity. See Section 7.5.96 "Hardware Wake-up control register".

Table 245. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Deep-sleep	Pin interrupts	Enable pin interrupts in NVIC and STARTER0 and/or STARTER1 registers.
	BOD interrupt	• Enable interrupt in NVIC and STARTER0 registers. • Enable interrupt in BODCTRL register. • Configure the BOD to keep running in this mode with the power API.
	BOD reset	Enable reset in BODCTRL register.
	Watchdog interrupt	• Enable the watchdog oscillator in the PDRUNCFG0 register. • Enable the watchdog interrupt in NVIC and STARTER0 registers. • Enable the watchdog in the WWDT MOD register and feed. • Enable interrupt in WWDT MOD register. • Configure the WDTOSC to keep running in this mode with the power API.
	Watchdog reset	• Enable the watchdog oscillator in the PDRUNCFG0 register. • Enable the watchdog and watchdog reset in the WWDT MOD register and feed.
	Reset pin	Always available.
	RTC 1 Hz alarm timer	• Enable the RTC 1 Hz oscillator in the RTCOSCCTRL register. • Enable the RTC bus clock in the AHBCLKCTRL0 register. • Start RTC alarm timer by writing a time-out value to the RTC COUNT register. • Enable the RTCALARM interrupt in the STARTER0 register.
	RTC 1 kHz timer time-out and alarm	• Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTC CTRL register. • Start RTC 1 kHz timer by writing a value to the WAKE register of the RTC. • Enable the RTC wake-up interrupt in the STARTER0 register.
	Micro-tick timer (intended for ultra-low power wake-up from deep-sleep mode	• Enable the watchdog oscillator in the PDRUNCFG0 register. • Enable the Micro-tick timer clock by writing to the AHBCLKCTRL1 register. • Start the Micro-tick timer by writing UTICK CTRL register. • Enable the Micro-tick timer interrupt in the STARTER0 register.
	I2C interrupt	Interrupt from I2C in slave mode. See Chapter 27 “LPC546xx I2C-bus interfaces”.
	SPI interrupt	Interrupt from SPI in slave mode. See Chapter 26 “LPC546xx Serial Peripheral Interfaces (SPI)”.
	USART interrupt	Interrupt from USART in slave or 32 kHz mode. See Chapter 25 “LPC546xx USARTs”.
	USB0 need clock interrupt	Interrupt from USB0 when activity is detected that requires a clock. See Section 37.7.6 “USB0 wake-up”.
	USB1 need clock interrupt	Interrupt from USB1 when activity is detected that requires a clock. See Section 39.7.6 “USB1 wake-up”.
	Ethernet interrupt	See Chapter 36 “LPC546xx Ethernet” for details of ethernet related interrupts.
	DMA interrupt	See Chapter 15 “LPC546xx DMA controller” for details of DMA related interrupts.
	HWWAKE	Certain Flexcomm Interface and DMIC subsystem activity. See Section 7.5.96 “Hardware Wake-up control register”.

Table 245. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Deep power-down	RTC 1 Hz alarm timer	• Enable the RTC 1 Hz oscillator in the RTC CTRL register. • Start RTC alarm timer by writing a time-out value to the RTC COUNT register.
	RTC 1 kHz timer time-out and alarm	• Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTCOSCCTRL register. • Enable the RTC bus clock in the AHBCLKCTRL0 register. • Start RTC 1 kHz timer by writing a value to the WAKE register of the RTC.
	Reset pin	Always available.

8.3 Functional description

8.3.1 Power management

The LPC546xx devices support a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are three processor power reduction with different peripherals running: sleep mode, deep-sleep mode, and deep power-down mode, activated by the power mode configure API.

Remark: The Debug mode is not supported in sleep, deep-sleep, or deep power-down modes.

8.3.2 Active mode

In Active mode, the CPU, memories, and peripherals are clocked by the AHB/CPU clock.

The chip is in Active mode after reset and the default power configuration is determined by the reset values of the PDRUNCFG0, PDRUNCFG1, AHBCLKCTRL0, AHBCLKCTRL1, and AHBCLKCTRL2 registers. The power configuration can be changed during run time.

8.3.2.1 Power configuration in active mode

Power consumption in active mode is determined by the following configuration choices:

- The AHBCLKCTRL registers control which memories and peripherals are running (Section 7.5.19 “AHB Clock Control register 0” and Section 7.5.20 “AHB Clock Control register 1”, and Section 7.5.21 “AHB Clock Control register 2”). To save power, turn off functions that are not needed by the application. If specific times are known when certain functions will not be needed, they can be turned off temporarily and turned back on when they will be needed.
- The power to various analog blocks (RAMs, PLL, oscillators, and the BOD circuit) can be controlled individually through the PDRUNCFG0 and PDRUNCFG1 registers (Section 7.5.84). As with clock controls, these blocks should generally be tuned off if not needed by the application. If turned off, time will be needed before these blocks can be used again after being turned on.
- The clock source for the system clock can be selected from the FRO (default), crystal oscillator, output of the PLL, the system oscillator, 32 kHz oscillator, or the watchdog oscillator (see Figure 7 and related registers).

- The system clock frequency can be selected (see Section 7.6.4 “System PLL functional description” and other clocking related sections). Generally speaking, everything uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If the PLL is not needed, it should be turned off to save power. Also, running the PLL at a lower CCO frequency saves power.
- Several peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements. See Chapter 9 “LPC546xx Power profiles/Power control API”.

8.3.3 Sleep mode

In sleep mode, the system clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the AHBCLKCTRL registers, continue operation during sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

As in active mode, the power API provides an easy way to optimize power consumption depending on CPU load and performance requirements in sleep mode. See Chapter 9 “LPC546xx Power profiles/Power control API”.

8.3.3.1 Power configuration in sleep mode

Power consumption in sleep mode is configured by the same settings as in Active mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in Active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in Active mode through the PDRUNCFG0, PDRUNCFG1, AHBCLKCTRL0, AHBCLKCTRL1, and AHBCLKCTRL2 registers.

8.3.3.2 Programming sleep mode

The following steps must be performed to enter sleep mode:

1. In the NVIC, enable all interrupts that are needed to wake up the part.
2. Call power API (see Section 9.4.2).

8.3.3.3 Wake-up from sleep mode

Sleep mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor or a reset occurs. After wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG and the AHBCLKCTRL registers. If a reset occurs, the microcontroller enters the default configuration in Active mode.

8.3.4 Deep-sleep mode

In deep-sleep mode, the system clock to the processor is disabled as in sleep mode. All analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. The main clock and all peripheral clocks are disabled. The FRO is disabled. The flash memory is put in standby mode.

Deep-sleep mode eliminates all power used by analog peripherals and all dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

GPIO Pin Interrupts, GPIO Group Interrupts, and selected peripherals such as USB0, USB1, DMIC, SPI, I2C, USART, WWDT, RTC, Micro-tick Timer, and BOD can be left running in deep-sleep mode The FRO, RTC oscillator, and the watchdog oscillator can be left running.

In some cases, DMA can operate in deep-sleep mode, see Section 15.5.9 “DMA in reduced power modes” and Section 7.5.96 “Hardware Wake-up control register”

8.3.4.1 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the STARTER registers and in the NVIC. In addition, any related analog block (for example, the RTC oscillator or the watchdog oscillator) must be explicitly enabled through the power API function. See Chapter 9 “LPC546xx Power profiles/Power control API”.

8.3.4.2 Programming deep-sleep mode

The following steps must be performed to enter deep-sleep mode:

1. Select wake-up sources and enable all selected wake-up events in the STARTER registers (Table 222 and Table 223) and in the NVIC.
2. Select the FRO 12 MHz as the main clock. See Table 148 and Table 149.
3. On power-up, the BOD is enabled. Power API disables BOD in deep-sleep mode. The user must disable BOD reset (bit 2 in the BODCTRL register) and clear bit 6 in the BODCTRL register before calling the power API to enter deep-sleep mode.
4. Call the power API with peripheral parameters to enable the analog peripherals as wake-up sources (see Chapter 9 “LPC546xx Power profiles/Power control API”). The peripheral parameter is a 64-bit value that corresponds to bits in the PDRUNCFG0 and PDRUNCFG1 registers.

8.3.4.3 Wake-up from deep-sleep mode

The part can wake up from deep-sleep mode in the following ways:

- Using a signal on one of the eight pin interrupts selected in Section 11.6.2 “Pin interrupt select registers”. Each pin interrupt must also be enabled in the STARTER0 register (Table 222) and in the NVIC.
- Using an interrupt from a block such as the watchdog interrupt or RTC interrupt, when enabled during the reduced power mode via the power API. Also enable the wake-up sources in the STARTER registers (Table 222 and Table 223) and the NVIC.
- Using a reset from the $\overline{\text{RESET}}$ pin, or the BOD or WWDT (if enabled in the power API).
- Using a wake-up signal from any of the serial peripherals that are operating in deep-sleep mode. Also enable the wake-up sources in the STARTER registers (Table 222 and Table 223) and the NVIC.
- GPIO group interrupt signal. The interrupt must also be enabled in the STARTER1 register (Table 223) and in the NVIC.
- RTC alarm signal or wake-up signal. See Chapter 19. Interrupts must also be enabled in the STARTER1 register (Table 223) and in the NVIC.

8.3.5 Deep power-down mode

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the RTC.

During deep power-down mode, the contents of the SRAM and registers (other than those in the RTC) are not retained. All functional pins are tri-stated in deep power-down mode as long as chip power supplied externally.

8.3.5.1 Power configuration in deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the device.

8.3.5.2 Wake-up from deep power-down mode:

Wake-up from deep power-down can be accomplished via the reset pin or the RTC.

8.3.5.3 Programming deep power-down mode using the RTC for wake-up:

The following steps must be performed to enter deep power-down mode when using the RTC for waking up:

1. Set up the RTC high resolution timer. Write to the RTC VAL register. This starts the high res timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Call the power API function. See Chapter 9 “LPC546xx Power profiles/Power control API”.

8.3.5.4 Wake-up from deep power-down mode using the RTC:

The part goes through the entire reset process when the RTC times out:

- The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip boots.
- All registers will be in their reset state.

9.1 How to read this chapter

The Power profiles and Power control APIs can be implemented using the power library from the SDK software package available on nxp.com.

9.2 Features

- Simple APIs to control power consumption and wake-up in all power modes.
- Manage power consumption for sleep and active modes
- Prepare the part to enter low power modes (sleep, deep-sleep, and deep power-down).
- Configure wake-up from deep-sleep via functions enabled by bits in the PDRUNCFG registers.

9.3 General description

Control of device power consumption or entry to low power modes can be configured through simple calls to the power profile. APIs exist to:

- Set up on-chip power based on the expected operating frequency.
- Set up reduced power modes.

Remark: Disable all interrupts before making calls to the power profile API. The interrupts can be re-enabled after the power profile API calls have completed.

9.4 API description

Power APIs provide functions to configure the system clock and set up the system for expected performance requirements. The Power APIs are available in the power library provided with SDK software package.

Table 246. Power API calls

Function prototype	API description	Section
`uint32_t POWER_SetVoltageForFreq (uint32_t desired_freq);`	Power API internal voltage configuration routine.This API configures the internal regulator for the desired active operating mode and frequency. Also sets up corresponding flash wait states.	9.4.1
`POWER_EnterPowerMode (power_mode_cfg_t mode, uint64_t exclude_from_pd);`	Power API power mode configuration routine.This API prepares the chip for reduced power modes: sleep, deep-sleep, or deep power-down mode. Also allows selection of which peripherals are kept alive in the reduced mode, and can therefore wake up the device from that mode.	9.4.2
`void CLOCK_SetupFROClocking(uint32_t iFreq);`	Setup the FRO high frequency output for either 48 MHz or 96 MHz. Updates the correct trim value and settings for high frequency FRO operation.	9.4.3

9.4.1 POWER_SetVoltageForFreq

This routine configures the device's internal power control settings according to the calling arguments. The goal is to prepare on-chip regulators to deliver the amount of power needed for the requested performance level, as defined by the CPU operating frequency.

Remark: The POWER_SetVoltageForFreq API should only be used when the system clock divider is 1 (AHBCLKDIV = 1, see Table 146).

Table 247. Chip_POWER_SetVoltage routine

Routine	Chip_POWER_SetVoltage
Prototype	`uint32_t POWER_SetVoltageForFreq (uint32_t desired_freq);`
Input parameter	**Param0:** desired frequency (in Hz)
Result	Error code. 0 = no error.
Description	Configures the internal device voltage in active mode, as well as setting up the corresponding flash wait states.

9.4.1.1 Param0: frequency

The frequency is the clock rate the CPU will be using during the selected mode. microcontroller uses to source the system and peripheral clocks. This operand must be an integer between 1 to 100 MHz inclusive.

9.4.1.2 Error or return codes

A return code of zero indicates that the operation was successful.

Table 248. Error codes

Return code	Error code	Description
0x000C 0004	PWR_ERROR_INVALID_CFG	Mode is invalid
0x000B 0002	ERR_CLK_INVALID_PARAM	Frequency is outside the supported range

9.4.2 Chip_POWER_EnterPowerMode

The Chip_POWER_EnterPowerMode API prepares the part, then enters any of the low power modes. Specifically for the deep-sleep mode, the API function configures which analog components remain running in those two modes, so that an interrupt from one of the analog peripherals can wake up the part.

Table 249. Chip_POWER_EnterPowerMode routine

Routine	Chip_POWER_EnterPowerMode
Prototype	`Void POWER_EnterPowerMode(power_mode_cfg_t mode, uint64_t exclude_from_pd);`
Input parameter	**Param0:** mode **Param1:** peripherals
Result	None
Description	Defines the low power mode (either sleep, deep-sleep, or deep power-down modes) and allows controlling which peripherals are powered up in the reduced power mode.

Remark: Aside from the analog peripherals listed with this parameter, the serial peripherals can also wake up the chip from deep-sleep mode from an interrupt triggered by an incoming signal. This wake-up scenario is not configured using the POWER_EnterPowerMode API. For details, see Section 25.3.2 "Configure the USART for

wake-up", Section 26.3.1 "Configure the SPI for wake-up", or Section 27.4.3 "Configure the I2C for wake-up".

9.4.2.1 Param0: mode

The mode parameter defines the low power mode and prepares the chip to enter the selected mode.

The following modes are valid:

```
typedef enum {
                POWER_SLEEP = 0,
                POWER_DEEP_SLEEP,
                POWER_DEEP_POWER_DOWN
} POWER_MODE_T;
```

9.4.2.2 Param1: peripherals

If sleep mode is selected with the mode parameter, the peripheral parameter is ignored.

The peripheral parameter defines which analog peripherals can wake up the chip from deep-sleep, or deep power-down mode. The selected peripherals remain running in deep-sleep mode. For example, the watchdog oscillator must be running if the WWDT is to remain active in deep-sleep mode or the RTC must be running if it is to remain active in deep power-down mode.

The peripheral parameter is a 64-bit value that corresponds to the PDRUNCFG0 and PDRUNCFG1 registers (see Table 216 "Power Configuration register (PDRUNCFG0, main syscon: offset 0x610) bit description" and Table 217 "Power Configuration register (PDRUNCFG1, main syscon: offset 0x614) bit description").

9.4.3 CLOCK_SetupFROClocking

This routine sets up the FRO high frequency output. The selected operating frequency must be either 48 MHz or 96 MHz, which are the two potential high frequency outputs of the FRO. The requested frequency is set up and the appropriate factory trim value will be used.

See Section 7.5.77 "FRO Control register" for more details to set-up the FRO high frequency output.

Table 250. Chip_POWER_SetFROHFRate routine

Routine	Chip_POWER_SetVoltage
Prototype	`void CLOCK_SetupFROClocking(uint32_t iFreq);`
Input parameter	**Param0:** desired frequency (in Hz)
Result	None.
Description	Setup the FRO high frequency output for the selected frequency, either 48 MHz or 96 MHz.

9.4.3.1 Param0: frequency

The frequency is the desired high frequency output clock for the FRO, 48 MHz or 96 MHz.

9.5 Functional description

9.5.1 Example low power mode control

9.5.1.1 Enter sleep mode

```
/* peripheral parameter is don't care*/
Chip_POWER_EnterPowerMode (POWER_SLEEP, 0x0 );
/* going to sleep mode. */
```

9.5.1.2 Enter deep-sleep mode and set up WWDT and BOD for wake-up

```
/* configure wwdt and bod event to wake up the chip from deep-sleep*/
LPC_SYSCON->STARTER0 = 0x3;
/* WDT_OSC and BOD are turned on */
Chip_POWER_EnterPowerMode (       POWER_DEEP_SLEEP, PDRUNCFG_PD_WDT_OSC |
                                  PDRUNCFG_PD_BOD_RESET | PDRUNCFG_PD_BOD_INTR);
/* going to deep-sleep mode. */
```

10.1 How to read this chapter

The IOCON block is included on all LPC546xx devices. Registers for pins that are not available on a specific package are reserved.

Remark: Some functions, such as SCTimer/PWM inputs, Frequency Measure, JTAG functions, and ADC triggers are not selected through IOCON. The connections for these function are described in either the Input Multiplexing chapter (Chapter 11 “LPC546xx Input multiplexing (INPUT MUX)”) or the chapter for the specific function. Refer to specific device data sheets for pinout details.

10.2 Features

The following electrical properties are configurable for standard port pins:

- Pull-up/pull-down resistor
- Open-drain mode
- Inverted function

Pins PIO0_13, PIO0_14, PIO3_23, and PIO3_24 are true open-drain pins that can be configured for different I^2C-bus speeds. Configuration options are somewhat different for these pins, as described in this chapter. Refer to a specific device data sheets for electrical details of these and other pins.

10.3 Basic configuration

Enable the clock to the IOCON in the AHBCLKCTRL0 register (Table 139). Once the pins are configured, the IOCON clock can be disabled in order to conserve power.

10.4 General description

10.4.1 Pin configuration

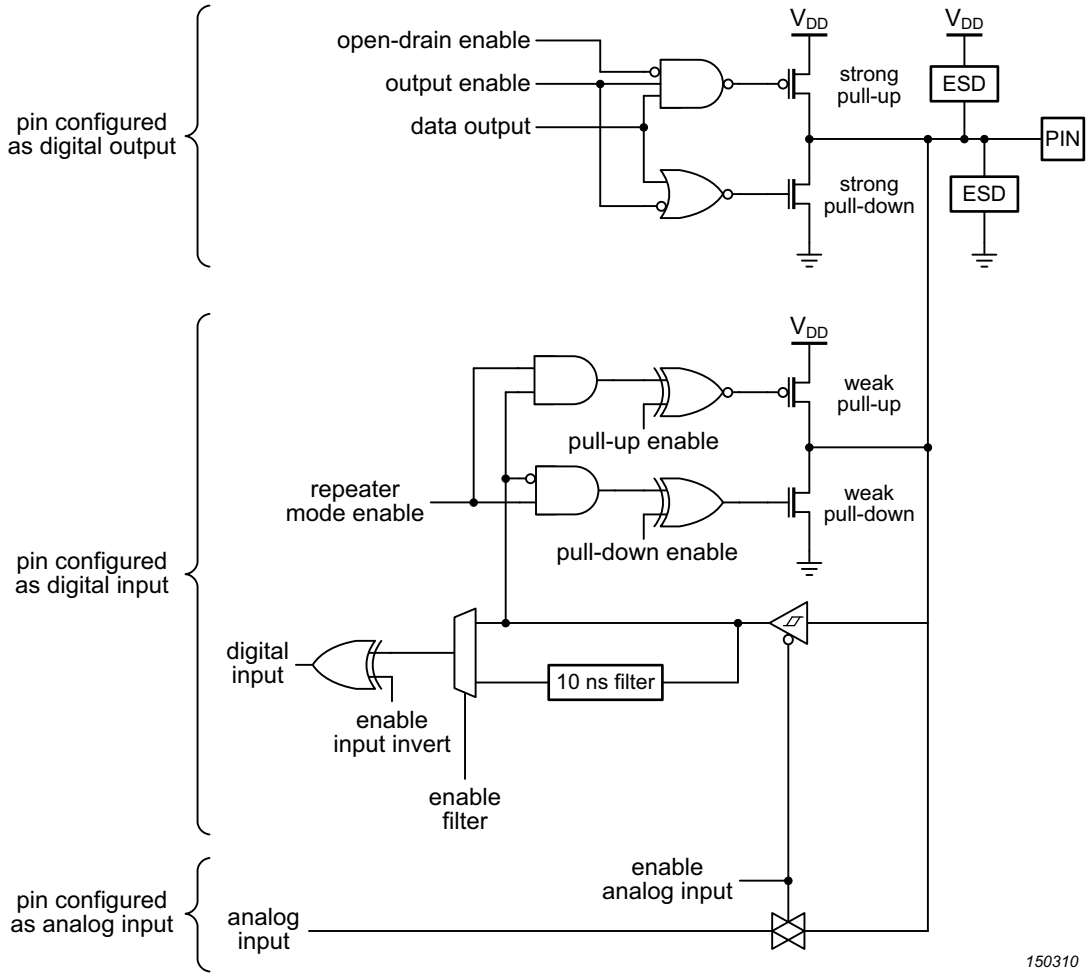


Fig 16. Pin configuration

10.4.2 IOCON registers

The IOCON registers control the functions of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I^2C function can be configured for different I^2C-bus modes, while pins that have an analog alternate function have an analog mode can be selected.Details of the IOCON registers are in Section 10.5. The following sections describe specific characteristics of pins.

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those pins. If a peripheral input function is defined as coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation. Therefore care should be taken to avoid this situation.

10.4.2.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (typically value 0) or to a special function. For pins set to GPIO, the DIR registers determine whether the pin is configured as an input or output (see Section 13.5.3). For any special function, the pin direction is controlled automatically depending on the function. The DIR registers have no effect for special functions.

10.4.2.2 Pin mode

The MODE bits in the IOCON register allow the selection of on-chip pull-up or pull-down resistors for each pin or select the repeater mode.

The possible on-chip resistor configurations are pull-up enabled, pull-down enabled, or no pull-up/pull-down. The default value is pull-up enabled.

The repeater mode enables the pull-up resistor if the pin is high and enables the pull-down resistor if the pin is low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. Such state retention is not applicable to the deep power-down mode. Repeater mode may typically be used to prevent a pin from floating (and potentially using significant power if it floats to an indeterminate state) if it is temporarily not driven.

10.4.2.3 Hysteresis

The input buffer for digital functions has built-in hysteresis. See the appropriate specific device data sheet for quantitative details.

10.4.2.4 Invert pin

This option is included to avoid having to include an external inverter on an input that is meant to be the opposite polarity of the external signal.

10.4.2.5 Analog/digital mode

When not in digital mode (DIGIMODE = 0) a pin is in analog mode, some digital pin functions are disabled and any analog pin functions are enabled. In digital mode (DIGIMODE = 1), any analog pin functions are disabled and digital pin functions are enabled. This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins, since they are typically 5V tolerant. All pin types include this control, even if they do not support any analog functions.

In order to use a pin that has an ADC input option for that purpose, select GPIO (FUNC field = 0) and disable the digital pin function (DIGIMODE = 0). The MODE field should also be set to 0.

In analog mode, the MODE field should be “Inactive” (00); the INVERT, FILTEROFF, and OD settings have no effect. For an unconnected pin that has an analog function, keep the DIGIMODE bit set to 1 (digital mode), and pull-up or pull-down mode selected in the MODE field.

10.4.2.6 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. The filter suppresses input pulses smaller than about 10 ns.

10.4.2.7 Output slew rate

The SLEW bits of digital outputs that do not need to switch state very quickly should be set to “standard”. This setting allows multiple outputs to switch simultaneously without noticeably degrading the power/ground distribution of the device, and has only a small effect on signal transition time. This is particularly important if analog accuracy is significant to the application. See the relevant specific device data sheet for more details.

10.4.2.8 I^2C modes

Pins that support I^2C with specialized pad electronics (PIO0_13, PIO0_14, PIO3_23, and PIO3_24) have additional configuration bits. These have multiple configurations to support I^2C variants. These are not hard-wired so that the pins can be more easily used for non-I^2C functions. See Table 254 for recommended mode settings.

For non-I^2C operation, these pins remain open-drain and can only drive low, regardless of how I2CSLEW and I2CDRIVE are set. They would typically be used with an external pull-up resistor if they are used as outputs unless they are used only to sink current. Leave I2CSLEW = 1, I2CDRIVE = 0, and I2CFILTER = 0 to maximize compatibility with other GPIO pins.

10.4.2.9 Open-drain mode

When output is selected, either by selecting a special function in the FUNC field, or by selecting the GPIO function for a pin having a 1 in the related bit of that port’s DIR register, a 1 in the OD bit selects open-drain operation, that is, a 1 disables the high-drive transistor. This option has no effect on the primary I2C pins. Note that the properties of a pin in this simulated open-drain mode are somewhat different than those of a true open drain output.

10.5 Register description

Each port pin PIOm_n has one IOCON register assigned to control the electrical characteristics of the pin.

Remark: See the Pinning information section of the appropriate device data sheet for details on which pins listed in Table 251 exist on each package configuration.

Table 251. Register overview: I/O configuration (base address 0x4000 1000)

Name	Access	Offset	Description	Reset value	Pin type	Section
PIO0_[0:31]	R/W	[0x000: 0x07C]	Digital I/O control for port 0 pins. These pins include an ADC input. PIO0_13 and PIO0_14 pins support I2C with true open-drain, drive and filtering for modes up to Fast-mode Plus.	PIO0_11/12: 0x0326 D, A: 0x320 I: 0x340	D, I, A	10.5.1 for type D 10.5.2 for type I 10.5.3 for type A
PIO1_[0:31]	R/W	[0x080: 0x0FC]	Digital I/O control for port 1 pins. These pins include an ADC input.	0x0320	D, A	10.5.1 for type D 10.5.3 for type A
PIO2_[0:31]	R/W	[0x100: 0x17C]	Digital I/O control for port 2 pins These pins include an ADC input.	0x0320	D, A	10.5.1 for type D 10.5.3 for type A
PIO3_[0:31]	R/W	[0x180: 0x1FC]	Digital I/O control for port 3 pins. These pins include an ADC input. PIO3_23 and PIO3_24 pins support I2C with true open-drain, drive and filtering for modes up to Fast-mode Plus.	D, A: 0x0320 I: 0x0340	D, I, A	10.5.1 for type D 10.5.2 for type I 10.5.3 for type A
PIO4_[0:31]	R/W	[0x200: 0x27C]	Digital I/O control for port 4 pins.	0x0320	D	10.5.1
PIO5_[0:10]	R/W	[0x280: 0x2A8]	Digital I/O control for port 5 pins.	0x0320	D	10.5.1

10.5.1 Type D IOCON registers

Table 252 applies to pins P0[0 to 9], P0[13 to 14], P0[17 to 22], P0[24 to 30], P1[1 to 31], P2[2 to 31], P3[0 to 20], P3[25 to 31], P4[0 to 31], P5[0 to 10].

Remark: The FUNC field for P0[11] and P0[12] resets to 0b110 (0x6), selecting the Serial Wire Debug function by default.

Table 252. Type D IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. See Table 256.	0 [1]
5:4	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
6	-	-	Reserved. Read value is undefined, only zero should be written.	NA
7	INVERT		Input polarity.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
8	DIGIMODE		Select Analog/Digital mode.	1
		0	Analog mode.	
		1	Digital mode.	
9	FILTEROFF		Controls input glitch filter.	1
		0	Filter enabled. Noise pulses below approximately 10 ns are filtered out.	
		1	Filter disabled. No input filtering is done.	
10	SLEW		Driver slew rate.	0
		0	Standard mode, output slew rate is slower, more outputs can be switched simultaneously.	
		1	Fast mode, output slew rate is faster, Refer to the appropriate specific device data sheet for details.	
11	OD		Controls open-drain mode.	0
		0	Normal. Normal push-pull output	
		1	Open-drain. Simulated open-drain output (high drive disabled).	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	NA

[1] For PIO0_11 and PIO0_12, the reset value is 0x5.

10.5.2 Type I IOCON registers

Table 253 applies to pins PIO0_13, PIO0_14, PIO3_23, and PIO3_24. See Table 254 for recommended setting for I2C operation.

Table 253. Type I IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. Table 257.	0
5:4	-	-	Reserved. Read value is undefined, only zero should be written.	NA
6	I2CSLEW		Controls slew rate of I^2C pad.	1
		0	I^2C mode.	
		1	GPIO mode.	
7	INVERT	1	Input polarity.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
8	DIGIMODE		Select Analog/Digital mode.	1
		0	Analog mode.	
		1	Digital mode.	
9	FILTEROFF		Controls input glitch filter.	1
		0	Filter enabled. Noise pulses below approximately 10 ns are filtered out.	
		1	Filter disabled. No input filtering is done.	
10	I2CDRIVE		Controls the current sink capability of the pin.	0
		0	Low drive. Output drive sink is 4 mA. This is sufficient for standard and Fast Mode I^2C.	
		1	High drive. Output drive sink is 20 mA. This is needed for Fast Mode Plus I^2C. See the appropriate specific device data sheet for details.	
11	I2CFILTEROFF		Configures I^2C features for standard mode, fast mode, and Fast Mode Plus operation.	0
		0	I^2C 50 ns glitch filter enabled. Typically used for Fast-mode and Fast-mode Plus I^2C.	
		1	I^2C 50 ns glitch filter not enabled. Typically used for Standard-mode I^2C.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 254. Suggested IOCON settings for I2C functions

	IOCON register bit					
Mode	**11: I2CFILTER**	**10: I2CDRIVE**	**9: FILTEROFF**	**8: DIGIMODE**	**7: INVERT**	**6: I2CSLEW**
GPIO 4 mA drive	0	0	0 [1]	1 [2]	0	1
GPIO 20 mA drive	0	1	0 [1]	1 [2]	0	1
Standard mode I^2C	1	0	1	1	0	0
Fast Mode Plus I^2C	0	1	1	1	0	0
High Speed slave I^2C	1	1	1	1	0	0

[1] The input filter may be turned by setting FILTEROFF off if it is not needed.

[2] The input may be turned off by clearing DIGIMODE if it is not needed.

10.5.3 Type A IOCON registers

Table 255 applies to pins P0[10 to 12], P0[15 to 16], P0[23], P0[31], P1[0], P2[0 to 1], and P3[21 to 22].

Table 255. Type A IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
3:0	FUNC	-	Selects pin function. Table 258.	0
5:4	MODE	-	Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
6	-	-	Reserved. Read value is undefined, only zero should be written.	NA
7	INVERT		Input polarity.	0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
8	DIGIMODE		Select Analog/Digital mode.	1
		0	Analog mode.	
		1	Digital mode.	
9	FILTEROFF		Controls input glitch filter.	1
		0	Filter enabled. Noise pulses below approximately 10 ns are filtered out.	
		1	Filter disabled. No input filtering is done.	
10	-	-	Reserved. Read value is undefined, only zero should be written.	NA
11	OD		Controls open-drain mode.	0
		0	Normal. Normal push-pull output	
		1	Open-drain. Simulated open-drain output (high drive disabled).	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Remark: To enable an ADC input, select the GPIO function and disable the digital functions of the pin by clearing the DIGIMODE bit in the related IOCON register.

Table 256. Type D I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO0_0	CAN1_RD	FC3_SCK	CTIMER0_MAT0	SCT0_GPI0	PDM0_CLK	-	-
PIO0_1	CAN1_TD	FC3_CTS_SDA_SSEL0	CTIMER0_CAP0	SCT0_GPI1	PDM0_DATA	-	-
PIO0_13	FC1_CTS_SDA_SSEL0	UTICK_CAP0	CTIMER0_CAP0	SCT0_GPI0	-	-	ENET_RXD0
PIO0_14	FC1_RTS_SCL_SSEL1	UTICK_CAP1	CTIMER0_CAP1	SCT0_GPI1	-	-	ENET_RXD1
PIO0_17	FC4_SSEL2	SD_CARD_DET_N	SCT0_GPI7	SCT0_OUT0	-	EMC_OEN	ENET_TXD1
PIO0_18	FC4_CTS_SDA_SSEL0	SD_WR_PRT	CTIMER1_MAT0	SCT0_OUT1	SCI1_SCLK	EMC_A[0]	-
PIO0_19	FC4_RTS_SCL_SSEL1	UTICK_CAP0	CTIMER0_MAT2	SCT0_OUT2	-	EMC_A[1]	FC7_TXD_SCL_MISO
PIO0_2	FC3_TXD_SCL_MISO	CTIMER0_CAP1	SCT0_OUT0	SCT0_GPI2	-	EMC_D[0]	-
PIO0_20	FC3_CTS_SDA_SSEL0	CTIMER1_MAT1	CTIMER3_CAP3	SCT0_GPI2	SCI0_IO	EMC_A[2]	FC7_RXD_SDA_MOSI
PIO0_21	FC3_RTS_SCL_SSEL1	UTICK_CAP3	CTIMER3_MAT3	SCT0_GPI3	SCI0_SCLK	EMC_A[3]	FC7_SCK
PIO0_22	FC6_TXD_SCL_MISO	UTICK_CAP1	CTIMER3_CAP3	SCT0_OUT3	-	-	USB0_VBUS
PIO0_24	FC0_RXD_SDA_MOSI	SD_D[0]	CTIMER2_CAP0	SCT0_GPI0	-	SPIFI_IO0	-
PIO0_25	FC0_TXD_SCL_MISO	SD_D[1]	CTIMER2_CAP1	SCT0_GPI1	-	SPIFI_IO1	-
PIO0_26	FC2_RXD_SDA_MOSI	CLKOUT	CTIMER3_CAP2	SCT0_OUT5	PDM0_CLK	SPIFI_CLK	USB0_IDVALUE
PIO0_27	FC2_TXD_SCL_MISO	-	CTIMER3_MAT2	SCT0_OUT6	PDM0_DATA	SPIFI_IO3	-
PIO0_28	FC0_SCK	-	CTIMER2_CAP3	SCT0_OUT7	TRACEDATA[3]	SPIFI_IO2	USB0_OVERCURRENTN
PIO0_29	FC0_RXD_SDA_MOSI	-	CTIMER2_MAT3	SCT0_OUT8	TRACEDATA[2]	-	-
PIO0_3	FC3_RXD_SDA_MOSI	CTIMER0_MAT1	SCT0_OUT1	SCT0_GPI3	-	EMC_D[1]	-
PIO0_30	FC0_TXD_SCL_MISO	-	CTIMER0_MAT0	SCT0_OUT9	TRACEDATA[1]	-	-
PIO0_4	CAN0_RD	FC4_SCK	CTIMER3_CAP0	SCT0_GPI4	-	EMC_D[2]	ENET_MDC
PIO0_5	CAN0_TD	FC4_RXD_SDA_MOSI	CTIMER3_MAT0	SCT0_GPI5	-	EMC_D[3]	ENET_MDIO
PIO0_6	FC3_SCK	CTIMER3_CAP1	CTIMER4_MAT0	SCT0_GPI6	-	EMC_D[4]	ENET_RX_DV
PIO0_7	FC3_RTS_SCL_SSEL1	SD_CLK	FC5_SCK	FC1_SCK	PDM1_CLK	EMC_D[5]	ENET_RX_CLK
PIO0_8	FC3_SSEL3	SD_CMD	FC5_RXD_SDA_MOSI	SWO	PDM1_DATA	EMC_D[6]	-
PIO0_9	FC3_SSEL2	SD_POW_EN	FC5_TXD_SCL_MISO	-	SCI1_IO	EMC_D[7]	-
PIO1_1	FC3_RXD_SDA_MOSI	-	CTIMER0_CAP3	SCT0_GPI5	-	-	USB1_OVERCURRENTN
PIO1_10	ENET_TXD1	FC1_RXD_SDA_MOSI	CTIMER1_MAT0	SCT0_OUT3	-	EMC_RASN	-
PIO1_11	ENET_TX_EN	FC1_TXD_SCL_MISO	CTIMER1_CAP1	USB0_VBUS	-	EMC_CLK[0]	-
PIO1_12	ENET_RXD0	FC6_SCK	CTIMER1_MAT1	USB0_PORTPWRN	-	EMC_DYCSN[0]	-
PIO1_13	ENET_RXD1	FC6_RXD_SDA_MOSI_DATA	CTIMER1_CAP2	USB0_OVERCURRENTN	USB0_FRAME	EMC_DQM[0]	-
PIO1_14	ENET_RX_DV	UTICK_CAP2	CTIMER1_MAT2	FC5_CTS_SDA_SSEL0	USB0_LEDN	EMC_DQM[1]	-

Table 256. Type D I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO1_15	ENET_RX_CLK	UTICK_CAP3	CTIMER1_CAP3	FC5_RTS_SCL_SSEL1	FC4_RTS_SCL_SSEL1	EMC_CKE[0]	-
PIO1_16	ENET_MDC	FC6_TXD_SCL_MISO_WS	CTIMER1_MAT3	SD_CMD	-	EMC_A[10]	-
PIO1_17	ENET_MDIO	FC8_RXD_SDA_MOSI	-	SCT0_OUT4	CAN1_TD	EMC_BLSN[0]	-
PIO1_18	-	FC8_TXD_SCL_MISO	-	SCT0_OUT5	CAN1_RD	EMC_BLSN[1]	-
PIO1_19	FC8_SCK	SCT0_OUT7	CTIMER3_MAT1	SCT0_GPI7	FC4_SCK	EMC_D[8]	-
PIO1_2	CAN0_TD	-	CTIMER0_MAT3	SCT0_GPI6	PDM1_CLK	-	USB1_PORTPWRN
PIO1_20	FC7_RTS_SCL_SSEL1	-	CTIMER3_CAP2	-	FC4_TXD_SCL_MISO	EMC_D[9]	-
PIO1_21	FC7_CTS_SDA_SSEL0	-	CTIMER3_MAT2	-	FC4_RXD_SDA_MOSI	EMC_D[10]	-
PIO1_22	FC8_RTS_SCL_SSEL1	SD_CMD	CTIMER2_MAT3	SCT0_GPI5	FC4_SSEL3	EMC_CKE[1]	-
PIO1_23	FC2_SCK	SCT0_OUT0	-	ENET_MDIO	FC3_SSEL2	EMC_A[11]	-
PIO1_24	FC2_RXD_SDA_MOSI	SCT0_OUT1	-	-	FC3_SSEL3	EMC_A[12]	-
PIO1_25	FC2_TXD_SCL_MISO	SCT0_OUT2	-	UTICK_CAP0	-	EMC_A[13]	-
PIO1_26	FC2_CTS_SDA_SSEL0	SCT0_OUT3	CTIMER0_CAP3	UTICK_CAP1	-	EMC_A[8]	-
PIO1_27	FC2_RTS_SCL_SSEL1	SD_D[4]	CTIMER0_MAT3	CLKOUT	-	EMC_A[9]	-
PIO1_28	FC7_SCK	SD_D[5]	CTIMER0_CAP2	-	-	EMC_D[12]	-
PIO1_29	FC7_RXD_SDA_MOSI_DATA	SD_D[6]	SCT0_GPI6	USB1_PORTPWRN	USB1_FRAME	EMC_D[13]	-
PIO1_3	CAN0_RD	-	-	SCT0_OUT4	PDM1_DATA	-	USB0_PORTPWRN
PIO1_30	FC7_TXD_SCL_MISO_WS	SD_D[7]	SCT0_GPI7	USB1_OVERCURRENTN	USB1_LEDN	EMC_D[14]	-
PIO1_31	MCLK	-	CT0_MAT2	SCT0_OUT6	FC8_CTS_SDA_SSEL0	EMC_D[15]	-
PIO1_4	FC0_SCK	SD_D[0]	CTIMER2_MAT1	SCT0_OUT0	FREQME_GPIO_CLK_A	EMC_D[11]	-
PIO1_5	FC0_RXD_SDA_MOSI	SD_D[2]	CTIMER2_MAT0	SCT0_GPI0	-	EMC_A[4]	-
PIO1_6	FC0_TXD_SCL_MISO	SD_D[3]	CTIMER2_MAT1	SCT0_GPI3	-	EMC_A[5]	-
PIO1_7	FC0_RTS_SCL_SSEL1	SD_D[1]	CTIMER2_MAT2	SCT0_GPI4	-	EMC_A[6]	-
PIO1_8	FC0_CTS_SDA_SSEL0	SD_CLK	-	SCT0_OUT1	FC4_SSEL2	EMC_A[7]	-
PIO1_9	ENET_TXD0	FC1_SCK	CTIMER1_CAP0	SCT0_OUT2	FC4_CTS_SDA_SSEL0	EMC_CASN	-
PIO2_10	ENET_RX_ER	SD_CARD_DET_N	-	-	-	-	-
PIO2_11	LCD_PWR	SD_VOLT[0]	-	-	FC5_SCK	-	-
PIO2_12	LCD_LE	SD_VOLT[1]	USB0_IDVALUE	-	FC5_RXD_SDA_MOSI	-	-
PIO2_13	LCD_DCLK	SD_VOLT[2]	-	-	FC5_TXD_SCL_MISO	-	-
PIO2_14	LCD_FP	USB0_FRAME	USB0_PORTPWRN	CTIMER0_MAT2	FC5_CTS_SDA_SSEL0	-	-
PIO2_15	LCD_AC	USB0_LEDN	USB0_OVERCURRENTN	CTIMER0_MAT3	FC5_RTS_SCL_SSEL1	-	-

Table 256. Type D I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO2_16	LCD_LP	USB1_FRAME	USB1_PORTPWRN	CTIMER1_MAT3	FC8_SCK	-	-
PIO2_17	LCD_CLKIN	USB1_LEDN	USB1_OVERCURRENTN	CTIMER1_CAP1	FC8_RXD_SDA_MOSI	-	-
PIO2_18	LCD_VD[0]	FC3_RXD_SDA_MOSI	FC7_SCK	CTIMER3_MAT0	-	-	-
PIO2_19	LCD_VD[1]	FC3_TXD_SCL_MISO	FC7_RXD_SDA_MOSI_DATA	CTIMER3_MAT1	-	-	-
PIO2_2	ENET_CRS	FC3_SSEL3	SCT0_OUT6	CTIMER1_MAT1	-	-	-
PIO2_20	LCD_VD[2]	FC3_RTS_SCL_SSEL1	FC7_TXD_SCL_MISO_WS	CTIMER3_MAT2	CTIMER4_CAP0	-	-
PIO2_21	LCD_VD[3]	FC3_CTS_SDA_SSEL0	MCLK	CTIMER3_MAT3	-	-	-
PIO2_22	LCD_VD[4]	SCT0_OUT7	-	CTIMER2_CAP0	-	-	-
PIO2_23	LCD_VD[5]	SCT0_OUT8	-	-	-	-	-
PIO2_24	LCD_VD[6]	SCT0_OUT9	-	-	-	-	-
PIO2_25	LCD_VD[7]	USB0_VBUS	-	-	-	-	-
PIO2_26	LCD_VD[8]	-	FC3_SCK	CTIMER2_CAP1	-	-	-
PIO2_27	LCD_VD[9]	FC9_SCK	FC3_SSEL2	-	-	-	-
PIO2_28	LCD_VD[10]	FC7_CTS_SDA_SSEL0	-	CTIMER2_CAP2	-	-	-
PIO2_29	LCD_VD[11]	FC7_RTS_SCL_SSEL1	FC8_TXD_SCL_MISO	CTIMER2_CAP3	CLKOUT	-	-
PIO2_3	ENET_TXD2	SD_CLK	FC1_RXD_SDA_MOSI	CTIMER2_MAT0	-	-	-
PIO2_30	LCD_VD[12]	-	-	CTIMER2_MAT2	-	-	-
PIO2_31	LCD_VD[13]	-	-	-	-	-	-
PIO2_4	ENET_TXD3	SD_CMD	FC1_TXD_SCL_MISO	CTIMER2_MAT1	-	-	-
PIO2_5	ENET_TX_ER	SD_POW_EN	FC1_CTS_SDA_SSEL0	CTIMER1_MAT2	-	-	-
PIO2_6	ENET_TX_CLK	SD_D[0]	FC1_RTS_SCL_SSEL1	CTIMER0_CAP0	-	-	-
PIO2_7	ENET_COL	SD_D[1]	FREQME_GPIO_CLK_B	CTIMER0_CAP1	-	-	-
PIO2_8	ENET_RXD2	SD_D[2]	-	CTIMER0_MAT0	-	-	-
PIO2_9	ENET_RXD3	SD_D[3]	-	CTIMER0_MAT1	-	-	-
PIO3_0	LCD_VD[14]	PDM0_CLK	-	CTIMER1_MAT0	-	-	-
PIO3_1	LCD_VD[15]	PDM0_DATA	-	CTIMER1_MAT1	-	-	-
PIO3_10	SCT0_OUT3	-	CTIMER3_MAT0	-	-	EMC_DYCSN[1]	TRACEDATA[0]
PIO3_11	MCLK	FC0_SCK	FC1_SCK	-	-	-	TRACEDATA[3]
PIO3_12	SCT0_OUT8	-	CTIMER3_CAP0	-	CLKOUT	EMC_CLK[1]	TRACECLK
PIO3_13	SCT0_OUT9	FC9_CTS_SDA_SSEL0	CTIMER3_CAP1	-	-	EMC_FBCK	TRACEDATA[1]
PIO3_14	SCT0_OUT4	FC9_RTS_SCL_SSEL1	CTIMER3_MAT1	-	-	-	TRACEDATA[2]

Table 256. Type D I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO3_15	FC8_SCK	SD_WR_PRT	-	-	-	-	-
PIO3_16	FC8_RXD_SDA_MOSI	SD_D[4]	-	-	-	-	-
PIO3_17	FC8_TXD_SCL_MISO	SD_D[5]	-	-	-	-	-
PIO3_18	FC8_CTS_SDA_SSEL0	SD_D[6]	CTIMER4_MAT0	CAN0_TD	SCT0_OUT5	-	-
PIO3_19	FC8_RTS_SCL_SSEL1	SD_D[7]	CTIMER4_MAT1	CAN0_RD	SCT0_OUT6	-	-
PIO3_2	LCD_VD[16]	FC9_RXD_SDA_MOSI	-	CTIMER1_MAT2	-	-	-
PIO3_20	FC9_SCK	SD_CARD_INT_N	CLKOUT	-	SCT0_OUT7	-	-
PIO3_23	FC2_CTS_SDA_SSEL0	-	UTICK_CAP3	-	-	-	-
PIO3_24	FC2_RTS_SCL_SSEL1	CTIMER4_CAP0	USB0_VBUS	-	-	-	-
PIO3_25	-	CTIMER4_CAP2	FC4_SCK	-	-	EMC_A[14]	-
PIO3_26	-	SCT0_OUT0	FC4_RXD_SDA_MOSI	-	-	EMC_A[15]	-
PIO3_27	-	SCT0_OUT1	FC4_TXD_SCL_MISO	-	-	EMC_A[16]	-
PIO3_28	-	SCT0_OUT2	FC4_CTS_SDA_SSEL0	-	-	EMC_A[17]	-
PIO3_29	-	SCT0_OUT3	FC4_RTS_SCL_SSEL1	-	-	EMC_A[18]	-
PIO3_3	LCD_VD[17]	FC9_TXD_SCL_MISO	-	-	-	-	-
PIO3_30	FC9_CTS_SDA_SSEL0	SCT0_OUT4	FC4_SSEL2	-	-	EMC_A[19]	-
PIO3_31	FC9_RTS_SCL_SSEL1	SCT0_OUT5	CTIMER4_MAT2	-	SCT0_GPI0	EMC_A[20]	-
PIO3_4	LCD_VD[18]	-	FC8_CTS_SDA_SSEL0	CTIMER4_CAP1	-	-	-
PIO3_5	LCD_VD[19]	-	FC8_RTS_SCL_SSEL1	CTIMER4_MAT1	-	-	-
PIO3_6	LCD_VD[20]	LCD_VD[0]	-	CTIMER4_MAT2	-	-	-
PIO3_7	LCD_VD[21]	LCD_VD[1]	-	CTIMER4_CAP2	-	-	-
PIO3_8	LCD_VD[22]	LCD_VD[2]	-	CTIMER4_CAP3	-	-	-
PIO3_9	LCD_VD[23]	LCD_VD[3]	-	CTIMER0_CAP2	-	-	-
PIO4_0	-	FC6_CTS_SDA_SSEL0	CTIMER4_CAP1	-	SCT0_GPI1	EMC_CSN[1]	-
PIO4_1	-	FC6_SCK	-	-	SCT0_GPI2	EMC_CSN[2]	-
PIO4_10	ENET_RX_DV	FC2_TXD_SCL_MISO	USB1_OVERCURRENTN	USB1_LEDN	SCT0_GPI3	-	-
PIO4_11	ENET_RXD0	FC2_CTS_SDA_SSEL0	USB0_IDVALUE	-	SCT0_GPI4	-	-
PIO4_12	ENET_RXD1	FC2_RTS_SCL_SSEL1	-	-	SCT0_GPI5	-	-
PIO4_13	ENET_TX_EN	CTIMER4_MAT0	-	-	SCT0_GPI6	-	-
PIO4_14	ENET_RX_CLK	CTIMER4_MAT1	FC9_SCK	-	SCT0_GPI7	-	-
PIO4_15	ENET_MDC	CTIMER4_MAT2	FC9_RXD_SDA_MOSI	-	-	-	-

Table 256. Type D I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO4_16	ENET_MDIO	CTIMER4_MAT3	FC9_TXD_SCL_MISO	-	-	-	-
PIO4_17	-	CAN1_TD	CTIMER1_CAP2	UTICK_CAP0	-	EMC_BLSN[2]	-
PIO4_18	-	CAN1_RD	CTIMER1_CAP3	UTICK_CAP1	-	EMC_BLSN[3]	-
PIO4_19	ENET_TXD0	SD_CLK	FC2_SCK	CTIMER4_CAP2	-	EMC_DQM[2]	-
PIO4_2	-	FC6_RXD_SDA_MOSI_DATA	-	-	SCT0_GPI3	EMC_CSN[3]	-
PIO4_20	ENET_TXD1	SD_CMD	FC2_RXD_SDA_MOSI	CTIMER4_CAP3	-	EMC_DQM[3]	-
PIO4_21	ENET_TXD2	SD_POW_EN	FC2_TXD_SCL_MISO	CTIMER2_MAT3	-	EMC_D[16]	-
PIO4_22	ENET_TXD3	SD_CARD_DET_N	FC2_RTS_SCL_SSEL1	CTIMER1_MAT3	-	EMC_D[17]	-
PIO4_23	ENET_RXD0	SD_WR_PRT	FC2_CTS_SDA_SSEL0	-	CTIMER1_MAT0	EMC_D[18]	-
PIO4_24	ENET_RXD1	SD_CARD_INT_N	FC7_RTS_SCL_SSEL1	-	CTIMER1_MAT1	EMC_D[19]	-
PIO4_25	ENET_RXD2	SD_D[0]	FC7_CTS_SDA_SSEL0	-	CTIMER1_MAT2	EMC_D[20]	-
PIO4_26	ENET_RXD3	SD_D[1]	-	UTICK_CAP2	CTIMER1_MAT3	EMC_D[21]	-
PIO4_27	ENET_TX_EN	SD_D[2]	-	FC1_SCK	CTIMER1_CAP0	EMC_D[22]	-
PIO4_28	ENET_TX_ER	SD_D[3]	-	FC1_RXD_SDA_MOSI	CTIMER1_CAP1	EMC_D[23]	-
PIO4_29	ENET_RX_ER	SD_D[4]	-	FC1_TXD_SCL_MISO	CTIMER1_CAP2	EMC_D[24]	-
PIO4_30	ENET_TX_CLK	SD_D[5]	CTIMER3_MAT0	FC1_RTS_SCL_SSEL1	CTIMER1_CAP3	EMC_D[25]	-
PIO4_31	ENET_RX_CLK	SD_D[6]	CTIMER3_MAT1	FC4_SCK	-	EMC_D[26]	-
PIO4_3	-	FC6_TXD_SCL_MISO_WS	CTIMER0_CAP3	-	SCT0_GPI4	EMC_DYCSN[2]	-
PIO4_4	-	FC4_SSEL3	FC0_RTS_SCL_SSEL1	-	SCT0_GPI5	EMC_DYCSN[3]	-
PIO4_5	-	FC9_CTS_SDA_SSEL0	FC0_CTS_SDA_SSEL0	CTIMER4_MAT3	SCT0_GPI6	EMC_CKE[2]	-
PIO4_6	-	FC9_RTS_SCL_SSEL1	-	-	SCT0_GPI7	EMC_CKE[3]	-
PIO4_7	-	CTIMER4_CAP3	USB0_PORTPWRN	USB0_FRAME	SCT0_GPI0	-	-
PIO4_8	ENET_TXD0	FC2_SCK	USB0_OVERCURRENTN	USB0_LEDN	SCT0_GPI1	-	-
PIO4_9	ENET_TXD1	FC2_RXD_SDA_MOSI	USB1_PORTPWRN	USB1_FRAME	SCT0_GPI2	-	-
PIO5_0	ENET_RX_DV	SD_D[7]	CTIMER3_MAT2	FC4_RXD_SDA_MOSI	-	EMC_D[27]	-
PIO5_1	ENET_CRS	SD_VOLT[0]	CTIMER3_MAT3	FC4_TXD_SCL_MISO	-	EMC_D[28]	-
PIO5_10	SCT0_GPI5	-	FC5_RTS_SCL_SSEL1	SCT0_OUT9	UTICK_CAP3	-	-
PIO5_2	ENET_COL	SD_VOLT[1]	CTIMER3_CAP0	FC4_CTS_SDA_SSEL0	-	EMC_D[29]	-
PIO5_3	ENET_MDC	SD_VOLT[2]	CTIMER3_CAP1	FC4_RTS_SCL_SSEL1	-	EMC_D[30]	-
PIO5_4	ENET_MDIO	SD_BACKEND_PWR	CTIMER3_CAP2	FC4_SSEL2	-	EMC_D[31]	-
PIO5_5	SCT0_GPI0	PDM1_CLK	CTIMER3_CAP3	FC4_SSEL3	TRACECLK	EMC_A[21]	-

Table 256. Type D I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO5_6	SCT0_GPI1	PDM1_DATA	FC5_SCK	SCT0_OUT5	TRACEDATA[0]	EMC_A[22]	-
PIO5_7	SCT0_GPI2	MCLK	FC5_RXD_SDA_MOSI	SCT0_OUT6	TRACEDATA[1]	EMC_A[23]	-
PIO5_8	SCT0_GPI3	PDM0_CLK	FC5_TXD_SCL_MISO	SCT0_OUT7	TRACEDATA[2]	EMC_A[24]	-
PIO5_9	SCT0_GPI4	PDM0_DATA	FC5_CTS_SDA_SSEL0	SCT0_OUT8	TRACEDATA[3]	EMC_A[25]	-

Table 257. Type I I/O Control registers: FUNC values and pin functions

Value of FUNC field in IOCON register							
Reg name/ FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO0_13	FC1_CTS_SDA_SSEL0	UTICK_CAP0	CTIMER0_CAP0	SCT0_GPI0	-	-	ENET_RXD0
PIO0_14	FC1_RTS_SCL_SSEL1	UTICK_CAP1	CTIMER0_CAP1	SCT0_GPI1	-	-	ENET_RXD1
PIO3_23	FC2_CTS_SDA_SSEL0	-	UTICK_CAP3	-	-	-	-
PIO3_24	FC2_RTS_SCL_SSEL1	CTIMER4_CAP0	USB0_VBUS	-	-	-	-

Table 258. Type A I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Reg name	FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO0_10	PIO0_10/ADC0_0	FC6_SCK	CTIMER2_CAP2	CTIMER2_MAT0	FC1_TXD_SCL_MISO	-	SWO	-
PIO0_11	PIO0_11/ADC0_1	FC6_RXD_SDA_MOSI_DATA	CTIMER2_MAT2	FREQME_GPIO_CLK_A	-	-	SWCLK	-
PIO0_12	PIO0_12/ADC0_2	FC3_TXD_SCL_MISO	-	FREQME_GPIO_CLK_B	SCT0_GPI7	-	SWDIO	-
PIO0_15	PIO0_15/ADC0_3	FC6_CTS_SDA_SSEL0	UTICK_CAP2	CTIMER4_CAP0	SCT0_OUT2	-	EMC_WEN	ENET_TX_EN
PIO0_16	PIO0_16/ADC0_4	FC4_TXD_SCL_MISO	CLKOUT	CTIMER1_CAP0	-	-	EMC_CSN[0]	ENET_TXD0
PIO0_31	PIO0_31/ADC0_5	FC0_CTS_SDA_SSEL0	SD_D[2]	CTIMER0_MAT1	SCT0_OUT3	TRACEDATA[0]	-	-
PIO1_0	PIO1_0/ADC0_6	FC0_RTS_SCL_SSEL1	SD_D[3]	CTIMER0_CAP2	SCT0_GPI4	TRACECLK	-	-
PIO2_0	PIO2_0/ADC0_7	-	FC0_RXD_SDA_MOSI	-	CTIMER1_CAP0	-	-	-
PIO2_1	PIO2_1/ADC0_8	-	FC0_TXD_SCL_MISO	-	CTIMER1_MAT0	-	-	-
PIO3_21	PIO3_21/ADC0_9	FC9_RXD_SDA_MOSI	SD_BACKEND_PWR	CTIMER4_MAT3	UTICK_CAP2	-	-	-
PIO3_22	PIO3_22/ADC0_10	FC9_TXD_SCL_MISO	-	-	-	-	-	-
PIO0_23	PIO0_23/ADC0_11	MCLK	CTIMER1_MAT2	CTIMER3_MAT3	SCT0_OUT4	-	SPIFI_CSN	-

11.1 How to read this chapter

Input multiplexing is present on all LPC546xx devices. Depending on the package, not all inputs from external pins may be available.

11.2 Features

- Configures the inputs to the SCT.
- Configures the inputs to the pin interrupt block and pattern match engine.
- Configures the inputs to the DMA triggers.
- Configures the inputs to the frequency measure function. This function is controlled by the FREQMECTRL register in the SYSCON block.

11.3 Basic configuration

Once set up, no clocks are needed for the input multiplexer to function. The system clock is needed only to write to or read from the INPUT MUX registers. Once the input multiplexer is configured, disable the clock to the INPUT MUX block in the AHBCLKCTRL register.

11.4 Pin description

The input multiplexer has no dedicated pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 259. INPUT MUX pin description

Pins	Peripheral	Section
Any existing pin on port 0 or 1	Pin interrupts 0 to 7	11.6.2
PIO0_11, PIO0_12, PIO1_4, PIO2_7	Frequency measure block	11.6.5
SCT0_GPI [0:7] pin functions selected from IOCON register (See the Pin descriptions in LPC546xx data sheet).	SCTimer/PWM	Chapter 16 “LPC546xx SCTimer/PWM”

11.5 General description

The inputs to the SCTimer/PWM, DMA triggers, to the eight pin interrupts, and to the frequency measure block are multiplexed to multiple input sources. The sources can be external pins, interrupts, or output signals of other peripherals.

The input multiplexing makes it possible to design event-driven processes without CPU intervention by connecting peripherals like the ADC and the SCTimer/PWM.

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines.

11.5.1 SCT0 input multiplexing

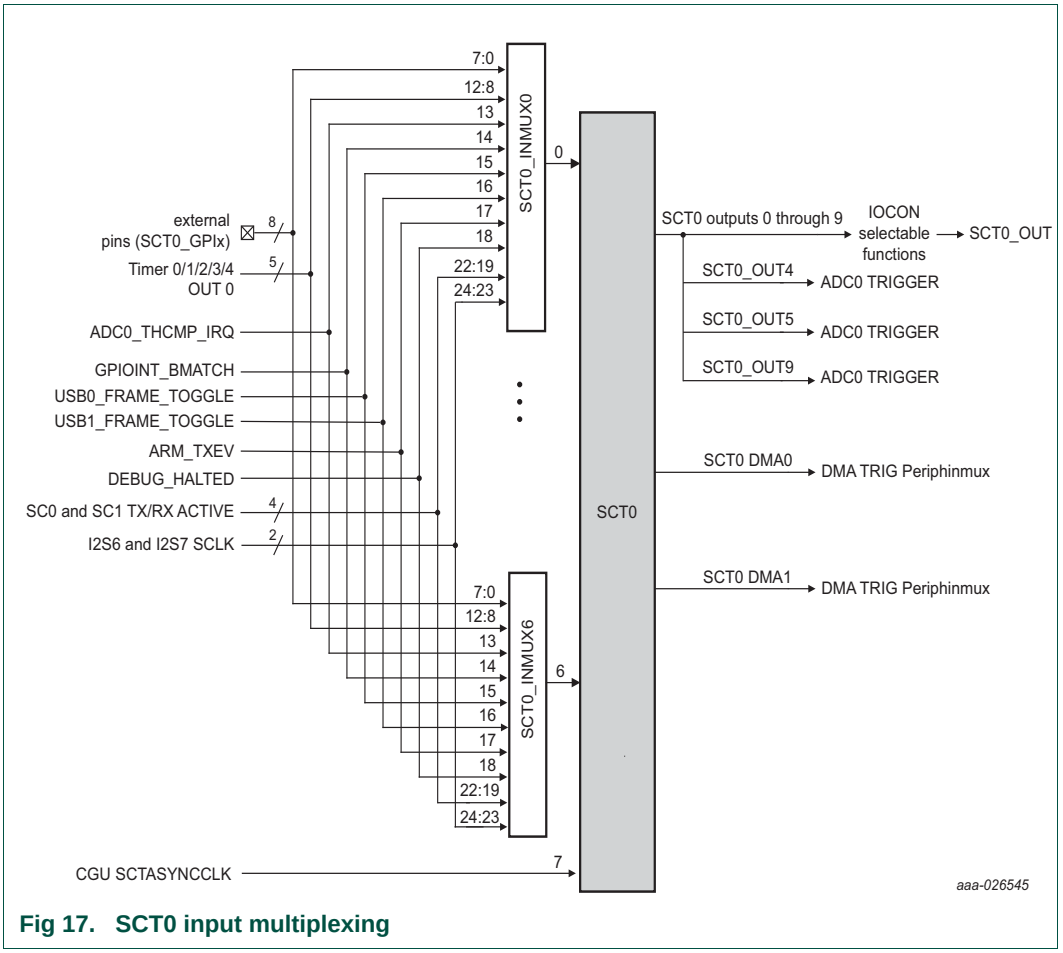


Fig 17. SCT0 input multiplexing

11.5.2 Pin interrupt input multiplexing

The input mux for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

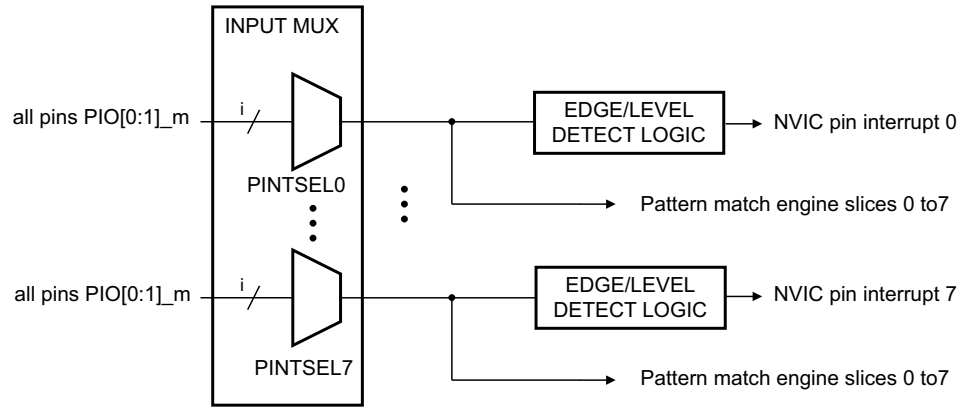


Fig 18. Pin interrupt multiplexing

11.5.3 DMA trigger input multiplexing

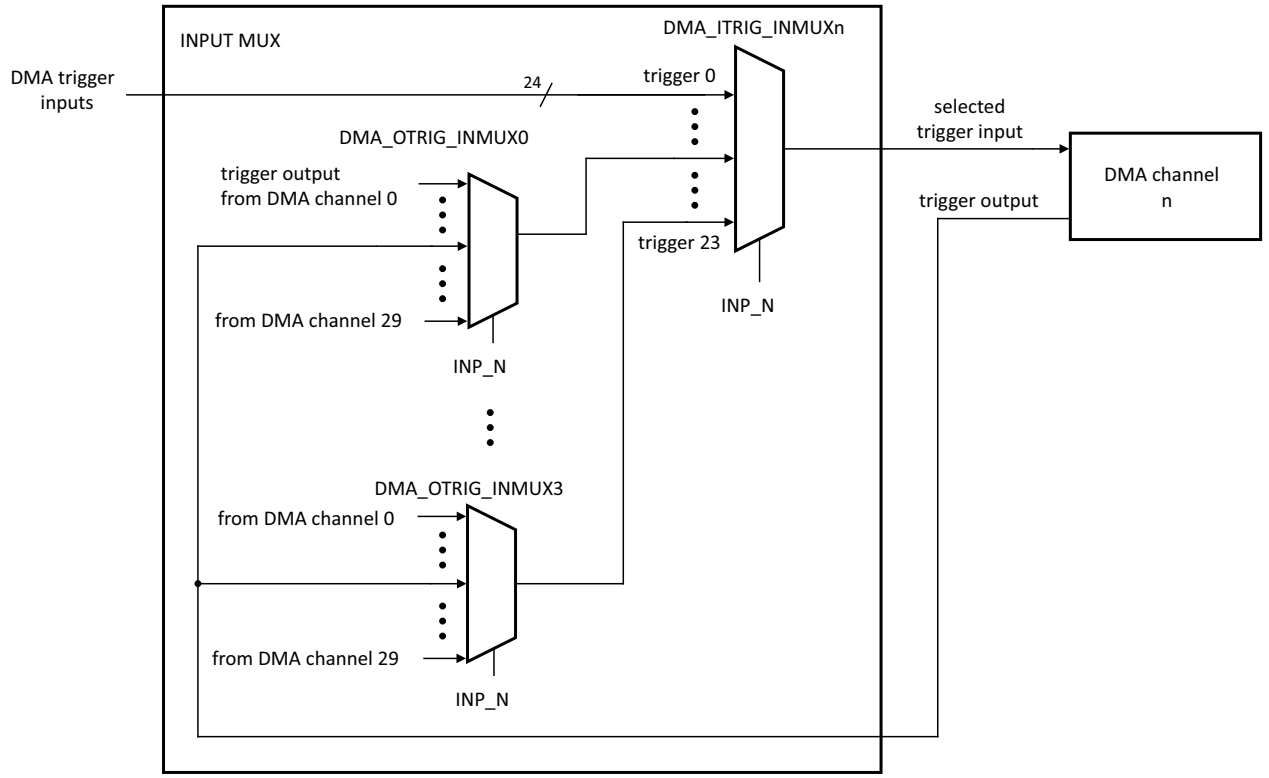


Fig 19. DMA trigger multiplexing

11.6 Register description

All input mux registers reside on word address boundaries. Details of the registers appear in the description of each function.

All address offsets not shown in Table 260 are reserved and should not be written to.

Table 260. Register overview: Input multiplexing (base address 0x4000 5000)

Name	Access	Offset	Description	Reset value	Section
SCT0_INMUX0	R/W	0x000	Input mux register for SCT0 input 0.	0x1F	11.6.1
SCT0_INMUX1	R/W	0x004	Input mux register for SCT0 input 1.	0x1F	11.6.1
SCT0_INMUX2	R/W	0x008	Input mux register for SCT0 input 2.	0x1F	11.6.1
SCT0_INMUX3	R/W	0x00C	Input mux register for SCT0 input 3.	0x1F	11.6.1
SCT0_INMUX4	R/W	0x010	Input mux register for SCT0 input 4.	0x1F	11.6.1
SCT0_INMUX5	R/W	0x014	Input mux register for SCT0 input 5.	0x1F	11.6.1
SCT0_INMUX6	R/W	0x018	Input mux register for SCT0 input 6.	0x1F	11.6.1
-	-	0x01C-0x0BC	Reserved	-	-
PINTSEL0	R/W	0x0C0	Pin interrupt select register 0.	0x0	11.6.2
PINTSEL1	R/W	0x0C4	Pin interrupt select register 1.	0x0	11.6.2
PINTSEL2	R/W	0x0C8	Pin interrupt select register 2.	0x0	11.6.2
PINTSEL3	R/W	0x0CC	Pin interrupt select register 3.	0x0	11.6.2
PINTSEL4	R/W	0x0D0	Pin interrupt select register 4.	0x0	11.6.2
PINTSEL5	R/W	0x0D4	Pin interrupt select register 5.	0x0	11.6.2
PINTSEL6	R/W	0x0D8	Pin interrupt select register 6.	0x0	11.6.2
PINTSEL7	R/W	0x0DC	Pin interrupt select register 7.	0x0	11.6.2
DMA_ITRIG_INMUX0	R/W	0x0E0	Trigger select register for DMA channel 0.	0x1F	11.6.3
DMA_ITRIG_INMUX1	R/W	0x0E4	Trigger select register for DMA channel 1.	0x1F	11.6.3
DMA_ITRIG_INMUX2	R/W	0x0E8	Trigger select register for DMA channel 2.	0x1F	11.6.3
DMA_ITRIG_INMUX3	R/W	0x0EC	Trigger select register for DMA channel 3.	0x1F	11.6.3
DMA_ITRIG_INMUX4	R/W	0x0F0	Trigger select register for DMA channel 4.	0x1F	11.6.3
DMA_ITRIG_INMUX5	R/W	0x0F4	Trigger select register for DMA channel 5.	0x1F	11.6.3
DMA_ITRIG_INMUX6	R/W	0x0F8	Trigger select register for DMA channel 6.	0x1F	11.6.3
DMA_ITRIG_INMUX7	R/W	0x0FC	Trigger select register for DMA channel 7.	0x1F	11.6.3
DMA_ITRIG_INMUX8	R/W	0x100	Trigger select register for DMA channel 8.	0x1F	11.6.3
DMA_ITRIG_INMUX9	R/W	0x104	Trigger select register for DMA channel 9.	0x1F	11.6.3
DMA_ITRIG_INMUX10	R/W	0x108	Trigger select register for DMA channel 10.	0x1F	11.6.3
DMA_ITRIG_INMUX11	R/W	0x10C	Trigger select register for DMA channel 11.	0x1F	11.6.3
DMA_ITRIG_INMUX12	R/W	0x110	Trigger select register for DMA channel 12.	0x1F	11.6.3
DMA_ITRIG_INMUX13	R/W	0x114	Trigger select register for DMA channel 13.	0x1F	11.6.3
DMA_ITRIG_INMUX14	R/W	0x118	Trigger select register for DMA channel 14.	0x1F	11.6.3
DMA_ITRIG_INMUX15	R/W	0x11C	Trigger select register for DMA channel 15.	0x1F	11.6.3
DMA_ITRIG_INMUX16	R/W	0x120	Trigger select register for DMA channel 16.	0x1F	11.6.3
DMA_ITRIG_INMUX17	R/W	0x124	Trigger select register for DMA channel 17.	0x1F	11.6.3

Table 260. Register overview: Input multiplexing (base address 0x4000 5000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DMA_ITRIG_INMUX18	R/W	0x128	Trigger select register for DMA channel 18.	0x1F	11.6.3
DMA_ITRIG_INMUX19	R/W	0x12C	Trigger select register for DMA channel 19.	0x1F	11.6.3
DMA_ITRIG_INMUX20	R/W	0x130	Trigger select register for DMA channel 20.	0x1F	11.6.3
DMA_ITRIG_INMUX21	R/W	0x134	Trigger select register for DMA channel 21.	0x1F	11.6.3
DMA_ITRIG_INMUX22	R/W	0x138	Trigger select register for DMA channel 22.	0x1F	11.6.3
DMA_ITRIG_INMUX23	R/W	0x13C	Trigger select register for DMA channel 23.	0x1F	11.6.3
DMA_ITRIG_INMUX24	R/W	0x140	Trigger select register for DMA channel 24.	0x1F	11.6.3
DMA_ITRIG_INMUX25	R/W	0x144	Trigger select register for DMA channel 25.	0x1F	11.6.3
DMA_ITRIG_INMUX26	R/W	0x148	Trigger select register for DMA channel 26.	0x1F	11.6.3
DMA_ITRIG_INMUX27	R/W	0x14C	Trigger select register for DMA channel 27.	0x1F	11.6.3
DMA_ITRIG_INMUX28	R/W	0x150	Trigger select register for DMA channel 28.	0x1F	11.6.3
DMA_ITRIG_INMUX29	R/W	0x154	Trigger select register for DMA channel 29.	0x1F	11.6.3
DMA_OTRIG_INMUX0	R/W	0x160	DMA output trigger selection to become DMA trigger 18.	0x1F	11.6.4
DMA_OTRIG_INMUX1	R/W	0x164	DMA output trigger selection to become DMA trigger 19.	0x1F	11.6.4
DMA_OTRIG_INMUX2	R/W	0x168	DMA output trigger selection to become DMA trigger 20.	0x1F	11.6.4
DMA_OTRIG_INMUX3	R/W	0x16C	DMA output trigger selection to become DMA trigger 21.	0x1F	11.6.4
FREQMEAS_REF	R/W	0x180	Selection for frequency measurement reference clock.	0x1F	11.6.5
FREQMEAS_TARGET	R/W	0x184	Selection for frequency measurement target clock.	0x1F	11.6.6

11.6.1 SCT0 Input mux registers 0 to 6

With the SCT0 Input mux registers you can select one input source for each SCT0 input from 24 external and internal sources. (An exception is SCT0 input SCT0_IN7, which is directly connected to the SCTASYNCCLK PLL clock and not multiplexed with any other signals.)

The output of SCT0 Input mux register 0 selects the source for SCT0 input 0. The output of SCT0 Input mux register 1 selects the source for SCT0 input 1, and so forth up to SCT0 Input mux register 6, which selects the input for SCT0 input 6.

Table 261. SCT0 Input mux registers 0 to 6 (SCT0_INMUX[0:6], offset [0x000: 0x018] bit description

Bit	Symbol	Value	Description	Reset value
4:0	INP_N		Input number to SCT0 inputs 0 to 6.	0x1F
		0x0	SCT_GPI0 function selected from IOCON register	
		0x1	SCT_GPI1 function selected from IOCON register	
		0x2	SCT_GPI2 function selected from IOCON register	
		0x3	SCT_GPI3 function selected from IOCON register	
		0x4	SCT_GPI4 function selected from IOCON register	
		0x5	SCT_GPI5 function selected from IOCON register	
		0x6	SCT_GPI6 function selected from IOCON register	
		0x7	SCT_GPI7 function selected from IOCON register	
		0x8	T0_OUT0	
		0x9	T1_OUT0	
		0xA	T2_OUT0	
		0xB	T3_OUT0	
		0xC	T4_OUT0	
		0xD	ADC_THCMP_IRQ	
		0xE	GPIOINT_BMATCH	
		0xF	USB0_FRAME_TOGGLE	
		0x10	USB1_FRAME_TOGGLE	
		0x11	ARM_TXEV	
		0x12	DEBUG_HALTED	
		0x13	SMARTCARD0_TX_ACTIVE	
		0x14	SMARTCARD0_RX_ACTIVE	
		0x15	SMARTCARD1_TX_ACTIVE	
		0x16	SMARTCARD1_RX_ACTIVE	
		0x17	I2S6_SCLK	
		0x18	I2S7_SCLK	
31:5	-	-	Reserved.	-

11.6.2 Pin interrupt select registers

Each of these 8 registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins PIO0_0 to PIO0_31 to the INTPIN bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Each of the pin interrupts must be enabled in the NVIC (see Table 88) before it becomes active.

To use the selected pins for pin interrupts or the pattern match engine, see Section 12.5.2 “Pattern match engine”.

Table 262. Pin interrupt select registers (PINTSEL[0:7], offsets [0x0C0:0x0DC]) bit description

Bit	Symbol	Description	Reset value
7:0	INTPIN	Pin number select for pin interrupt or pattern match engine input. (For PIOx_y: INTPIN = (x * 32) + y. PIO0_0 to PIO1_31 correspond to numbers 0 to 63.	0
31:8	-	Reserved	-

11.6.3 DMA trigger input mux registers 0 to 29

With the DMA trigger input mux registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Table 263. DMA trigger Input mux registers (DMA_ITRIG_INMUX[0:29], offsets [0x0E0:0x154]) bit description

Bit	Symbol	Description	Reset value
4:0	INP	Trigger input number (decimal value) for DMA channel n (n = 0 to 29). 0 = ADC0 Sequence A interrupt 1 = ADC0 Sequence B interrupt 2 = SCT0 DMA request 0 3 = SCT0 DMA request 1 4 = Pin interrupt 0 5 = Pin interrupt 1 6 = Pin interrupt 2 7 = Pin interrupt 3 8 = Timer CTIMER0 Match 0 9 = Timer CTIMER0 Match 1 10 = Timer CTIMER1 Match 0 11 = Timer CTIMER1 Match 1 12 = Timer CTIMER2 Match 0 13 = Timer CTIMER2 Match 1 14 = Timer CTIMER3 Match 0 15 = Timer CTIMER3 Match 1 16 = Timer CTIMER4 Match 0 17 = Timer CTIMER4 Match 1 18 = DMA output trigger mux 0 19 = DMA output trigger mux 1 20 = DMA output trigger mux 2 21 = DMA output trigger mux 3	0x1F
31:5	-	Reserved.	-

11.6.4 DMA output trigger feedback mux registers 0 to 3

This register provides a multiplexer for inputs 18 to 21 of each DMA trigger input mux register DMA_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by the each DMA channel. By default, none of the triggers are selected.

Table 264. DMA output trigger feedback mux registers (DMA_OTRIG_INMUX[0:3], offset [0x160:0x16C]) bit description

Bit	Symbol	Description	Reset value
4:0	INP	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 29).	0x1F
31:5	-	Reserved.	-

11.6.5 Frequency measure function reference clock select register

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Also see:

- Section 7.6.7 “Frequency measure function”
- Section 7.2.3 “Measure the frequency of a clock signal”
- Frequency measure control register (FREQMECTRL) - Section 7.5.67
- Frequency target clock select register (FREQMEAS_REF) - Section 11.6.6

Table 265. Frequency measure function frequency clock select register (FREQMEAS_REF, offset 0x180) bit description

Bit	Symbol	Description	Reset value
4:0	CLKIN	Clock source number (decimal value) for frequency measure function target clock: 0 = External crystal oscillator (clk_in) 1 = FRO 12 MHz oscillator (fro_12m) 2 = FRO 96 or 48 MHz (fro_hf) 3 = Watchdog oscillator (wdt_clk) 4 = 32 kHz RTC oscillator (32k_clk) 5 = Main clock (main_clk) 6 = FREQME_GPIO_CLK_A 7 = FREQME_GPIO_CLK_B	0x1F
31:5	-	Reserved.	-

11.6.6 Frequency measure function target clock select register

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected. See Section 7.6.7 “Frequency measure function”, Section 7.2.3 “Measure the frequency of a clock signal”, Section 7.5.67 “Frequency measure function control register”, and Section 11.6.5 above for more on this function.

Table 266. Frequency measure function target clock select register (FREQMEAS_TARGET, offset 0x184) bit description

Bit	Symbol	Description	Reset value
4:0	CLKIN	0 = External crystal oscillator (clk_in) 1 = FRO 12 MHz oscillator (fro_12m) 2 = FRO 96 or 48 MHz (fro_hf) 3 = Watchdog oscillator (wdt_clk) 4 = 32 kHz RTC oscillator (32k_clk) 5 = Main clock (see Section 7.5.29) 6 = FREQME_GPIO_CLK_A 7 = FREQME_GPIO_CLK_B	0x1F
31:5	-	Reserved.	-

12.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC546xx devices.

12.2 Features

- Pin interrupts
  - Up to eight pins can be selected from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine
  - Up to 8 pins can be selected from all digital pins on ports 0 and 1 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
  - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
  - Any occurrence of a pattern match can be programmed to also generate an RXEV notification to the CPU.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

12.3 Basic configuration

- Pin interrupts:
  - Select up to eight external interrupt pins from all digital port pins on ports 0 and 1 in the Input Mux block (Table 262). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
  - Enable the clock to the pin interrupt register block in the AHBCLKCTRL0 register (Table 139).
  - In order to use the pin interrupts to wake up the part from deep-sleep mode, enable the pin interrupt wake-up feature in the STARTER0 register for pin interrupt 0 through 3 and the STARTER1 register for pin interrupt 4 through 7 (Table 222 and Table 223 respectively).
- Pattern match engine:
  - Select up to eight external pins from all digital port pins on ports 0 and 1 in the Input mux block (Table 262). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
  - Enable the clock to the pin interrupt register block in the AHBCLKCTRL0 register (Table 139).
  - Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupts #5 through #8 for pin interrupts 0 to 3, and 33 through 36 for pin interrupts 4 through 7).

12.3.1 Configure pins as pin interrupts or as inputs to the pattern match engine

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the LPC546xx package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number from ports 0 and 1 into one of the eight PINTSEL registers in the Input mux block.

   Remark: The port pin number serves to identify the pin to the PINTSEL register. Any function, including GPIO, can be assigned to this pin via IOCON.
3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can set up.

See Section 11.6.2 “Pin interrupt select registers” in the Input mux block for the PINTSEL registers.

Remark: The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

12.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the Input mux. See Section 11.6.2 “Pin interrupt select registers”.

12.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to eight pins can be configured using the PINTSEL registers in the Input mux block for these features.

12.5.1 Pin interrupts

From all available GPIO pins, up to eight pins can be selected in the system control block to serve as external interrupt pins (see Table 262). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

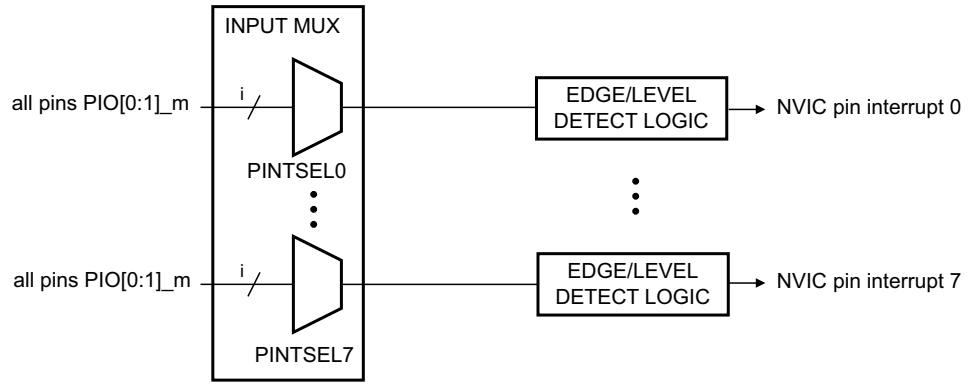


Fig 20. Pin interrupt connections

12.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic that monitors the selected input continuously and creates a HIGH output if the input qualifies as detected, that is as true. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

The detect logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.

- Event (non-sticky): Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the detect logic can detect another edge,
- Level: A HIGH or LOW level on the selected input.

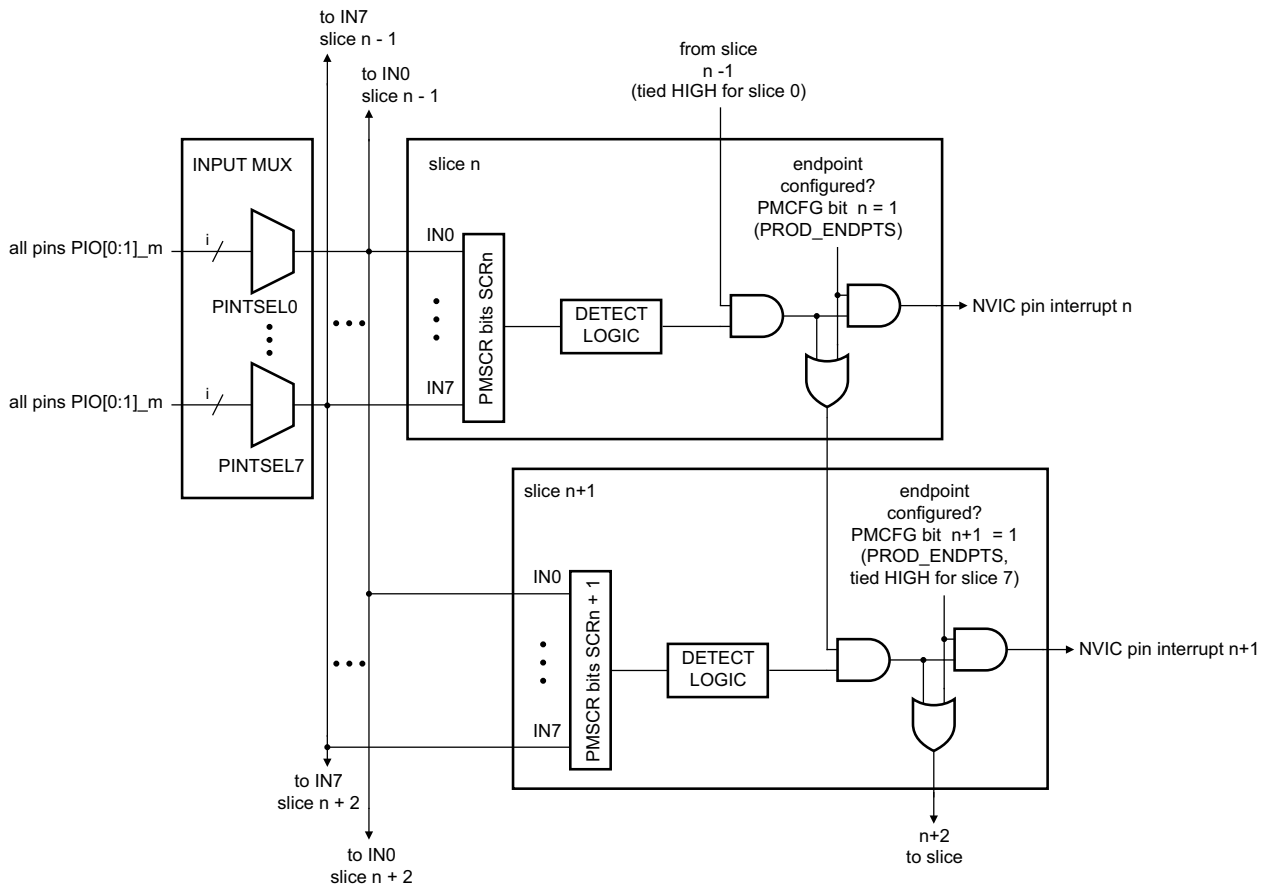
Figure 22 shows the details of the edge detection logic for each slice.

Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 12.7.3 for details.

The connections between the pins and the pattern match engine are shown in Figure 21. All pins that are inputs to the pattern match engine are selected in the Syscon block and can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during deep-sleep mode.



See Figure 22 for the detect logic block.

Fig 21. Pattern match engine connections

The pattern match logic continuously monitors the eight inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for each individual minterm.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the ARM core when the entire boolean expression is true (i.e. when any minterm is matched).

The pattern match function utilizes the same eight interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from reduced power modes. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of eight bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched.
(See bit slice drawing Figure 22).

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

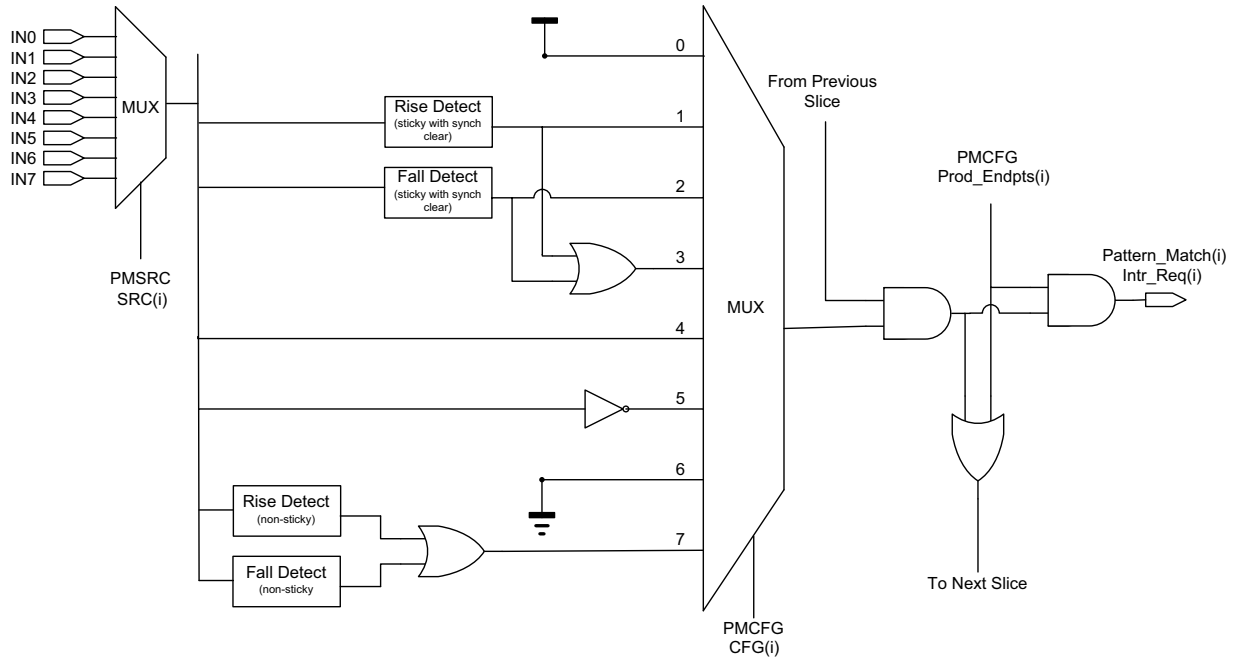


Fig 22. Pattern match bit slice with detect logic

12.5.2.1 Example

Assume the expression: (IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) is specified through the registers PMSRC (Table 279) and PMCFG (Table 280). Each term in the boolean expression, (IN0), ~(IN1), (IN3)^, etc., represents one bit slice of the pattern match engine.

- In the first minterm (IN0)~(IN1)(IN3)^, bit slice 0 monitors for a high-level on input (IN0), bit slice 1 monitors for a low level on input (IN1) and bit slice 2 monitors for a rising-edge on input (IN3). If this combination is detected, that is if all three terms are true, the interrupt associated with bit slice 2 will be asserted.
- In the second minterm (IN1)(IN2), bit slice 3 monitors input (IN1) for a high level, bit slice 4 monitors input (IN2) for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.
- In the third minterm (IN0)~(IN3)~(IN4), bit slice 5 monitors input (IN0) for a high level, bit slice 6 monitors input (IN3) for a low level, and bit slice 7 monitors input (IN4) for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.
- The ORed result of all three minterms asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

Related links:

Section 12.7.2

12.6 Register description

Table 267. Register overview: Pin interrupts/pattern match engine (base address: 0x4000 4000)

Name	Access	Offset	Description	Reset value	Section
ISEL	R/W	0x000	Pin interrupt mode.	0	12.6.1
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable.	0	12.6.2
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt enable set.	NA	12.6.3
CIENR	WO	0x00C	Pin interrupt level or rising edge interrupt enable clear.	NA	12.6.4
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable.	0	12.6.5
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set.	NA	12.6.6
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear.	NA	12.6.7
RISE	R/W	0x01C	Pin interrupt rising edge.	0	12.6.8
FALL	R/W	0x020	Pin interrupt falling edge.	0	12.6.9
IST	R/W	0x024	Pin interrupt status.	0	12.6.10
PMCTRL	R/W	0x028	Pattern match interrupt control.	0	12.6.11
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source.	0	12.6.12
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration.	0	12.6.13

12.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 268. Pin interrupt mode register (ISEL, offset 0x000) bit description

Bit	Symbol	Description	Reset value	Access
7:0	PMODE	Selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Edge sensitive 1 = Level sensitive	0	R/W
31:8	-	Reserved. Read value is undefined, only zero should be written.	-	-

12.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the IENR register enables the interrupt depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the rising edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the level interrupt is enabled. The IENF register configures the active level (HIGH or LOW) for this interrupt.

Table 269. Pin interrupt level or rising edge interrupt enable register (IENR, offset 0x004) bit description

Bit	Symbol	Description	Reset value	Access
7:0	ENRL	Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable rising edge or level interrupt. 1 = Enable rising edge or level interrupt.	0	R/W
31:8	-	Reserved. Read value is undefined, only zero should be written.	-	-

12.6.3 Pin interrupt level or rising edge interrupt enable set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the SIENR register sets the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register.

Table 270. Pin interrupt level or rising edge interrupt enable set register (SIENR, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access
7:0	SETENRL	Ones written to this address set bits in the IENR, thus enabling interrupts. Bit n sets bit n in the IENR register. 0 = No operation. 1 = Enable rising edge or level interrupt.	NA	WO
31:8	-	Reserved.	-	-

12.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the CIENR register clears the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register.

Table 271. Pin interrupt level or rising edge interrupt clear register (CIENR, offset 0x00C) bit description

Bit	Symbol	Description	Reset value	Access
7:0	CENRL	Ones written to this address clear bits in the IENR, thus disabling the interrupts. Bit n clears bit n in the IENR register. 0 = No operation. 1 = Disable rising edge or level interrupt.	NA	WO
31:8	-	Reserved.	-	-

12.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Table 272. Pin interrupt active level or falling edge interrupt enable register (IENF, offset 0x010) bit description

Bit	Symbol	Description	Reset value	Access
7:0	ENAF	Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable falling edge interrupt or set active interrupt level LOW. 1 = Enable falling edge interrupt enabled or set active interrupt level HIGH.	0	R/W
31:8	-	Reserved.	-	-

12.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the SIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is set.
- If the pin interrupt mode is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Table 273. Pin interrupt active level or falling edge interrupt set register (SIENF, offset 0x014) bit description

Bit	Symbol	Description	Reset value	Access
7:0	SETENAF	Ones written to this address set bits in the IENF, thus enabling interrupts. Bit n sets bit n in the IENF register. 0 = No operation. 1 = Select HIGH-active interrupt or enable falling edge interrupt.	NA	WO
31:8	-	Reserved.	-	-

12.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 262), one bit in the CIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is cleared.
- If the pin interrupt mode is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Table 274. Pin interrupt active level or falling edge interrupt clear register (CIENF, offset 0x018) bit description

Bit	Symbol	Description	Reset value	Access
7:0	CENAF	Ones written to this address clears bits in the IENF, thus disabling interrupts. Bit n clears bit n in the IENF register. 0 = No operation. 1 = LOW-active interrupt selected or falling edge interrupt disabled.	NA	WO
31:8	-	Reserved.	-	-

12.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Table 262) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 275. Pin interrupt rising edge register (RISE, offset 0x01C) bit description

Bit	Symbol	Description	Reset value	Access
7:0	RDET	Rising edge detect. Bit n detects the rising edge of the pin selected in PINTSELn. Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear rising edge detection for this pin.	0	R/W
31:8	-	Reserved.	-	-

12.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Table 262) on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 276. Pin interrupt falling edge register (FALL, offset 0x020) bit description

Bit	Symbol	Description	Reset value	Access
7:0	FDET	Falling edge detect. Bit n detects the falling edge of the pin selected in PINTSELn. Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear falling edge detection for this pin.	0	R/W
31:8	-	Reserved.	-	-

12.6.10 Pin interrupt status register

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge-sensitive in the Interrupt Select register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the Active level register, thus switching the active level on the pin.

Table 277. Pin interrupt status register (IST, offset 0x024) bit description

Bit	Symbol	Description	Reset value	Access
7:0	PSTAT	Pin interrupt status. Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINTSELn. Read 0: interrupt is not being requested for this interrupt pin. Write 0: no operation. Read 1: interrupt is being requested for this interrupt pin. Write 1 (edge-sensitive): clear rising- and falling-edge detection for this pin. Write 1 (level-sensitive): switch the active level for this pin (in the IENF register).	0	R/W
31:8	-	Reserved.	-	-

12.6.11 Pattern Match Interrupt Control Register

The pattern match control register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read.

If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

Remark: Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

Remark: note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Table 278. Pattern match interrupt control register (PMCTRL, offset 0x028) bit description

Bit	Symbol	Value	Description	Reset value
0	SEL_PMATCH		Specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function.	0
		0	Pin interrupt. Interrupts are driven in response to the standard pin interrupt function.	
		1	Pattern match. Interrupts are driven in response to pattern matches.	
1	ENA_RXEV		Enables the RXEV output to the CPU when the specified boolean expression evaluates to true.	0
		0	Disabled. RXEV output to the CPU is disabled.	
		1	Enabled. RXEV output to the CPU is enabled.	
23:2	-	-	Reserved. Do not write 1s to unused bits.	0
31:24	PMAT	-	This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs.	0x0

12.6.12 Pattern Match Interrupt Bit-Slice Source register

The bit-slice source register specifies the input source for each of the eight pattern match bit slices.

Each of the possible eight inputs is selected in the pin interrupt select registers in the SYSCON block. See Section 11.6.2. Input 0 corresponds to the pin selected in the PINTSEL0 register, input 1 corresponds to the pin selected in the PINTSEL1 register, and so forth.

Remark: Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Table 279. Pattern match bit-slice source register (PMSRC, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
7:0	Reserved	-	Software should not write 1s to unused bits.	0
10:8	SRC0		Selects the input source for bit slice 0	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	

Table 279. Pattern match bit-slice source register (PMSRC, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
13:11	SRC1		Selects the input source for bit slice 1	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 1.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 1.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 1.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 1.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 1.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 1.	
16:14	SRC2		Selects the input source for bit slice 2	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 2.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 2.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 2.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 2.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 2.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 2.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 2.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 2.	
19:17	SRC3		Selects the input source for bit slice 3	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 3.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 3.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 3.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 3.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 3.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 3.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 3.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 3.	
22:20	SRC4		Selects the input source for bit slice 4	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 4.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 4.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 4.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 4.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 4.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 4.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 4.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 4.	

Table 279. Pattern match bit-slice source register (PMSRC, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
25:23	SRC5		Selects the input source for bit slice 5	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 5.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 5.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 5.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 5.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 5.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 5.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 5.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 5.	
28:26	SRC6		Selects the input source for bit slice 6	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 6.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 6.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 6.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 6.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 6.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 6.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 6.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 6.	
31:29	SRC7		Selects the input source for bit slice 7	0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 7.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 7.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 7.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 7.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 7.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 7.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 7.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 7.	

12.6.13 Pattern Match Interrupt Bit Slice Configuration register

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (i.e. where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.

- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge,

Remark: To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENPTSn bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression (i.e. an OR will be inserted in the boolean expression following the element controlled by this bit slice).

Table 280. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description

Bit	Symbol	Value	Description	Reset value
0	PROD_ENDPTS0		Determines whether slice 0 is an endpoint.	0
		0	No effect. Slice 0 is not an endpoint.	
		1	endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true.	
1	PROD_ENDPTS1		Determines whether slice 1 is an endpoint.	0
		0	No effect. Slice 1 is not an endpoint.	
		1	endpoint. Slice 1 is the endpoint of a product term (minterm). Pin interrupt 1 in the NVIC is raised if the minterm evaluates as true.	
2	PROD_ENDPTS2		Determines whether slice 2 is an endpoint.	0
		0	No effect. Slice 2 is not an endpoint.	
		1	endpoint. Slice 2 is the endpoint of a product term (minterm). Pin interrupt 2 in the NVIC is raised if the minterm evaluates as true.	
3	PROD_ENDPTS3		Determines whether slice 3 is an endpoint.	0
		0	No effect. Slice 3 is not an endpoint.	
		1	endpoint. Slice 3 is the endpoint of a product term (minterm). Pin interrupt 3 in the NVIC is raised if the minterm evaluates as true.	
4	PROD_ENDPTS4		Determines whether slice 4 is an endpoint.	0
		0	No effect. Slice 4 is not an endpoint.	
		1	endpoint. Slice 4 is the endpoint of a product term (minterm). Pin interrupt 4 in the NVIC is raised if the minterm evaluates as true.	
5	PROD_ENDPTS5		Determines whether slice 5 is an endpoint.	0
		0	No effect. Slice 5 is not an endpoint.	
		1	endpoint. Slice 5 is the endpoint of a product term (minterm). Pin interrupt 5 in the NVIC is raised if the minterm evaluates as true.	

Table 280. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
6	PROD_ENDPTS6		Determines whether slice 6 is an endpoint.	0
		0	No effect. Slice 6 is not an endpoint.	
		1	endpoint. Slice 6 is the endpoint of a product term (minterm). Pin interrupt 6 in the NVIC is raised if the minterm evaluates as true.	
7	-	-	Reserved. Bit slice 7 is automatically considered a product end point.	0
10:8	CFG0		Specifies the match contribution condition for bit slice 0.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
13:11	CFG1		Specifies the match contribution condition for bit slice 1.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 280. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
16:14	CFG2		Specifies the match contribution condition for bit slice 2.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
19:17	CFG3		Specifies the match contribution condition for bit slice 3.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 280. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
22:20	CFG4		Specifies the match contribution condition for bit slice 4.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
25:23	CFG5		Specifies the match contribution condition for bit slice 5.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 280. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
28:26	CFG6		Specifies the match contribution condition for bit slice 6.	0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
31:29	CFG7		Specifies the match contribution condition for bit slice 7.	-
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

12.7 Functional description

12.7.1 Pin interrupts

In this interrupt facility, up to 8 pins are identified as interrupt sources by the Pin Interrupt Select registers (PINTSEL0-7). All registers in the pin interrupt block contain 8 bits, corresponding to the pins called out by the PINTSEL0-7 registers. The ISEL register defines whether each interrupt pin is edge- or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin, and can be written to clear (and set) edge detection. The IST register indicates whether each interrupt pin is currently requesting an interrupt, and this register can also be written to clear interrupts.

The other pin interrupt registers play different roles for edge-sensitive and level-sensitive pins, as described in Table 281.

Table 281. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

12.7.2 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:
(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register (Table 279):
  - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2
  - SRC3: 010 - select input 2 for bit slice 3
  - SRC4: 011 - select input 3 for bit slice 4
  - SRC5: 110 - select input 6 for bit slice 5
  - SRC6: 101 - select input 5 for bit slice 6

- SRC7: 111 - select input 7 for bit slice 7

- PMCFG register (Table 280):
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS02 = 1
  - PROD_ENDPTS5 = 1
  - All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
  - = 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
  - CFG0: 000 - high level on the selected input (input 1) for bit slice 0
  - CFG1: 000 - high level on the selected input (input 1) for bit slice 1
  - CFG2: 000 - high level on the selected input (input 2) for bit slice 2
  - CFG3: 101 - low level on the selected input (input 2) for bit slice 3
  - CFG4: 101 - low level on the selected input (input 3) for bit slice 4
  - CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
  - CFG6: 000 - high level on the selected input (input 5) for bit slice 6
  - CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register (Table 278):
  - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.

    For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

    Pin interrupt 2 will be asserted in response to a match on the second product term.

    Pin interrupt 5 will be asserted when there is a match on the third product term.

    Pin interrupt 7 will be asserted on a match on the last term.
  - Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
  - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits will always be low.

12.7.3 Pattern match engine edge detect examples

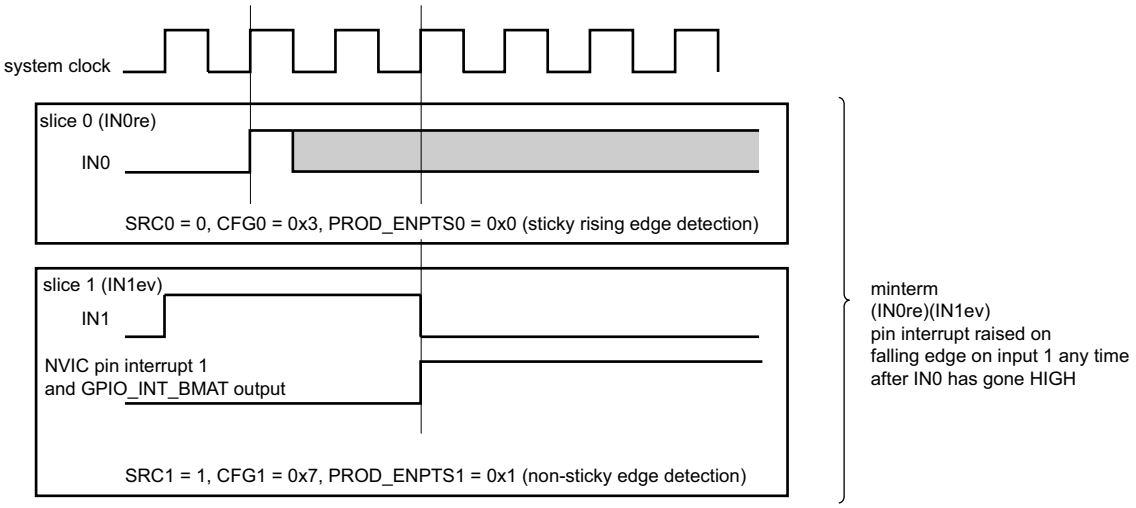


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 23. Pattern match engine examples: sticky edge detect

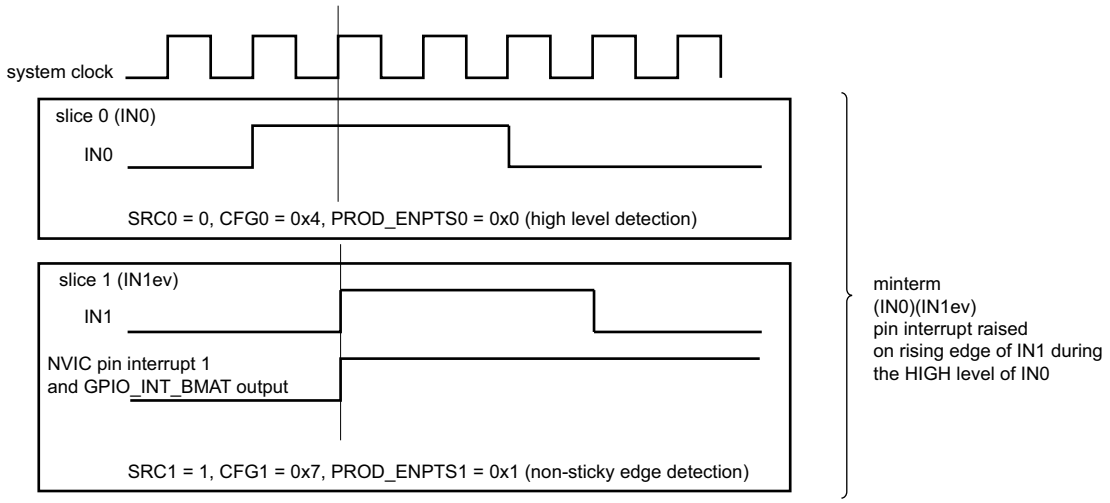


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 24. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

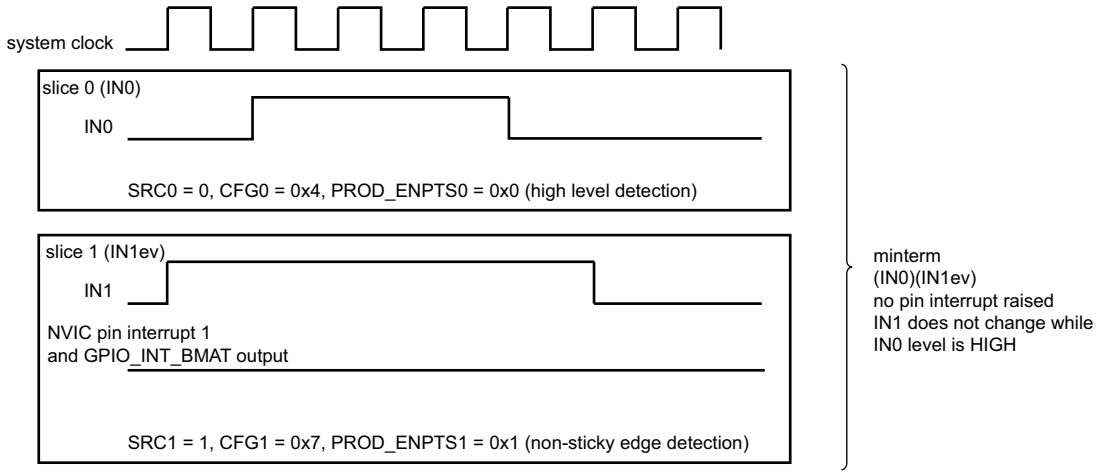


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 25. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

13.1 How to read this chapter

GPIO registers support up to 32 pins on each port. Depending on the device and package type, a subset of those pins may be available, and the unused bits in GPIO registers are reserved (see Table 283).

Table 282. GPIO pins available

Package	Total GPIOs	GPIO Port 0	GPIO Port 1	GPIO Port 2	GPIO Port 3	GPIO Port 4	GPIO Port 5
208-pin device	171	PIO0_0 to PIO0_31	PIO1_0 to PIO1_31	PIO2_0 to PIO2_31	PIO3_0 to PIO3_31	PIO4_0 to PIO4_31	PIO5_0 to PIO5_10
180-pin device	145	PIO0_0 to PIO0_31	PIO1_0 to PIO1_31	PIO2_0 to PIO2_31	PIO3_0 to PIO3_31	PIO4_0 to PIO4_16	-
100-pin device	64	PIO0_0 to PIO0_31	PIO1_0 to PIO1_31	-	-	-	-

13.2 Features

- GPIO pins can be configured as input or output by software.
- All GPIO pins default to inputs with interrupt disabled at reset.
- Pin registers allow pins to be sensed and set individually.
- Direction (input/output) can be set and cleared individually.

13.3 Basic configuration

For the GPIO port registers, enable the clock to each GPIO port in the AHBCLKCTRL0 register (Table 139).

13.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks, see Chapter 12 and Chapter 14.

The GPIO port registers configure each GPIO pin as input or output and read the state of each pin if the pin is configured as input or set the state of each pin if the pin is configured as output.

13.5 Register description

Note: In all GPIO registers, bits that are not shown are reserved.

GPIO port addresses can be read and written as bytes, halfwords, or words.

Remark: A reset value noted as “ext” in this table and subsequent tables indicates that the data read after reset depends on the state of the pin, which in turn may depend on an external source.

Table 283. Register overview: GPIO port (base address 0x4008 C000)

Name	Access	Offset	Description	Reset value	Section
B[0:182]	R/W	[0x0000:0x00B6]	Byte pin registers for ports 0, 1, 2, 3, 4, and 5 GPIO pins.	ext	13.5.1
W[0:182]	R/W	[0x1000:0x12D8]	Word pin registers for ports 0, 1, 2, 3, 4, and 5 GPIO pins.	ext	13.5.2
DIR0	R/W	0x2000	Direction registers port 0.	0	13.5.3
DIR1	R/W	0x2004	Direction registers port 1.	0	13.5.3
DIR2	R/W	0x2008	Direction registers port 2.	0	13.5.3
DIR3	R/W	0x200C	Direction registers port 3.	0	13.5.3
DIR4	R/W	0x2010	Direction registers port 4.	0	13.5.3
DIR5	R/W	0x2014	Direction registers port 5.	0	13.5.3
MASK0	R/W	0x2080	Mask register port 0.	0	13.5.4
MASK1	R/W	0x2084	Mask register port 1.	0	13.5.4
MASK2	R/W	0x2088	Mask register port 2.	0	13.5.4
MASK3	R/W	0x208C	Mask register port 3.	0	13.5.4
MASK4	R/W	0x2090	Mask register port 4.	0	13.5.4
MASK5	R/W	0x2094	Mask register port 5.	0	13.5.4
PIN0	R/W	0x2100	Port pin register port 0.	ext	13.5.5
PIN1	R/W	0x2104	Port pin register port 1.	ext	13.5.5
PIN2	R/W	0x2108	Port pin register port 2.	ext	13.5.5
PIN3	R/W	0x210C	Port pin register port 3.	ext	13.5.5
PIN4	R/W	0x2110	Port pin register port 4.	ext	13.5.5
PIN5	R/W	0x2114	Port pin register port 5.	ext	13.5.5
MPIN0	R/W	0x2180	Masked port register port 0.	ext	13.5.6
MPIN1	R/W	0x2184	Masked port register port 1.	ext	13.5.6
MPIN2	R/W	0x2188	Masked port register port 2.	ext	13.5.6
MPIN3	R/W	0x218C	Masked port register port 3.	ext	13.5.6
MPIN4	R/W	0x2190	Masked port register port 4.	ext	13.5.6
MPIN5	R/W	0x2194	Masked port register port 5.	ext	13.5.6
SET0	R/W	0x2200	Write: Set register for port 0. Read: output bits for port 0.	0	13.5.7
SET1	R/W	0x2204	Write: Set register for port 1. Read: output bits for port 1.	0	13.5.7
SET2	R/W	0x2208	Write: Set register for port 2. Read: output bits for port 2.	0	13.5.7
SET3	R/W	0x220C	Write: Set register for port 3. Read: output bits for port 3.	0	13.5.7

Table 283. Register overview: GPIO port (base address 0x4008 C000)

Name	Access	Offset	Description	Reset value	Section
SET4	R/W	0x2210	Write: Set register for port 4. Read: output bits for port 4.	0	13.5.7
SET5	R/W	0x2214	Write: Set register for port 5. Read: output bits for port 5.	0	13.5.7
CLR0	WO	0x2280	Clear port 0.	NA	13.5.8
CLR1	WO	0x2284	Clear port 1.	NA	13.5.8
CLR2	WO	0x2288	Clear port 2.	NA	13.5.8
CLR3	WO	0x228C	Clear port 3.	NA	13.5.8
CLR4	WO	0x2290	Clear port 4.	NA	13.5.8
CLR5	WO	0x2294	Clear port 5.	NA	13.5.8
NOT0	WO	0x2300	Toggle port 0.	NA	13.5.9
NOT1	WO	0x2304	Toggle port 1.	NA	13.5.9
NOT2	WO	0x2308	Toggle port 2.	NA	13.5.9
NOT3	WO	0x230C	Toggle port 3.	NA	13.5.9
NOT4	WO	0x2310	Toggle port 4.	NA	13.5.9
NOT5	WO	0x2314	Toggle port 5.	NA	13.5.9
DIRSET0	WO	0x2380	Set pin direction bits for port 0.	0	13.5.10
DIRSET1	WO	0x2384	Set pin direction bits for port 1.	0	13.5.10
DIRSET2	WO	0x2388	Set pin direction bits for port 2.	0	13.5.10
DIRSET3	WO	0x238C	Set pin direction bits for port 3.	0	13.5.10
DIRSET4	WO	0x2390	Set pin direction bits for port 4.	0	13.5.10
DIRSET5	WO	0x2394	Set pin direction bits for port 5.	0	13.5.10
DIRCLR0	WO	0x2400	Clear pin direction bits for port 0.	-	13.5.11
DIRCLR1	WO	0x2404	Clear pin direction bits for port 1.	-	13.5.11
DIRCLR2	WO	0x2408	Clear pin direction bits for port 2.	-	13.5.11
DIRCLR3	WO	0x240C	Clear pin direction bits for port 3.	-	13.5.11
DIRCLR4	WO	0x2410	Clear pin direction bits for port 4.	-	13.5.11
DIRCLR5	WO	0x2414	Clear pin direction bits for port 5.	-	13.5.11
DIRNOT0	WO	0x2480	Toggle pin direction bits for port 0.	-	13.5.12
DIRNOT1	WO	0x2484	Toggle pin direction bits for port 1.	-	13.5.12
DIRNOT2	WO	0x2488	Toggle pin direction bits for port 2.	-	13.5.12
DIRNOT3	WO	0x248C	Toggle pin direction bits for port 3.	-	13.5.12
DIRNOT4	WO	0x2490	Toggle pin direction bits for port 4.	-	13.5.12
DIRNOT5	WO	0x2494	Toggle pin direction bits for port 5.	-	13.5.12

13.5.1 GPIO port byte pin registers

Each GPIO pin has a byte register in this address range. Software typically reads and writes bytes to access individual pins, but can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

Table 284. GPIO port byte pin registers (B[0:182], offset [0x0000:0x00B6]) bit description

Bit	Symbol	Description	Reset value	Access
0	PBYTE	Read: state of the pin PIOm_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. One register for each port pin. Supported pins depends on the specific device and package. Write: loads the pin's output bit. **Remark:** One register for each port pin. Supported pins depends on the specific device and package.	ext	R/W
7:1	-	Reserved (0 on read, ignored on write)	0	-

13.5.2 GPIO port word pin registers

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as zeros. Any write will clear the pin's output bit if the value written is all zeros, else it will set the pin's output bit.

Table 285. GPIO port word pin registers (W[0:182], offsets [0x1000:0x12D8]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PWORD	Read 0: pin PIOm_n is LOW. Write 0: clear output bit. Read 0xFFFF FFFF: pin PIOm_n is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit. **Remark:** Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. One register for each port pin. Supported pins depends on the specific device and package.	ext	R/W

13.5.3 GPIO port direction registers

Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Table 286. GPIO direction port register (DIR[0:5], offset [0x2000:0x2014]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIRP	Selects pin direction for pin PIOm_n (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = input. 1 = output.	0	R/W

13.5.4 GPIO port mask registers

These registers affect writing and reading the MPORT registers. Zeroes in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Table 287. GPIO mask port register (MASK[0:5], offset [0x2080:0x2094]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MASKP	Controls which bits corresponding to PIOm_n are active in the MPORT register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected.	0	R/W

13.5.5 GPIO port pin registers

Reading these registers returns the current state of the pins read, regardless of direction, masking, or alternate functions, except that pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

Table 288. GPIO port pin register (PIN[0:5], offset [0x2100:0x2114]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PORT	Reads pin states or loads output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	R/W

13.5.6 GPIO masked port pin registers

These registers are similar to the PORT registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register

Table 289. GPIO masked port pin register (MPIN[0:5], offset [0x2180:0x2194]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MPORTP	Masked port register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read: pin is LOW and/or the corresponding bit in the MASK register is 1; write: clear output bit if the corresponding bit in the MASK register is 0. 1 = Read: pin is HIGH and the corresponding bit in the MASK register is 0; write: set output bit if the corresponding bit in the MASK register is 0.	ext	R/W

13.5.7 GPIO port set registers

Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Table 290. GPIO set port register (SET[0:5], offset [0x2200:0x2214]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	SETP	Read or set output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0	R/W

13.5.8 GPIO port clear registers

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Table 291. GPIO clear port register (CLR[0:5], offset [0x2280:0x2294]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	CLRP	Clear output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = No operation. 1 = Clear output bit.	NA	WO

13.5.9 GPIO port toggle registers

Output bits can be toggled/inverted/complemented by writing ones to these write-only registers, regardless of MASK registers.

Table 292. GPIO toggle port register (NOT[0:5], offset [0x2300:0x2314]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	NOTP	Toggle output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = no operation. 1 = Toggle output bit.	NA	WO

13.5.10 GPIO port direction set registers

Direction bits can be set by writing ones to these registers.

Table 293. GPIO port direction set register (DIRSET[0:5], offset 0x2380:0x2394) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRSETP	Set direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = No operation. 1 = Set direction bit.	0	WO
31:29	-	Reserved.	0	-

13.5.11 GPIO port direction clear registers

Direction bits can be cleared by writing ones to these write-only registers.

Table 294. GPIO port direction clear register (DIRCLR[0:5], offset 0x2400:0x2414) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRCLRP	Clear direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = No operation. 1 = Clear direction bit.	NA	WO
31:29	-	Reserved.	0	-

13.5.12 GPIO port direction toggle registers

Direction bits can be set by writing ones to these write-only registers.

Table 295. GPIO port direction toggle register (DIRNOT[0:5], offset 0x2480:0x2494) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRNOTP	Toggle direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = no operation. 1 = Toggle direction bit.	NA	WO
31:29	-	Reserved.	0	-

13.6 Functional description

13.6.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the “I/O Configuration” logic. A pin does not have to be selected for GPIO in “I/O Configuration” in order to read its state. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins in a port can be read as a byte, halfword, or word from a PORT register.
- The state of a selected subset of the pins in a port can be read from a Masked Port (MPORT) register. Pins having a 1 in the port’s Mask register will read as 0 from its MPORT register.

13.6.2 GPIO output

Each GPIO pin has an output bit in the GPIO block. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin’s output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOCON (this is the default), and
2. the pin must be selected for output by a 1 in its port’s DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to a Byte Pin register loads the output bit from the least significant bit.
- Writing to a Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to a port’s PORT register loads the output bits of all the pins written to.
- Writing to a port’s MPORT register loads the output bits of pins identified by zeros in corresponding positions of the port’s MASK register.
- Writing ones to a port’s SET register sets output bits.
- Writing ones to a port’s CLR register clears output bits.
- Writing ones to a port’s NOT register toggles/complements/inverts output bits.

The state of a port’s output bits can be read from its SET register. Reading any of the registers described in 13.6.1 returns the state of pins, regardless of their direction or alternate functions.

13.6.3 Masked I/O

A port's MASK register defines which of its pins should be accessible in its MPORT register. Zeroes in MASK enable the corresponding pins to be read from and written to MPORT. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPORT. When a port's MASK register contains all zeros, its PORT and MPORT registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPORT or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers, and set/capture the semaphore controlling exclusive use of the MASK registers before setting the MASK registers, and release the semaphore after the last operation that uses the MPORT or MASK registers.

13.6.4 GPIO direction

Each pin in a GPIO port can be configured as input or output using the DIR registers. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers.

13.6.5 Recommended practices

The following lists some recommended uses for using the GPIO port registers:

- For initial setup after Reset or re-initialization, write the PORT registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte Pin or Word Pin register.
- To make a decision based on multiple pins, read and mask a PORT register.

14.1 Features

- The inputs from any number of digital pins can be enabled to contribute to a combined group interrupt.
- The polarity of each input enabled for the group interrupt can be configured HIGH or LOW.
- Enabled interrupts can be logically combined through an OR or AND operation.
- Two group interrupts are supported to reflect two distinct interrupt patterns.
- The grouped interrupts can wake up the part from sleep or deep-sleep modes.

14.2 Basic configuration

For the group interrupt feature, enable the clock to both the GROUP0 and GROUP1 register interfaces in the AHBCLKCTRL0 register ((Table 139). The group interrupt wake-up feature is enabled in the STARTER0 register for GINT0 and GINT1 (Table 222 and Table 223 respectively). The interrupt must also be enabled in the NVIC (see Table 88).

The pins can be configured as GPIO pins through IOCON, but they don't have to be. The GINT block reads the input from the pin bypassing IOCON multiplexing. Make sure that no analog function is selected on pins that are input to the group interrupts. Selecting an analog function in IOCON disables the digital pad and the digital signal is tied to 0.

14.3 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

For each port/pin connected to one of the two the GPIO Grouped Interrupt blocks (GROUP0 and GROUP1), the GPIO grouped interrupt registers determine which pins are enabled to generate interrupts and what the active polarities of each of those inputs are.

The GPIO grouped interrupt registers also select whether the interrupt output will be level or edge triggered and whether it will be based on the OR or the AND of all of the enabled inputs.

When the designated pattern is detected on the selected input pins, the GPIO grouped interrupt block generates an interrupt. If the part is in a power-savings mode, it l first asynchronously wakes the part up prior to asserting the interrupt request. The interrupt request line can be cleared by writing a one to the interrupt status bit in the control register.

14.4 Register description

Note: In all registers, bits that are not shown are reserved.

Table 296. Register overview: GROUP0 interrupt (base address 0x4000 2000 (GINT0) and 0x4000 3000 (GINT1))

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	GPIO grouped interrupt control.	0	14.4.1
PORT_POL0	R/W	0x020	GPIO grouped interrupt port 0 polarity.	0xFFFF FFFF	14.4.2
PORT_POL1	R/W	0x024	GPIO grouped interrupt port 1 polarity.	0xFFFF FFFF	14.4.2
PORT_ENA0	R/W	0x040	GPIO grouped interrupt port 0 enable.	0	14.4.3
PORT_ENA1	R/W	0x044	GPIO grouped interrupt port 1 enable.	0	14.4.3

14.4.1 Grouped interrupt control register

Table 297. GPIO grouped interrupt control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	INT		Group interrupt status. This bit is cleared by writing a one to it. Writing zero has no effect.	0
		0	No request. No interrupt request is pending.	
		1	Request active. Interrupt request is active.	
1	COMB		Combine enabled inputs for group interrupt	0
		0	OR functionality: A grouped interrupt is generated when any one of the enabled inputs is active (based on its programmed polarity).	
		1	AND functionality: An interrupt is generated when all enabled bits are active (based on their programmed polarity).	
2	TRIG		Group interrupt trigger	0
		0	Edge-triggered.	
		1	Level-triggered.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	0

14.4.2 GPIO grouped interrupt port polarity registers

The grouped interrupt port polarity registers determine how the polarity of each enabled pin contributes to the grouped interrupt. Each port is associated with its own port polarity register, and the values of both registers together determine the grouped interrupt.

Each register PORT_POLm controls the polarity of pins in port m.

Table 298. GPIO grouped interrupt port polarity registers (PORT_POL[0:1], offset 0x020 for PORT_POL0; 0x024 for PORT_POL1) bit description

Bit	Symbol	Description	Reset value
31:0	POL	Configure pin polarity of port m pins for group interrupt. Bit n corresponds to pin PIOm_n of port m. 0 = the pin is active LOW. If the level on this pin is LOW, the pin contributes to the group interrupt. 1 = the pin is active HIGH. If the level on this pin is HIGH, the pin contributes to the group interrupt.	1

14.4.3 GPIO grouped interrupt port enable registers

The grouped interrupt port enable registers enable the pins which contribute to the grouped interrupt. Each port is associated with its own port enable register, and the values of both registers together determine which pins contribute to the grouped interrupt.

Each register PORT_ENm enables pins in port m.

Table 299. GPIO grouped interrupt port enable registers (PORT_ENA[0:1], offset 0x040 for PORT_ENA0; 0x044 PORT_ENA1) bit description

Bit	Symbol	Description	Reset value
31:0	ENA	Enable port 0 pin for group interrupt. Bit n corresponds to pin Pm_n of port m. 0 = the port 0 pin is disabled and does not contribute to the grouped interrupt. 1 = the port 0 pin is enabled and contributes to the grouped interrupt.	0

14.5 Functional description

Any subset of the pins in each port can be selected to contribute to a common group interrupt (GINT) and can be enabled to wake the part up from deep-sleep mode.

An interrupt can be requested for each port, based on any selected subset of pins within each port. The pins that contribute to each port interrupt are selected by 1s in the port's Enable register, and an interrupt polarity can be selected for each pin in the port's Polarity register. The level on each pin is exclusive-ORed with its polarity bit, and the result is ANDed with its enable bit. These results are then inclusive-ORed among all the pins in the port to create the port's raw interrupt request.

The raw interrupt request from each of the two group interrupts is sent to the NVIC, which can be programmed to treat it as level- or edge-sensitive, or it can be edge-detected by the wake-up interrupt logic (see Table 222).

15.1 How to read this chapter

The DMA controller is available on all LPC546xx devices.

15.2 Features

- 30 channels, 28 of which are connected to peripheral DMA requests. These come from the Flexcomm Interface (USART, SPI, I^2C, and I^2S), digital microphone, SPIFI, SHA, and SmartCard interfaces. Two spare channels have no DMA request connected, and can be used for functions such as memory-to-memory moves. Any otherwise unused channel can also be used for other purposes.
- DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 24 sources. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with four entries (each entry is a pair of transfer addresses).
- Efficient use of data bus.
- Supports single transfers up to 1,024 words. A single transfer is defined as one read from the source address, followed by one write to the destination address.
- Address increment options allow packing and/or unpacking data.

15.3 Basic configuration

Configure the DMA as follows:

- Use the AHBCLKCTRL0 register (Table 139) to enable the clock to the DMA registers interface.
- Clear the DMA peripheral reset using the PRESETCTRL0 register (Table 129).
- The DMA controller provides an interrupt to the NVIC, see Chapter 6.
- Most peripherals that support DMA, the ADC being an exception, have at least one DMA request line associated with them. The related channel(s) must be set up according to the desired operation. the ADC uses a trigger instead of a DMA request. DMA requests and triggers are described in detail in Section 15.5.1
- For peripherals using DMA requests, DMA operation must be triggered before any transfer will occur. This can be done by software, or can optionally be signalled by one of 20 hardware triggers, through the input mux registers DMA_ITRIG_INMUX[0:19]. DMA requests and triggers are described in detail in Section 15.5.1
- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to DMA_INMUX_INMUX[0:3] as inputs to DMA triggers.

For details on the trigger input and output multiplexing, see Section 11.5.3 “DMA trigger input multiplexing”.

15.4 Pin description

The DMA controller has no direct pin connections. However, some DMA triggers can be associated with pin functions (see Section 15.5.1.2).

15.5 General description

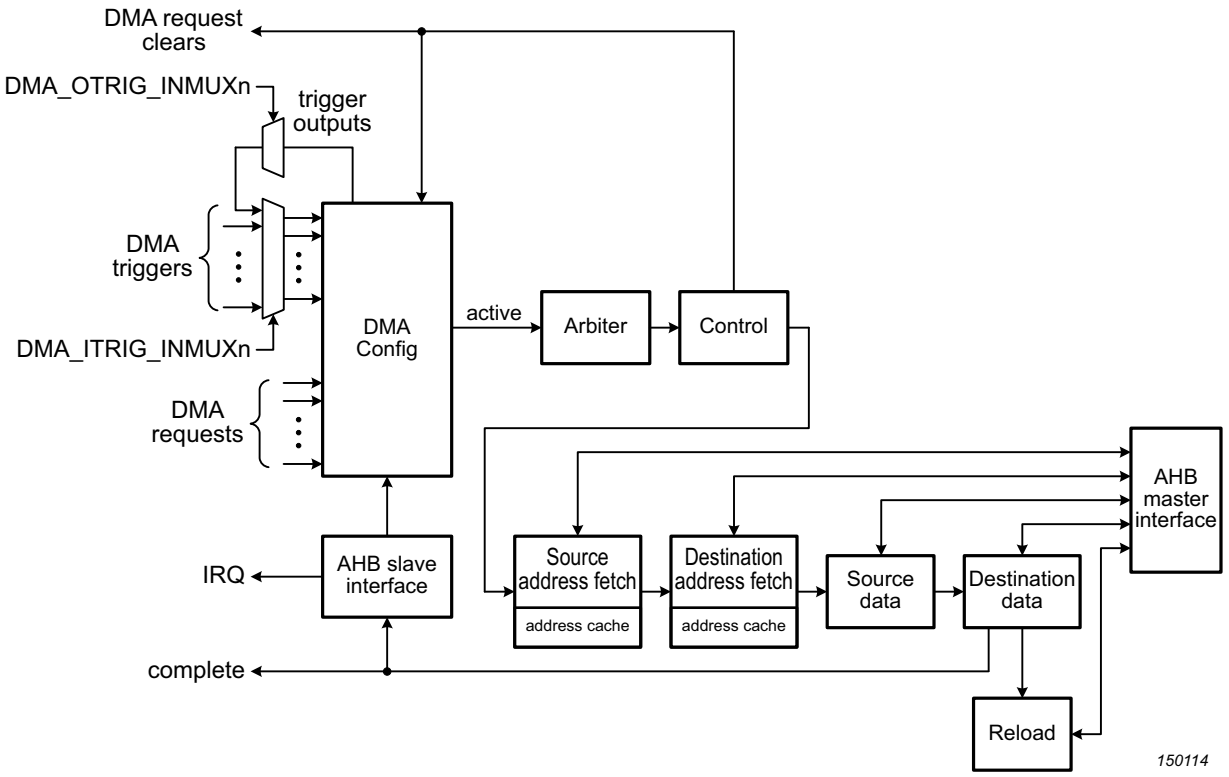


Fig 26. DMA block diagram

15.5.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty. DMA requests are summarized in Table 300.

Triggers start the transfer. In typical cases, only a software trigger will probably be used. Other possibilities are provided for, such as starting a DMA transfer when certain timer or pin related events occur. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Note that no DMA activity will take place for any particular DMA channel unless that channel has been triggered, either by software or hardware. DMA triggers are summarized in Table 300.

There is one specific exception to the above description, which is the ADC. The ADC doesn't fit the simple pacing signal model very well because of the possibilities represented by the programmable conversion sequences. A sequence complete DMA

request is likely to require transferring several non-contiguous result registers at once (see Chapter 44 “LPC546xx 12-bit ADC controller (ADC)”). It might also require other things to be done that can be done by the DMA without software intervention. This model fits better with the trigger facility, so that is how the ADC is connected to the DMA controller.

Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel’s DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral’s transmit buffer.

Each DMA channel also has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

15.5.1.1 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input. Some DMA requests allow a selection of requests sources. DMA triggers are selected from many possible input sources. The possibilities are shown in Table 300.

Table 300. DMA requests and trigger muxes

DMA channel #	Request input	DMA trigger mux	Software trigger
0	Flexcomm Interface 0 RX / I2C Slave [1]	DMA_ITRIG_INMUX0	Yes
1	Flexcomm Interface 0 TX / I2C Master [1]	DMA_ITRIG_INMUX1	Yes
2	Flexcomm Interface 1 RX / I2C Slave [1]	DMA_ITRIG_INMUX2	Yes
3	Flexcomm Interface 1 TX / I2C Master [1]	DMA_ITRIG_INMUX3	Yes
4	Flexcomm Interface 2 RX / I2C Slave [1]	DMA_ITRIG_INMUX4	Yes
5	Flexcomm Interface 2 TX / I2C Master [1]	DMA_ITRIG_INMUX5	Yes
6	Flexcomm Interface 3 RX / I2C Slave [1]	DMA_ITRIG_INMUX6	Yes
7	Flexcomm Interface 3 TX / I2C Master [1]	DMA_ITRIG_INMUX7	Yes
8	Flexcomm Interface 4 RX / I2C Slave [1]	DMA_ITRIG_INMUX8	Yes
9	Flexcomm Interface 4 TX / I2C Master [1]	DMA_ITRIG_INMUX9	Yes
10	Flexcomm Interface 5 RX / I2C Slave [1]	DMA_ITRIG_INMUX10	Yes
11	Flexcomm Interface 5 TX / I2C Master [1]	DMA_ITRIG_INMUX11	Yes
12	Flexcomm Interface 6 RX / I2C Slave [1]	DMA_ITRIG_INMUX12	Yes
13	Flexcomm Interface 6 TX / I2C Master [1]	DMA_ITRIG_INMUX13	Yes
14	Flexcomm Interface 7 RX / I2C Slave [1]	DMA_ITRIG_INMUX14	Yes
15	Flexcomm Interface 7 TX / I2C Master [1]	DMA_ITRIG_INMUX15	Yes
16	DMIC0	DMA_ITRIG_INMUX16	Yes
17	DMIC1	DMA_ITRIG_INMUX17	Yes
18	SPIFI	DMA_ITRIG_INMUX18	Yes
19	SHA	DMA_ITRIG_INMUX19	Yes

Table 300. DMA requests and trigger muxes

DMA channel #	Request input	DMA trigger mux	Software trigger
20	Flexcomm Interface 8_RX / I2C Slave [1]	DMA_ITRIG_INMUX20	Yes
21	Flexcomm Interface 8_TX / I2C Slave [1]	DMA_ITRIG_INMUX21	Yes
22	Flexcomm Interface 9_RX / I2C Slave [1]	DMA_ITRIG_INMUX22	Yes
23	Flexcomm Interface 9_TX / I2C Slave [1]	DMA_ITRIG_INMUX23	Yes
24	SMARTCARD0_RX	DMA_ITRIG_INMUX24	Yes
25	SMARTCARD0_TX	DMA_ITRIG_INMUX25	Yes
26	SMARTCARD1_RX	DMA_ITRIG_INMUX26	Yes
27	SMARTCARD1_TX	DMA_ITRIG_INMUX27	Yes
28	(no DMA request)	DMA_ITRIG_INMUX28	Yes
29	(no DMA request)	DMA_ITRIG_INMUX29	Yes

[1] See Section 15.5.1.1.1 for information about DMA for the I2C Monitor function.

15.5.1.1.1 DMA with I2C monitor mode

The I2C monitor function may be used with DMA if one of the channels related to the same Flexcomm Interface is available.

Table 301. DMA with the I2C Monitor function

I2C Master DMA	I2C Slave DMA	I2C Monitor DMA
Not enabled	-	If I2C Monitor DMA is enabled, it will use the DMA channel for the Master function of the same Flexcomm Interface.
Enabled	Not enabled	If I2C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same Flexcomm Interface.
Enabled	Enabled	The I2C Monitor function cannot use DMA.

15.5.1.2 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 20 possible internal trigger sources for each DMA channel. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See Table 302 and Chapter 11 “LPC546xx Input multiplexing (INPUT MUX)”.

Note that the ADC is unique in that it uses DMA triggers only, and has no DMA requests.

Table 302. DMA trigger sources

DMA trigger #	Trigger input
0	ADC0 sequence A interrupt
1	ADC0 sequence B interrupt
2	SCT0 DMA request 0
3	SCT0 DMA request 1
4	Pin interrupt 0
5	Pin interrupt 1
6	Pin interrupt 2
7	Pin interrupt 3
8	Timer CTIMER0 Match 0 DMA request

Table 302. DMA trigger sources

DMA trigger #	Trigger input
9	Timer CTIMER0 Match 1 DMA request
10	Timer CTIMER1 Match 0 DMA request
11	Timer CTIMER1 Match 1 DMA request
12	Timer CTIMER2 Match 1 DMA request
13	Timer CTIMER2 Match 0 DMA request
14	Timer CTIMER3 Match 0 DMA request
15	Timer CTIMER3 Match 1 DMA request
16	Timer CTIMER4 Match 0 DMA request
17	Timer CTIMER4 Match 1 DMA request
18	DMA output trigger 0
19	DMA output trigger 1
20	DMA output trigger 2
21	DMA output trigger 3

15.5.1.3 Trigger operation detail

A trigger of some kind is always needed to start a transfer on a DMA channel. This can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the SWTRIG bit equal to 0, the channel can be later triggered either by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate bit in the SETTRIG register. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the CFG register for the related channel. When a channel is initially set up, the SWTRIG bit in the XFERCFG register can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer will proceed at full speed.

The TRIG bit in the CTLSTAT register can be cleared at the end of a transfer, determined by the value CLRTRIG (bit 0) in the XFERCFG register. When a 1 is found in CLRTRIG, the trigger is cleared when the descriptor is exhausted.

15.5.1.4 Trigger output detail

Each channel of the DMA controller provides a trigger output. This allows the possibility of using the trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, for example, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called “chained operation” or “channel chaining”.

15.5.2 DMA Modes

The DMA controller does not really have separate operating modes, but there are ways of using the DMA controller that have commonly used terminology in the industry.

Using any specific DMA channel requires initializing the device registers associated with that channel (see Table 300), and setting the channel descriptor. The channel descriptor is shown in Table 303. The channel descriptor is an entry in the channel descriptor table. This table is located somewhere in memory, typically in on-chip SRAM (see Section 15.6.3).

Table 303: Channel descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

The source and destination end addresses, as well as the link to the next descriptor are just memory addresses that can point to any valid address on the device. The link to the next descriptor is used only if it is a linked transfer.

The DMA transfer is initiated by writing the channels CFG and XFERCFG registers and setting the channels ENABLESET bit. Note that once a DMA operation is initiated, the descriptor entry in the DMA descriptor table may be modified by the DMA controller. Therefore, when a subsequent DMA transaction is initiated, the DMA descriptor entry must be re-initialized. This is not the case with the linked descriptors. The linked list entries, that are not in the main DMA descriptor table, do not need to be re-written, for subsequent DMA requests. These entries are not modified by the controller.

When a DMA transfer involves a fixed peripheral data register, such as, when moving data from memory to a peripheral or moving data from a peripheral to memory, the address used for SRCINC or DSTINC (whichever corresponds to the fixed peripheral data address) is the address of the peripheral data register. The memory address for such a transfer is based on the end (upper) address of the memory buffer. The value can be calculated from the starting address of the buffer and the length of the buffer, where the transfer increment is the value specified by SRCINC or DSTINC (whichever corresponds to the memory buffer):

Buffer ending address = buffer starting address + (XFERCOUNT * the transfer increment)

See Section 15.6.18 for the description of SRCINC and DSTINC. Note that XFERCOUNT is defined as the actual count minus 1 and that is why it is not necessary to subtract 1 from the count in the equation above.

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the channel descriptor will be reloaded with data from memory pointed to by the “Link to next descriptor” entry of the initial channel descriptor. Descriptors loaded in this manner look slightly different the channel descriptor, as shown in Table 304. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

Table 304: Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor).

15.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 305 is needed.

Table 305: Channel descriptor for a single transfer

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

This case is identified by the Reload bit in the XFERCFG register = 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the channel descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

15.5.4 Ping-Pong

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 306 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 306: Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor	
+ 0x0	(not used)
+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B

Descriptor B	
+ 0x0	Buffer B transfer configuration
+ 0x4	Peripheral data end address
+ 0x8	Buffer B memory end address
+ 0xC	Address of descriptor A

Descriptor A	
+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B

In this example, the channel descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the channel descriptor is exhausted, descriptor B is loaded using the link to descriptor B, and a transfer interrupt informs the CPU that buffer A is available.

Descriptor B is then used until it is also exhausted, when descriptor A is loaded using the link to descriptor A contained in descriptor B. Then a transfer interrupt informs the CPU that buffer B is available for processing. The process repeats when descriptor A is exhausted, alternately using each of the 2 memory buffers.

15.5.5 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFERCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if 4 data samples from several peripherals need to be interleaved into a single data structure, this may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for 4 samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using DSTINC to de-interleave combined data from the buffer and send it to several peripherals or locations.

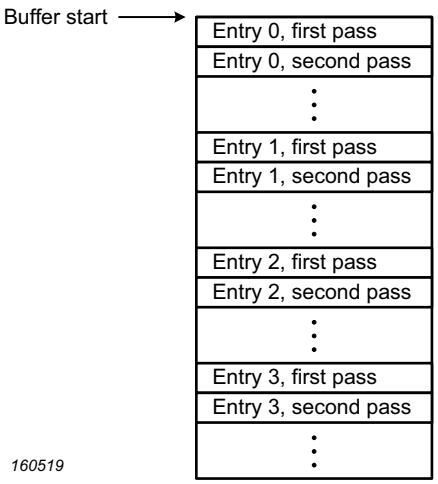


Fig 27. Interleaved transfer in a single buffer

15.5.6 Linked transfers (linked list)

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 306). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as

desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

15.5.7 Address alignment for data transfers

Transfers of 16-bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

15.5.8 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature can for example be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the ARM core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:
  - If channel x is configured to auto reload the descriptor on exhausting of the descriptor (bit RELOAD in the transfer configuration of the descriptor is set), then enable 'clear trigger on descriptor exhausted' by setting bit CLRTRIG in the channel's transfer configuration in the descriptor.
- For channel y:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use any of the available DMA trigger muxes (DMA trigger mux 0/1).
  - Configure the chosen DMA trigger mux to select DMA channel x.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

Note that after completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above. In addition to that:

- A ping-pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.
- For channel x:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use the same DMA trigger mux as chosen for channel y.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.

- Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
- Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

15.5.9 DMA in reduced power modes

DMA in sleep mode

In sleep mode, the DMA can operate and access all enabled SRAM blocks, without waking up the CPU.

DMA in deep-sleep mode

Some peripherals support DMA service during deep-sleep mode without waking up the CPU or the rest of the device. These peripherals are the Flexcomm Interface functions that include FIFO support (USART, SPI, and I2S), and the DMIC.

These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts. See Section 7.5.96 for more information.

15.6 Register description

The DMA registers are grouped into DMA control, interrupt and status registers and DMA channel registers. DMA transfers are controlled by a set of three registers per channel, the CFG[0:29], CTRLSTAT[0:29], and XFERCFG[0:29] registers.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 307. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
Global control and status registers					
CTRL	R/W	0x000	DMA control.	0	15.6.1
INTSTAT	RO	0x004	Interrupt status.	0	15.6.2
SRAMBASE	R/W	0x008	SRAM address of the channel configuration table.	0	15.6.3
Shared registers					
ENABLESET0	R/W	0x020	Channel enable read and Set for all DMA channels.	0	15.6.4
ENABLECLR0	WO	0x028	Channel enable clear for all DMA channels.	NA	15.6.5
ACTIVE0	RO	0x030	Channel active status for all DMA channels.	0	15.6.6
BUSY0	RO	0x038	Channel busy status for all DMA channels.	0	15.6.7
ERRINT0	R/W	0x040	Error interrupt status for all DMA channels.	0	15.6.8
INTENSET0	R/W	0x048	Interrupt enable read and Set for all DMA channels.	0	15.6.9
INTENCLR0	WO	0x050	Interrupt enable clear for all DMA channels.	NA	15.6.10
INTA0	R/W	0x058	Interrupt A status for all DMA channels.	0	15.6.11
INTB0	R/W	0x060	Interrupt B status for all DMA channels.	0	15.6.12
SETVALID0	WO	0x068	Set ValidPending control bits for all DMA channels.	NA	15.6.13
SETTRIG0	WO	0x070	Set trigger control bits for all DMA channels.	NA	15.6.14
ABORT0	WO	0x078	Channel abort control for all DMA channels.	NA	15.6.15
Channel 0 registers					
CFG0	R/W	0x400	Configuration register for DMA channel 0.	0	15.6.16
CTLSTAT0	RO	0x404	Control and status register for DMA channel 0.	0	15.6.17
XFERCFG0	R/W	0x408	Transfer configuration register for DMA channel 0.	0	15.6.18
Channel 1 registers					
CFG1	R/W	0x410	Configuration register for DMA channel 1.	0	15.6.16
CTLSTAT1	RO	0x414	Control and status register for DMA channel 1.	0	15.6.17
XFERCFG1	R/W	0x418	Transfer configuration register for DMA channel 1.	0	15.6.18
Channel 2 registers					
CFG2	R/W	0x420	Configuration register for DMA channel 2.	0	15.6.16
CTLSTAT2	RO	0x424	Control and status register for DMA channel 2.	0	15.6.17
XFERCFG2	R/W	0x428	Transfer configuration register for DMA channel 2.	0	15.6.18
Channel 3 registers					
CFG3	R/W	0x430	Configuration register for DMA channel 3.	0	15.6.16
CTLSTAT3	RO	0x434	Control and status register for DMA channel 3.	0	15.6.17
XFERCFG3	R/W	0x438	Transfer configuration register for DMA channel 3.	0	15.6.18

Table 307. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
Channel 4 registers					
CFG4	R/W	0x440	Configuration register for DMA channel 4.	0	15.6.16
CTLSTAT4	RO	0x444	Control and status register for DMA channel 4.	0	15.6.17
XFERCFG4	R/W	0x448	Transfer configuration register for DMA channel 4.	0	15.6.18
Channel 5 registers					
CFG5	R/W	0x450	Configuration register for DMA channel 5.	0	15.6.16
CTLSTAT5	RO	0x454	Control and status register for DMA channel 5.	0	15.6.17
XFERCFG5	R/W	0x458	Transfer configuration register for DMA channel 5.	0	15.6.18
Channel 6 registers					
CFG6	R/W	0x460	Configuration register for DMA channel 6.	0	15.6.16
CTLSTAT6	RO	0x464	Control and status register for DMA channel 6.	0	15.6.17
XFERCFG6	R/W	0x468	Transfer configuration register for DMA channel 6.	0	15.6.18
Channel 7 registers					
CFG7	R/W	0x470	Configuration register for DMA channel 7.	0	15.6.16
CTLSTAT7	RO	0x474	Control and status register for DMA channel 7.	0	15.6.17
XFERCFG7	R/W	0x478	Transfer configuration register for DMA channel 7.	0	15.6.18
Channel 8 registers					
CFG8	R/W	0x480	Configuration register for DMA channel 8.	0	15.6.16
CTLSTAT8	RO	0x484	Control and status register for DMA channel 8.	0	15.6.17
XFERCFG8	R/W	0x488	Transfer configuration register for DMA channel 8.	0	15.6.18
Channel 9 registers					
CFG9	R/W	0x490	Configuration register for DMA channel 9.	0	15.6.16
CTLSTAT9	RO	0x494	Control and status register for DMA channel 9.	0	15.6.17
XFERCFG9	R/W	0x498	Transfer configuration register for DMA channel 9.	0	15.6.18
Channel 10 registers					
CFG10	R/W	0x4A0	Configuration register for DMA channel 10.	0	15.6.16
CTLSTAT10	RO	0x4A4	Control and status register for DMA channel 10.	0	15.6.17
XFERCFG10	R/W	0x4A8	Transfer configuration register for DMA channel 10.	0	15.6.18
Channel 11 registers					
CFG11	R/W	0x4B0	Configuration register for DMA channel 11.	0	15.6.16
CTLSTAT11	RO	0x4B4	Control and status register for DMA channel 11.	0	15.6.17
XFERCFG11	R/W	0x4B8	Transfer configuration register for DMA channel 11.	0	15.6.18
Channel 12 registers					
CFG12	R/W	0x4C0	Configuration register for DMA channel 12.	0	15.6.16
CTLSTAT12	RO	0x4C4	Control and status register for DMA channel 12.	0	15.6.17
XFERCFG12	R/W	0x4C8	Transfer configuration register for DMA channel 12.	0	15.6.18
Channel 13 registers					
CFG13	R/W	0x4D0	Configuration register for DMA channel 13.	0	15.6.16
CTLSTAT13	RO	0x4D4	Control and status register for DMA channel 13.	0	15.6.17
XFERCFG13	R/W	0x4D8	Transfer configuration register for DMA channel 13.	0	15.6.18
Channel 14 registers					

Table 307. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
CFG14	R/W	0x4E0	Configuration register for DMA channel 14.	0	15.6.16
CTLSTAT14	RO	0x4E4	Control and status register for DMA channel 14.	0	15.6.17
XFERCFG14	R/W	0x4E8	Transfer configuration register for DMA channel 14.	0	15.6.18
Channel 15 registers					
CFG15	R/W	0x4F0	Configuration register for DMA channel 15.	0	15.6.16
CTLSTAT15	RO	0x4F4	Control and status register for DMA channel 15.	0	15.6.17
XFERCFG15	R/W	0x4F8	Transfer configuration register for DMA channel 15.	0	15.6.18
Channel 16 registers					
CFG16	R/W	0x500	Configuration register for DMA channel 16.	0	15.6.16
CTLSTAT16	RO	0x504	Control and status register for DMA channel 16.	0	15.6.17
XFERCFG16	R/W	0x508	Transfer configuration register for DMA channel 16.	0	15.6.18
Channel 17 registers					
CFG17	R/W	0x510	Configuration register for DMA channel 17.	0	15.6.16
CTLSTAT17	RO	0x514	Control and status register for DMA channel 17.	0	15.6.17
XFERCFG17	R/W	0x518	Transfer configuration register for DMA channel 17.	0	15.6.18
Channel 18 registers					
CFG18	R/W	0x520	Configuration register for DMA channel 18.	0	15.6.16
CTLSTAT18	RO	0x524	Control and status register for DMA channel 18.	0	15.6.17
XFERCFG18	R/W	0x528	Transfer configuration register for DMA channel 18.	0	15.6.18
Channel 19 registers					
CFG19	R/W	0x530	Configuration register for DMA channel 19.	0	15.6.16
CTLSTAT19	RO	0x534	Control and status register for DMA channel 19.	0	15.6.17
XFERCFG19	R/W	0x538	Transfer configuration register for DMA channel 19.	0	15.6.18
Channel 20 registers					
CFG20	R/W	0x540	Configuration register for DMA channel 20.	0	15.6.16
CTLSTAT20	RO	0x544	Control and status register for DMA channel 20.	0	15.6.17
XFERCFG20	R/W	0x548	Transfer configuration register for DMA channel 20.	0	15.6.18
Channel 21 registers					
CFG21	R/W	0x550	Configuration register for DMA channel 21.	0	15.6.16
CTLSTAT21	RO	0x554	Control and status register for DMA channel 21.	0	15.6.17
XFERCFG21	R/W	0x558	Transfer configuration register for DMA channel 21.	0	15.6.18
Channel 22 registers					
CFG22	R/W	0x560	Configuration register for DMA channel 22.	0	15.6.16
CTLSTAT22	RO	0x564	Control and status register for DMA channel 22.	0	15.6.17
XFERCFG22	R/W	0x568	Transfer configuration register for DMA channel 22.	0	15.6.18
Channel 23 registers					
CFG23	R/W	0x570	Configuration register for DMA channel 23.	0	15.6.16
CTLSTAT23	RO	0x574	Control and status register for DMA channel 23.	0	15.6.17
XFERCFG23	R/W	0x578	Transfer configuration register for DMA channel 23.	0	15.6.18

Table 307. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
Channel 24 registers					
CFG24	R/W	0x580	Configuration register for DMA channel 24.	0	15.6.16
CTLSTAT24	RO	0x584	Control and status register for DMA channel 24.	0	15.6.17
XFERCFG24	R/W	0x588	Transfer configuration register for DMA channel 24.	0	15.6.18
Channel 25 registers					
CFG25	R/W	0x590	Configuration register for DMA channel 25.	0	15.6.16
CTLSTAT25	RO	0x594	Control and status register for DMA channel 25.	0	15.6.17
XFERCFG25	R/W	0x598	Transfer configuration register for DMA channel 25.	0	15.6.18
Channel 26 registers					
CFG26	R/W	0x5A0	Configuration register for DMA channel 26.	0	15.6.16
CTLSTAT26	RO	0x5A4	Control and status register for DMA channel 26.	0	15.6.17
XFERCFG26	R/W	0x5A8	Transfer configuration register for DMA channel 26.	0	15.6.18
Channel 27 registers					
CFG27	R/W	0x5B0	Configuration register for DMA channel 27.	0	15.6.16
CTLSTAT27	RO	0x5B4	Control and status register for DMA channel 27.	0	15.6.17
XFERCFG27	R/W	0x5B8	Transfer configuration register for DMA channel 27.	0	15.6.18
Channel 28 registers					
CFG28	R/W	0x5C0	Configuration register for DMA channel 28.	0	15.6.16
CTLSTAT28	RO	0x5C4	Control and status register for DMA channel 28.	0	15.6.17
XFERCFG28	R/W	0x5C8	Transfer configuration register for DMA channel 28.	0	15.6.18
Channel 29 registers					
CFG29	R/W	0x5D0	Configuration register for DMA channel 29.	0	15.6.16
CTLSTAT29	RO	0x5D4	Control and status register for DMA channel 29.	0	15.6.17
XFERCFG29	R/W	0x5D8	Transfer configuration register for DMA channel 29.	0	15.6.18

15.6.1 Control register

The CTRL register contains global the control bit for a enabling the DMA controller.

Table 308. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		DMA controller master enable.	0
		0	Disabled. The DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled.	
		1	Enabled. The DMA controller is enabled.	
31:1	-	-	Reserved. Read value is undefined, only zero should be written.	NA

15.6.2 Interrupt Status register

The read-only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved are found in the interrupt type specific registers.

Table 309. Interrupt Status register (INTSTAT, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved. Read value is undefined, only zero should be written.	NA
1	ACTIVEINT		Summarizes whether any enabled interrupts (other than error interrupts) are pending.	0
		0	Not pending. No enabled interrupts are pending.	
		1	Pending. At least one enabled interrupt is pending.	
2	ACTIVEERRINT		Summarizes whether any error interrupts are pending.	0
		0	Not pending. No error interrupts are pending.	
		1	Pending. At least one error interrupt is pending.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	NA

15.6.3 SRAM Base address register

The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Table 310. SRAM Base address register (SRAMBASE, offset 0x008) bit description

Bit	Symbol	Description	Reset value
8:0	-	Reserved. Read value is undefined, only zero should be written.	NA
31:9	OFFSET	Address bits 31:9 of the beginning of the DMA descriptor table. For 18 channels, the table must begin on a 512 byte boundary.	0

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 311. Only the descriptors for channels defined at extraction are used. The contents of each channel descriptor are described in Table 303.

Table 311. Channel descriptor map

Descriptor	Table offset
Channel descriptor for DMA channel 0	0x000
Channel descriptor for DMA channel 1	0x010
Channel descriptor for DMA channel 2	0x020
Channel descriptor for DMA channel 3	0x030
Channel descriptor for DMA channel 4	0x040
Channel descriptor for DMA channel 5	0x050
Channel descriptor for DMA channel 6	0x060
Channel descriptor for DMA channel 7	0x070
Channel descriptor for DMA channel 8	0x080
Channel descriptor for DMA channel 9	0x090
Channel descriptor for DMA channel 10	0x0A0
Channel descriptor for DMA channel 11	0x0B0
Channel descriptor for DMA channel 12	0x0C0
Channel descriptor for DMA channel 13	0x0D0
Channel descriptor for DMA channel 14	0x0E0
Channel descriptor for DMA channel 15	0x0F0
Channel descriptor for DMA channel 16	0x100
Channel descriptor for DMA channel 17	0x110
Channel descriptor for DMA channel 18	0x120
Channel descriptor for DMA channel 19	0x130
Channel descriptor for DMA channel 20	0x140
Channel descriptor for DMA channel 21	0x150
Channel descriptor for DMA channel 22	0x160
Channel descriptor for DMA channel 23	0x170
Channel descriptor for DMA channel 24	0x180
Channel descriptor for DMA channel 25	0x190
Channel descriptor for DMA channel 26	0x1A0
Channel descriptor for DMA channel 27	0x1B0
Channel descriptor for DMA channel 28	0x1C0
Channel descriptor for DMA channel 29	0x1D0

15.6.4 Enable read and Set registers

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel.

Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Table 312. Enable read and Set register 0 (ENABLESET0, offset 0x020) bit description

Bit	Symbol	Description	Reset value
31:0	ENA	Enable for DMA channels. Bit n enables or disables DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = disabled. 1 = enabled.	0

15.6.5 Enable Clear register

The ENABLECLR0 register is used to clear the enable of one or more DMA channels. This register is write-only.

Table 313. Enable Clear register 0 (ENABLECLR0, offset 0x028) bit description

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET0. Bit n clears the channel enable bit n. The number of bits = number of DMA channels in this device. Other bits are reserved.	NA

15.6.6 Active status register

The ACTIVE0 register indicates which DMA channels are active at the point when the read occurs. The register is read-only.

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see Section 15.6.15).

Table 314. Active status register 0 (ACTIVE0, offset 0x030) bit description

Bit	Symbol	Description	Reset value
31:0	ACT	Active flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = not active. 1 = active.	0

15.6.7 Busy status register

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. This registers is read-only.

A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Table 315. Busy status register 0 (BUSY0, offset 0x038) bit description

Bit	Symbol	Description	Reset value
31:0	BSY	Busy flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = not busy. 1 = busy.	0

15.6.8 Error Interrupt register

The ERRINT0 register contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Table 316. Error Interrupt register 0 (ERRINT0, offset 0x040) bit description

Bit	Symbol	Description	Reset value
31:0	ERR	Error Interrupt flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = error interrupt is not active. 1 = error interrupt is active.	0

15.6.9 Interrupt Enable read and Set register

The INTENSET0 register controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Table 317. Interrupt Enable read and Set register 0 (INTENSET0, offset 0x048) bit description

Bit	Symbol	Description	Reset value
31: 0	INTEN	Interrupt Enable read and set for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0

15.6.10 Interrupt Enable Clear register

The INTENCLR0 register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Table 318. Interrupt Enable Clear register 0 (INTENCLR0, offset 0x050) bit description

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET0. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved.	NA

15.6.11 Interrupt A register

The IntA0 register contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Table 319. Interrupt A register 0 (INTA0, offset 0x058) bit description

Bit	Symbol	Description	Reset value
31:0	IA	Interrupt A status for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	0

15.6.12 Interrupt B register

The INTB0 register contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Table 320. Interrupt B register 0 (INTB0, offset 0x060) bit description

Bit	Symbol	Description	Reset value
31:0	IB	Interrupt B status for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	0

15.6.13 Set Valid register

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. See Section 15.6.17 for a description of the VALID bit. This register is write-only.

The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Table 321. Set Valid 0 register (SETVALID0, offset 0x068) bit description

Bit	Symbol	Description	Reset value
31:0	SV	SETVALID control for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n	NA

15.6.14 Set Trigger register

The SETTRIG0 register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See Section 15.6.17 for a description of the TRIG bit, and Section 15.5.1 for a general description of triggering. This register is write-only.

Table 322. Set Trigger 0 register (SETTRIG0, offset 0x070) bit description

Bit	Symbol	Description	Reset value
31:0	TRIG	Set Trigger control bit for DMA channel 0. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	NA

15.6.15 Abort register

The Abort0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again. This register is write-only.

Table 323. Abort 0 register (ABORT0, offset 0x078) bit description

Bit	Symbol	Description	Reset value
31:0	ABORTCTRL	Abort control for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = aborts DMA operations on channel n.	NA

15.6.16 Channel configuration registers

The CFGn register contains various configuration options for DMA channel n.

See Table 325 for a summary of trigger options.

Table 324. Channel configuration registers bit description

Bit	Symbol	Value	Description	Reset value
0	PERIPHREQEN		Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory move, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller.	0
		0	Disabled. Peripheral DMA requests are disabled.	
		1	Enabled. Peripheral DMA requests are enabled.	
1	HWTRIGEN		Hardware Triggering Enable for this channel.	0
		0	Disabled. Hardware triggering is not used.	
		1	Enabled. Use hardware triggering.	
3:2	-		Reserved. Read value is undefined, only zero should be written.	NA
4	TRIGPOL		Trigger Polarity. Selects the polarity of a hardware trigger for this channel.	0
		0	Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE.	
		1	Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.	
5	TRIGTYPE		Trigger Type. Selects hardware trigger as edge triggered or level triggered.	0
		0	Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger.	
		1	Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is deasserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.	
6	TRIGBURST		Trigger Burst. Selects whether hardware triggers cause a single or burst transfer.	0
		0	Single transfer. Hardware trigger causes a single transfer.	
		1	Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.	
7	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 324. Channel configuration registers bit description

Bit	Symbol	Value	Description	Reset value
11:8	BURSTPOWER		Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register). When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). ... 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size. Note that the total number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field).	0
13:12	-	-	Reserved. Read value is undefined, only zero should be written.	NA
14	SRCBURSTWRAP		Source Burst Wrap. When enabled, the source data address for the DMA is “wrapped”, meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst.	0
		0	Disabled. Source burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Source burst wrapping is enabled for this DMA channel.	
15	DSTBURSTWRAP		Destination Burst Wrap. When enabled, the destination data address for the DMA is “wrapped”, meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.	0
		0	Disabled. Destination burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Destination burst wrapping is enabled for this DMA channel.	
18:16	CHPRIORITY		Priority of this channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.	0
31:19	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 325. Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.

Table 325. Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

15.6.17 Channel control and status registers

The CTLSTATn register provides status flags specific to DMA channel n.These registers are read-only.

Table 326. Channel control and Status registers bit description

Bit	Symbol	Value	Description	Reset value
0	VALIDPENDING		Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel.	0
		0	No effect. No effect on DMA operation.	
		1	Valid pending.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	NA
2	TRIG		Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1.	0
		0	Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out.	
		1	Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	NA

15.6.18 Channel transfer configuration registers

The XFERCFGn register contains transfer related configuration information for DMA channel n. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

See Section 15.5.1.3 “Trigger operation detail”.

Table 327. Channel transfer configuration registers bit description

Bit	Symbol	Value	Description	Reset value
0	CFGVALID		Configuration Valid flag. This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled.	0
		0	Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting.	
		1	Valid. The current channel descriptor is considered valid.	
1	RELOAD		Indicates whether the channel’s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers.	0
		0	Disabled. Do not reload the channels’ control structure when the current descriptor is exhausted.	
		1	Enabled. Reload the channels’ control structure when the current descriptor is exhausted.	
2	SWTRIG		Software Trigger.	0
		0	Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel.	
		1	Set. Not set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.	
3	CLRTRIG		Clear Trigger.	0
		0	Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started.	
		1	Cleared. The trigger is cleared when this descriptor is exhausted.	
4	SETINTA		Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0
		0	No effect.	
		1	Set. The INTA flag for this channel will be set when the current descriptor is exhausted.	
5	SETINTB		Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0
		0	No effect.	
		1	Set. The INTB flag for this channel will be set when the current descriptor is exhausted.	
7:6	-	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 327. Channel transfer configuration registers bit description

Bit	Symbol	Value	Description	Reset value
9:8	WIDTH		Transfer width used for this DMA channel.	0
		0x0	8-bit. 8-bit transfers are performed (8-bit source reads and destination writes).	
		0x1	16-bit. 6-bit transfers are performed (16-bit source reads and destination writes).	
		0x2	32-bit. 32-bit transfers are performed (32-bit source reads and destination writes).	
		0x3	Reserved. Reserved setting, do not use.	
11:10	-	-	Reserved. Read value is undefined, only zero should be written.	NA
13:12	SRCINC		Determines whether the source address is incremented for each DMA transfer.	0
		0x0	No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device.	
		0x1	1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory.	
		0x2	2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.	
15:14	DSTINC		Determines whether the destination address is incremented for each DMA transfer.	0
		0x0	No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device.	
		0x1	1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory.	
		0x2	2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.	
25:16	XFERCOUNT		Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). XFERCOUNT is used to count down during DMA transfer. When one DMA transfer is completed, XFERCOUNT decrements by 1. Example: The total number of DMA transfer to complete is N. The initial value for XFERCOUNT is N-1. XFERCOUNT = N - 1 means there are N transfers to complete. XFEFCOUNT = N - 2 means there are N-1 transfers to complete. ... XFERCOUNT = 1 means there are 2 transfers to complete. XFERCOUNT = 0 means there is 1 transfer to complete. XFERCOUNT = 0x3FF means all transfers are completed. **Remark:** When all transfers are completed, XFERCOUNT value changes from 0 to 0x3FF. The last value 0x3FF does not mean there are 1024 transfers left to complete. If the initial value for XFERCOUNT is 0x3FF (that is, when the XFERCFGn register is programmed), then there are 1024 transfers to complete. The size of each DMA transfer is determined by the WIDTH field of the same XFERCFGn register.	0
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-

16.1 How to read this chapter

The SCTimer/PWM is available on all LPC546xx devices.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see Ref. 2 “AN11538”.

16.2 Features

- The SCTimer/PWM supports:
  - Eight inputs.
  - Ten outputs.
  - Ten match/capture registers.
  - Ten events.
  - Ten states.
- Counter/timer features:
  - Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
  - Counters clocked by system clock or selected input.
  - Configurable as up counters or up-down counters.
  - Configurable number of match and capture registers. Up to ten match and capture registers total.
  - When there is a match and/or an input or output transition or level, create events to accomplish any or all of the following: stop, limit or halt the timer; change counting direction; set, clear or toggle outputs; change the state; capture the counter value; generate an interrupt or DMA request.
  - Counter value can be loaded into capture register triggered by a match or input/output toggle.
- PWM features:
  - Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals.
  - PWM behavior can change based on the current state to create very complex, variable waveforms. In effect, states are a means of context switching for the entire SCTimer/PWM.
  - Up to 8 single-edge or 4 dual-edge PWM outputs with independent duty cycle and common PWM cycle length.
- Event creation features:
  - The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition. Event creation is qualified by States (“contexts”).
  - In bi-directional mode, events can be enabled based on the count direction.
  - Selected events can limit, halt, start, or stop a counter or change its direction.

- Events trigger state changes, output transitions, timer captures, interrupts, and DMA transactions.
- Match register 0 can be used as an automatic limit.
- Matches can be defined as “greater/less-than-or-equal-to” the counter value for purposes of event generation.

- State control features:
  - States have no pre-defined meaning. Entirely determined by the user. States provide a mechanism for context switching for the SCTimer/PWM including creation of complex state machines.SCTimer/PWM
  - The only function a State serves is to define which events can occur in that state.
  - A state changes to some other state in response to an event.
  - Each event can be enabled to occur in one or more states.
  - State variable allows sequencing across multiple counter cycles.

16.3 Basic configuration

Configure the SCTimer/PWM as follows:

- Select a function clock using SCTCLKSEL (Section 7.5.41).

  Remark: The maximum frequency for the SCTimer/PWM clock is 100 MHz.
- Enable the clock to the SCTimer/PWM in the AHBCLKCTRL1 register (Section 7.5.20) to enable the register interface and the peripheral clock.
- Clear the SCTimer/PWM peripheral reset using the PRESETCTRL register (Section 7.5.10).
- The SCTimer/PWM provides an interrupt to the NVIC, see Chapter 6.
- SCTimer/PWM inputs are selected from the SCTimer/PWM input mux registers. See Chapter 11 “LPC546xx Input multiplexing (INPUT MUX)”.
- The SCTimer/PWM DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 11.6.3 “DMA trigger input mux registers 0 to 29”.

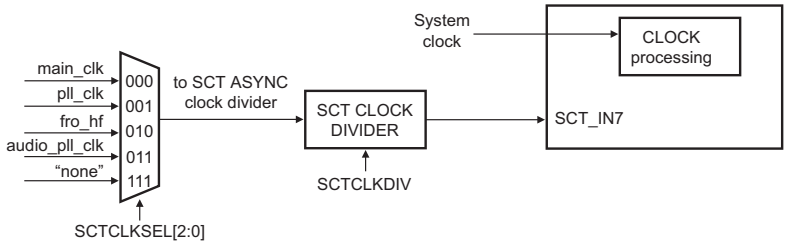


Fig 28. SCTimer/PWM clocking

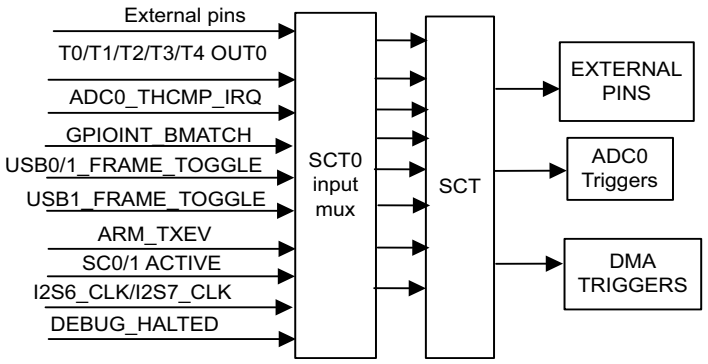


Fig 29. SCTimer/PWM connections

16.4 Pin description

Remark: Availability of inputs or outputs related to a particular peripheral function might be package dependent.

See Chapter 10 “LPC546xx I/O pin configuration (IOCON)” to assign the SCTimer/PWM functions to external pins.

SCTimer/PWM input signals may come from external pins and internal signals. These are connected to the SCTimer/PWM via input multiplexers (Chapter 11 “LPC546xx Input multiplexing (INPUT MUX)”) and IOCON (Chapter 10 “LPC546xx I/O pin configuration (IOCON)”).

SCTimer/PWM outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time. See Figure 30.

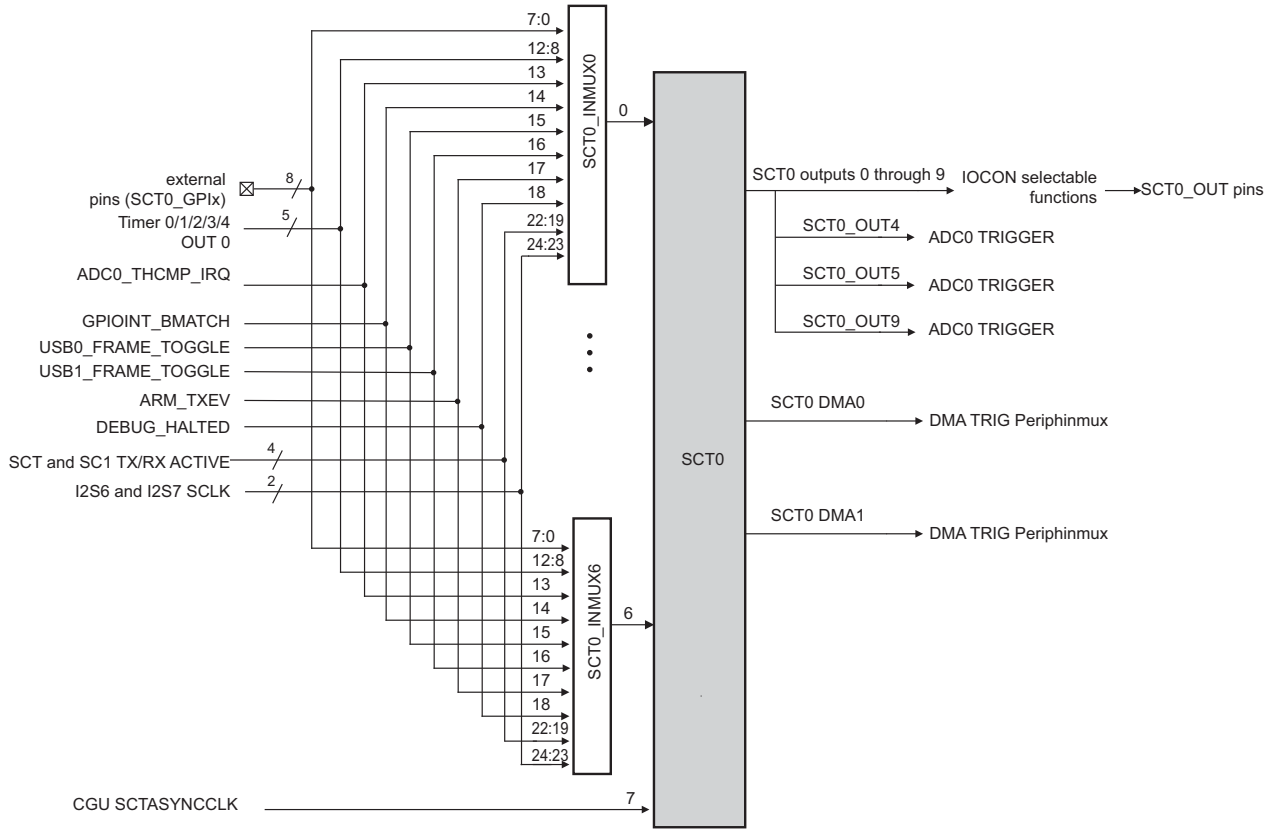


Fig 30. SCTimer/PWM inputs and outputs

Table 328. SCT0 pin description (internal signals)

Type	Connect to	Reference
internal signals	ADC0_THCMP_IRQ, DEBUG_HALTED, T0_OUT0, T1_OUT0, T2_OUT0, T3_OUT0, T4_OUT0, GPIOINT_BMATCH, USB0_FRAME_TOGGLE, USB1_FRAME_TOGGLE, ARM_TXEV, SMARTCARD0_TX_ACTIVE, SMARTCARD0_RX_ACTIVE, SMARTCARD1_TX_ACTIVE, SMARTCARD1_RX_ACTIVE, I2S6_SCLK, I2S7_SCLK.	Figure 29

Table 329. SCT0 pin description (inputs)

Type	Function	Connect to	Use register	Reference
external from pin	SCT0_GPI0	PIO0_0, PIO0_13, PIO0_24, PIO1_5, PIO3_31, PIO4_7, PIO5_5	IOCON register for the related pin	See Chapter 10
	SCT0_GPI1	PIO0_1, PIO0_14, PIO0_25, PIO4_0, PIO4_8, PIO5_6		
	SCT0_GPI2	PIO0_2, PIO0_20, PIO4_1, PIO4_9, PIO5_7		
	SCT0_GPI3	PIO0_3, PIO0_21, PIO1_6, PIO4_2, PIO4_10, PIO5_8		
	SCT0_GPI4	PIO0_4, PIO1_0, PIO1_7, PIO4_3, PIO4_11, PIO5_9		
	SCT0_GPI5	PIO0_5, PIO1_1, PIO1_22, PIO4_4, PIO4_12, PIO5_10		
	SCT0_GPI6	PIO0_6, PIO1_2, PIO1_29, PIO4_5, PIO4_13		
	SCT0_GPI7	PIO0_12, PIO0_17, PIO1_19, PIO1_30, PIO4_6, PIO4_14		
internal	-	ADC0 trigger	SCT0 output 4, SCT0 output 5, SCT0 output 6	Table 1044
internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	Table 300

Table 330. SCT0 pin description (outputs)

Type	Function	Connect to	Use register	Reference
external to pin	SCT0_OUT0	PIO0_2, PIO0_17, PIO1_4, PIO1_23, PIO3_26	IOCON register for the related pin	See Chapter 10
	SCT0_OUT1	PIO0_3, PIO0_18, PIO1_8, PIO1_24, PIO3_27		
	SCT0_OUT2	PIO0_15, PIO0_19, PIO1_9, PIO1_25, PIO3_28		
	SCT0_OUT3	PIO0_22, PIO0_31, PIO1_10, PIO1_26, PIO3_10, PIO3_29		
	SCT0_OUT4	PIO0_23, PIO1_3, PIO1_17, PIO3_14, PIO3_30		
	SCT0_OUT5	PIO0_26, PIO5_22, PIO1_18, PIO3_18, PIO3_31, PIO5_6		
	SCT0_OUT6	PIO0_27, PIO5_23, PIO1_31, PIO2_2, PIO3_19, PIO5_7		
	SCT0_OUT7	PIO0_28, PIO5_24, PIO1_19, PIO2_22, PIO3_20, PIO5_8		
	SCT0_OUT8	PIO2_23, PIO0_29, PIO3_12, PIO5_9, PIO5_25		
	SCT0_OUT9	PIO2_24, PIO0_30, PIO3_13, PIO5_10, PIO5_26		
internal	-	ADC0 trigger	SCT0 output 4, SCT0 output 5, SCT0 output 6	Table 1044
internal	-	SDMA trigger	SCT_DMA0, SCT_DMA1	Table 300

Table 331. Suggested SCTimer/PWM input pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	OD: Set to 0	Same as type D.	I2CFILTER: Set to 1
10	SLEW: Set to 0.	Not used, set to 0	I2CDRIVE: Set to 0.
9	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.

Table 331. Suggested SCTimer/PWM input pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
5:4	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 00.
3:0	FUNC: not used, set to 0. Specific pin inputs are directly connected to the SCTimer/PWM.	Same as type D.	Same as type D.
General comment	A good choice for an SCTimer/PWM input.	A reasonable choice for an SCTimer/PWM input.	A reasonable choice for an SCTimer/PWM input.

Recommended IOCON settings are shown in Table 331 and Table 332.

Table 332. Suggested SCTimer/PWM output pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1
10	SLEW: Set to 0.	Not used, set to 0	I2CDRIVE: Set to 0.
9	FILTEROFF: Set to 1.	Same as type D.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
5:4	MODE: set to 0.	Same as type D.	Not used, set to 0.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for an SCTimer/PWM output.	A reasonable choice for an SCTimer/PWM output.	Not recommended for SCTimer/PWM outputs.

16.5 General description

The SCTimer/PWM is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention.

The SCTimer/PWM can operate as a single 32-bit counter or as two independent, 16-bit counters in uni-directional or bi-directional mode. As with most timers, the SCTimer/PWM supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

An additional feature contributing to the versatility of the SCTimer/PWM is the concept of “events”. The SCTimer/PWM module supports multiple separate events that can be defined by the user based on some combination of parameters including a match on one of the match registers, and/or a transition on one of the SCTimer/PWM inputs or outputs, the direction of count, and other factors.

Every action that the SCTimer/PWM block can perform occurs in direct response to one of these user-defined events without any software overhead. Any event can be enabled to:

- Start, stop, or halt the counter.
- Limit the counter which means to clear the counter in unidirectional mode or change its direction in bi-directional mode.
- Set, clear, or toggle any SCTimer/PWM output.
- Force a capture of the count value into any capture registers.
- Generate an interrupt or DMA request.

The SCTimer/PWM allows the user to group and filter events, thereby selecting some events to be enabled together while others are disabled in a given context. A group of enabled and disabled events can be described as a state (or a “context”), and multiple states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven as well and can therefore happen without software intervention. Any event can dictate whether to remain in the current state or switch to a new one. By defining these states, the SCTimer/PWM provides the means to periodically alter the entire behavior of the machine based on whatever criteria the use chooses. It is also possible to generate finite state machines in hardware with any desired level of complexity to accomplish complex waveform and timing tasks.

In a simple system, such as a basic timer/counter with capture and match capabilities, there is no need to use more than a single state. All events that could cause the timer to capture the timer value or toggle a match output are enabled at all times while the counter is running. In this case, no events are filtered and the system is described by a single state that does not change. This is the default configuration of the SCTimer/PWM.

In a slightly more complex system, two states could be set up that allow certain events in one state and not in the other. An event enabled in both states can then be used to move from one state to the other and back while filtering out other events in either state. In such a two-state system different waveforms at the SCTimer/PWM output can be created depending on the event history. Changing between states is event-driven and happens without any intervention by the CPU.

For even more advanced applications, up to 32 different states/contexts can be defined (depending on the number of states available on a particular part). If required, the use of states can permit the SCTimer/PWM to serve as finite state machine generator. The ability to perform switching between groups of events provides the SCTimer/PWM the unique capability to be utilized as a highly complex State Machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides an extremely powerful control tool - particularly when the SCTimer/PWM inputs and outputs are connected to other on-chip resources (such as ADC triggers, other timers etc.) in addition to general-purpose I/O.

In addition to events and states, the SCTimer/PWM provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode.
- Selection of any SCTimer/PWM input as a clock source or a clock gate.
- Capability of selecting a “greater-than-or-equal-to” match condition for the purpose of event generation.

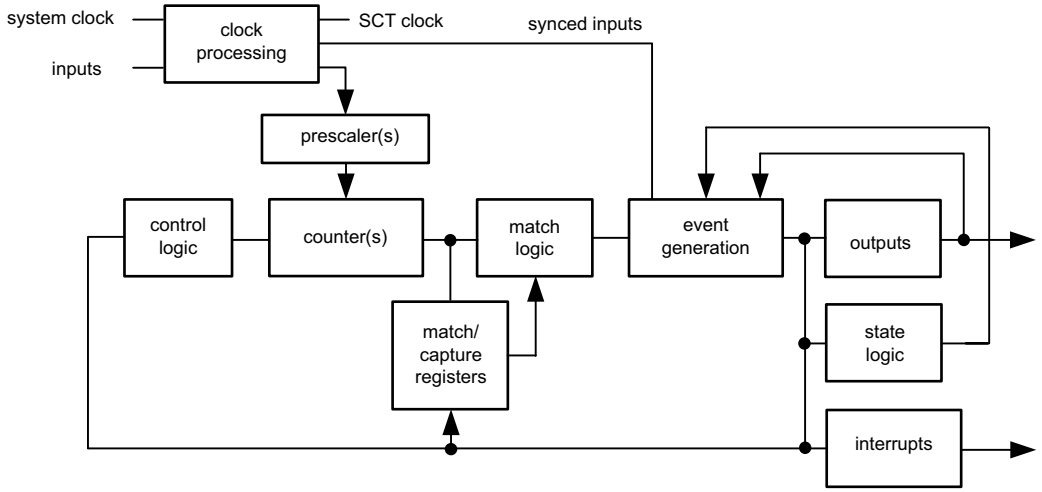


Fig 31. SCTimer/PWM block diagram

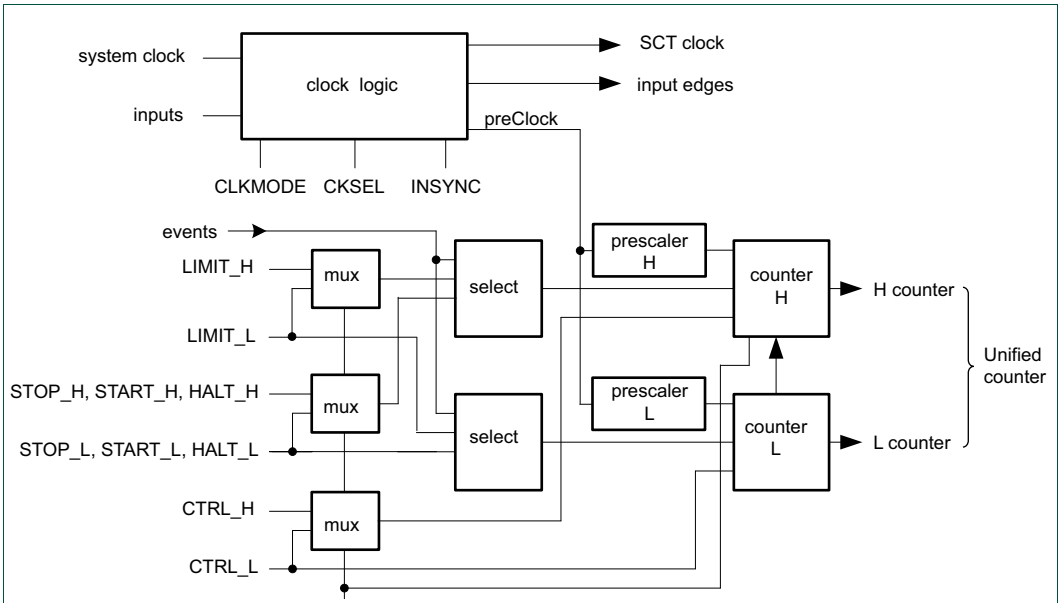


Fig 32. SCTimer/PWM counter and select logic

Remark: In this chapter, the term bus error indicates an SCTimer/PWM response that makes the processor take an exception.

16.6 Register description

The register addresses of the SCTimer/PWM are shown in Table 333. For most of the SCTimer/PWM registers, the register function depends on the setting of certain other register bits:

1. The UNIFY bit in the CONFIG register determines whether the SCTimer/PWM is used as one 32-bit register (for operation as one 32-bit counter/timer) or as two 16-bit counter/timers named L and H. The setting of the UNIFY bit is reflected in the register map:
   - UNIFY = 1: Only one register is used (for operation as one 32-bit counter/timer).
   - UNIFY = 0: Access the L and H registers by a 32-bit read or write operation or can be read or written to individually (for operation as two 16-bit counter/timers).

   Typically, the UNIFY bit is configured by writing to the CONFIG register before any other registers are accessed.
2. The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:
   - REGMODEn = 0: Registers operate as match and reload registers.
   - REGMODEn = 1: Registers operate as capture and capture control registers.

Table 333. Register overview: SCTimer/PWM (base address 0x4008 5000)

Name	Access	Offset	Description	Reset value	Section
CONFIG	R/W	0x000	Configuration register	0x0001 FE00	16.6.2
CTRL	R/W	0x004	Control register	0x0004 0004	16.6.3
CTRL_L	R/W	0x004	Control register low counter 16-bit	0x0000 0004	16.6.3
CTRL_H	R/W	0x006	Control register high counter 16-bit	0x0000 0004	16.6.3
LIMIT	R/W	0x008	Limit event select register	0x0000 0000	16.6.4
LIMIT_L	R/W	0x008	Limit event select register low counter 16-bit	0x0000 0000	16.6.4
LIMIT_H	R/W	0x00A	Limit event select register high counter 16-bit	0x0000 0000	16.6.4
HALT	R/W	0x00C	Halt event select register	0x0000 0000	16.6.5
HALT_L	R/W	0x00C	Halt event select register low counter 16-bit	0x0000 0000	16.6.5
HALT_H	R/W	0x00E	Halt event select register high counter 16-bit	0x0000 0000	16.6.5
STOP	R/W	0x010	Stop event select register	0x0000 0000	16.6.6
STOP_L	R/W	0x010	Stop event select register low counter 16-bit	0x0000 0000	16.6.6
STOP_H	R/W	0x012	Stop event select register high counter 16-bit	0x0000 0000	16.6.6
START	R/W	0x014	Start event select register	0x0000 0000	16.6.7
START_L	R/W	0x014	Start event select register low counter 16-bit	0x0000 0000	16.6.7
START_H	R/W	0x016	Start event select register high counter 16-bit	0x0000 0000	16.6.7
COUNT	R/W	0x040	Counter register	0x0000 0000	16.6.8
COUNT_L	R/W	0x040	Counter register low counter 16-bit	0x0000 0000	16.6.8
COUNT_H	R/W	0x042	Counter register high counter 16-bit	0x0000 0000	16.6.8
STATE	R/W	0x044	State register	0x0000 0000	16.6.9
STATE_L	R/W	0x044	State register low counter 16-bit	0x0000 0000	16.6.9
STATE_H	R/W	0x046	State register high counter 16-bit	0x0000 0000	16.6.9
INPUT	RO	0x048	Input register	0x0000 0000	16.6.10
REGMODE	R/W	0x04C	Match/capture mode register	0x0000 0000	16.6.11
REGMODE_L	R/W	0x04C	Match/capture mode register low counter 16-bit	0x0000 0000	16.6.11
REGMODE_H	R/W	0x04E	Match/capture registers mode register high counter 16-bit	0x0000 0000	16.6.11
OUTPUT	R/W	0x050	Output register	0x0000 0000	16.6.12
OUTPUTDIRCTRL	R/W	0x054	Output counter direction control register	0x0000 0000	16.6.13

Table 333. Register overview: SCTimer/PWM (base address 0x4008 5000) *…continued*

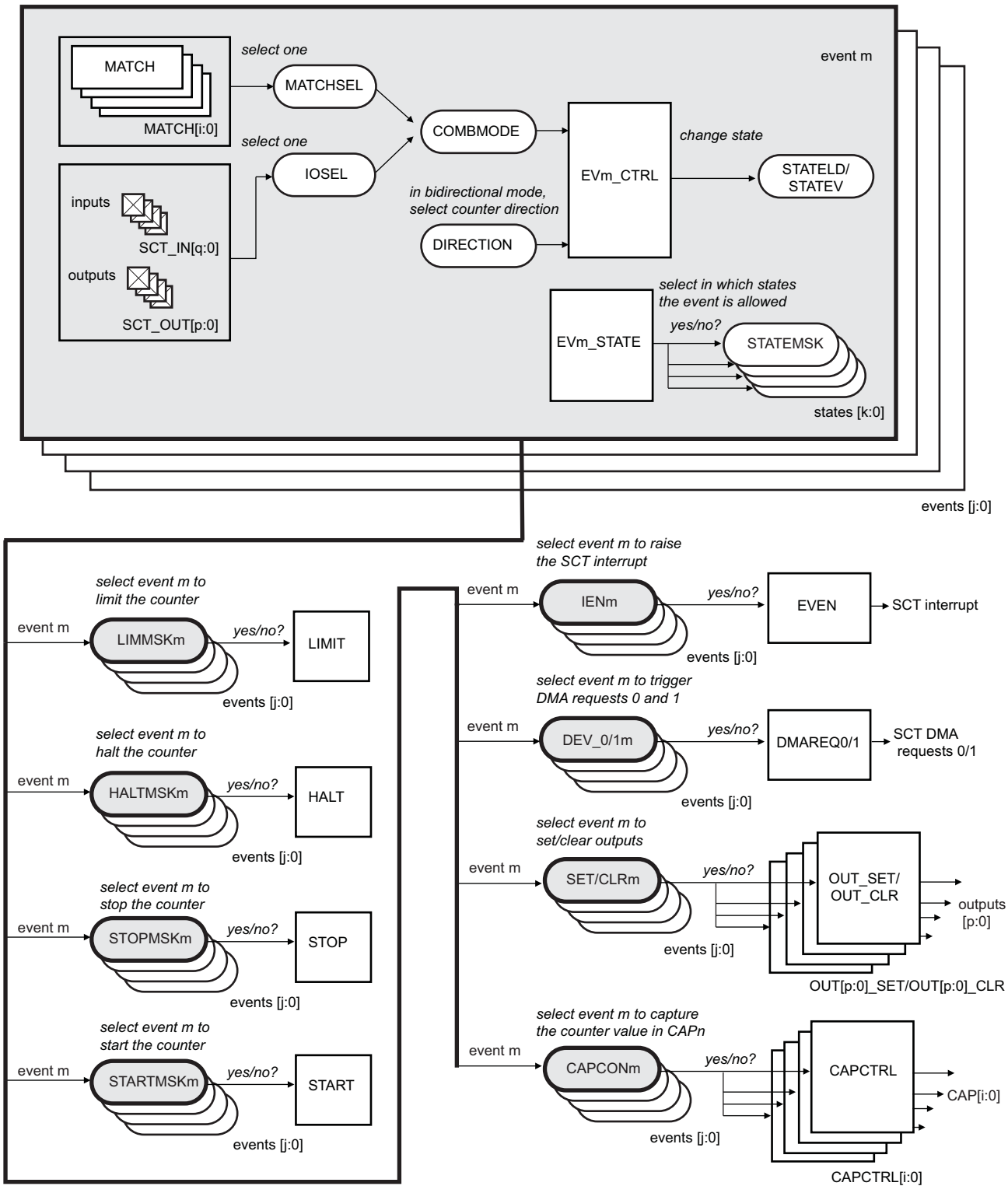
Name	Access	Offset	Description	Reset value	Section
RES	R/W	0x058	Conflict resolution register	0x0000 0000	16.6.14
DMAREQ0	R/W	0x05C	DMA request 0 register	0x0000 0000	16.6.15
DMAREQ1	R/W	0x060	DMA request 1 register	0x0000 0000	16.6.15
EVEN	R/W	0x0F0	Event interrupt enable register	0x0000 0000	16.6.16
EVFLAG	R/W	0x0F4	Event flag register	0x0000 0000	16.6.17
CONEN	R/W	0x0F8	Conflict interrupt enable register	0x0000 0000	16.6.18
CONFLAG	R/W	0x0FC	Conflict flag register	0x0000 0000	16.6.19
MATCH0 to MATCH9	R/W	0x100 to 0x124	Match value register of match channels 0 to 9; REGMODE0 to REGMODE9 = 0	0x0000 0000	16.6.20
MATCH0_L to MATCH9_L	R/W	0x100 to 0x124	Match value register of match channels 0 to 9; low counter 16-bit; REGMODE0_L to REGMODE9_L = 0	0x0000 0000	16.6.20
MATCH0_H to MATCH9_H	R/W	0x102 to 0x126	Match value register of match channels 0 to 9; high counter 16-bit; REGMODE0_H to REGMODE9_H = 0	0x0000 0000	16.6.20
CAP0 to CAP9	R/W	0x100 to 0x124	Capture register of capture channel 0 to 9; REGMODE0 to REGMODE9 = 1	0x0000 0000	16.6.21
CAP0_L to CAP9_L	R/W	0x100 to 0x124	Capture register of capture channel 0 to 9; low counter 16-bit; REGMODE0_L to REGMODE9_L = 1	0x0000 0000	16.6.21
CAP0_H to CAP9_H	R/W	0x102 to 0x126	Capture register of capture channel 0 to 9; high counter 16-bit; REGMODE0_H to REGMODE9_H = 1	0x0000 0000	16.6.21
MATCHREL0 to MATCHREL9	R/W	0x200 to 0x224	Match reload value register 0 to 9; REGMODE0 = 0 to REGMODE9 = 0	0x0000 0000	16.6.22
MATCHREL0_L to MATCHREL9_L	R/W	0x200 to 0x224	Match reload value register 0 to 9; low counter 16-bit; REGMODE0_L = 0 to REGMODE9_L = 0	0x0000 0000	16.6.22
MATCHREL0_H to MATCHREL9_H	R/W	0x202 to 0x226	Match reload value register 0 to 9; high counter 16-bit; REGMODE0_H = 0 to REGMODE9_H = 0	0x0000 0000	16.6.22
CAPCTRL0 to CAPCTRL9	R/W	0x200 to 0x224	Capture control register 0 to 9; REGMODE0 = 1 to REGMODE9 = 1	0x0000 0000	16.6.23
CAPCTRL0_L to CAPCTRL9_L	R/W	0x200 to 0x224	Capture control register 0 to 9; low counter 16-bit; REGMODE0_L = 1 to REGMODE9_L = 1	0x0000 0000	16.6.23
CAPCTRL0_H to CAPCTRL9_H	R/W	0x202 to 0x226	Capture control register 0 to 9; high counter 16-bit; REGMODE0 = 1 to REGMODE9 = 1	0x0000 0000	16.6.23
EV0_STATE	R/W	0x300	Event state register 0	0x0000 0000	16.6.24
EV0_CTRL	R/W	0x304	Event control register 0	0x0000 0000	16.6.25
EV1_STATE	R/W	0x308	Event state register 1	0x0000 0000	16.6.24
EV1_CTRL	R/W	0x30C	Event control register 1	0x0000 0000	16.6.25
EV2_STATE	R/W	0x310	Event state register 2	0x0000 0000	16.6.24
EV2_CTRL	R/W	0x314	Event control register 2	0x0000 0000	16.6.25
EV3_STATE	R/W	0x318	Event state register 3	0x0000 0000	16.6.24
EV3_CTRL	R/W	0x31C	Event control register 3	0x0000 0000	16.6.25
EV4_STATE	R/W	0x320	Event state register 4	0x0000 0000	16.6.24
EV4_CTRL	R/W	0x324	Event control register4	0x0000 0000	16.6.25
EV5_STATE	R/W	0x328	Event state register 5	0x0000 0000	16.6.24
EV5_CTRL	R/W	0x32C	Event control register 5	0x0000 0000	16.6.25
EV6_STATE	R/W	0x330	Event state register 6	0x0000 0000	16.6.24

Table 333. Register overview: SCTimer/PWM (base address 0x4008 5000) *…continued*

Name	Access	Offset	Description	Reset value	Section
EV6_CTRL	R/W	0x334	Event control register 6	0x0000 0000	16.6.25
EV7_STATE	R/W	0x338	Event state register 7	0x0000 0000	16.6.24
EV7_CTRL	R/W	0x33C	Event control register 7	0x0000 0000	16.6.25
EV8_STATE	R/W	0x340	Event state register 8	0x0000 0000	16.6.24
EV8_CTRL	R/W	0x344	Event control register 8	0x0000 0000	16.6.25
EV9_STATE	R/W	0x348	Event state register 9	0x0000 0000	16.6.24
EV9_CTRL	R/W	0x34C	Event control register 9	0x0000 0000	16.6.25
OUT0_SET	R/W	0x500	Output 0 set register	0x0000 0000	16.6.26
OUT0_CLR	R/W	0x504	Output 0 clear register	0x0000 0000	16.6.27
OUT1_SET	R/W	0x508	Output 1 set register	0x0000 0000	16.6.26
OUT1_CLR	R/W	0x50C	Output 1 clear register	0x0000 0000	16.6.27
OUT2_SET	R/W	0x510	Output 2 set register	0x0000 0000	16.6.26
OUT2_CLR	R/W	0x514	Output 2 clear register	0x0000 0000	16.6.27
OUT3_SET	R/W	0x518	Output 3 set register	0x0000 0000	16.6.26
OUT3_CLR	R/W	0x51C	Output 3 clear register	0x0000 0000	16.6.27
OUT4_SET	R/W	0x520	Output 4 set register	0x0000 0000	16.6.26
OUT4_CLR	R/W	0x524	Output 4 clear register	0x0000 0000	16.6.27
OUT5_SET	R/W	0x528	Output 5 set register	0x0000 0000	16.6.26
OUT5_CLR	R/W	0x52C	Output 5 clear register	0x0000 0000	16.6.27
OUT6_SET	R/W	0x530	Output 6 set register	0x0000 0000	16.6.26
OUT6_CLR	R/W	0x534	Output 6 clear register	0x0000 0000	16.6.27
OUT7_SET	R/W	0x538	Output 7 set register	0x0000 0000	16.6.26
OUT7_CLR	R/W	0x53C	Output 7 clear register	0x0000 0000	16.6.27

16.6.1 Register functional grouping

Most SCTimer/PWM registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. Figure 33 shows the registers and register bits that need to be configured for each event.



Note: in this figure, letters are used to represent the maximum quantity of certain SCTimer/PWM features as noted below.

i = match/captures, j = events, k = states, p = outputs, q = inputs

Fig 33. SCTimer/PWM event configuration and selection registers

16.6.1.1 Counter configuration and control registers

The SCTimer/PWM contains two registers for configuring the SCTimer/PWM and monitor and control its operation by software.

- The configuration register (CONFIG) configures the SCTimer/PWM in single, 32-bit counter mode or in dual, 16-bit counter mode, configures the clocking and clock synchronization, and configures automatic limits and the use of reload registers.
- The control register (CTRL) allows to monitor and set the counter direction, and to clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter if in dual-counter mode.

16.6.1.2 Event configuration registers

Each event is associated with two registers:

- One EVn_CTRL register per event to define what triggers the event.
- One EVn_STATE register per event to enable the event.

16.6.1.3 Match and capture registers

The SCTimer/PWM includes a set of registers to store the SCTimer/PWM match or capture values. Each match register is associated with a match reload register which automatically reloads the match register at the beginning of each counter cycle. This register group includes the following registers:

- One REGMODE register per match/capture register to configure each match/capture register for either storing a match value or a capture value.
- A set of match/capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter value.
- One reload register for each match register.

16.6.1.4 Event select registers for the counter operations

This group contains the registers that select the events which affect the counter. Counter actions are limit, halt, and start or stop and apply to the unified counter or to the two 16-bit counters. Also included is the counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter: HALT
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

16.6.1.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCTimer/PWM output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One OUTn_SET register for each output to select the events which set the output.
- One OUTn_CLR register for each output to select the events which clear the output.
- The conflict resolution register which defines an action when more than one event try to control an output at the same time.
- The conflict flag and conflict interrupt enable registers that monitor interrupts arising from output set and clear conflicts.
- The output direction control register that interchanges the set and clear output operation caused by an event in bi-directional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state a given output n.

In the dual-counter mode, the events can be selected independently for each output.

16.6.1.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAP registers. Each capture register m has one associated CAPCTRLm register which in turn selects the events to capture the counter value.

16.6.1.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can trigger a DMA request.

The DMAREQn register contain one bit for each event that selects whether this event triggers a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

16.6.1.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that when they occur request an interrupt.

- The event flag register provides one flag for each event that is set when the event occurs.
- The event flag interrupt enable register provides one bit for each event to be enabled for the SCTimer/PWM interrupt.

16.6.1.9 Registers for controlling SCTimer/PWM inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCTimer/PWM:

- The SCTimer/PWM input register to read the state of any of the SCTimer/PWM inputs.

- The SCTimer/PWM output register to set or clear any of the SCTimer/PWM outputs or to read the state of the outputs.

16.6.2 SCTimer/PWM configuration register

This register configures the overall operation of the SCTimer/PWM. Write to this register before any other registers. Only word-writes are permitted to this register. Attempting to write a half-word value results in a bus error.

Table 334. SCTimer/PWM configuration register (CONFIG, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCTimer/PWM operation	0
		0	The SCTimer/PWM operates as two 16-bit counters named COUNTER_L and COUNTER_H.	
		1	The SCTimer/PWM operates as a unified 32-bit counter.	
2:1	CLKMODE		SCTimer/PWM clock mode	0
		0x0	System Clock Mode. The system clock clocks the entire SCTimer/PWMSCTimer/PWM module including the counter(s) and counter prescalers.	
		0x1	Sampled System Clock Mode. The system clock clocks the SCTimer/PWM module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode.	
		0x2	SCTimer/PWM Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCTimer/PWM module, including the counters and prescalers, after first being synchronized to the system clock. The minimum width of the positive and negative phases of the clock input must each be greater than one full period of the bus/system clock.	
		0x3	Asynchronous Mode. The entire SCTimer/PWM module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCTimer/PWM outputs are switched synchronously to the SCTimer/PWM input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.	

Table 334. SCTimer/PWM configuration register (CONFIG, offset 0x000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
6:3	CKSEL		CSCTimer/PWMlock select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register.	0
		0x0	Rising edges on input 0.	
		0x1	Falling edges on input 0.	
		0x2	Rising edges on input 1.	
		0x3	Falling edges on input 1.	
		0x4	Rising edges on input 2.	
		0x5	Falling edges on input 2.	
		0x6	Rising edges on input 3.	
		0x7	Falling edges on input 3.	
		0x8	Rising edges on input 4.	
		0x9	Falling edges on input 4.	
		0xA	Rising edges on input 5.	
		0xB	Falling edges on input 5.	
		0xC	Rising edges on input 6.	
		0xD	Falling edges on input 6	
		0xE	Rising edges on input 7.	
		0xF	Falling edges on input 7.	
7	NORELOAD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	0
8	NORELOAD_H	-	A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	0
16:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1,..., bit 12 = input 3); all other bits are reserved. A 1 in one of these bits subjects the corresponding input to synchronization to the SCTimer/PWM clock, before it is used to create an event. This synchronization injects a delay of two SCTimer/PWMSCTimer/PWM clocks in the input path. Clearing this bit bypasses synchronization on the corresponding input. This bit may be cleared for faster input response time if both of the following conditions are met (for all Clock Modes): The corresponding input is already synchronous to the SCTimer/PWM clock. The SCTimer/PWM clock frequency does not exceed 100 MHz. Note: The SCTimer/PWM clock is the bus/system clock for CKMODE 0-2 or the selected, asynchronous input clock for CKMODE3. Alternatively, for CKMODE2 only, it is also allowable to bypass synchronization if both of the following conditions are met: The corresponding input is synchronous to the designated CKMODE2 input clock. The CKMODE2 input clock frequency is less than one-third the frequency of the bus/system clock.	1

Table 334. SCTimer/PWM configuration register (CONFIG, offset 0x000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
17	AUTOLIMIT_L	-	This bit applies to the lower registers when the UNIFY bit = 0, and both the higher and lower registers when the UNIFY bit is set. Software can write to set or clear this bit at any time. A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode.	0
18	AUTOLIMIT_H	-	This bit applies to the upper registers when the UNIFY bit = 0, and is not used when the UNIFY bit is set. Software can write to set or clear this bit at any time. A one in this bit will cause a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode.	0
31:19	-	-	Reserved	-

16.6.3 SCTimer/PWM control register

If bit UNIFY = 1 in the CONFIG register, only the _L bits are used.

If bit UNIFY = 0 in the CONFIG register, this register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

All bits in this register can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

Remark: If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 335. SCTimer/PWM control register (CTRL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
0	DOWN_L	-	This read-only bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter is counting up, counter limit occurs, and BIDIR = 1.Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
1	STOP_L	-	When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes.	0
2	HALT_L	-	When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCTimer/PWM in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, only software can clear this bit to restore counter operation. This bit is set on reset.	1
3	CLRCTR_L	-	When the counter is halted (not just stopped), writing a 1 to this bit will clear the L or unified counter. This bit always reads as 0.	0
4	BIDIR_L		L or unified counter direction select	0
		0	Up. The counter counts up to a limit condition, then is cleared to zero.	
		1	Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.	
12:5	PRE_L	-	Specifies the factor by which the SCTimer/PWM clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCTimer/PWM clock divided by PRE_L+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
15:13	-	-	Reserved	-
16	DOWN_H	-	This read-only bit is 1 when the H counter is counting down. Hardware sets this bit when the counter is counting, a counter limit condition occurs, and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
17	STOP_H	-	When this bit is 1 and HALT is 0, the H counter does not, run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0
18	HALT_H	-	When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCTimer/PWM in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, this bit can only be cleared by software to restore counter operation. This bit is set on reset.	1
19	CLRCTR_H	-	When the counter is halted (not just stopped), writing a 1 to this bit will clear the H counter. This bit always reads as 0.	0

Table 335. SCTimer/PWM control register (CTRL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
20	BIDIR_H		Direction select	0
		0	The H counter counts up to its limit condition, then is cleared to zero.	
		1	The H counter counts up to its limit, then counts down to a limit condition or to 0.	
28:21	PRE_H	-	Specifies the factor by which the SCTimer/PWM clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCTimer/PWM clock divided by PRELH+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
31:29	-	-	Reserved	-

16.6.4 SCTimer/PWM limit event select register

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes its direction of count in bi-directional mode and keeps running.To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature (see Table 334).

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 336. SCTimer/PWM limit event select register (LIMIT, offset 0x008) bit description

Bit	Symbol	Description	Reset value
15:0	LIMMSK_L	If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCTimer/PWM.	0
31:16	LIMMSK_H	If bit n is one, event n is used as a couSCTimer/PWMnter limit for the H counter (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCTimer/PWM.	0

16.6.5 SCTimer/PWM halt event select register

The running counter can be disabled (halted) by an event. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

Remark: A HALT condition can only be removed when software clears the HALT bit in the CTRL register (Table 335).

If UNIFY = 1 in the CONFIG register, only the L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers HALT_L and HALT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 337. SCTimer/PWM halt event select register (HALT, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
15:0	HALTMSK_L	If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCTimer/PWM.	0
31:16	HALTMSK_H	If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCTimer/PWM.	0

16.6.6 SCTimer/PWM stop event select register

The running counter can be stopped by an event. When any of the events selected in this register occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

This register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Software can stop and restart the counter by writing to the CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STOPT_L and STOP_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 338. SCTimer/PWM stop event select register (STOP, offset 0x010) bit description

Bit	Symbol	Description	Reset value
15:0	STOPMSK_L	If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCTimer/PWM.	0
31:16	STOPMSK_H	If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCTimer/PWM.	0

16.6.7 SCTimer/PWM start event select register

The stopped counter can be re-started by an event. When any of the events selected in this register occur, counting is restarted from the current counter value.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a START event. When any START event occurs, hardware will clear the STOP bit in the Control Register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers START_L and START_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 339. SCTimer/PWM start event select register (START, offset 0x014) bit description

Bit	Symbol	Description	Reset value
15:0	STARTMSK_L	If bit n is one, event n clears the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCTimer/PWM.	0
31:16	STARTMSK_H	If bit n is one, event n clears the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCTimer/PWM.	0

16.6.8 SCTimer/PWM counter register

If UNIFY = 1 in the CONFIG register, the counter is a unified 32-bit register and both the _L and _H bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Table 340. SCTimer/PWM counter register (COUNT, offset 0x040) bit description

Bit	Symbol	Description	Reset value
15:0	CTR_L	When UNIFY = 0, read or write the 16-bit L counter value. When UNIFY = 1, read or write the lower 16 bits of the 32-bit unified counter.	0
31:16	CTR_H	When UNIFY = 0, read or write the 16-bit H counter value. When UNIFY = 1, read or write the upper 16 bits of the 32-bit unified counter.	0

16.6.9 SCTimer/PWM state register

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

Remark: The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L, STATE_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register).

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/ PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- set and clear outputs
- limit, stop, and start the counter
- cause interrupts and DMA requests
- modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCTimer/PWM configuration. See Section 16.6.24 and 16.6.25 for more about the relationship between states and events.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 341. SCTimer/PWM state register (STATE, offset 0x044) bit description

Bit	Symbol	Description	Reset value
4:0	STATE_L	State variable.	0
15:5	-	Reserved.	-
20:16	STATE_H	State variable.	0
31:21	-	Reserved.	-

16.6.10 SCTimer/PWM input register

Software can read the state of the SCTimer/PWM inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCTimer/PWM clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the INSYNC field in the CONFIG register:
   - If the INSYNC bit is set for the input, the input is triple-synchronized to the SCTimer/PWM clock resulting in a stable signal that is delayed by three SCTimer/PWM clock cycles.
   - If the INSYNC bit is not set, the SIN bit value is identical to the AIN bit value.

Table 342. SCTimer/PWM input register (INPUT, offset 0x048) bit description

Bit	Symbol	Description	Reset value
0	AIN0	Input 0 state. Input 0 state on the last SCTimer/PWM clock edge.	-
1	AIN1	Input 1 state. Input 1 state on the last SCTimer/PWM clock edge.	-
2	AIN2	Input 2 state. Input 2 state on the last SCTimer/PWM clock edge.	-
3	AIN3	Input 3 state. Input 3 state on the last SCTimer/PWM clock edge.	-
15:4	AIN…	Input state for the remainder of inputs implemented in this SCTimer/PWM.	-
16	SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC0.	-
17	SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC1.	-
18	SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC2.	-
19	SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC3.	-
31:20	SIN…	Input state for the remainder of states implemented in this SCTimer/PWM.	-

16.6.11 SCTimer/PWM match/capture mode register

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. In this case, REGMODE_H is not used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. The _L bits/registers control the L match/capture registers, and the _H bits/registers control the H match/capture registers.

The SCTimer/PWM contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register (see Section 16.6.20) or as a Capture register (see Section 16.6.21). Each Match/Capture register has an

accompanying register which functions as a Reload register when the primary register is used as a Match register (see Section 16.6.22) or as a Capture-Control (event select) register when the register is used as a capture register (see Section 16.6.23). REGMODE_H is used only when the UNIFY bit is 0.

Table 343. SCTimer/PWM match/capture mode register (REGMODE, offset 0x04C) bit description

Bit	Symbol	Description	Reset value
15:0	REGMOD_L	Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1, …). The number of bits = number of match/captures supported by this SCTimer/PWM. 0 = register operates as match register. 1 = register operates as capture register.	0
31:16	REGMOD_H	Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17, …). The number of bits = number of match/captures supported by this SCTimer/PWM. 0 = register operates as match registers. 1 = register operates as capture registers.	0

16.6.12 SCTimer/PWM output register

Each SCTimer/PWM output has a corresponding bit in this register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using this register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (HALT bits are set to 1 in the CTRL register).

Software can read this register at any time to sense the state of the outputs.

Table 344. SCTimer/PWM output register (OUTPUT, offset 0x050) bit description

Bit	Symbol	Description	Reset value
15:0	OUT	Writing a 1 to bit n forces the corresponding output HIGH. Writing a 0 forces the corresponding output LOW (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs in this SCTimer/PWM.	0
31:16	-	Reserved	-

16.6.13 SCTimer/PWM bi-directional output control register

For bi-directional mode, this register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output (see Section 16.6.26 and Section 16.6.27). The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Table 345. SCTimer/PWM bidirectional output control register (OUTPUTDIRCTRL, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
3:2	SETCLR1		Set/clear operation on output 1.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
9:8	SETCLR4		Set/clear operation on output 4.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:10	SETCLR…		Set/clear operation controls for the remainder of outputs on this SCTimer/PWM. [1]	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

16.6.14 SCTimer/PWM conflict resolution register

The output conflict resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Table 346. SCTimer/PWM conflict resolution register (RES, offset 0x058) bit description

Bit	Symbol	Value	Description	Reset value
1:0	O0RES		Effect of simultaneous set and clear on output 0.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR0 field).	
		0x3	Toggle output.	

Table 346. SCTimer/PWM conflict resolution register (RES, offset 0x058) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
3:2	O1RES		Effect of simultaneous set and clear on output 1.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR1 field).	
		0x3	Toggle output.	
5:4	O2RES		Effect of simultaneous set and clear on output 2.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output n (or set based on the SETCLR2 field).	
		0x3	Toggle output.	
7:6	O3RES		Effect of simultaneous set and clear on output 3.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR3 field).	
		0x3	Toggle output.	
9:8	O4RES		Effect of simultaneous set and clear on output 4.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR4 field).	
		0x3	Toggle output.	
31:10	O…RES		Resolution controls for the remainder of outputs on this SCTimer/PWM. [1]	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

16.6.15 SCTimer/PWM DMA request 0 and 1 registers

The SCTimer/PWM includes two DMA request outputs. These registers enable the DMA requests to be triggered when a particular event occurs or when counter Match registers are loaded from its Reload registers. The DMA request registers are word-write only. Attempting to write a half-word value to these registers result in a bus error.

Event-triggered DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCTimer/PWM.

Table 347. SCTimer/PWM DMA 0 request register (DMAREQ0, offset 0x05C) bit description

Bit	Symbol	Description	Reset value
15:0	DEV_0	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events in this SCTimer/PWM.	0
29:16	-	Reserved	-
30	DRL0	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/Unified registers from the RELOAD_L/Unified registers.	0
31	DRQ0	This read-only bit indicates the state of DMA Request 0. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	0

Table 348. SCTimer/PWM DMA 1 request register (DMAREQ1, offset 0x060) bit description

Bit	Symbol	Description	Reset value
15:0	DEV_1	If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events in this SCTimer/PWM.	0
29:16	-	Reserved	-
30	DRL1	A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.	0
31	DRQ1	This read-only bit indicates the state of DMA Request 1. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	0

16.6.16 SCTimer/PWM event interrupt enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCTimer/PWM event flag register (Section 16.6.17) is also set.

Table 349. SCTimer/PWM event interrupt enable register (EVEN, offset 0x0F0) bit description

Bit	Symbol	Description	Reset value
15:0	IEN	The SCTimer/PWM requests an interrupt when bit n of this register and the event flag register are both one (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCTimer/PWM.	0
31:16	-	Reserved	-

16.6.17 SCTimer/PWM event flag register

This register records events. Writing ones to this register clears the corresponding flags and negates the SCTimer/PWM interrupt request if all enabled flag register bits are zero.

Table 350. SCTimer/PWM event flag register (EVFLAG, offset 0x0F4) bit description

Bit	Symbol	Description	Reset value
15:0	FLAG	Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCTimer/PWM.	0
31:16	-	Reserved	-

16.6.18 SCTimer/PWM conflict interrupt enable register

This register enables the no-change conflict events specified in the SCTimer/PWM conflict resolution register to generate an interrupt request.

Table 351. SCTimer/PWM conflict interrupt enable register (CONEN, offset 0x0F8) bit description

Bit	Symbol	Description	Reset value
15:0	NCEN	The SCTimer/PWM requests an interrupt when bit n of this register and the SCTimer/PWM conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCTimer/PWM.	0
31:16	-	Reserved	

16.6.19 SCTimer/PWM conflict flag register

This register records a no-change conflict occurrence and provides details of a bus error. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCTimer/PWM interrupt request if all enabled Flag bits are zero.

Table 352. SCTimer/PWM conflict flag register (CONFLAG, offset 0x0FC) bit description

Bit	Symbol	Description	Reset value
15:0	NCFLAG	Bit n is one if a no-change conflict event occurred on output n since reset or a 1 was last written to this bit (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCTimer/PWM.	0
29:16	-	Reserved.	-
30	BUSERRL	The most recent bus error from this SCTimer/PWM involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.	0
31	BUSERRH	The most recent bus error from this SCTimer/PWM involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.	0

16.6.20 SCTimer/PWM match registers 0 to 9 (REGMODEn bit = 0)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCH_L, MATCH_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Match events occur in the SCTimer/PWM clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in Section 16.6.4, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Table 353. SCTimer/PWM match registers 0 to 9 (MATCH[0:9], offset 0x100 (MATCH0) to 0x124 (MATCH9)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	MATCHn_L	When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.	0
31:16	MATCHn_H	When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.	0

16.6.21 SCTimer/PWM capture registers 0 to 9 (REGMODEn bit = 1)

These registers allow software to record the counter values upon occurrence of the events selected by the corresponding Capture Control registers occurred.

Table 354. SCTimer/PWM capture registers 0 to 9 (CAP[0:9], offset 0x100 (CAP0) to 0x124 (CAP9)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPn_L	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value at which this register was last captured.	0
31:16	CAPn_H	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.	0

16.6.22 SCTimer/PWM match reload registers 0 to 9 (REGMODEn bit = 0)

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is:

- when BIDIR = 0 and the counter is cleared to zero upon reaching its limit condition.
- when BIDIR = 1 and the counter counts down to 0.

In either case, reloading does not occur if the corresponding NORELOAD bit is set in the CFG register.

Table 355. SCTimer/PWM match reload registers 0 to 9 (MATCHREL[0:9], offset 0x200 (MATCHREL0) to 0x224 (MATCHREL9)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOADn_L	When UNIFY = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When UNIFY = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.	0
31:16	RELOADn_H	When UNIFY = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.	0

16.6.23 SCTimer/PWM capture control registers 0 to 9 (REGMODEn bit = 1)

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

The capture registers can be loaded with the current counter value when any of the specified events occur.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause the load of corresponding Capture register from the counter.

Table 356. SCTimer/PWM capture control registers 0 to 9 (CAPCTRL[0:9], offset 0x200 (CAPCTRL0) to 0x224 (CAPCTRL9)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPCONn_L	If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of match/captures supported by this SCTimer/PWM.	0
31:16	CAPCONn_H	If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of match/captures supported by this SCTimer/PWM.	0

16.6.24 SCTimer/PWM event enable registers 0 to 9

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRL register has one associated event enable register that can enable or disable the event for each available state.

An event n is completely disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state. Unused events should be disabled in this manner.

In simple applications that do not use states, writing 0x01 (or any other value with a 1 in bit 0) will enable the event. Since the state doesn't change (that is, the state variable always remains at its reset value of 0), setting bit 0 permanently enables this event. Conversely, clearing bit 0 will disable the event.

Table 357. SCTimer/PWM event state mask registers 0 to 9 (EV[0:9]_STATE, offsets 0x300 (EV0_STATE) to 0x348 (EV9_STATE)) bit description

Bit	Symbol	Description	Reset value
15:0	STATEMSKn	If bit m is one, event n is enabled to occur whenever the state = m. When the UNIFY bit is 0, the pertinent state is the one associated with the counter selected by the HEVENT bit in the Event Control Register. (n = event number, m = state number; state 0 = bit 0, state 1= bit 1, …). The number of bits = number of states in this SCTimer/PWM.	0
31:16	-	Reserved.	-

16.6.25 SCTimer/PWM event control registers 0 to 9

This register defines the conditions for an event to occur based on the counter values or input and output states.Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCTimer/PWM or the counter is halted. To block a particular event from occurring, use the EV_STATE register. To block all events for a given counter, set the HALT bit in the CTRL register or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register. In bi-directional mode, events can also be enabled based on the direction of count.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Table 358. SCTimer/PWM event control register 0 to 9 (EV[0:9]_CTRL, offset 0x304 (EV0_CTRL) to 0x34C (EV9_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	MATCHSEL	-	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.	0
4	HEVENT		Select L/H counter. Do not set this bit if UNIFY = 1.	0
		0	Selects the L state and the L match register selected by MATCHSEL.	
		1	Selects the H state and the H match register selected by MATCHSEL.	
5	OUTSEL		Input/output select	0
		0	Selects the inputs selected by IOSEL.	
		1	Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal number associated with this event (if any). Do not select an input in this register if CKMODE is 1x. In this case the clock input is an implicit ingredient of every event.	0
11:10	IOCOND		Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCTimer/PWM clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCTimer/PWM clock period .	0
		0x0	LOW	
		0x1	Rise	
		0x2	Fall	
		0x3	HIGH	
13:12	COMBMODE		Selects how the specified match and I/O condition are used and combined.	0
		0x0	OR. The event occurs when either the specified match or I/O condition occurs.	
		0x1	MATCH. Uses the specified match only.	
		0x2	IO. Uses the specified I/O condition only.	
		0x3	AND. The event occurs when the specified match and I/O condition occur simultaneously.	
14	STATELD		This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state.	0
		0	STATEV value is added into STATE (the carry-out is ignored).	
		1	STATEV value is loaded into STATE.	
19:15	STATEV	-	This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.	0
20	MATCHMEM	-	If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.	0

Table 358. SCTimer/PWM event control register 0 to 9 (EV[0:9]_CTRL, offset 0x304 (EV0_CTRL) to 0x34C (EV9_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
22:21	DIRECTION		Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCTimer/PWM ignores this field. Value 0x3 is reserved.	0
		0x0	Direction independent. This event is triggered regardless of the count direction.	
		0x1	Counting up. This event is triggered only during up-counting when BIDIR = 1.	
		0x2	Counting down. This event is triggered only during down-counting when BIDIR = 1.	
31:23	-	-	Reserved	-

16.6.26 SCTimer/PWM output set registers 0 to 9

Each SCTimer/PWM output can be set upon the occurrence of one or more specified events.

There is one output set register for each SCTimer/PWM output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.).

Note that it is possible to reverse the action specified by “SET” and “CLR” when counting down in bi-directional mode depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the creation of the actual event(s) that sets an output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 359. SCTimer/PWM output set register (OUT[0:9]_SET, offset 0x500 (OUT0_SET) to 0x548 (OUT9_SET) bit description

Bit	Symbol	Description	Reset value
15:0	SET	A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) output 0 = bit 0, output 1 = bit 1, … The number of bits = number of events supported by this SCTimer/PWM. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0
31:16	-	Reserved	-

16.6.27 SCTimer/PWM output clear registers 0 to 9

Each SCTimer/PWM output can be cleared upon the occurrence of one or more specified events.

There is one register for each SCTimer/PWM output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (for example, bit 0 with event 0).

Note that it is possible to reverse the action specified by “SET” and “CLR” when counting down in bi-directional mode depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the creation of the actual event(s) that sets an output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 360. SCTimer/PWM output clear register (OUT[0:9]_CLR, offset 0x504 (OUT0_CLR) to 0x54C (OUT9_CLR)) bit description

Bit	Symbol	Description	Reset value
15:0	CLR	A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, … The number of bits = number of events in this SCTimer/PWM. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0
31:16	-	Reserved	-

16.7 Functional description

16.7.1 Match logic

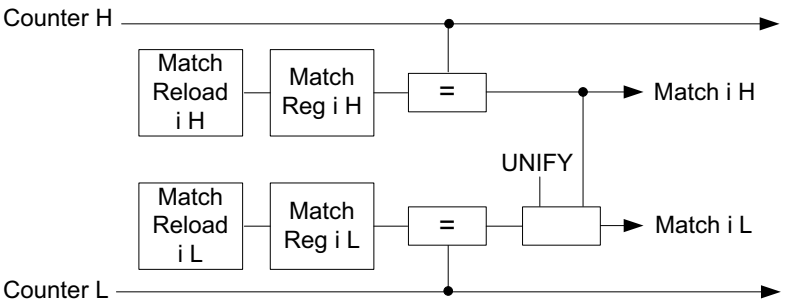


Fig 34. Match logic

16.7.2 Capture logic

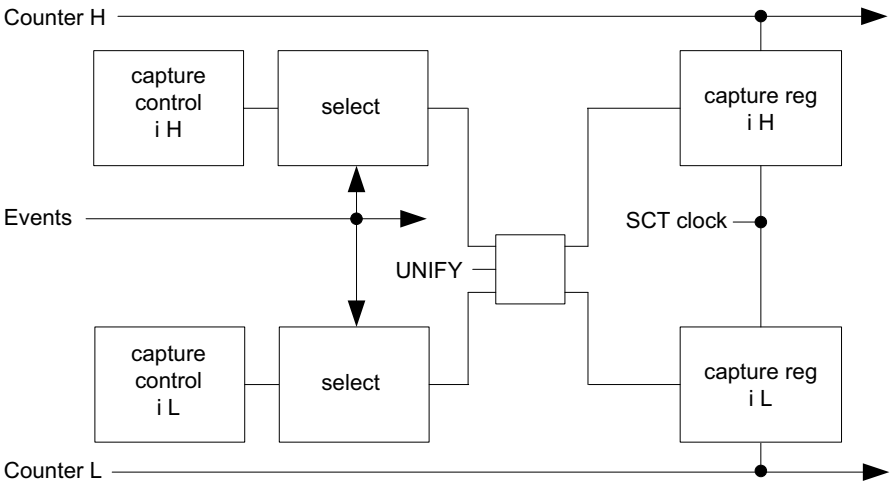


Fig 35. Capture logic

16.7.3 Event selection

State variables allow control of the SCTimer/PWM across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

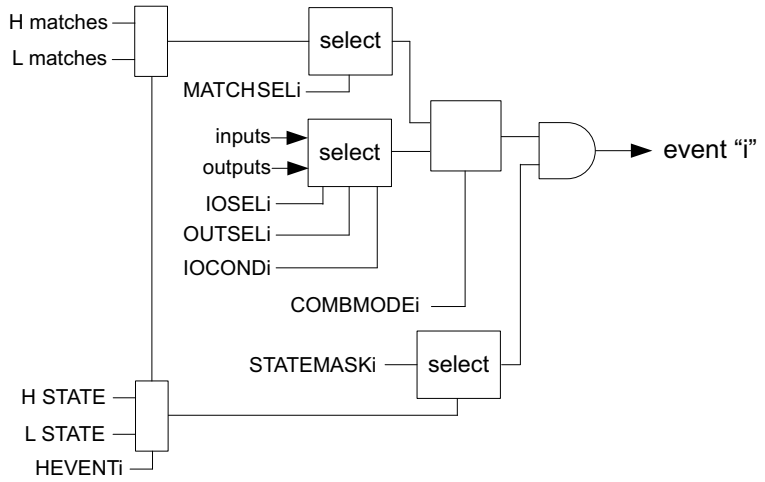


Fig 36. Event selection

16.7.4 Output generation

Figure 37 shows one output slice of the SCTimer/PWM.

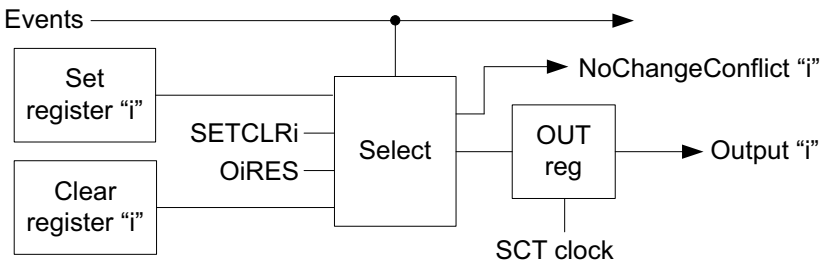


Fig 37. Output slice i

16.7.5 State logic

The SCTimer/PWM can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCTimer/PWM, the SCTimer/PWM can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event’s EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, the SCTimer/PWM enters a locked state in which all further events are ignored while the counter is still running. Software must intervene to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCTimer/PWM into a locked state occurs. Later, while the SCTimer/PWM is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number

If the SCTimer/PWM registers an event that forces an abort, putting the SCTimer/PWM in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

16.7.6 Interrupt generation

The SCTimer/PWM generates one interrupt to the NVIC.

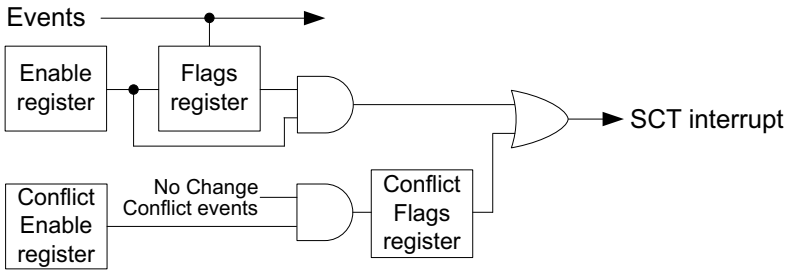


Fig 38. SCTimer/PWM interrupt generation

16.7.7 Clearing the prescaler

When enabled by a non-zero PRE field in the Control register, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset.
- Software writing to the counter register.
- Software writing a 1 to the CLRCTR bit in the control register.
- an event selected by a 1 in the counter limit register when BIDIR = 0.

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described Section 16.7.8.

A limit event when BIDIR is 1 does not clear the prescaler. Rather it clears the DOWN bit in the Control register, and decrements the counter on the same clock if the counter is enabled in that clock.

16.7.8 Match versus I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCTimer/PWM clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CLKSEL field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCTimer/PWM clock when its counter HALT bit is 0. In general, a Match component of an event can only occur in an SCTimer/PWM clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

Table 361 shows when the various kinds of events can occur.

Table 361. Event conditions

COMBMODE	IOMODE	Event can occur on clock:
IO	Any	Event can occur whenever HALT = 0 (type A).
MATCH	Any	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (type C).
OR	Any	From the IO component: Event can occur whenever HALT = 0 (A). From the match component: Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	LOW or HIGH	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	RISE or FALL	Event can occur whenever HALT = 0 (A).

16.7.9 SCTimer/PWM operation

In its simplest, single-state configuration, the SCTimer/PWM operates as an event controlled unidirectional or bidirectional counter. Events can be configured to occur on counter match events, an input or output level, transitions on an input or output pin, or a combination of match and input/output behavior. In response to an event, the SCTimer/PWM output or outputs can transition, or the SCTimer/PWM can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCTimer/PWM.

An action or multiple actions of the SCTimer/PWM uniquely define an event. A state is defined by which events are enabled to trigger an SCTimer/PWM action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCTimer/PWM can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCTimer/PWM. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCTimer/PWM outputs and/or interrupts.

Once configured, the SCTimer/PWM can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCTimer/PWM, see Section 16.7.10.
- To start, run, and stop the SCTimer/PWM, see Section 16.7.11.

- To configure the SCTimer/PWM as simple event controlled counter/timer, see Section 16.7.12.

16.7.10 Configure the SCTimer/PWM

To set up the SCTimer/PWM for multiple events and states, perform the following configuration steps:

16.7.10.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCTimer/PWM clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

16.7.10.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (not more than what is available on this device):
   - In the REGMODE register, select for each of the match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
   - Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
   - Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

16.7.10.3 Configure events and event responses

1. Define when each event can occur in the following way in the EVn_CTRL registers (up to 6, one register per event):
   - Select whether the event occurs on an input or output changing, on an input or output level, a match condition of the counter, or a combination of match and input/output conditions in field COMBMODE.
   - For a match condition:

     Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field MATCHSEL.

     If using L and H counters, define whether the event occurs on matching the L or the H counter in field HEVENT.
   - For an SCTimer/PWM input or output level or transition:

     Select the input number or the output number that is associated with this event in fields IOSEL and OUTSEL.

     Define how the selected input or output triggers the event (edge or level sensitive) in field IOCOND.
2. Define what the effect of each event is on the SCTimer/PWM outputs in the OUTn_SET or OUTn_CLR registers (up to the maximum number of outputs on this device, one register per output):

- For each SCTimer/PWM output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.

3. Define how each event affects the counter:
   - Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.

     When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.

     When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   - Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
   - Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
   - Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCTimer/PWM interrupt:
   - Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCTimer/PWM interrupt.

16.7.10.4 Configure multiple states

1. In the EVn_STATE register for each event (up to the maximum number of events on this device, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:

   In the EVn_CTRL registers (up to the maximum number of events on this device, one register per event), set the new state value in the STATEV field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field STATELD.

   Remark: If there are higher numbered events in the current state, this event cannot change the state.

   If the STATEV and STATELD values are set to zero, the state does not change.

16.7.10.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

16.7.11 Run the SCTimer/PWM

1. Configure the SCTimer/PWM (see Section 16.7.10 “Configure the SCTimer/PWM”).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCTimer/PWM, write to the CTRL register:
   - Clear the counters.
   - Clear or set the STOP_L and/or STOP_H bits.

     Remark: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCTimer/PWM waits instead for an event to occur that is configured to start the counter.
   - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
   - Select the prescale factor for the counter clock (CTRL register).
   - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters by software at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
   - When the counters are stopped, both an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
   - When the counter are halted, only a software write to clear the HALT bit can start the counter again. No events can occur.
   - When the counters are halted, software can set any SCTimer/PWM output HIGH or LOW directly by writing to the OUT register.

The current state can be read at any time by reading the STATE register.

To change the current state by software (that is independently of any event occurring), set the HALT bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter is halted (the HALT_L and/or HALT_H bits are set) and no events can occur.

16.7.12 Configure the SCTimer/PWM without using states

The SCTimer/PWM can be used as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCTimer/PWM without states, configure the SCTimer/PWM as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the STATELD and STATEV fields in the EVCTRL registers for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.

  In effect, the event is allowed to occur in a single state which never changes while the counter is running.

16.7.13 SCTimer/PWM PWM Example

Figure 39 shows a simple application of the SCTimer/PWM using two sets of match events (EV0/1 and EV3/4) to set/clear SCTimer/PWM output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0. The SCTimer/PWM input 0 is monitored: If input0 is found LOW by the next time the timer is reset(EV2), the state is changed to state 1, and EV3/4 are enabled, which create the same output but triggered by different match values. If input 0 is found HIGH by the next time the timer is reset, the associated event (EV5) causes the state to change back to state 0 where the events EV0 and EV1 are enabled.

The example uses the following SCTimer/PWM configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

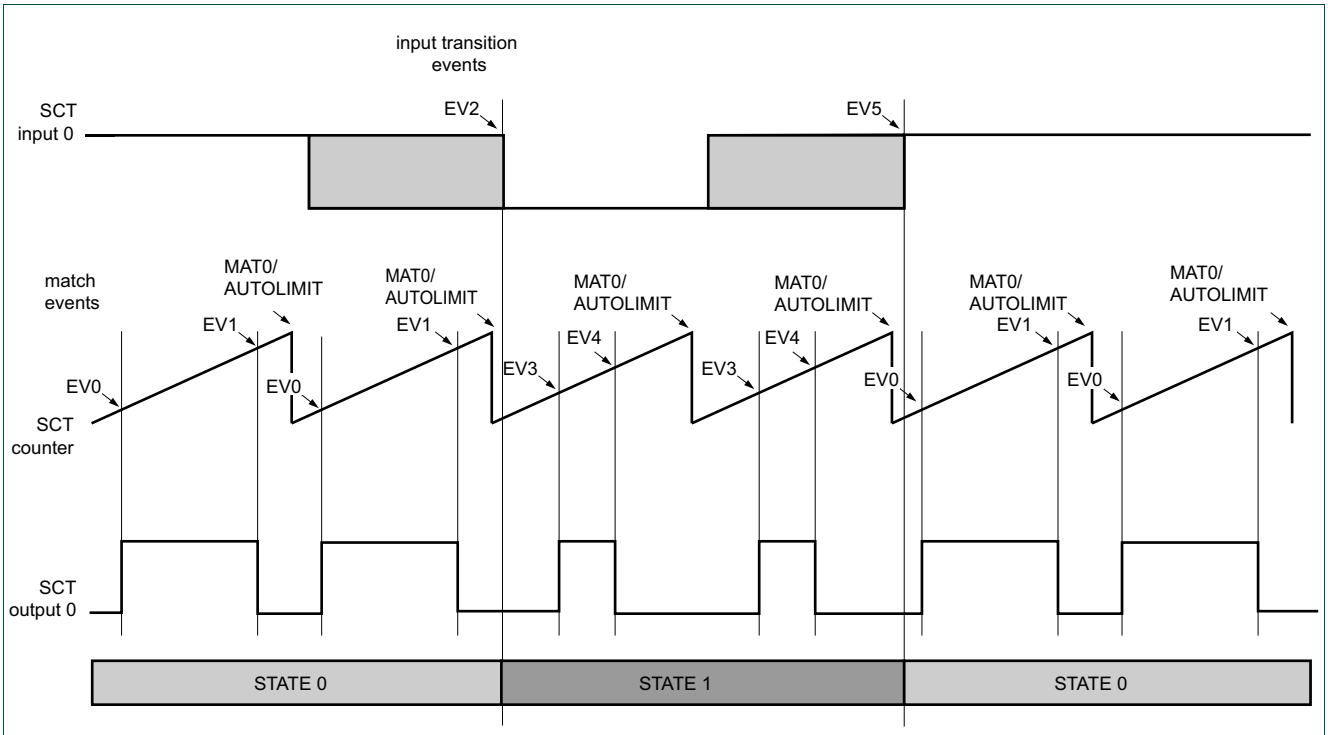


Fig 39. SCTimer/PWM configuration example

This application of the SCTimer/PWM uses the following configuration (all register values not listed in Table 362 are set to their default values):

Table 362. SCTimer/PWM configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT = 1.)
	CTRL	Uses unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0,1, 2, 3, 4 to 0. This is the default.
Define match values	MATCH 0/1/2/3/4	Set a match value MATCH0/1/2/3/4_L in each register. The match 0 register serves as an automatic limit event that resets the counter. without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register.
Define match reload values	MATCHREL 0/1/2/3/4	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set COMBMODE = 0x1. Event 0 uses match condition only. • Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set COMBMODE = 0x1. Event 1 uses match condition only. • Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1.
Define when event 2 occurs	EV2_CTRL	• Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x0. Input 0 is LOW. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set COMBMODE = 0x1. Event 3 uses match condition only. • Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.
Define when event 4 occurs	EV4_CTRL	• Set COMBMODE = 0x1. Event 4 uses match condition only. • Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x3. Input 0 is HIGH. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states in which event 0 is enabled	EV0_STATE	Set STATEMSK0 bit 0 to 1. Set all other bits to 0. Event 0 is enabled in state 0.
Configure states in which event 1 is enabled	EV1_STATE	Set STATEMSK1 bit 0 to 1. Set all other bits to 0. Event 1 is enabled in state 0.
Configure states in which event 2 is enabled	EV2_STATE	Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0.

Table 362. SCTimer/PWM configuration example

Configuration	Registers	Setting
Configure states in which event 3 is enabled	EV3_STATE	Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1.
Configure states in which event 4 is enabled	EV4_STATE	Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1.
Configure states in which event 5 is enabled	EV5_STATE	Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1.

17.1 How to read this chapter

These five standard timers are available on all LPC546xx devices.

17.2 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. The timers include external capture and match pin connections.
- Counter or timer operation.
- Up to four 32-bit captures can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt. The number of capture inputs for each timer that are actually available on device pins may vary by device.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Optional auto-reload from match shadow registers when counter is reset.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- For each timer, up to 4 external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by device):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 3 single edged controlled PWM outputs. (The number of match outputs for each timer that are actually available on device pins may vary by device.)
- Up to 2 match registers can be used to generate DMA requests.

17.3 Basic configuration

- Set the appropriate bits to enable clocks to timers that will be used: CTIMER0 and CTIMER1, and CTIMER2 in the AHBCLKCTRL1 register (Section 7.5.20), CTIMER3 and CTIMER4 in the ASYNCAPBCLKCTRL register (Section 7.5.104).

- Clear the timer reset using the ASYNCPRESETCTRL register (Table 234 for CTIMER0 and 1) and the PRESETCTRL1 register (Table 130 for CTIMER2, 3, and 4). Note that bit positions in the reset control registers match the bit positions in the clock control registers.
- Pins: Select timer pins and pin modes as needed through the relevant IOCON registers (Chapter 10).
- Interrupts: See register MCR (Table 371) and CCR (Table 373) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. For interrupt connections, see Table 88.
- DMA: Some timer match conditions can be used to generate timed DMA requests, see Table 300.

17.4 General description

Each Counter/timer is designed to count cycles of the APB bus clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length. All match registers can optionally be auto-reloaded from a companion shadow register whenever the counter is reset to zero. This permits modifying the match values for the next counter cycle without risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload will occur whenever the counter is reset either due to a match event or a write to bit 1 of the Timer Control Register (TCR).

17.4.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the APB bus clock. For more details see Section 17.6.11.

17.4.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMCON) control the functionality of this output.

17.4.3 Applications

- Interval timer for counting internal events
- Pulse Width Modulator via match outputs
- Pulse Width Demodulator via capture input
- Free running timer

17.4.4 Architecture

The block diagram for the timers is shown in Figure 40.

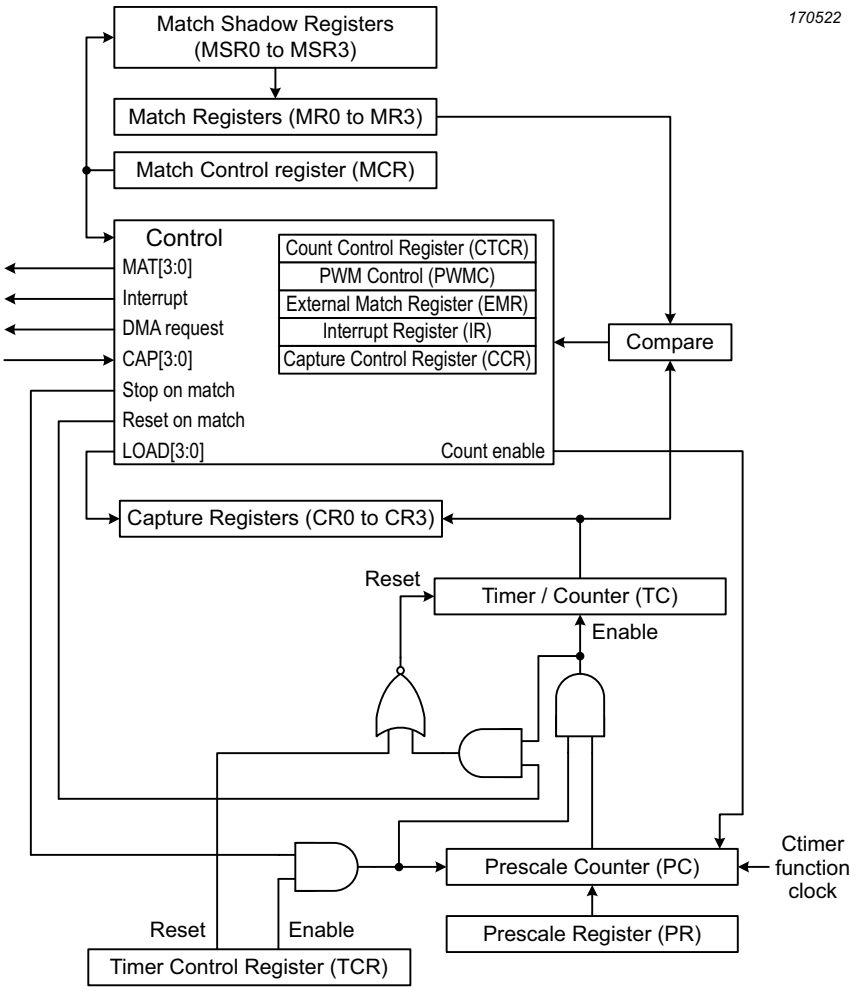


Fig 40. 32-bit counter/timer block diagram

17.5 Pin description

Table 363 gives a brief summary of each of the Timer/Counter related pins. Recommended IOCON settings are shown in Table 364.

Table 363. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP3:0 CTIMER1_CAP3:0 CTIMER2_CAP3:0 CTIMER3_CAP3:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details see Section 17.6.11.
CTIMER0_MAT3:0 CTIMER1_MAT3:0 CTIMER2_MAT3:0 CTIMER3_MAT3:0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

Table 364. Suggested CTIMER timer pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1.
10	SLEW: Generally set to 0.	Not used, set to 0.	I2CDRIVE: Set to 0.
9	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
5:4	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for timer capture input or match output.	A reasonable choice for timer capture input or match output.	Not recommended for timer match outputs.

17.5.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the IOCON registers, which are described in Chapter 10. Note that match conditions may be used internally without the use of a device pin.

17.6 Register description

Each Timer/Counter contains the registers shown in Table 365.

Table 365. Register overview: CTIMER0/1/2/3 (register base addresses 0x4000 8000 (CTIMER0), 0x4000 9000 (CTIMER1), 0x4002 8000 (CTIMER2), 0x4004 8000 (CTIMER3), 0x4004 9000 (CTIMER4))

Name	Access	Offset	Description	Reset value[1]	Section
IR	R/W	0x00	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0	17.6.1
TCR	R/W	0x04	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0	17.6.2
TC	R/W	0x08	Timer Counter. The 32 bit TC is incremented every PR+1 cycles of the APB bus clock. The TC is controlled through the TCR.	0	17.6.3
PR	R/W	0x0C	Prescale Register. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.	0	17.6.4
PC	R/W	0x10	Prescale Counter. The 32 bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0	17.6.5
MCR	R/W	0x14	The MCR is used to control whether an interrupt is generated, whether the TC is reset when a Match occurs, and whether the match register is reloaded from its shadow register when the TC is reset.	0	17.6.6
MR0	R/W	0x18	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0	17.6.7
MR1	R/W	0x1C	Match Register 1. See MR0 description.	0	17.6.7
MR2	R/W	0x20	Match Register 2. See MR0 description.	0	17.6.7
MR3	R/W	0x24	Match Register 3. See MR0 description.	0	17.6.7
CCR	R/W	0x28	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0	17.6.8
CR0	RO	0x2C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 input.	0	17.6.9
CR1	RO	0x30	Capture Register 1. See CR0 description.	0	17.6.9
CR2	RO	0x34	Capture Register 2. See CR0 description.	0	17.6.9
CR3	RO	0x38	Capture Register 3. See CR0 description.	0	17.6.9
EMR	R/W	0x3C	External Match Register. The EMR controls the match function and the external match pins.	0	17.6.10
CTCR	R/W	0x70	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	17.6.11
PWMC	R/W	0x74	PWM Control Register. The PWMCON enables PWM mode for the external match pins.	0	17.6.12
MSR0	R/W	0x78	Match 0 Shadow Register. If enabled, the Match 0 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	17.6.13

Table 365. Register overview: CTIMER0/1/2/3 (register base addresses 0x4000 8000 (CTIMER0), 0x4000 9000 (CTIMER1), 0x4002 8000 (CTIMER2), 0x4004 8000 (CTIMER3), 0x4004 9000 (CTIMER4))

Name	Access	Offset	Description	Reset value[1]	Section
MSR1	R/W	0x7C	Match 1 Shadow Register. If enabled, the Match 1 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	17.6.13
MSR2	R/W	0x80	Match 2 Shadow Register. If enabled, the Match 2 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	17.6.13
MSR3	R/W	0x84	Match 3 Shadow Register. If enabled, the Match 3 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.	0	17.6.13

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

17.6.1 Interrupt Register

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect.

Table 366. Interrupt Register (IR, offset 0x000) bit description

Bit	Symbol	Description	Reset Value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
7	CR3INT	Interrupt flag for capture channel 3 event.	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	-

17.6.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 367. Timer Control Register (TCR, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	Disabled.The counters are disabled.	
		1	Enabled. The Timer Counter and Prescale Counter are enabled.	
1	CRST		Counter reset.	0
		0	Disabled. Do nothing.	
		1	Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the APB bus clock. The counters remain reset until TCR[1] is returned to zero.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.3 Timer Counter registers

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed.

Table 368. Timer counter registers (TC, offset 0x08) bit description

Bit	Symbol	Description	Reset value
31:0	TCVAL	Timer counter value.	0

17.6.4 Prescale register

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

Table 369. Timer prescale registers (PR, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
31:0	PRVAL	Prescale counter value.	0

17.6.5 Prescale Counter register

The 32-bit Prescale Counter controls division of the APB bus clock by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every APB bus clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next APB bus clock. This causes the Timer Counter to increment on every APB bus clock when PR = 0, every 2 APB bus clocks when PR = 1, etc.

Table 370. Timer prescale counter registers (PC, offset 0x010) bit description

Bit	Symbol	Description	Reset value
31:0	PCVAL	Prescale counter value.	0

17.6.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Table 371. Match Control Register (MCR, offset 0x014) bit description

Bit	Symbol	Description	Reset Value
0	MR0I	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC. 0 = disabled. 1 = enabled.	0
1	MR0R	Reset on MR0: the TC will be reset if MR0 matches it. 0 = disabled. 1 = enabled.	0
2	MR0S	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0 = disabled. 1 = enabled.	0
3	MR1I	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC. 0 = disabled. 1 = enabled. 0 = disabled. 1 = enabled.	0
4	MR1R	Reset on MR1: the TC will be reset if MR1 matches it. 0 = disabled. 1 = enabled.	0
5	MR1S	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0 = disabled. 1 = enabled.	0
6	MR2I	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC. 0 = disabled. 1 = enabled.	0
7	MR2R	Reset on MR2: the TC will be reset if MR2 matches it. 0 = disabled. 1 = enabled.	0
8	MR2S	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0 = disabled. 1 = enabled.	0
9	MR3I	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC. 0 = disabled. 1 = enabled.	0
10	MR3R	Reset on MR3: the TC will be reset if MR3 matches it. 0 = disabled. 1 = enabled.	0

Table 371. Match Control Register (MCR, offset 0x014) bit description

Bit	Symbol	Description	Reset Value
11	MR3S	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0 = disabled. 1 = enabled.	0
23:12	-	Reserved. Read value is undefined, only zero should be written.	NA
24	MR0RL	Reload MR0 with the contents of the Match 0 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
25	MR1RL	Reload MR1 with the contents of the Match 1 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
26	MR2RL	Reload MR2 with the contents of the Match 2 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
27	MR3RL	Reload MR3 with the contents of the Match 3 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled.	0
31:28	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

If the associated MRxRL bit in the Match Control Register is set, the Match Register will be automatically reloaded with the current contents of its corresponding Match Shadow register whenever the TC is cleared to zero. This transfer will take place on the same clock edge that clocks the TC to zero.

Note: The TC is typically reset in response to an occurrence of a match on the Match Register being used to set the cycle counter rate. A reset can also occur due to software writing a 1 to bit 1 of the Timer Control Register.

Table 372. Timer match registers (MR[0:3], offset [0x018:0x024]) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

17.6.8 Capture Control Register

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Table 373. Capture Control Register (CCR, offset 0x028) bit description

Bit	Symbol	Description	Reset Value
0	CAP0RE	Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
1	CAP0FE	Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
2	CAP0I	Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt.	0
3	CAP1RE	Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
4	CAP1FE	Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
5	CAP1I	Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt.	0
6	CAP2RE	Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
7	CAP2FE	Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
8	CAP2I	Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt.	0
9	CAP3RE	Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
10	CAP3FE	Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0
11	CAP3I	Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.9 Capture Registers

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges.

Table 374. Timer capture registers (CR[0:3], offsets [0x02C:0x038]) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

17.6.10 External Match Register

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the timer number, 0 or 1, and “m” represent a Match number, 0 through 3.

Match events for Match 0 and Match 1 in each timer can cause a DMA request, see Section 17.7.2.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 17.7.1 “Rules for single edge controlled PWM outputs” on page 326).

Table 375. Timer external match registers (EMR, offset 0x03C) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0	-	External Match 0. This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
1	EM1	-	External Match 1. This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
2	EM2	-	External Match 2. This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
3	EM3	-	External Match 3. This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT1 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT1 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT2 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT2 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3.	00
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT3 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the APB bus clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the APB bus clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the APB bus clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than 1/APB bus clock.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 376. Count Control Register (CTCR, offset 0x070) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CTMODE		Counter/Timer Mode This field selects which rising APB bus clock edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register.	00
		0x0	Timer Mode. Incremented every rising APB bus clock edge.	
		0x1	Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	CINSEL		Count Input Select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. **Note:** If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	0
		0x0	Channel 0. CAPn.0 for CTIMERn	
		0x1	Channel 1. CAPn.1 for CTIMERn	
		0x2	Channel 2. CAPn.2 for CTIMERn	
		0x3	Channel 3. CAPn.3 for CTIMERn	

Table 376. Count Control Register (CTCR, offset 0x070) bit description

Bit	Symbol	Value	Description	Reset Value
4	ENCC	-	Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0
7:5	SELCC		Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Note that different part number and package variations may provide different capture input pin functions.	0
		0x0	Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x1	Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x2	Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x3	Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x4	Channel 2 Rising Edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x5	Channel 2 Falling Edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x6	Channel 3 Rising Edge. Rising edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
		0x7	Channel 3 Falling Edge. Falling edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	NA

17.6.12 PWM Control Register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 377. PWM Control Register (PWMC, offset 0x074)) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	Match. CTIMERn_MAT0 is controlled by EM0.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0
		0	Match. CTIMERn_MAT01 is controlled by EM1.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT1.	

Table 377. PWM Control Register (PWMC, offset 0x074)) bit description

Bit	Symbol	Value	Description	Reset value
2	PWMEN2		PWM mode enable for channel2.	0
		0	Match. CTIMERn_MAT2 is controlled by EM2.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT2.	
3	PWMEN3		PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0
		0	Match. CTIMERn_MAT3 is controlled by EM3.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT3.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

17.6.13 Match Shadow Registers

The Match Shadow registers contain the values that the corresponding Match Registers are (optionally) reloaded with at the start of each new counter cycle. Typically, the match that causes the counter to be reset (and instigates the match reload) will also be programmed to generate an interrupt or DMA request. Software or the DMA engine will then have one full counter cycle to modify the contents of the Match Shadow Register(s) before the next reload occurs.

Table 378. Timer match shadow registers (MSR[0:3], offset [0x78:0x84]) bit description

Bit	Symbol	Description	Reset value
31:0	SHADOW	Timer counter match shadow value.	0x0

17.7 Functional description

Figure 41 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 42 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

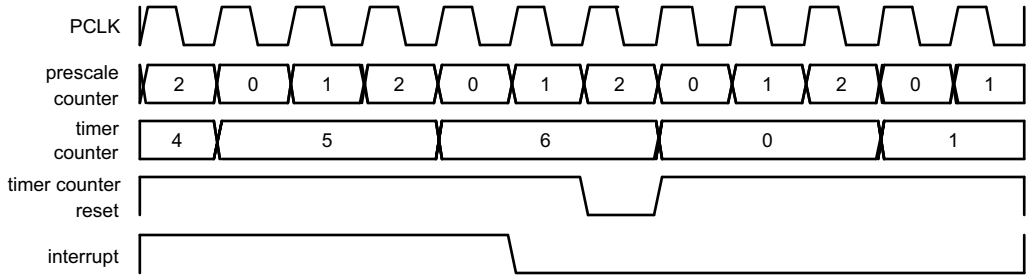


Fig 41. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

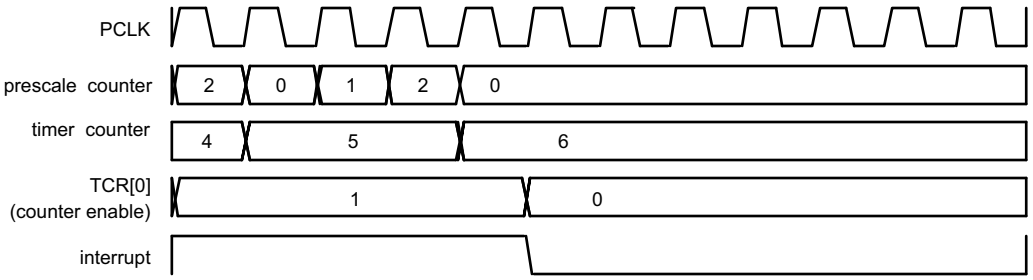


Fig 42. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

17.7.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.

4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

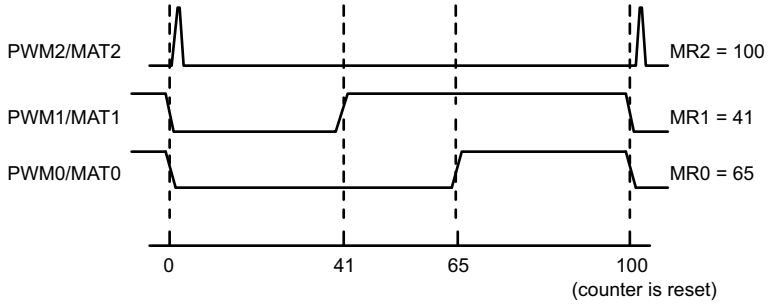


Fig 43. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

17.7.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See Section 17.6.1. A DMA request will be cleared automatically when it is acted upon by the DMA controller.

Note: because timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

18.1 How to read this chapter

The watchdog timer is available on all LPC546xx devices.

18.2 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- Incorrect feed sequence causes immediate watchdog event if enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is a selectable frequency in the range of 6 kHz to 1.5 MHz (see Section 7.5.79 “Watchdog oscillator control register”). The accuracy of this clock is limited to +/- 40% over temperature, voltage, and silicon processing variations. To determine the actual watchdog oscillator output, use the frequency measure block. See Section 7.2.3 “Measure the frequency of a clock signal”.
- The Watchdog timer can be configured to run in deep-sleep mode.
- Debug mode.

18.3 Basic configuration

Configuration of the WWDT is accomplished as follows:

- Turn on and configure the Watchdog oscillator. See the PDEN_WDT_OSC bit in the PDRUNCG0 register (Section 7.5.84), and the Watchdog oscillator control register (Section 7.5.79).
- Enable the register interface (WWDT bus clock): set the WWDT bit in the AHBCLKCTRL0 register, Table 139.
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the STARTER0 register (Table 222).

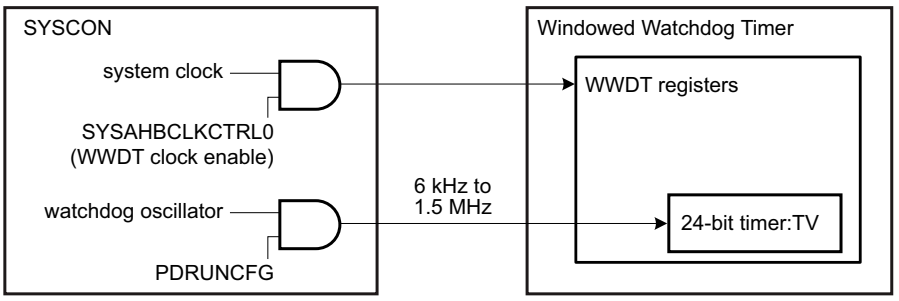


Fig 44. WWDT timing

18.4 Pin description

The WWDT has no external pins.

18.5 General description

The purpose of the Watchdog Timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the Watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Enable and configure the Watchdog oscillator as described in Section 18.3.
- Set the Watchdog timer constant reload value in the TC register.

- Set the Watchdog timer operating mode in the MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.
- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the FEED register.
- The Watchdog must be fed again before the Watchdog counter reaches zero in order to prevent a watchdog event. If a window value is programmed, the feed must also occur after the watchdog counter passes that value.

When the Watchdog Timer is configured so that a watchdog event will cause a reset and the counter reaches zero, the CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

When the Watchdog Timer is configured to generate a warning interrupt, the interrupt will occur when the counter is no longer greater than the value defined by the WARNINT register.

18.5.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 45. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.

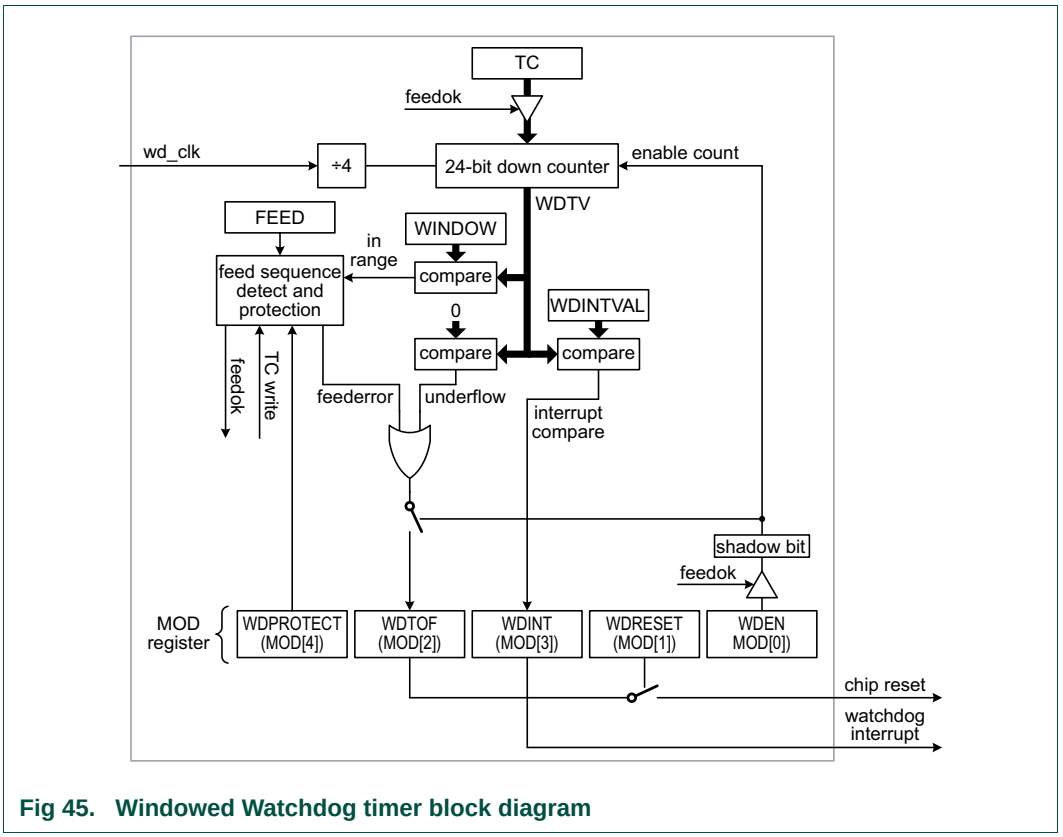


Fig 45. Windowed Watchdog timer block diagram

18.5.2 Clocking and power control

The watchdog timer block uses two clocks: APB bus clock and WDCLK. The APB bus clock is used for the APB accesses to the watchdog registers and is derived from the system clock (see Figure 7). The WDCLK is used for the watchdog timer counting and is derived from the watchdog oscillator.

The synchronization logic between the two clock domains works as follows: When the MOD and TC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the APB bus clock, so that the CPU can read the TV register.

Remark: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time the WDPROTECT bit is enabled in the MOD register. The length of the delay depends on the selected watchdog clock WDCLK.

18.5.3 Using the WWDT lock features

The WWDT supports several lock features which can be enabled to ensure that the WWDT is running at all times:

- Disabling the WWDT clock source
- Changing the WWDT reload value

18.5.3.1 Disabling the WWDT clock source

If bit 5 in the WWDT MOD register is set, the WWDT clock source is locked and can not be disabled either by software or by hardware when sleep or deep-sleep modes are entered. Therefore, the user must ensure that the watchdog oscillator for each power mode is enabled **before** setting bit 5 in the MOD register.

18.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

18.6 Register description

The Watchdog Timer contains the registers shown in Table 379.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 379. Register overview: Watchdog timer (base address 0x4000 C000)

Name	Access	Offset	Description	Reset value	Section
MOD	R/W	0x000	Watchdog mode. This register contains the basic mode and status of the Watchdog Timer.	0	18.6.1
TC	R/W	0x004	Watchdog timer constant. This 24-bit register determines the time-out value.	0xFF	18.6.2
FEED	WO	0x008	Watchdog feed sequence. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in TC.	NA	18.6.3
TV	RO	0x00C	Watchdog timer value. This 24-bit register reads out the current value of the Watchdog timer.	0xFF	18.6.4
-	-	0x010	Reserved	-	-
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0	18.6.5
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF	18.6.6

18.6.1 Watchdog mode register

The WDMOD register controls the operation of the Watchdog. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 380. Watchdog mode register (MOD, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. Once this bit is set to one and a watchdog feed is performed, the watchdog timer will run permanently.	0
		0	Stop. The watchdog timer is stopped.	
		1	Run. The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. Once this bit has been written with a 1 it cannot be re-written with a 0.	0
		0	Interrupt. A watchdog time-out will not cause a chip reset.	
		1	Reset. A watchdog time-out will cause a chip reset.	
2	WDTOF	-	Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software writing a 0 to this bit position. Causes a chip reset if WDRESET = 1.	0 [1]
3	WDINT	-	Warning interrupt flag. Set when the timer is at or below the value in WDWARNINT. Cleared by software writing a 1 to this bit position. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT is 0 and the WDRESET bit is 0 when TV decrements to 0.	0

Table 380. Watchdog mode register (MOD, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
4	WDPROTECT		Watchdog update mode. This bit can be set once by software and is only cleared by a reset.	0
		0	Flexible. The watchdog time-out value (TC) can be changed at any time.	
		1	Threshold. The watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW.	
5	LOCK	-	Once this bit is set to one and a watchdog feed is performed, disabling or powering down the watchdog oscillator is prevented by hardware. This bit can be set once by software and is only cleared by any reset.	0
31:6	-	-	Reserved. Read value is undefined, only zero should be written.	-

[1] Only an external or power-on reset has this effect.

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. All three bits are cleared by an external reset or a Watchdog timer reset.

WDTOF The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when PROTECT =1 and an attempt is made to write to the TC register. This flag is cleared by software writing a 0 to this bit.

WDINT The Watchdog interrupt flag is set when the Watchdog counter is no longer greater than the value specified by WARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 1 to this bit.

In all power modes except deep power-down mode, a Watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog oscillator can be configured to keep running in sleep and deep-sleep modes.

If a watchdog interrupt occurs in sleep or deep-sleep mode, and the WWDT interrupt is enabled in the NVIC, the device will wake up. Note that in deep-sleep mode, the WWDT interrupt must be enabled in the STARTER0 register in addition to the NVIC.

See the following registers:

Table 222 “Start enable register 0 (STARTER0, main syscon: offset 0x680) bit description”

Table 381. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WDWINDOW will also cause a watchdog reset.

18.6.2 Watchdog Timer Constant register

The TC register determines the time-out value. Every time a feed sequence occurs the value in the TC is loaded into the Watchdog timer. The TC resets to 0x00 00FF. Writing a value below 0xFF will cause 0x00 00FF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in WDMOD = 1, an attempt to change the value of TC before the watchdog counter is below the values of WDWARNINT and WDWINDOW will cause a watchdog reset and set the WDTOF flag.

Table 382. Watchdog Timer Constant register (TC, offset 0x04) bit description

Bit	Symbol	Description	Reset value
23:0	COUNT	Watchdog time-out value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

18.6.3 Watchdog Feed register

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the TC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

After writing 0xAA to WDFEED, access to any Watchdog register other than writing 0x55 to WDFEED causes an immediate reset/interrupt when the Watchdog is enabled, and sets the WDTOF flag. The reset will be generated during the second APB bus clock following an incorrect access to a Watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Table 383. Watchdog Feed register (FEED, offset 0x08) bit description

Bit	Symbol	Description	Reset value
7:0	FEED	Feed value should be 0xAA followed by 0x55.	NA
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

18.6.4 Watchdog Timer Value register

The TV register is used to read the current value of Watchdog timer counter.

When reading the value of the 24-bit counter, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 APB bus clock cycles, so the value of TV is older than the actual value of the timer when it's being read by the CPU.

Table 384. Watchdog Timer Value register (TV, offset 0x0C) bit description

Bit	Symbol	Description	Reset value
23:0	COUNT	Counter timer value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

18.6.5 Watchdog Timer Warning Interrupt register

The WDWARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter is no longer greater than the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 385. Watchdog Timer Warning Interrupt register (WARNINT, offset 0x14) bit description

Bit	Symbol	Description	Reset value
9:0	WARNINT	Watchdog warning interrupt compare value.	0
31:10	-	Reserved, only zero should be written.	-

18.6.6 Watchdog Timer Window register

The WINDOW register determines the highest TV value allowed when a watchdog feed is performed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event will occur.

WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Table 386. Watchdog Timer Window register (WINDOW, offset 0x18) bit description

Bit	Symbol	Description	Reset value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved, only zero should be written.	-

18.7 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.

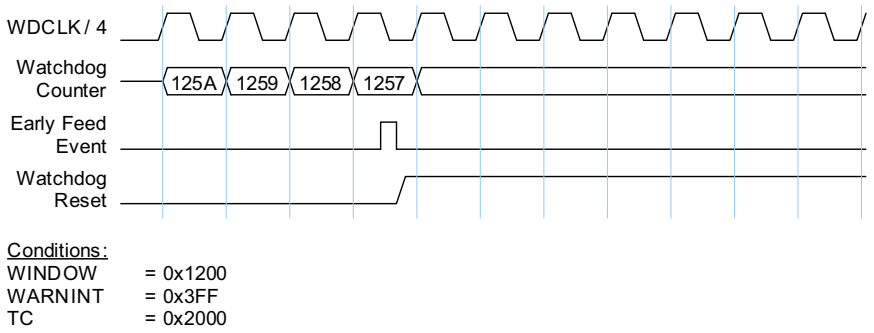


Fig 46. Early watchdog feed with windowed mode enabled

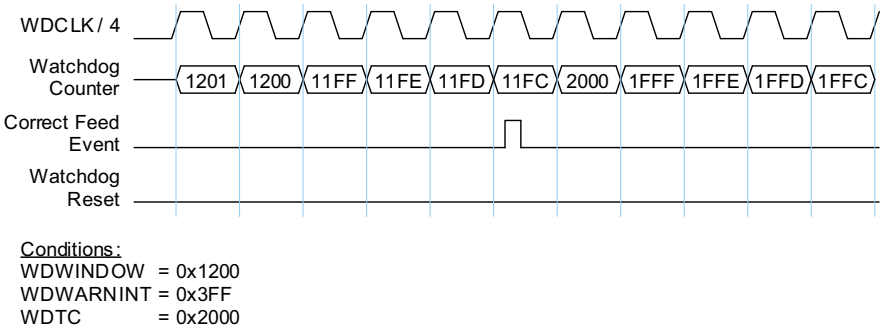


Fig 47. Correct watchdog feed with windowed mode enabled

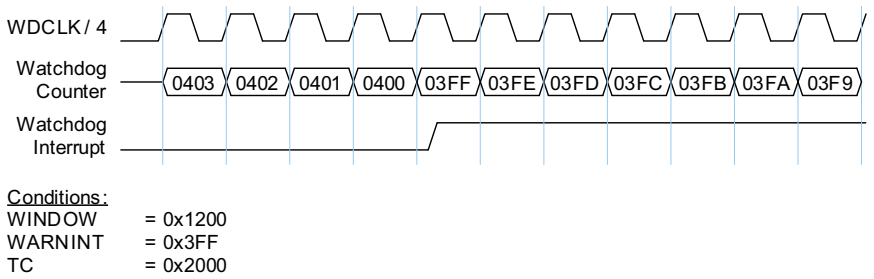


Fig 48. Watchdog warning interrupt

19.1 How to read this chapter

The RTC is available on all LPC546xx devices.

19.2 Features

- The RTC and its independent oscillator operate directly from the device power pins, not using the on-chip regulator. The RTC oscillator has the following clock outputs:
  - 32 kHz clock, selectable for system clock and CLKOUT pin.
  - 1 Hz clock for RTC timing.
  - 1 kHz clock for high-resolution RTC timing.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- Separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 ms resolution with a more that one minute maximum time-out period.
- RTC alarm and high-resolution/wake-up timer time-out each generate independent interrupt requests that go to one NVIC channel. Either time-out can wake up the part from any of the low power modes, including deep power-down.
- Eight 32-bit general purpose registers can retain data in deep power-down or in the event of a power failure, provided there is battery backup.

19.3 Basic configuration

Configure the RTC as follows:

- Use the AHBCLKCTRL0 register (Table 139) to enable the clock to the RTC register interface and peripheral clock.
- For RTC software reset use the RTC CTRL register. See Table 389. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, it is not initialized by other system resets.
- The RTC provides an interrupt to the NVIC for the RTC_WAKE and RTC_ALARM functions, see Chapter 6 “LPC546xx Nested Vectored Interrupt Controller (NVIC)”.
- To enable the RTC interrupts for waking up from deep-sleep mode, enable the interrupts in the STARTER0 register (Table 222) and the NVIC.
- To enable the RTC interrupts for waking up from deep power-down, enable the appropriate RTC clock and wake-up in the RTC CTRL register (Table 389).
- If enabled, the RTC and its oscillator continue running in all reduced power modes as long as power is supplied to the device. So, the 32 kHz output is always available to be enabled for syscon clock generation (see Table 200). Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the CLKOUT pin. The 1 Hz output is enabled in the RTC CTRL register (RTC_EN bit). Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL register (RTC1KHZ_EN bit).

- If the 32 kHz output of the RTC is used by another part of the system, enable it via the EN bit in the RTCOSCCTRL register. See Table 200.

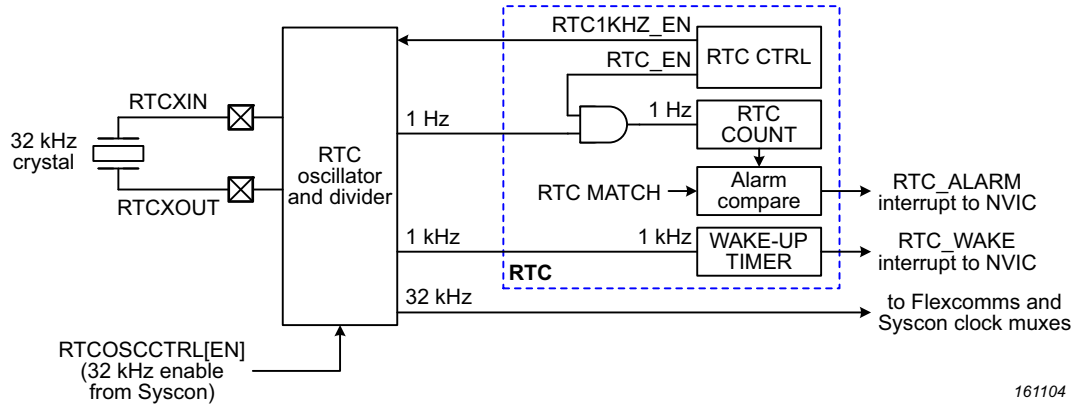


Fig 49. RTC clocking and block diagram

19.3.1 RTC timers

The RTC contains two counters:

1. The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the part from any low power mode if enabled.
2. The high-resolution/wake-up timer. This 16-bit timer uses a 1 kHz clock and operates as a one-shot down timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the part from any low power mode if enabled. This timer is intended to be used for timed wake-up from deep-sleep or deep power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used.

19.4 General description

19.4.1 Real-time clock

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, it counts continuously at a 1 Hz clock rate as long as the device is powered up and the RTC remains enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit match register.

If the part is in one of the reduced-power modes (deep-sleep, deep power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

19.4.2 High-resolution/wake-up timer

The time interval required for many applications, including waking the part up from a low-power mode, will often demand a greater degree of resolution than the one-second minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-Resolution/Wake-up Timer is a 16-bit down counter which is clocked at a 1 kHz rate when it is enabled. Writing any non-zero value to this timer will automatically enable the counter and launch a countdown sequence. When the counter is being used as a wake-up timer, this write can occur just prior to entering a reduced power mode.

When a starting count value is loaded, the High-Resolution/Wake-up Timer will turn on, count from the pre-loaded value down to zero, generate an interrupt and/or a wake-up command, and then turn itself off until re-launched by a subsequent software write.

19.4.3 RTC power

The RTC module and the oscillator that drives it, run directly from device power pins. As a result, the RTC will continue operating in deep power-down mode when power is internally turned off to the rest of the device.

19.4.4 General purpose backup registers

The RTC module and the oscillator that drives it, run directly from device power pins. As a result, the RTC will continue operating in deep power-down mode when power is internally turned off to the rest of the device.

The general purpose registers retain data through the deep power-down mode or loss of main power as long as the V_{BAT} supply is powered. Only a complete removal of power from the chip (including V_{BAT}) or a software reset of the RTC can clear the general purpose registers.

19.5 Pin description

Table 387 gives a summary of pins related to the RTC.

Table 387. RTC pin description

Pin	Type	Description
RTCXIN	Input	RTC oscillator input.
RTCXOUT	Output	RTC oscillator output.

19.6 Register description

Reset Values pertain to initial power-up of the device or when an RTC software reset is applied (except where noted). This block is not initialized by any other system reset.

Table 388. Register overview: RTC (base address 0x4002 C000)

Name	Access	Offset	Description	SWRESET bit in CTRL = 1	Reset value	Section
CTRL	R/W	0x00	RTC control.	0x1	0x1	19.6.1
MATCH	R/W	0x04	RTC match.	0xFFFF FFFF	0xFFFF FFFF	19.6.2
COUNT	R/W	0x08	RTC counter.	0	0	19.6.3
WAKE	R/W	0x0C	High-resolution/wake-up timer control.	0	0	19.6.4
GPREG0	R/W	0x40	General Purpose register 0.	0	0	19.6.5
GPREG1	R/W	0x44	General Purpose register 1.	0	0	19.6.5
GPREG2	R/W	0x48	General Purpose register 2.	0	0	19.6.5
GPREG3	R/W	0x4C	General Purpose register 3.	0	0	19.6.5
GPREG4	R/W	0x50	General Purpose register 4.	0	0	19.6.5
GPREG5	R/W	0x54	General Purpose register 5.	0	0	19.6.5
GPREG6	R/W	0x58	General Purpose register 6.	0	0	19.6.5
GPREG7	R/W	0x5C	General Purpose register 7.	0	0	19.6.5

19.6.1 RTC CTRL register

This register controls which clock the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from deep power-down. To wake up the part from deep-sleep mode, enable the RTC interrupts in the system control block STARTLOGIC1 register.

Table 389. RTC control register (CTRL, offset 0x00) bit description

Bit	Symbol	Value	Description	Reset value
0	SWRESET		Software reset control	1
		0	Not in reset. The RTC is not held in reset. This bit must be cleared prior to configuring or initiating any operation of the RTC.	
		1	In reset. The RTC is held in reset. All register bits within the RTC will be forced to their reset value except the RTC_OSC_PD and RTC_OSC_BYPASS bits in this register. This bit must be cleared before writing to any register in the RTC - including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register at the same time that the reset bit is being cleared.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
2	ALARM1HZ		RTC 1 Hz timer alarm flag status.	0
		0	No match. No match has occurred on the 1 Hz RTC timer. Writing a 0 has no effect.	
		1	Match. A match condition has occurred on the 1 Hz RTC timer. This flag generates an RTC alarm interrupt request RTC_ALARM which can also wake up the part from any low power mode. Writing a 1 clears this bit.	

Table 389. RTC control register (CTRL, offset 0x00) bit description

Bit	Symbol	Value	Description	Reset value
3	WAKE1KHZ		RTC 1 kHz timer wake-up flag status.	0
		0	Run. The RTC 1 kHz timer is running. Writing a 0 has no effect.	
		1	Time-out. The 1 kHz high-resolution/wake-up timer has timed out. This flag generates an RTC wake-up interrupt request RTC-WAKE which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
4	ALARMDPD_EN		RTC 1 Hz timer alarm enable for deep power-down.	0
		0	Disable. A match on the 1 Hz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 Hz RTC timer bring the part out of deep power-down mode.	
5	WAKEDPD_EN		RTC 1 kHz timer wake-up enable for deep power-down.	0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 kHz RTC timer bring the part out of deep power-down mode.	
6	RTC1KHZ_EN		RTC 1 kHz clock enable. This bit can be set to 0 to conserve power if the 1 kHz timer is not used. This bit has no effect when the RTC is disabled (bit 7 of this register is 0).	0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode. Disabling the RTC 1 kHz clock also clears the WAKE1KHZ flag.	
		1	Enable. The 1 kHz RTC timer is enabled.	
7	RTC_EN		RTC enable.	0
		0	Disable. The RTC 1 Hz and 1 kHz clocks are shut down and the RTC operation is disabled. This bit should be 0 when writing to load a value in the RTC counter register.	
		1	Enable. The 1 Hz RTC clock is enabled and RTC operation is enabled. This bit must be set to initiate operation of the RTC. The first clock to the RTC counter occurs 1 s after this bit is set. To also enable the high-resolution, 1 kHz clock, set bit 6 in this register.	
8	RTC_OSC_PD		RTC oscillator power-down control.	0
		0	RTC oscillator is powered up.	
		1	RTC oscillator is powered-down.	
31:9	-	-	Reserved. Read value is undefined, only zero should be written.	-

19.6.2 RTC match register

Table 390. RTC match register (MATCH, offset 0x04) bit description

Bit	Symbol	Description	Reset value
31:0	MATVAL	Contains the match value against which the 1 Hz RTC timer will be compared to set the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled.	0xFFFF FFFF

19.6.3 RTC counter register

Table 391. RTC counter register (COUNT, offset 0x08) bit description

Bit	Symbol	Description	Reset value
31:0	VAL	A read reflects the current value of the main, 1 Hz RTC timer. A write loads a new initial value into the timer. The RTC counter will count up continuously at a 1 Hz rate once the RTC Software Reset is removed (by clearing bit 0 of the CTRL register). **Remark:** No synchronization is provided to prevent a read of the counter register during a count transition. The suggested method to read a counter is to read the location twice and compare the results. If the values match, the time can be used. If they do not match, then the read should be repeated until two consecutive reads produce the same result. **Remark:** Only write to this register when the RTC_EN bit in the RTC CTRL Register is 0. The counter increments one second after the RTC_EN bit is set.	0

19.6.4 RTC high-resolution/wake-up register

Table 392. RTC high-resolution/wake-up register (WAKE, offset 0x0C) bit description

Bit	Symbol	Description	Reset value
15:0	VAL	A read reflects the current value of the high-resolution/wake-up timer. A write pre-loads a start count value into the wake-up timer and initializes a count-down sequence. Do not write to this register while counting is in progress.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

19.6.5 RTC General purpose backup registers

These register retain contents even during deep power-down mode as long as device power is maintained. They can be used to preserve application data or configuration that will always be available.

Table 393. RTC general purpose registers 0 to 7 (GPREG[0:7], offset 0x40:0x5C) bit description

Bit	Symbol	Description	Reset value
31:0	GPDATA	Data retained during deep power-down mode or loss of main power as long as V_{BAT} is supplied.	0

20.1 How to read this chapter

The MRT is available on all LPC546xx devices.

20.2 Features

- 24-bit interrupt timer
- Four channels independently counting down from individually set values
- Repeat interrupt, one-shot interrupt, and one-shot bus stall modes

20.3 Basic configuration

Configuration of the MRT is accomplished as followings:

- In the AHBCLKCTRL1 register (Table 140), set the MRT bit to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL1 register (Table 130).
- The global MRT interrupt is connected to an interrupt slot in the NVIC (see Table 88).

20.4 Pin description

The MRT is not associated with any device pins.

20.5 General description

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with four channels. Each channel can be programmed with an independent time interval.

Each channel operates independently from the other channels in one of the following modes:

- Repeat interrupt mode. See Section 20.5.1.
- One-shot interrupt mode. See Section 20.5.2.
- One-shot stall mode. See Section 20.5.3.

The modes for each timer are set in the timer's control register. See Table 397.

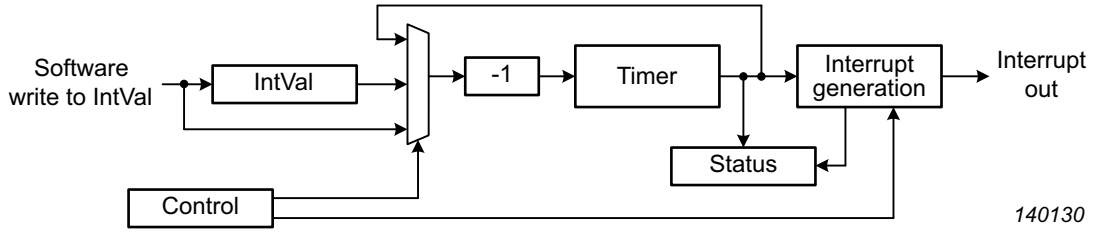


Fig 50. MRT block diagram

20.5.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer begins to count down from this value. When the timer reaches zero, an interrupt is generated, the value in the INTVALn register IVALUE - 1 is reloaded automatically, and the timer starts to count down again.

While the timer is running in repeat interrupt mode, the following actions can be performed:

- Change the interval value on the next timer cycle by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero. On the next cycle, the timer counts down from the new value.
- Change the interval value on-the-fly immediately by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. An interrupt is generated when the timer reaches 0.
- Stop the timer at the end of time interval by writing a 0 to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero.
- Stop the timer immediately by writing a 0 to the INTVALn register and setting the LOAD bit to 1. No interrupt is generated when the INTVALn register is written.

20.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, a single interrupt can be generated at any point. This mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer starts to count down. When the timer reaches 0, an interrupt is generated and the timer stops and enters the idle state.

While the timer is running in the one-shot interrupt mode, the following actions can be performed:

- Update the INTVALn register with a new time interval value (>0) and set the LOAD bit to 1. The timer immediately reloads the new time interval, and starts counting down from the new value. No interrupt is generated when the TIME_INTVALn register is updated.
- Write a 0 to the INTVALn register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. No interrupt is generated when the INTVALn register is updated.

20.5.3 One-shot stall mode

One-shot stall mode is similar to one-shot interrupt mode, except that it is intended for very short delays, for instance when the delay needed is less than the time it takes to get to an interrupt service routine. This mode is designed for very low software overhead, requiring only a single write to the INTVAL register (if the channel is already configured for one-shot stall mode). The MRT times the requested delay while stalling the bus write operation, concluding the write when the delay is complete. No interrupt or status polling is needed.

Bus stall mode can be used when a short delay is need between two software controlled events, or when a delay is expected before software can continue. Since in this mode there are no bus transactions while the MRT is counting down, the CPU consumes a minimum amount of power during that time.

Note that bus stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL and the time that software continues. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.

20.6 Register description

The reset values shown in Table 394 are POR reset values.

Table 394. Register overview: MRT (base address 0x4000 D000)

Name	Access	Offset	Description	Reset value	Section
MRT Timer 0 registers					
INTVAL0	R/W	0x0	MRT0 Time interval value. This value is loaded into the TIMER0 register.	0	20.6.1
TIMER0	RO	0x4	MRT0 Timer. This register reads the value of the down-counter.	0xFF FFFF	20.6.2
CTRL0	R/W	0x8	MRT0 Control. This register controls the MRT0 modes.	0	20.6.3
STAT0	R/W	0xC	MRT0 Status.	0	20.6.4
MRT Timer 1 registers					
INTVAL1	R/W	0x10	MRT1 Time interval value. This value is loaded into the TIMER1 register.	0	20.6.1
TIMER1	R/W	0x14	MRT1 Timer. This register reads the value of the down-counter.	0xFF FFFF	20.6.2
CTRL1	R/W	0x18	MRT1 Control. This register controls the MRT1 modes.	0	20.6.3
STAT1	R/W	0x1C	MRT1 Status.	0	20.6.4
MRT Timer 2 registers					
INTVAL2	R/W	0x20	MRT2 Time interval value. This value is loaded into the TIMER2 register.	0	20.6.1
TIMER2	R/W	0x24	MRT2 Timer. This register reads the value of the down-counter.	0xFF FFFF	20.6.2
CTRL2	R/W	0x28	MRT2 Control. This register controls the MRT2 modes.	0	20.6.3
STAT2	R/W	0x2C	MRT2 Status.	0	20.6.4
MRT Timer 3 registers					
INTVAL3	R/W	0x30	MRT3 Time interval value. This value is loaded into the TIMER3 register.	0	20.6.1
TIMER3	R/W	0x34	MRT3 Timer. This register reads the value of the down-counter.	0xFF FFFF	20.6.2
CTRL3	R/W	0x38	MRT3 Control. This register controls the MRT modes.	0	20.6.3
STAT3	R/W	0x3C	MRT3 Status.	0	20.6.4
Global MRT registers					
MODCFG	R/W	0xF0	Module Configuration. This register provides information about this particular MRT instance, and allows choosing an overall mode for the idle channel feature.	0	20.6.5
IDLE_CH	RO	0xF4	Idle channel. This register returns the number of the first idle channel.	0	20.6.6
IRQ_FLAG	R/W	0xF8	Global interrupt flag.	0	20.6.7

20.6.1 Time interval register

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 395. Time interval register (INTVAL[0:3], offset 0x000 (INTVAL0) to 0x030 (INTVAL3)) bit description

Bit	Symbol	Value	Description	Reset value
23:0	IVALUE		Time interval load value. This value is loaded into the TIMERn register and the MRT channel n starts counting down from IVALUE -1. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: • If LOAD = 1, the timer stops immediately. • If LOAD = 0, the timer stops at the end of the time interval.	0
30:24	-	-	Reserved. Read value is undefined, only zero should be written.	-
31	LOAD		Determines how the timer interval value IVALUE -1 is loaded into the TIMERn register. This bit is write-only. Reading this bit always returns 0.	0
		0	No force load. The load from the INTVALn register to the TIMERn register is processed at the end of the time interval if the repeat mode is selected.	
		1	Force load. The INTVALn interval value IVALUE -1 is immediately loaded into the TIMERn register while TIMERn is running.	

20.6.2 Timer register

The timer register holds the current timer value. This register is read-only.

Table 396. Timer register (TIMER[0:3], offset 0x004 (TIMER0) to 0x034 (TIMER3)) bit description

Bit	Symbol	Description	Reset value
23:0	VALUE	Holds the current timer value of the down-counter. The initial value of the TIMERn register is loaded as IVALUE - 1 from the INTVALn register either at the end of the time interval or immediately in the following cases: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF).	0x00FF FFFF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

20.6.3 Control register

The control register configures the mode for each MRT and enables the interrupt.

Table 397. Control register (CTRL[0:3], offset 0x08 (CTRL0) to 0x38 (CTRL3)) bit description

Bit	Symbol	Value	Description	Reset value
0	INTEN		Enable the TIMERn interrupt.	0
		0	Disabled. TIMERn interrupt is disabled.	
		1	Enabled. TIMERn interrupt is enabled.	
2:1	MODE		Selects timer mode.	0
		0x0	Repeat interrupt mode.	
		0x1	One-shot interrupt mode.	
		0x2	One-shot stall mode.	
		0x3	Reserved.	
31:3	-	-	Reserved.	-

20.6.4 Status register

This register indicates the status of each MRT.

Table 398. Status register (STAT[0:3], offset 0x0C (STAT0) to 0x3C (STAT3)) bit description

Bit	Symbol	Value	Description	Reset value
0	INTFLAG		Monitors the interrupt flag.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMERn has reached the end of the time interval. If the INTEN bit in the CONTROLn is also set to 1, the interrupt for timer channel n and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	RUN		Indicates the state of TIMERn. This bit is read-only.	0
		0	Idle state. TIMERn is stopped.	
		1	Running. TIMERn is running.	
2	INUSE		Channel In Use flag. Operating details depend on the MULTITASK bit in the MODCFG register, and affects the use of IDLE_CH. See Section 20.6.6 “Idle channel register” for details of the two operating modes.	0
		0	This channel **is not** in use.	
		1	This channel **is** in use.	
31:2	-	-	Reserved.	-

20.6.5 Module Configuration register

The MODCFG register provides the configuration (number of channels and timer width) for this MRT. See Section 20.6.6 “Idle channel register” for details.

Table 399. Module Configuration register (MODCFG, offset 0xF0) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	NOC	-	Identifies the number of channels in this MRT. (4 channels on this device.)	0x3
8:4	NOB	-	Identifies the number of timer bits in this MRT. (24 bits wide on this device.)	0x17
30:9	-	-	Reserved. Read value is undefined, only zero should be written.	NA
31	MULTITASK		Selects the operating mode for the INUSE flags and the IDLE_CH register.	0
		0	Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset.	
		1	Multi-task mode.	

20.6.6 Idle channel register

The idle channel register can be used to assist software in finding available channels in the MRT. This allows more flexibility by not giving hard assignments to software that makes use of the MRT, without the need to search for an available channel. Generally, IDLE_CH returns the lowest available channel number.

IDLE_CH can be used in two ways, controlled by the value of the MULTITASK bit in the MODCFG register. MULTITASK affects both the function of IDLE_CH, and the function of the INUSE bit for each MRT channel as follows:

- MULTITASK = 0: hardware status mode. The INUSE flags for all MRT channels are reset. IDLECH returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: multi-task mode. In this mode, the INUSE flags allow more control over when MRT channels are released for further use. When IDLE_CH is read, returning a channel number of an idle channel, the INUSE flag for that channel is set by hardware. That channel will not be considered idle until its RUN flag = 0, there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and no need to start the channel before it is no longer considered idle by IDLE_CH. It also allows software to identify a specific MRT channel that it can use, then use it more than once without releasing it, removing the need to ask for an available channel for every use.

Table 400. Idle channel register (IDLE_CH, offset 0xF4) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	0
7:4	CHAN	Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned such that it can be used as an offset from the MRT base address in order to access the registers for the allocated channel. If all timer channels are running, CHAN = 0xF. See text above for more details.	0
31:8	-	Reserved.	-

20.6.7 Global interrupt flag register

The global interrupt register combines the interrupt flags from the individual timer channels in one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUSn registers.

Table 401. Global interrupt flag register (IRQ_FLAG, offset 0xF8) bit description

Bit	Symbol	Value	Description	Reset value
0	GFLAG0		Monitors the interrupt flag of TIMER0.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, the interrupt for timer channel 0 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	GFLAG1	-	Monitors the interrupt flag of TIMER1. See description of channel 0.	0
2	GFLAG2	-	Monitors the interrupt flag of TIMER2. See description of channel 0.	0
3	GFLAG3	-	Monitors the interrupt flag of TIMER3. See description of channel 0.	0
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	0

21.1 How to read this chapter

This timer is available on all LPC546xx devices.

21.2 Features

- 48-bit counter running from the main clock. Counter can be free-running or be reset by a generated interrupt.
- 48-bit compare value.
- 48-bit compare mask. An interrupt is generated when the counter value equals the compare value, after masking. This allows for combinations not possible with a simple compare.

21.3 Basic configuration

The RI timer is configured through the following registers:

- Power to the register interface: set the RIT bit in the AHBCLKCTRL1 register, Table 140.

21.4 General description

The Repetitive Interrupt Timer (RIT) provides a versatile means of generating interrupts at specified time intervals, without using a standard timer. It is intended for repeating interrupts that aren't related to Operating System interrupts. The RIT could also be used as an alternative to the System Tick Timer if there are different system requirements.

The RI timer can be used in conjunction with the Embedded Trace Macrocell (ETM) for time-stamping. The RI timer allows periodic insertion of a time value based on specific events (exceptions, return from exceptions, trace FIFO flush) into the trace data stream of the ETM. With the ETM time-stamping feature, multiple trace data streams can be correlated to obtain a rough measure of code performance.

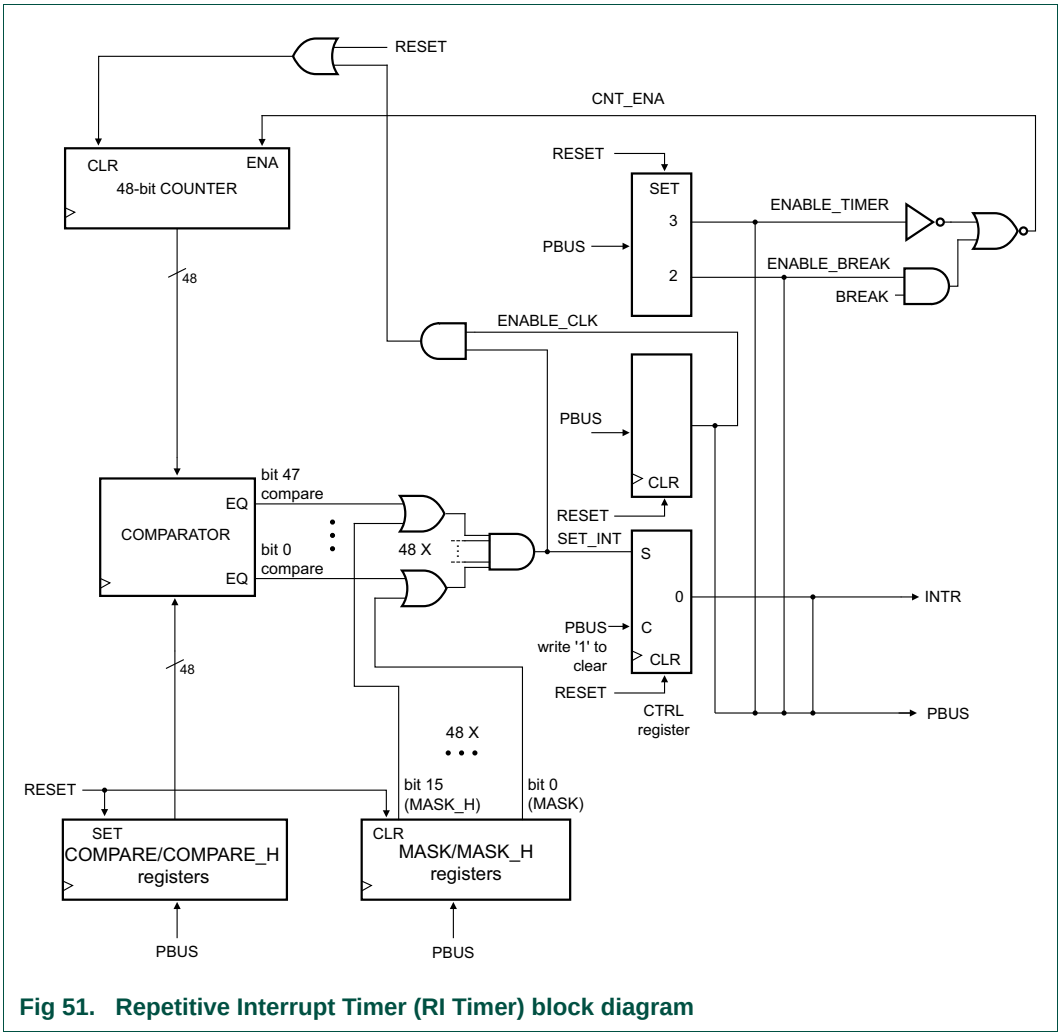


Fig 51. Repetitive Interrupt Timer (RI Timer) block diagram

21.5 Register description

Table 402. Register overview: Repetitive Interrupt Timer (RIT) (base address 0x4002 D000)

Name	Access	Address	Description	Reset value[1]	Reference
COMPVAL	R/W	0x000	Compare value LSB register. Holds the 32 LSBs of the compare value.	0xFFFF FFFF	21.5.1
MASK	R/W	0x004	Mask LSB register. This register holds the 32 LSB s of the mask value. A 1 written to any bit will force the compare to be true for the corresponding bit of the counter and compare register.	0	21.5.2
CTRL	R/W	0x008	Control register.	0xC	21.5.3
COUNTER	R/W	0x00C	Counter LSB register. 32 LSBs of the counter.	0	21.5.4
COMPVAL_H	R/W	0x010	Compare value MSB register. Holds the 16 MSBs of the compare value.	0x0000 FFFF	21.5.5
MASK_H	R/W	0x014	Mask MSB register. This register holds the 16 MSBs of the mask value. A 1 written to any bit will force a compare on the corresponding bit of the counter and compare register.	0	21.5.6
COUNTER_H	R/W	0x01C	Counter MSB register. 16 MSBs of the counter.	0	21.5.7

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

21.5.1 RI Compare Value LSB register

Table 403. RI Compare Value LSB register (COMPVAL, address 0x4002 D000) bit description

Bit	Symbol	Description	Reset value
31:0	RICOMP	Compare register. Holds the 32 LSBs of the value which is compared to the counter.	0xFFFF FFFF

21.5.2 RI Mask LSB register

Table 404. RI Mask LSB register (MASK, address0x4002 D004) bit description

Bit	Symbol	Description	Reset value
31:0	RIMASK	Mask register. This register holds the 32 LSBs of the mask value. A one written to any bit overrides the result of the comparison for the corresponding bit of the counter and compare register (causes the comparison of the register bits to be always true).	0

21.5.3 RI Control register

Table 405. RI Control register (CTRL, address 0x4002 D008) bit description

Bit	Symbol	Value	Description	Reset value
0	RITINT		Interrupt flag	0
		1	This bit is set to 1 by hardware whenever the counter value equals the masked compare value specified by the contents of RICOMPVAL and RIMASK registers. Writing a 1 to this bit will clear it to 0. Writing a 0 has no effect.	
		0	The counter value does not equal the masked compare value.	

Table 405. RI Control register (CTRL, address 0x4002 D008) bit description

Bit	Symbol	Value	Description	Reset value
1	RITENCLR		Timer enable clear	
		1	The timer will be cleared to 0 whenever the counter value equals the masked compare value specified by the contents of COMPVAL/COMPVAL_H and MASK/MASK_H registers. This will occur on the same clock that sets the interrupt flag.	0
		0	The timer will not be cleared to 0.	
2	RITENBR		Timer enable for debug	1
		1	The timer is halted when the processor is halted for debugging.	
		0	Debug has no effect on the timer operation.	
3	RITEN		Timer enable.	1
		1	Timer enabled. **Remark:** This can be overruled by a debug halt if enabled in bit 2.	
		0	Timer disabled.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

21.5.4 RI Counter LSB register

Table 406. RI Counter register (COUNTER, address 0x4002 D00C) bit description

Bit	Symbol	Description	Reset value
31:0	RICOUNTER	32 LSBs of the up counter. Counts continuously unless RITEN bit in CTRL register is cleared or debug mode is entered (if enabled by the RITNEBR bit in RICTRL). Can be loaded to any value in software.	0

21.5.5 RI Compare Value MSB register

Table 407. RI Compare Value MSB register (COMPVAL_H, address 0x4002 D010) bit description

Bit	Symbol	Description	Reset value
15:0	RICOMP	Compare value MSB register. Holds the 16 MSBs of the value which is compared to the counter.	0x0000 FFFF
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

21.5.6 RI Mask MSB register

Table 408. RI Mask MSB register (MASK_H, address 0x4002 D014) bit description

Bit	Symbol	Description	Reset value
15:0	RIMASK	Mask register. This register holds the 16 MSBs of the mask value. A one written to any bit overrides the result of the comparison for the corresponding bit of the counter and compare register (causes the comparison of the register bits to be always true).	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

21.5.7 RI Counter MSB register

Table 409. RI Counter MSB register (COUNTER_H, address 0x4002 D01C) bit description

Bit	Symbol	Description	Reset value
15:0	RICOUNTER	16 LSBs of the up counter. Counts continuously unless RITEN bit in RICTRL register is cleared or debug mode is entered (if enabled by the RITNEBR bit in RICTRL). Can be loaded to any value in software.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

21.6 RI timer operation

Following reset, the counter begins counting up from 0. (The RIT bit must be set in the AHBCLKCTRL1 register to enable the clock to the RIT.) Whenever the counter value equals the 48-bit value programmed into the COMPVAL and COMPVAL_H registers, the interrupt flag will be set. Any bit or combination of bits can be removed from this comparison (i.e. forced to compare) by writing a 1 to the corresponding bit(s) in the MASK and MASK_H registers. If the RITENCLR bit is low (default state), a valid comparison ONLY causes the interrupt flag to be set. It has no effect on the count sequence. Counting continues as usual. When the counter reaches 0xFFFF FFFF FFFF it rolls-over to 0 on the next clock and continues counting. If the RITENCLR bit is set to 1 a valid comparison will also cause the counter to be reset to zero. Counting will resume from there on the next clock edge.

Counting can be halted in software by writing a 0 to the RITEN bit. Counting will also be halted when the processor is halted for debugging provided the RITENBR bit is set. Both the RITEN and RITENBR bits are set on reset.

The interrupt flag can be cleared in software by writing a 1 to the RITINT bit.

Software must stop the counter before reloading it with a new value.

The counter (COUNTER/COUNTER_H), COMPVAL/COMPVAL_H registers, MASK/MASK_H registers, and the CTRL register can all be read by software at any time.

22.1 How to read this chapter

The system tick timer (SysTick timer) is present on all LPC546xx devices.

22.2 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the SYSTICKCLK.

22.3 Basic configuration

Configuration of the system tick timer is accomplished as follows:

1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register). The system tick timer clock is fixed to half the frequency of the system clock.
3. Enable the clock source for the SysTick timer in the SYST_CSR register.

22.4 General description

The block diagram of the SysTick timer is shown below in the Figure 52.

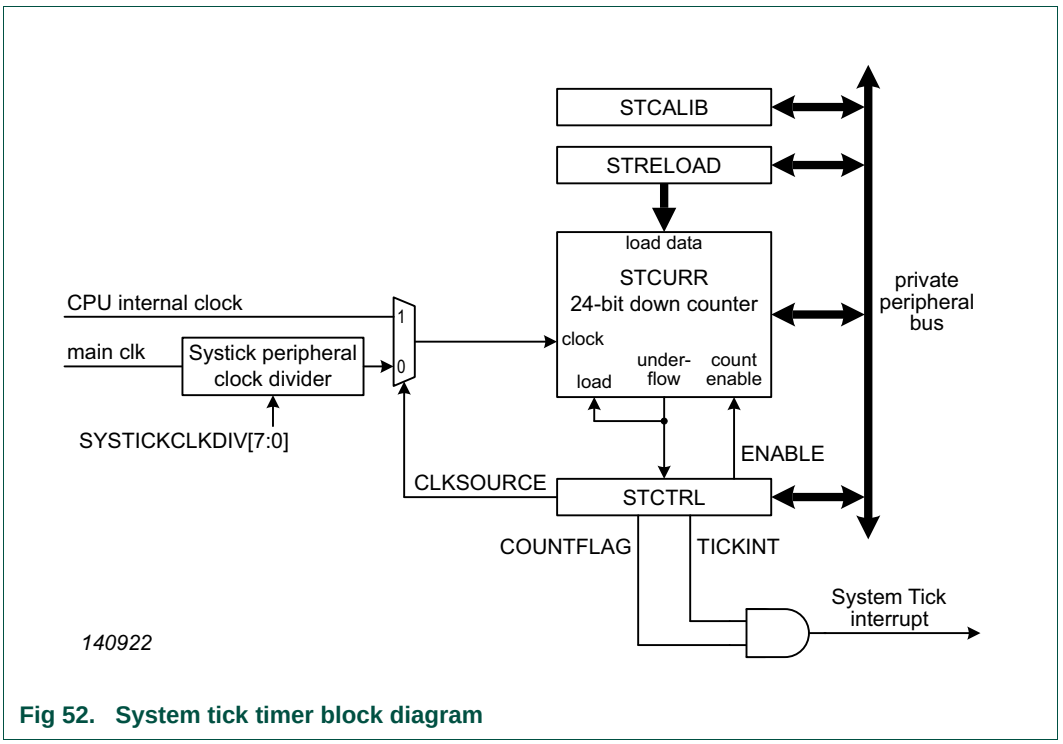


Fig 52. System tick timer block diagram

The SysTick timer is an integral part of the Cortex-M4. The SysTick timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-based devices. The SysTick timer can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to the appropriate ARM Cortex User Guide for details.

22.5 Register description

The systick timer registers are located on the private peripheral bus of each CPU (see Figure 7).

Table 410. Register overview: SysTick timer (base address 0xE000 E000)

Name	Access	Offset	Description	Reset value[1]	Section
SYST_CSR	R/W	0x010	System Timer Control and status register	0	22.5.1
SYST_RVR	R/W	0x014	System Timer Reload value register	0	22.5.2
SYST_CVR	R/W	0x018	System Timer Current value register	0	22.5.3
SYST_CALIB	RO	0x01C	System Timer Calibration value register	0	22.5.4

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

22.5.1 System Timer Control and status register

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the CPU.

This register determines the clock source for the system tick timer.

Table 411. SysTick Timer Control and status register (SYST_CSR, offset 0x010) bit description

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0
2	CLKSOURCE	System Tick clock source selection. When 1, the system clock is selected. When 0, the output clock from the system tick clock divider (SYSTICKCLKDIV, see Figure 52 and Table 164) is selected as the reference clock. **Remark:** When the system tick clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output.	0
15:3	-	Reserved. Read value is undefined, only zero should be written.	NA
16	COUNTFLAG	Returns 1 if the SysTick timer counted to 0 since the last read of this register. Cleared automatically when read or when the counter is cleared.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

22.5.2 System Timer Reload value register

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 412. System Timer Reload value register (SYST_RVR, offset 0x014) bit description

Bit	Symbol	Description	Reset value
23:0	RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.	0
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

22.5.3 System Timer Current value register

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 413. System Timer Current value register (SYST_CVR, offset 0x018) bit description

Bit	Symbol	Description	Reset value
23:0	CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in SYST_CSR.	0
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

22.5.4 System Timer Calibration value register

The value of the SYST_CALIB register is read-only and is provided by the value of the SYSTCKCAL register in the system configuration block (see Table 122).

Table 414. System Timer Calibration value register (SYST_CALIB, offset 0x01C) bit description

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value from the SYSTCKCAL register in the SYSCON block. This field is loaded from the SYSTCKCAL register in Syscon.	0
29:24	-	Reserved. Read value is undefined, only zero should be written.	NA
30	SKEW	Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is loaded from the SYSTCKCAL register in Syscon. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.	0
31	NOREF	Indicates whether an external reference clock is available. This bit is loaded from the SYSTCKCAL register in Syscon. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0

22.6 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Figure 7) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

22.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value calculated as shown below to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System clock = 72 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

SYST_RVR = (system clock frequency × 10 ms) −1 = (72 MHz × 10 ms) −1 = 720000 −1 = 719999 = 0x000A FC7F

System tick timer clock = 24 MHz

Program the SYST_CSR register with the value 0x3 which selects the clock from the system tick clock divider (see Section 7.5.44 “SYSTICK clock divider register”) as the clock source and enables the SysTick timer and the SysTick timer interrupt. Use DIV = 3.

SYST_RVR = (system tick timer clock frequency × 10 ms) −1 = (24 MHz × 10 ms) −1 = 240000 −1 = 239999 = 0x0003 A97F

System clock = 12 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

In this case the system clock is derived from the FRO 12 MHz clock.

SYST_RVR = (system clock frequency × 10 ms) −1 = (12 MHz × 10 ms) −1 = 120000 −1 = 119999 = 0x0001 D4BF

23.1 How to read this chapter

The Micro-Tick Timer is available on all LPC546xx devices.

23.2 Features

- Ultra simple, ultra-low power timer that can run and wake up the device in reduced power modes other than deep power-down.
- Write once to start.
- Interrupt or software polling.
- Four capture registers that can be triggered by external pin transitions.

23.3 Basic configuration

Configure the Micro-Tick Timer as follows:

- Set the UTICK bit in the AHBCLKCTRL1 register (Table 140) to enable the clock to the Micro-Tick Timer register interface.
- The Micro-Tick Timer provides an interrupt to the NVIC, see Chapter 6 “LPC546xx Nested Vectored Interrupt Controller (NVIC)”.
- To enable Micro-Tick timer interrupts for waking up from deep-sleep, enable the interrupts in the STARTER0 register (Table 222) and the NVIC.
- Configure the pin functions of any Micro-tick Timer capture pins that will be used via IOCON, see Chapter 10 “LPC546xx I/O pin configuration (IOCON)”.
- Enable the Watchdog oscillator that provides the Micro-Tick Timer clock in the syscon block using the Power API. See Chapter 9 “LPC546xx Power profiles/Power control API”.

23.4 General description

Figure 53 shows a conceptual view of the Micro-tick Timer.

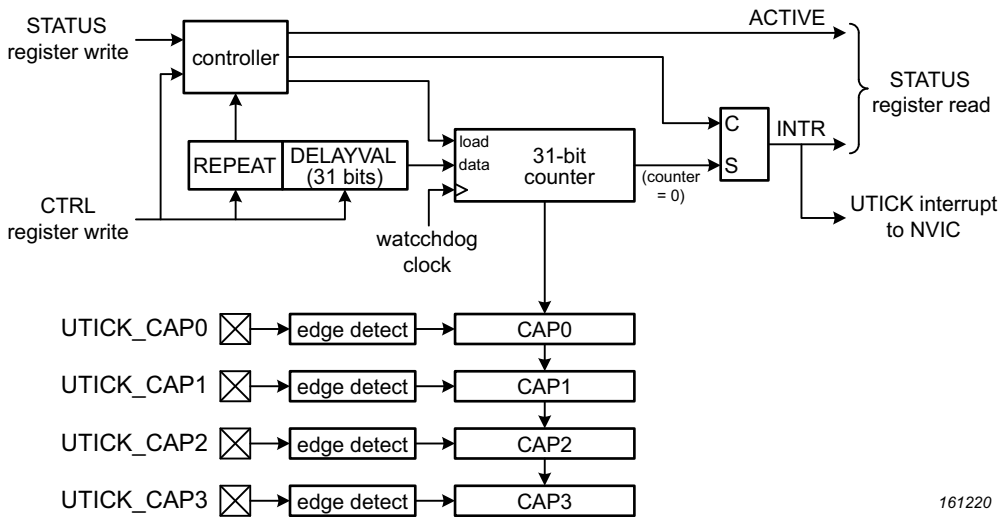


Fig 53. MIcro-Tick Timer block diagram

23.5 Pin description

Table 415 gives a summary of pins related to the Micro-tick Timer.

Table 415. Micro-tick Timer pin description

Pin	Type	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	Input	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.

23.6 Register description

The Micro-Tick Timer contains the registers shown in Table 416. Note that the Micro-Tick Timer operates from a different (typically slower) clock than the CPU and bus systems. This means there may be a synchronization delay when accessing Micro-Tick Timer registers.

Table 416. Register overview: Micro-Tick Timer (base address 0x4000 E000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register.	0	23.6.1
STAT	R/W	0x004	Status register.	0	23.6.2
CFG	R/W	0x008	Capture configuration register.	0	23.6.3
CAPCLR	WO	0x00C	Capture clear register.	NA	23.6.4
CAP0	RO	0x010	Capture register 0.	0	23.6.5
CAP1	RO	0x014	Capture register 1.	0	23.6.5
CAP2	RO	0x018	Capture register 2.	0	23.6.5
CAP3	RO	0x01C	Capture register 3.	0	23.6.5

23.6.1 CTRL register

This register controls the Micro-tick Timer. Any write to the CTRL register resets the counter, meaning a new interval will be measured if one was in progress.

Table 417. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Description	Reset value
30:0	DELAYVAL	Tick interval value. The delay will be equal to DELAYVAL + 1 periods of the timer clock. The minimum usable value is 1, for a delay of 2 timer clocks. A value of 0 stops the timer.	0
31	REPEAT	Repeat delay. 0 = One-time delay. 1 = Delay repeats continuously.	0

23.6.2 Status register

This register provides status for the Micro-tick Timer.

Table 418. Status register (STAT, offset 0x004) bit description

Bit	Symbol	Description	Reset value
0	INTR	Interrupt flag. 0 = No interrupt is pending. 1 = An interrupt is pending. A write of any value to this register clears this flag.	0
1	ACTIVE	Active flag. 0 = The Micro-Tick Timer is stopped. 1 = The Micro-Tick Timer is currently active.	0
31:2	-	Reserved	-

23.6.3 Capture configuration register

This register allows enabling Micro-tick capture functions and selects the polarity of the capture triggers.

Table 419. Capture configuration register (CFG, offset 0x008) bit description

Bit	Symbol	Description	Reset value
0	CAPEN0	Enable Capture 0. 1 = Enabled, 0 = Disabled.	0
1	CAPEN1	Enable Capture 1. 1 = Enabled, 0 = Disabled.	0
2	CAPEN2	Enable Capture 2. 1 = Enabled, 0 = Disabled.	0
3	CAPEN3	Enable Capture 3. 1 = Enabled, 0 = Disabled.	0
7:4	-	Reserved	-
8	CAPPOL0	Capture Polarity 0. 0 = Positive edge capture, 1 = Negative edge capture.	0
9	CAPPOL1	Capture Polarity 1. 0 = Positive edge capture, 1 = Negative edge capture.	0
10	CAPPOL2	Capture Polarity 2. 0 = Positive edge capture, 1 = Negative edge capture.	0
11	CAPPOL3	Capture Polarity 3. 0 = Positive edge capture, 1 = Negative edge capture.	0
31:12	-	Reserved	-

23.6.4 Capture clear register

This read-only register allows clearing previous capture values, allowing new captures to take place.

Table 420. Capture clear register (CAPCLR, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
0	CAPCLR0	Clear capture 0. Writing 1 to this bit clears the CAP0 register value.	NA
1	CAPCLR1	Clear capture 1. Writing 1 to this bit clears the CAP1 register value.	NA
2	CAPCLR2	Clear capture 2. Writing 1 to this bit clears the CAP2 register value.	NA
3	CAPCLR3	Clear capture 3. Writing 1 to this bit clears the CAP3 register value.	NA
31:4	-	Reserved	-

23.6.5 Capture registers

This register contains the Micro-tick time value based on any previously capture events.Each Capture register is associated with one of the capture trigger inputs.

Table 421. Capture registers (CAP[0:3], offsets [0x010:0x01C]) bit description

Bit	Symbol	Description	Reset value
30:0	CAP_VALUE	Capture value for the related capture event (UTICK_CAPn. Note: the value is 1 lower than the actual value of the Micro-tick Timer at the moment of the capture event.	0
31	VALID	Capture Valid. When 1, a value has been captured based on a transition of the related UTICK_CAPn pin. Cleared by writing to the related bit in the CAPCLR register.	0

24.1 How to read this chapter

Multiple Flexcomm Interfaces are available on all LPC546xx devices.

24.2 Introduction

Each Flexcomm Interface provides one peripheral function from a choice of several, chosen by the user. This chapter describes the overall Flexcomm Interface and how to choose peripheral functions. Details of the different peripherals are found in separate chapters for each type.

24.3 Features

Each Flexcomm Interface provides a choice of peripheral functions, one of which must be chosen by the user before the function can be configured and used.

- USART with asynchronous operation or synchronous master or slave operation.
- SPI master or slave, with up to 4 slave selects.
- I^2C, including separate master, slave, and monitor functions.
- Some Flexcomm Interfaces may also provide an I^2S function. If so, there may be from one to four I2S channel pairs, one of which may optionally be a master and the rest slaves, configured together for either transmit or receive.
- Data for USART, SPI, and I^2S traffic uses the Flexcomm Interface FIFO. The I^2C function does not use the FIFO.

24.4 Basic configuration

The Flexcomm Interface is configured as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 or AHBCLKCTRL2 register (Section 7.5.20 or Section 7.5.21).
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 7. Also see Section 7.5.37.
3. Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 24.7.1).
4. See specific peripheral chapters for information on configuring those peripherals: Chapter 25 “LPC546xx USARTs”, Chapter 26 “LPC546xx Serial Peripheral Interfaces (SPI)”, Chapter 27 “LPC546xx I^2C-bus interfaces”, Chapter 28 “LPC546xx I^2S interface”.

   Remark: The Flexcomm Interface function clock frequency should not be higher than 48 MHz.

24.5 Architecture

The overall structure of one Flexcomm Interface is shown in Figure 54.

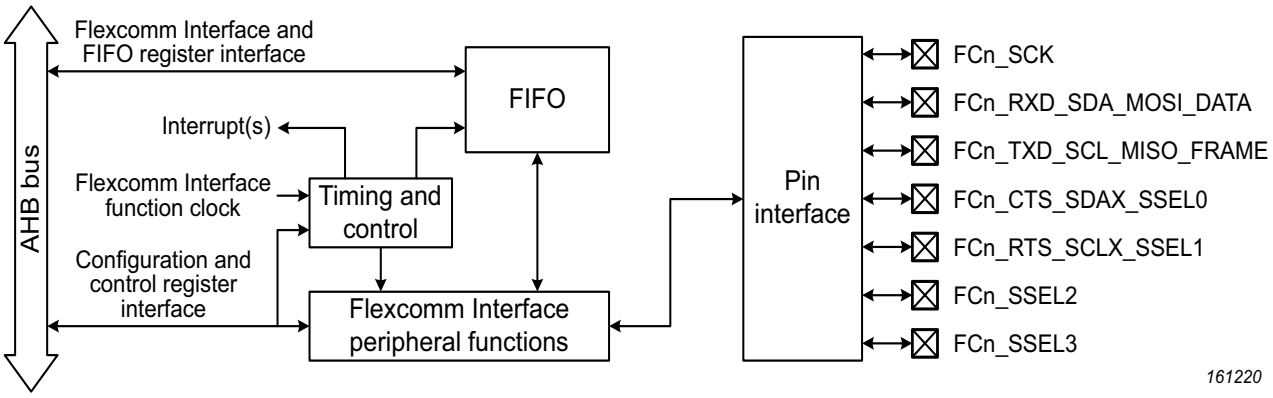


Fig 54. Flexcomm Interface block diagram

24.5.1 Function Summary

LPC546xx devices include Flexcomm Interfaces and functions as shown in Table 422. Specific part numbers and package variations may include a subset of this list.

Table 422. Flexcomm Interface base addresses and functions

Flexcomm Interface number	Base address	USART Chapter 25	SPI Chapter 26	I^2C Chapter 27	I^2S Chapter 28
0	0x4008 6000	Yes	Yes	Yes	
1	0x4008 7000	Yes	Yes	Yes, special I^2C pins available	
2	0x4008 8000	Yes	Yes	Yes, special I^2C pins available	
3	0x4008 9000	Yes	Yes	Yes	
4	0x4008 A000	Yes	Yes	Yes	
5	0x4009 6000	Yes	Yes	Yes	
6	0x4009 7000	Yes	Yes	Yes	Yes, 4 channel pair
7	0x4009 8000	Yes	Yes	Yes	Yes, 4 channel pair
8	0x4009 9000	Yes	Yes	Yes	
9	0x4009 A000	Yes	Yes	Yes	

24.5.2 Choosing a peripheral function

A specific peripheral function, from among those supported by a particular Flexcomm Interface, is selected by software writing to the PSELID register. Reading the PSELID register provides information on which peripheral functions are available on that Flexcomm Interface.

Once a specific peripheral function has been selected, the PID register will supply an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

24.5.3 FIFO usage

Refer to the chapter for a specific peripheral function for information on how the FIFO is used (see Table 422).

24.5.4 DMA

The Flexcomm Interface generates DMA requests if desired, based on a selectable FIFO level. Refer to the chapter for a specific peripheral function for information on how the FIFO is used (see Table 422).

24.5.5 AHB bus access

Generally, the bus interface to the registers contained in the Flexcomm Interface (including its serial peripheral functions) support only word writes. Byte and halfword writes should not be used. An exception is that the FIFOWR register, when the Flexcomm Interface is configured for use as an SPI interface, also allows byte and halfword writes. This allows support for control information embedded in DMA buffers, for example. See Section 26.6.14 “FIFO write data register” for more information.

24.6 Pin description

Each Flexcomm Interface allows up to 7 pin connections. Specific uses of a Flexcomm Interface typically do not use all of these, and some Flexcomm Interface instances may not provide a means to connect all functions to device pins. Pin usage for a specific peripheral function is described in the chapter for that peripheral.

Table 423. Flexcomm Interface pin description

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I^2S function (if present).
RXD_SDA_MOSI or RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I^2S function (if present).
TXD_SCL_MISO or TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I^2C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I^2S function (if present).
CTS_SDA_SSEL0	Input	Clear To Send input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Slave Select 0 input or output for the SPI function.
RTS_SCL_SSEL1	Output	Request To Send output for the USART function.
	I/O	SCL (clock) input/output for the I^2C function.
	I/O	Slave Select 1 input or output for the SPI function.
SSEL2	I/O	Slave Select 2 input or output for the SPI function.
SSEL3	I/O	Slave Select 3 input or output for the SPI function.

24.7 Register description

Each Flexcomm Interface contains registers that are related to configuring the Flexcomm Interface to do a specific peripheral function and other registers related to peripheral FIFOs and data access. The latter depend somewhat on the chosen peripheral functions and are described in the chapters for each specific function (USART, SPI, I2C, and I^2S if present in a specific Flexcomm Interface). The Flexcomm Interface registers that identify and configure the Flexcomm Interface are shown in Table 424.

The base addresses of all Flexcomm Interfaces may be found in Table 422.

Table 424. Register map for the first channel pair within one Flexcomm Interface

Name	Access	Offset	Description	Reset Value [1]	Section
PSELID	R/W	0xFF8	Peripheral Select and Flexcomm Interface ID register.	0	24.7.1
PID	RO	0xFFC	Peripheral identification register.	Section 24.7.2	24.7.2

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

24.7.1 Peripheral Select and Flexcomm Interface ID register

The PSELID register identifies the Flexcomm Interface and provides information about which peripheral functions are supported by each Flexcomm Interface. It also provides the means to select one peripheral function for each Flexcomm Interface.

Table 425. Peripheral Select and Flexcomm Interface ID register (PSELID - offset 0xFF8) bit description

Bit	Symbol	Value	Description	Reset Value
2:0	PERSEL		Peripheral Select. This field is writable by software.	0
		0x0	No peripheral selected.	
		0x1	USART function selected.	
		0x2	SPI function selected.	
		0x3	I^2C function selected.	
		0x4	I^2S transmit function selected.	
		0x5	I^2S receive function selected.	
		0x6	Reserved	
		0x7	Reserved	
3	LOCK		Lock the peripheral select. This field is writable by software.	0
		0	Peripheral select can be changed by software.	
		1	Peripheral select is locked and cannot be changed until this Flexcomm Interface or the entire device is reset.	
4	USARTPRESENT		USART present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the USART function.	
		1	This Flexcomm Interface includes the USART function.	
5	SPIPRESENT		SPI present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the SPI function.	
		1	This Flexcomm Interface includes the SPI function.	
6	I2CPRESENT		I^2C present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the I^2C function.	
		1	This Flexcomm Interface includes the I^2C function.	
7	I2SPRESENT		I^2S present indicator. This field is Read-only.	0
		0	This Flexcomm Interface does not include the I^2S function.	
		1	This Flexcomm Interface includes the I^2S function.	
11:8	-		Reserved. Read value is undefined, only zero should be written.	NA
31:12	ID		Flexcomm Interface ID.	0x00101

24.7.2 Peripheral identification register

This register is read-only and will read as 0 until a specific Flexcomm Interface function is selected via the PID register. Once the Flexcomm Interface is configured for a function, this register confirms the selection by returning the module ID for that function, and identifies the revision of that function. A software driver could make use of this information register to implement module type or revision specific behavior.

Table 426. Peripheral identification register (PID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	-	-	0
11:8	Minor_Rev	Minor revision of module implementation.	See specific device chapter
15:12	Major_Rev	Major revision of module implementation.	See specific device chapter
31:16	ID	Module identifier for the selected function.	See specific device chapter

25.1 How to read this chapter

USART functions are available on all LPC546xx devices as a selectable function in each Flexcomm Interface peripheral. Up to 10 Flexcomm Interfaces are available.

25.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits.
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare.
- RS-485 transceiver output enable.
- Parity generation and checking: odd, even, or none.
- Software selectable oversampling from 5 to 16 clocks in asynchronous mode.
- One transmit and one receive data buffer.
- The USART function supports separate transmit and receive FIFO with 16 entries each.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator.
- Autobaud mode for automatic baud rate detection.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected (among others).
- USART transmit and receive functions can operated with the system DMA controller.
- Loopback mode for testing of data and flow control.

25.3 Basic configuration

Initial configuration of a USART peripheral is accomplished as follows:

- If needed, use the PRESETCTRL1 or PRESETCTRL2 register (Table 130 or Table 131) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 24.7.1).
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
  - In the AHBCLKCTRL1 or AHBCLKCTRL2 register (Table 140 or Table 141), set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC (Table 88).
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 10 “LPC546xx I/O pin configuration (IOCON)”.
  - Configure the Flexcomm Interface clock and USART baud rate. See Section 25.3.1.

    Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.
- Configure the USART to wake up the part from low power modes. See Section 25.3.2.
- Configure the USART to receive and transmit data in synchronous slave mode. See Section 25.3.2.

25.3.1 Configure the Flexcomm Interface clock and USART baud rate

Each Flexcomm Interface has a separate clock selection, which can include a shared fractional divider (also see Section 25.7.2.3 “32 kHz mode”). The function clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows:

1. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider (FRG, controlled by Syscon register FRGCTRL). The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

   Flexcomm Interface clock = (FRG input clock) / (1+(MULT / DIV))

   The following rules apply for MULT and DIV:

   - Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
   - Set the MULT to any value between 0 and 255.

   See Table 177 for more information on the FRG.
2. In asynchronous mode: configure the baud rate divider BRGVAL in the BRG register. The baud rate divider divides the Flexcomm Interface function clock (FCLK) to create the clock needed to produce the desired baud rate.

   Generally: baud rate = [FCLK / oversample rate] / BRG divide

   With specific register values: baud rate = [FCLK / (OSRVAL+1)] / (BRGVAL + 1)

Generally: BRG divide = [FCLK / oversample rate] / baud rate

With specific register values: BRGVAL = [[FCLK / (OSRVAL + 1)] / baud rate] - 1

See Section 25.6.6 “USART Baud Rate Generator register”.

3. In synchronous master mode: The serial clock is Un_SCLK = FCLK / (BRGVAL+1).

The USART can also be clocked by the 32 kHz RTC oscillator. Set the MODE32K bit to enable this 32 kHz mode. See also Section 25.7.2.3 “32 kHz mode”.

For details on the clock configuration see:

Section 25.7.2 “Clocking and baud rates”

25.3.2 Configure the USART for wake-up

A USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in deep-sleep mode.

  As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.
- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

25.3.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 430.
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from sleep mode.

25.3.2.2 Wake-up from deep-sleep mode

- Configure the USART in synchronous slave mode. See Table 430. The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt in the STARTER0 register. See Table 222.
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from deep-sleep mode on all events that cause an interrupt and are enabled. Typical wake-up events are:
  - A start bit has been received.
  - Received data becomes available.

- In synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
- A change in the state of the CTS pin if the CTS function is connected.
- **Remark:** By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

25.4 Pin description

The USART receive, transmit, and control signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. See the IOCON description (Chapter 10 “LPC546xx I/O pin configuration (IOCON)”) to assign the USART functions to pins on the device package. Recommended IOCON settings are shown in Table 428.

Table 427. USART pin description

Pin	Type	Name used in Pin Configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO_WS	Transmitter output for USART on Flexcomm Interface n. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI_DATA	Receiver input for USART on Flexcomm Interface n. Serial receive data.
$\overline{\text{RTS}}$	O	FCn_RTS_SCL_SSEL1	Request To Send output for USART on Flexcomm Interface n. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
$\overline{\text{CTS}}$	I	FCn_CTS_SDA_SSEL0	Clear To Send input for USART on Flexcomm Interface n. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	FCn_SCK	Serial clock input/output for USART on Flexcomm Interface n in synchronous mode. Clock input or output in synchronous mode. **Remark:** When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the USART to work properly.

Table 428. Suggested USART pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1.
10	SLEW: Generally set to 0.	Not used, set to 0.	I2CDRIVE: Set to 0.
9	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Not used, set to 0.	I2CSLEW: Set to 1.
5:4	MODE: Set 00 (to pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for USART input or output.	A reasonable choice for USART input or output.	Not recommended for USART functions that can be outputs in the chosen mode.

25.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver FIFO to await access by the CPU or DMA controller.

The USART transmitter block accepts data written by the CPU or DMA controller to the transmit FIFO. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

The Baud Rate Generator block divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the USART function clock. The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

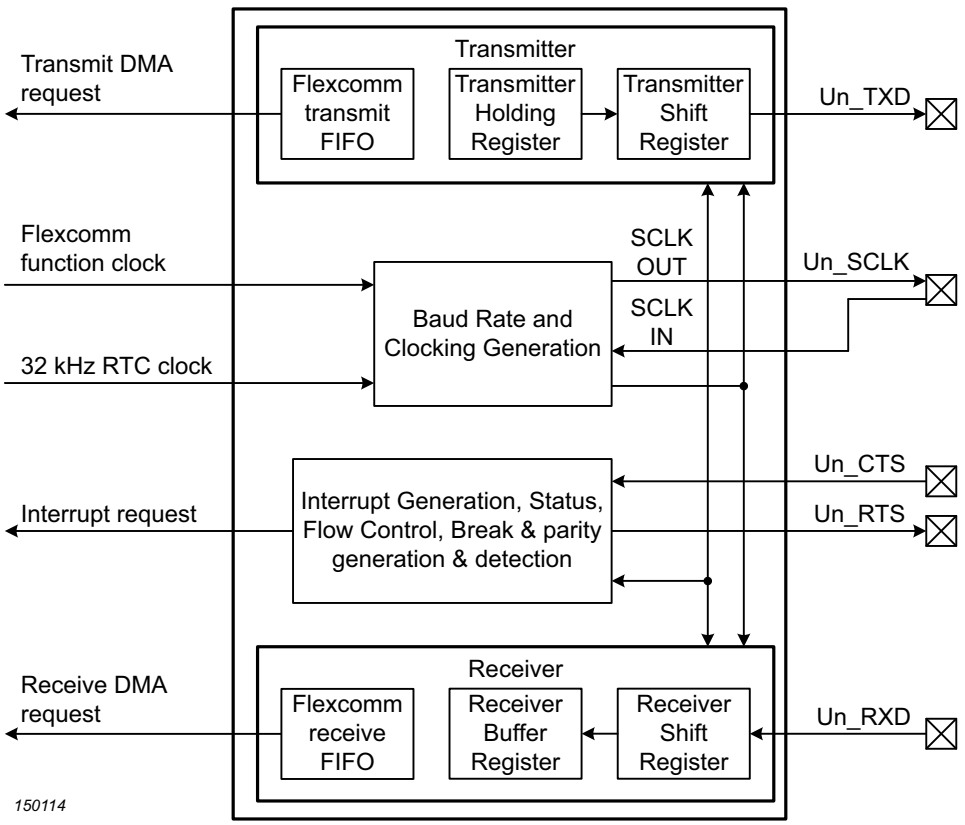


Fig 55. USART block diagram

25.6 Register description

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits. Address offsets are within the related Flexcomm Interface address space **after** the USART function has been selected for that Flexcomm Interface (see Section 24.5.1 for a summary of Flexcomm Interface addresses).

Table 429. USART register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the USART function:					
CFG	R/W	0x000	USART Configuration register. Basic USART configuration settings that typically are not changed during operation.	0	25.6.1
CTL	R/W	0x004	USART Control register. USART control settings that are more likely to change during operation.	0	25.6.2
STAT	R/W	0x008	USART Status register. The complete status value can be read here. Writing ones clears some bits in the register. Some bits can be cleared by writing a 1 to them.	0x0A	25.6.3
INTENSET	R/W	0x00C	Interrupt Enable read and Set register for USART (not FIFO) status. Contains individual interrupt enable bits for each potential USART interrupt. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0	25.6.4
INTENCLR	WO	0x010	Interrupt Enable Clear register. Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.	-	25.6.5
BRG	R/W	0x020	Baud Rate Generator register. 16-bit integer baud rate divisor value.	0	25.6.6
INTSTAT	RO	0x024	Interrupt status register. Reflects interrupts that are currently enabled.	0x05	25.6.7
OSR	R/W	0x028	Oversample selection register for asynchronous communication.	0xF	25.6.8
ADDR	R/W	0x02C	Address register for automatic address matching.	0	25.6.9
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0	25.6.10
FIFOSTAT	R/W	0xE04	FIFO status register.	0x18	25.6.11
FIFOTRIG	R/W	0xE08	FIFO trigger settings for interrupt and DMA request.	0	25.6.12
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0	25.6.13
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0	25.6.14
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0	25.6.15
FIFOWR	WO	0xE20	FIFO write data.	NA	25.6.16
FIFORD	RO	0xE30	FIFO read data.	NA	25.6.17
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	NA	25.6.18
ID register:					
ID	RO	0xFFC	USART module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when USART is selected.	0xE000 0000	25.6.19

25.6.1 USART Configuration register

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured once in an application.

Remark: Only the CFG register can be written when the ENABLE bit = 0. CFG can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

Remark: If software needs to change configuration values, the following sequence should be used: 1) Make sure the USART is not currently sending or receiving data. 2) Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register). 3) Write the new configuration value, with the ENABLE bit set to 1.

Table 430. USART Configuration register (CFG, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset Value
0	ENABLE		USART Enable.	0
		0	Disabled. The USART is disabled and the internal state machine and counters are reset. While Enable = 0, all USART interrupts and DMA transfers are disabled. When Enable is set again, CFG and most other control bits remain unchanged. When re-enabled, the USART will immediately be ready to transmit because the transmitter has been reset and is therefore available.	
		1	Enabled. The USART is enabled for operation.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
3:2	DATALEN		Selects the data size for the USART.	0
		0x0	7 bit Data length.	
		0x1	8 bit Data length.	
		0x2	9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in the CTL register.	
		0x3	Reserved.	
5:4	PARITYSEL		Selects what type of parity is used by the USART.	0
		0x0	No parity.	
		0x1	Reserved.	
		0x2	Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even.	
		0x3	Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.	
6	STOPLEN		Number of stop bits appended to transmitted data. Only a single stop bit is required for received data.	0
		0	1 stop bit.	
		1	2 stop bits. This setting should only be used for asynchronous communication.	
7	MODE32K		Selects standard or 32 kHz clocking mode.	0
		0	Disabled. USART uses standard clocking.	
		1	Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.	
8	LINMODE		LIN break mode enable.	0
		0	Disabled. Break detect and generate is configured for normal operation.	
		1	Enabled. Break detect and generate is configured for LIN bus operation.	

Table 430. USART Configuration register (CFG, offset 0x000) bit description *...continued*

Bit	Symbol	Value	Description	Reset Value
9	CTSEN		CTS Enable. Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if loopback mode is enabled.	0
		0	No flow control. The transmitter does not receive any automatic flow control signal.	
		1	Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.	
10	-	-	Reserved. Read value is undefined, only zero should be written.	-
11	SYNCEN		Selects synchronous or asynchronous operation.	0
		0	Asynchronous mode.	
		1	Synchronous mode.	
12	CLKPOL		Selects the clock polarity and sampling edge of received data in synchronous mode.	0
		0	Falling edge. Un_RXD is sampled on the falling edge of SCLK.	
		1	Rising edge. Un_RXD is sampled on the rising edge of SCLK.	
13	-	-	Reserved. Read value is undefined, only zero should be written.	-
14	SYNCMST		Synchronous mode Master select.	0
		0	Slave. When synchronous mode is enabled, the USART is a slave.	
		1	Master. When synchronous mode is enabled, the USART is a master.	
15	LOOP		Selects data loopback mode.	0
		0	Normal operation.	
		1	Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (Un_TXD) is connected internally to serial input of the receive (Un_RXD). Un_TXD and Un_RTS activity will also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN.	
17, 16	-	-	Reserved. Read value is undefined, only zero should be written.	-
18	OETA		Output Enable Turnaround time enable for RS-485 operation.	0
		0	Disabled. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission.	
		1	Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is deasserted.	
19	AUTOADDR		Automatic Address matching enable.	0
		0	Disabled. When addressing is enabled by ADDRDET, address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address).	
		1	Enabled. When addressing is enabled by ADDRDET, address matching is done by hardware, using the value in the ADDR register as the address to match.	
20	OESEL		Output Enable Select.	0
		0	Standard. The RTS signal is used as the standard flow control function.	
		1	RS-485. The RTS signal configured to provide an output enable signal to control an RS-485 transceiver.	
21	OEPOL		Output Enable Polarity.	0
		0	Low. If selected by OESEL, the output enable is active low.	
		1	High. If selected by OESEL, the output enable is active high.	

Table 430. USART Configuration register (CFG, offset 0x000) bit description *...continued*

Bit	Symbol	Value	Description	Reset Value
22	RXPOL		Receive data polarity.	0
		0	Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
23	TXPOL		Transmit data polarity.	0
		0	Standard. The TX signal is sent out without change. This means that the TX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
31:24	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.2 USART Control register

The CTL register controls aspects of USART operation that are more likely to change during operation.

Table 431. USART Control register (CTL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset Value
0	-	-	Reserved. Read value is undefined, only zero should be written.	-
1	TXBRKEN		Break Enable.	0
		0	Normal operation.	
		1	Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN.	
2	ADDRDET		Enable address detect mode.	0
		0	Disabled. The USART presents all incoming data.	
		1	Enabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.	
5:3	-	-	Reserved. Read value is undefined, only zero should be written.	-
6	TXDIS		Transmit Disable.	0
		0	Not disabled. USART transmitter is not disabled.	
		1	Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.	
7	-	-	Reserved. Read value is undefined, only zero should be written.	-
8	CC		Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode.	0
		0	Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received.	
		1	Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD).	
9	CLRCCONRX		Clear Continuous Clock.	0
		0	No effect. No effect on the CC bit.	
		1	Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.	
15:10	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 431. USART Control register (CTL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset Value
16	AUTOBAUD		Autobaud enable.	0
		0	Disabled. USART is in normal operating mode.	
		1	Enabled. USART is in autobaud mode. This bit should only be set when the USART receiver is idle. The first start bit of RX is measured and used the update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR.	
31:17	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.3 USART Status register

The STAT register primarily provides a set of USART status flags (not including FIFO status) for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register (see Table 434).

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Table 432. USART Status register (STAT, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access [1]
0	-	Reserved. Read value is undefined, only zero should be written.	-	-
1	RXIDLE	Receiver Idle. When 0, indicates that the receiver is currently in the process of receiving data. When 1, indicates that the receiver is not currently in the process of receiving data.	1	RO
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	TXIDLE	Transmitter Idle. When 0, indicates that the transmitter is currently in the process of sending data.When 1, indicate that the transmitter is not currently in the process of sending data.	1	RO
4	CTS	This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. This will be the value of the CTS input pin unless loopback mode is enabled.	NA	RO
5	DELTACTS	This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software.	0	R/W1C
6	TXDISSTAT	Transmitter Disabled Status flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS bit in the CFG register (TXDIS = 1).	0	RO
9:7	-	Reserved. Read value is undefined, only zero should be written.	-	-
10	RXBRK	Received Break. This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16 bit times. Note that FRAMERRINT will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high.	0	RO
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs. Cleared by software.	0	R/W1C
12	START	This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from deep-sleep mode immediately when a start is detected. Cleared by software.	0	R/W1C
13	FRAMERRINT	Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0	R/W1C
14	PARITYERRINT	Parity Error interrupt flag. This flag is set when a parity error is detected in a received character.	0	R/W1C

Table 432. USART Status register (STAT, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access [1]
15	RXNOISEINT	Received Noise interrupt flag. Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.	0	R/W1C
16	ABERR	Auto baud Error. An auto baud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out.	0	R/W1C
31:17	-	Reserved. Read value is undefined, only zero should be written.	-	-

[1] RO = Read-Only, R/W1C = Write 1 to Clear.

25.6.4 USART Interrupt Enable read and set register

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register.

Table 433. USART Interrupt Enable read and set register (INTENSET, offset 0x00C) bit description

Bit	Symbol	Description	Reset Value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLEEN	When 1, enables an interrupt when the transmitter becomes idle (TXIDLE = 1).	0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTSEN	When 1, enables an interrupt when there is a change in the state of the CTS input.	0
6	TXDISEN	When 1, enables an interrupt when the transmitter is fully disabled as indicated by the TXDISINT flag in STAT. See description of the TXDISINT bit for details.	0
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRKEN	When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).	0
12	STARTEN	When 1, enables an interrupt when a received start bit has been detected.	0
13	FRAMERREN	When 1, enables an interrupt when a framing error has been detected.	0
14	PARITYERREN	When 1, enables an interrupt when a parity error has been detected.	0
15	RXNOISEEN	When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Table 432.	0
16	ABERREN	When 1, enables an interrupt when an auto baud error occurs.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.5 USART Interrupt Enable Clear register

The INTENCLR register is used to clear bits in the INTENSET register.

Table 434. USART Interrupt Enable clear register (INTENCLR, offset 0x010) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTSCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
6	TXDISCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRKCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
12	STARTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
13	FRAMERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
14	PARITYERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
15	RXNOISECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
16	ABERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.6 USART Baud Rate Generator register

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm Interface clock (FCLK) in order to produce the clock used for USART internal operations.

A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz.

Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data.

Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

For more information on USART clocking, see Section 25.7.2 and Section 25.3.1.

Remark: To change a baud rate after a USART is running, the following sequence should be used:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register).
3. Write the new BRGVAL.
4. Write to the CFG register to set the Enable bit to 1.

Table 435. USART Baud Rate Generator register (BRG, offset 0x020) bit description

Bit	Symbol	Description	Reset value
15:0	BRGVAL	This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0 = FCLK is used directly by the USART function. 1 = FCLK is divided by 2 before use by the USART function. 2 = FCLK is divided by 3 before use by the USART function. ... 0xFFFF = FCLK is divided by 65,536 before use by the USART function.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.7 USART Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 432 for detailed descriptions of the interrupt flags.

Table 436. USART Interrupt Status register (INTSTAT, offset 0x024) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLE	Transmitter Idle status.	0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTS	This bit is set when a change in the state of the CTS input is detected.	0
6	TXDISINT	Transmitter Disabled Interrupt flag.	0
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs.	0
12	START	This bit is set when a start is detected on the receiver input.	0
13	FRAMERRINT	Framing Error interrupt flag.	0
14	PARITYERRINT	Parity Error interrupt flag.	0
15	RXNOISEINT	Received Noise interrupt flag.	0
16	ABERRINT	Auto baud Error Interrupt flag.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.8 Oversample selection register

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Table 437. Oversample selection register (OSR, offset 0x028) bit description

Bit	Symbol	Description	Reset value
3:0	OSRVAL	Oversample Selection Value. 0 to 3 = not supported 0x4 = 5 function clocks are used to transmit and receive each data bit. 0x5 = 6 function clocks are used to transmit and receive each data bit. ... 0xF= 16 function clocks are used to transmit and receive each data bit.	0xF
31:4	-	Reserved, the value read from a reserved bit is not defined.	-

25.6.9 Address register

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

Table 438. Address register (ADDR, offset 0x02C) bit description

Bit	Symbol	Description	Reset value
7:0	ADDRESS	8-bit address used with automatic address matching. Used when address detection is enabled (ADDRDET in CTL = 1) and automatic address matching is enabled (AUTOADDR in CFG = 1).	0
31:8	-	Reserved, the value read from a reserved bit is not defined.	-

25.6.10 FIFO Configuration register

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 439. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0 = FIFO is configured as 16 entries of 8 bits. 0x1, 0x2, 0x3 = not applicable to USART.	NA	RO
11:6	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 7.5.96 “Hardware Wake-up control register”.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 7.5.96 “Hardware Wake-up control register”.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		

Table 439. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

25.6.11 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 440. FIFO status register (FIFOSTAT - offset 0xE04) bit description

Bit	Symbol	Description	Access	Reset value
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	R/W1C	0
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	R/W1C	0
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	RO	0
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	RO	1
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	RO	1
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	RO	0
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	RO	0
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	RO	0
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	RO	0
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

25.6.12 FIFO trigger level settings register

This register allows selecting when FIFO-level related interrupts occur.

Table 441. FIFO trigger level settings register (FIFOTRIG - offset 0xE08) bit description

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
10:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 “Hardware Wake-up control register”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 15 = generate an interrupt when the TX FIFO level decreases to 15 entries (is no longer full).	0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 “Hardware Wake-up control register”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 15 = generate an interrupt when the RX FIFO increases to 16 entries (has become full).	0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.13 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Table 442. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10) bit description

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.14 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 443. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14) bit description

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.15 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 25.6.11 and Section 25.6.12 for details.

Table 444. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18) bit description

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0
1	RXERR	RX FIFO error.	0
2	TXLVL	Transmit FIFO level interrupt.	0
3	RXLVL	Receive FIFO level interrupt.	0
4	PERINT	Peripheral interrupt.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.16 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO.

Table 445. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Description	Reset value
8:0	TXDATA	Transmit data to the FIFO.	NA

25.6.17 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO.

Table 446. FIFO read data register (FIFORD - offset 0xE30) bit description

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO. The number of bits used depends on the DATALEN and PARITYSEL settings.	NA
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag. This bit reflects the status for the data it is read along with from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	NA
14	PARITYERR	Parity Error status flag. This bit reflects the status for the data it is read along with from the FIFO. This bit will be set when a parity error is detected in a received character.	NA
15	RXNOISE	Received Noise flag. See description of the RxNoiseInt bit in Table 432.	NA
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

25.6.18 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 447. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO.	NA
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag.	NA
14	PARITYERR	Parity Error status flag.	NA
15	RXNOISE	Received Noise flag.	NA
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

25.6.19 Module identification register

The ID register identifies the type and revision of the USART module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 448. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE010

25.7 Functional description

25.7.1 AHB bus access

The bus interface to the USART registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the USART function.

25.7.2 Clocking and baud rates

In order to use the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see Section 25.3.1 “Configure the Flexcomm Interface clock and USART baud rate”.

25.7.2.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the function clock is not a good multiple of standard (or otherwise desirable) baud rates.

The FRG is typically set up to produce an integer multiple of the highest required baud rate, or a very close approximation. The BRG is then used to obtain the actual baud rate needed.

The FRG register controls the Fractional Rate Generator, which provides the base clock that may be used by any Flexcomm Interface. The Fractional Rate Generator creates a lower rate output clock by suppressing selected input clocks. When not needed, the value of 0 can be set for the FRG, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by 1 + (MULT / 256), where MULT is in the range of 1 to 255. This allows producing an output clock that ranges from the input clock divided by 1+1/256 to 1+255/256 (just more than 1 to just less than 2). Any further division can be done specific to each USART block by the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since USARTs normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

For setting up the fractional divider, see Section 7.5.37 and Section 7.5.57.

25.7.2.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see Section 25.6.6) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

25.7.2.3 32 kHz mode

In order to use a 32 kHz clock to operate a USART at any reasonable speed, a number of adaptations need to be made. First, 16x overclocking has to be abandoned. Otherwise, the maximum data rate would be very low. For the same reason, multiple samples of each

data bit must be reduced to one. Finally, special clocking has to be used for individual bit times because 32 kHz is not particularly close to an even multiple of any standard baud rate.

When 32 kHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error will be less than one half percent in this mode, provided the RTC oscillator is operating at the intended frequency of 32.768 kHz.

25.7.3 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling a that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

25.7.4 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and BRG, which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking in either case.

25.7.5 Flow control

The USART supports both hardware and software flow control.

25.7.5.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signalling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together.

Figure 56 shows an overview of RTS and CTS within the USART.

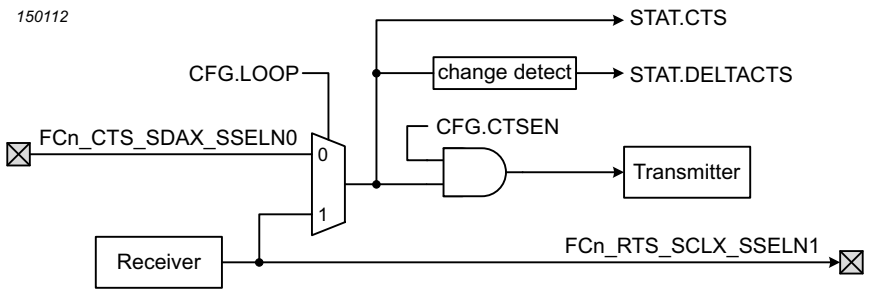


Fig 56. Hardware flow control using RTS and CTS

25.7.5.2 Software flow control

Software flow control could include XON / XOFF flow control, or other mechanisms. these are supported by the ability to check the current state of the CTS input, and/or have an interrupt when CTS changes state (via the CTS and DELTACTS bits, respectively, in the STAT register), and by the ability of software to gracefully turn off the transmitter (via the TXDIS bit in the CTL register).

25.7.6 Autobaud function

The autobaud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. Before an autobaud operation is requested, the BRG value must be set to 0. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Autobaud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that autobaud only be enabled when the USART receiver is idle. Once enabled, either data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off autobaud.

Autobaud has no meaning and should not be enabled when the USART is in synchronous mode.

25.7.7 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in Section 25.6.1 and the ADDR register in Section 25.6.9), as well as software address recognition (see the ADDRDET bit in the CTL register in Section 25.6.2).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register (Section 25.6.1). Data direction control can also be implemented in software using a GPIO pin.

25.7.8 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock. There is no oversampling in synchronous modes.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. In asynchronous modes, the USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

25.7.9 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the RX input remains low for 16 bit times. Both the beginning and end of a received break are noted by the DELTARXBRK status flag, which can be used as an interrupt. See Section 25.7.10 for details of LIN mode break.

In order to avoid corrupting any character currently being transmitted, it is recommended that the USART transmitter be disabled by setting the TXDIS bit in the CTL register, then waiting for the TXDISSTAT flag to be set prior to sending a break. Then a 1 may be written to the TXBRKEN bit in the CTL register. This sends a break until TXBRKEN is cleared, allowing any length break to be sent.

25.7.10 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way that break generation and detection is performed (see Section 25.7.9 for details of the standard break). When a break is requested by setting the TXBRKEN bit in the CTL register, then sending a dummy character, a 13 bit time break is sent. A received break is flagged when the RX input remains low for 11 bit times. As for non-LIN mode, a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the autobaud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as confirmation of success.

Wake-up for LIN can potentially be handled in a number of ways, depending on the system, and what clocks may be running in a slave device. For instance, as long as the USART is receiving internal clocks allowing it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, the GPIO function of the USART RX pin can be programmed to wake up the device.

26.1 How to read this chapter

SPI functions are available on all LPC546xx devices as a selectable function in each Flexcomm Interface. Up to 10 Flexcomm Interfaces are available.

26.2 Features

- Master and slave operation.
- Data transmits of 4 to 16 bits supported directly. Larger frames supported by software.
- The SPI function supports separate transmit and receive FIFOs with 8 entries each.
- Supports DMA transfers: SPIn transmit and receive functions can operated with the system DMA controller.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Up to four Slave Select input/outputs with selectable polarity and flexible usage.

Remark: Texas Instruments SSI and National Microwire modes are not supported.

26.3 Basic configuration

Initial configuration of an SPI peripheral is accomplished as follows:

- If needed, use the PRESETCTRL1 or PRESETCTRL2 register (Table 130) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 24.7.1).
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
  - In the AHBCLKCTRL1 or AHBCLKCTRL2 (Table 139) register, set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupts in the NVIC (Table 88).
  - Configure the required Flexcomm Interface pin functions through IOCON. See Section 26.4.
  - Configure the Flexcomm Interface clock and SPI data rate (see Section 26.7.4).

    Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.
  - Set the RXIGNORE bit to only transmit data and not read the incoming data. Otherwise, the transmit halts when the FIFORD buffer is full.
- Configure the SPI function to wake up the part from low power modes. See Section 26.3.1.

26.3.1 Configure the SPI for wake-up

In sleep mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the NVIC.

26.3.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode. See Table 452.
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from sleep mode.

26.3.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. See Table 452. The SCK function must be connected to a pin and the pin connected to the master.
- Enable the SPI interrupt in the STARTER0 register. See Table 222.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. Examples are the following wake-up events:
  - A change in the state of the SSEL pins.
  - Data available to be received.
  - Receive FIFO overflow.

26.4 Pin description

The SPI signals are movable Flexcomm Interface functions and are assigned to external pins via IOCON. See Chapter 10 “LPC546xx I/O pin configuration (IOCON)”. Recommended IOCON settings are shown in Table 450.

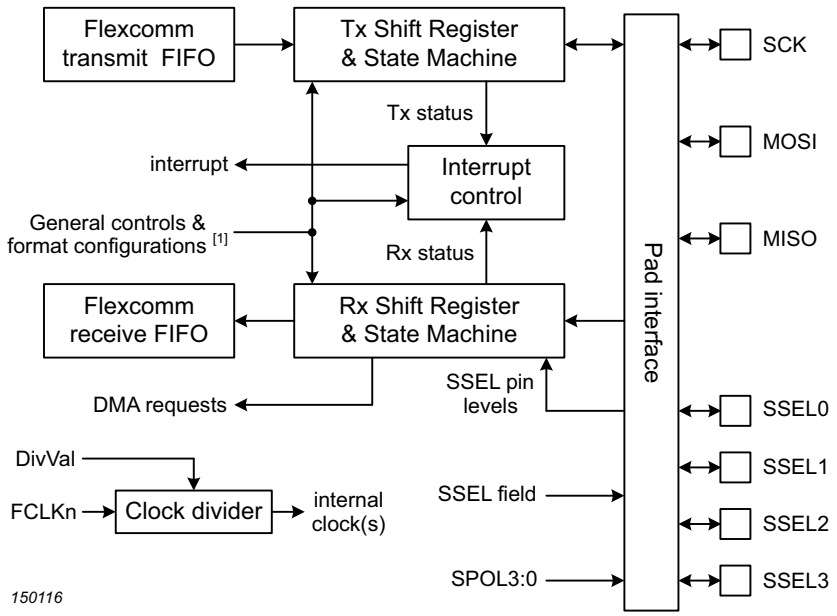
Table 449. SPI Pin Description

Function	Type	Pin name used in Pin Description chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for SPI on Flexcomm Interface n. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MOSI	I/O	FCn_RXD_SDA_MOSI or FCn_RXD_SDA_MOSI_DATA	Master Out Slave In for SPI on Flexcomm Interface n. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MISO	I/O	FCn_TXD_SCL_MISO or FCn_TXD_SCL_MISO_WS	Master In Slave Out for SPI on Flexcomm Interface n. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	FCn_CTS_SDA_SSEL0	Slave Select 0 for SPI on Flexcomm Interface n. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.
SSEL1	I/O	FCn_RTS_SCL_SSEL1	Slave Select 1 for SPI on Flexcomm Interface n.
SSEL2	I/O	FCn_SSEL2	Slave Select 2 for SPI on Flexcomm Interface n.
SSEL3	I/O	FCn_SSEL3	Slave Select 3 for SPI on Flexcomm Interface n.

Table 450. Suggested SPI pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1.
10	SLEW: Generally set to 0. Setting to 1 at higher SPI rates may improve performance.	Not used, set to 0.	I2CDRIVE: Set to 0
9	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
8	DIGIMODE: Set to 1.	DIGIMODE: Set to 1.	DIGIMODE: Set to 1.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
5:4	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for SPI input or output.	A reasonable choice for SPI input or output.	Not recommended for SPI functions that can be outputs in the chosen mode.

26.5 General description



(1) Includes CPOL, CPHA, LSBF, LEN, master, enable, transfer_delay, frame_delay, pre_delay, post_delay, SOT, EOT, EOF, RXIGNORE, individual interrupt enables.

Fig 57. SPI block diagram

26.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 451. SPI register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the SPI function:					
CFG	R/W	0x400	SPI Configuration register	0	26.6.1
DLY	R/W	0x404	SPI Delay register	0	26.6.2
STAT	R/W	0x408	SPI Status. Some status flags can be cleared by writing a 1 to that bit position.	0x0100	26.6.3
INTENSET	R/W	0x40C	SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0	26.6.4
INTENCLR	WO	0x410	SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.	-	26.6.5
DIV	R/W	0x424	SPI clock Divider	0	26.6.6
INTSTAT	RO	0x428	SPI Interrupt Status	0	26.6.7
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0	26.6.8
FIFOSTAT	R/W	0xE04	FIFO status register.	0x18	26.6.9
FIFOTRIG	R/W	0xE08	FIFO trigger level settings for interrupt and DMA request.	0	26.6.10
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0	26.6.11
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0	26.6.12
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0	26.6.13
FIFOWR	WO	0xE20	FIFO write data.	-	26.6.14
FIFORD	RO	0xE30	FIFO read data.	-	26.6.15
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	26.6.16
ID register:					
ID	RO	0xFFC	SPI module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when SPI is selected.	0xE020 0000	26.6.17

26.6.1 SPI Configuration register

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. See the description of the master idle status (MSTIDLE in Table 454) for more information.

Remark: A setup sequence is recommended for initial SPI setup (after the SPI function has been selected (see Chapter 26 "LPC546xx Serial Peripheral Interfaces (SPI)"), and when changes need to be made to settings in the CFG register after the interface has been in use. See the list below. In the case of changing existing settings, the interface should first be disabled by clearing the ENABLE bit once the interface is fully idle. See the description of the master idle status (MSTIDLE in Table 454) for more information.

- Disable the FIFO by clearing the ENABLETX and ENABLERX bits in FIFOCFG
- Setup the SPI interface in the CFG register, leaving ENABLE = 0.
- Enable the FIFO by setting the ENABLETX and/or ENABLERX bits in FIFOCFG
- Enable the SPI by setting the ENABLE bit in CFG.

Table 452. SPI Configuration register (CFG, offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		SPI enable.	0
		0	Disabled. The SPI is disabled and the internal state machine and counters are reset.	
		1	Enabled. The SPI is enabled for operation.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
2	MASTER		Master mode select.	0
		0	Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output.	
		1	Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.	
3	LSBF		LSB First mode enable.	0
		0	Standard. Data is transmitted and received in standard MSB first order.	
		1	Reverse. Data is transmitted and received in reverse order (LSB first).	
4	CPHA		Clock Phase select.	0
		0	Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	
		1	Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	
5	CPOL		Clock Polarity select.	0
		0	Low. The rest state of the clock (between transfers) is low.	
		1	High. The rest state of the clock (between transfers) is high.	
6	-	-	Reserved. Read value is undefined, only zero should be written.	-
7	LOOP		Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing.	0
		0	Disabled.	
		1	Enabled.	

Table 452. SPI Configuration register (CFG, offset 0x400) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
8	SPOL0		SSEL0 Polarity select.	0
		0	Low. The SSEL0 pin is active low.	
		1	High. The SSEL0 pin is active high.	
9	SPOL1		SSEL1 Polarity select.	0
		0	Low. The SSEL1 pin is active low.	
		1	High. The SSEL1 pin is active high.	
10	SPOL2		SSEL2 Polarity select.	0
		0	Low. The SSEL2 pin is active low.	
		1	High. The SSEL2 pin is active high.	
11	SPOL3		SSEL3 Polarity select.	0
		0	Low. The SSEL3 pin is active low.	
		1	High. The SSEL3 pin is active high.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.2 SPI Delay register

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Timing details are shown in Section 26.7.3.1 “Pre_delay and Post_delay”, Section 26.7.3.2 “Frame_delay”, and Section 26.7.3.3 “Transfer_delay”.

Table 453. SPI Delay register (DLY, offset 0x404) bit description

Bit	Symbol	Description	Reset value
3:0	PRE_DELAY	Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0
7:4	POST_DELAY	Controls the amount of time between the end of a data transfer and SSEL deassertion. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0
11:8	FRAME_DELAY	If the EOF flag is set, controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion if EOT). 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0
15:12	TRANSFER_ DELAY	Controls the minimum amount of time that the SSEL is deasserted between transfers. 0x0 = The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.) 0x1 = The minimum time that SSEL is deasserted is 2 SPI clock times. 0x2 = The minimum time that SSEL is deasserted is 3 SPI clock times. ... 0xF = The minimum time that SSEL is deasserted is 16 SPI clock times.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.3 SPI Status register

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

In this register, the following notation is used: RO = read-only, W1C = write 1 to clear.

Table 454. SPI Status register (STAT, offset 0x408) bit description

Bit	Symbol	Description	Reset value	Access [1]
3:0	-	Reserved. Read value is undefined, only zero should be written.	-	-
4	SSA	Slave Select Assert. This flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. This flag is cleared by software.	0	W1C
5	SSD	Slave Select Deassert. This flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. This flag is cleared by software.	0	W1C
6	STALLED	Stalled status flag. This indicates whether the SPI is currently in a stall condition.	0	RO
7	ENDTRANSFER	End Transfer control bit. Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOT flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.	0	R/W1C
8	MSTIDLE	Master idle status flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.	1	RO
31:9	-	Reserved. Read value is undefined, only zero should be written.	-	-

[1] RO = read-only, W1C = write 1 to clear.

26.6.4 SPI Interrupt Enable read and Set register

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See Table 454 for details of the interrupts.

Table 455. SPI Interrupt Enable read and Set register (INTENSET, offset 0x40C) bit description

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSAEN		Slave select assert interrupt enable. Determines whether an interrupt occurs when the Slave Select is asserted.	0
		0	Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
		1	Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
5	SSDEN		Slave select deassert interrupt enable. Determines whether an interrupt occurs when the Slave Select is deasserted.	0
		0	Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted.	
		1	Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted.	

Table 455. SPI Interrupt Enable read and Set register (INTENSET, offset 0x40C) bit description

Bit	Symbol	Value	Description	Reset value
7:6	-	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLEEN		Master idle interrupt enable.	0
		0	No interrupt will be generated when the SPI master function is idle.	
		1	An interrupt will be generated when the SPI master function is fully idle.	
31:9	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.5 SPI Interrupt Enable Clear register

The INTENCLR register is used to clear interrupt enable bits in the INTENSET register.

Table 456. SPI Interrupt Enable clear register (INTENCLR, offset 0x410) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSAEN	Writing 1 clears the corresponding bit in the INTENSET register.	0
5	SSDEN	Writing 1 clears the corresponding bit in the INTENSET register.	0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLE	Writing 1 clears the corresponding bit in the INTENSET register.	0
31:9	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.6 SPI Divider register

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see Section 26.7.4 “Clocking and data rates”.

Table 457. SPI Divider register (DIV, offset 0x424) bit description

Bit	Symbol	Description	Reset Value
15:0	DIVVAL	Rate divider value. Specifies how the Flexcomm Interface clock (FCLK) is divided to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in FCLK/1, the value 1 results in FCLK/2, up to the maximum possible divide value of 0xFFFF, which results in FCLK/65536.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.7 SPI Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 454 for detailed descriptions of the interrupt flags.

Table 458. SPI Interrupt Status register (INTSTAT, offset 0x428) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSA	Slave Select Assert.	0
5	SSD	Slave Select Deassert.	0

Table 458. SPI Interrupt Status register (INTSTAT, offset 0x428) bit description

Bit	Symbol	Description	Reset value
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLE	Master Idle status flag.	0
31:9	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.8 FIFO Configuration register

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 459. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x1 = FIFO is configured as 8 entries of 16 bits. 0x0, 0x2, 0x3 = not applicable to SPI.	-	RO
11:6	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 7.5.96 “Hardware Wake-up control register”.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 7.5.96 “Hardware Wake-up control register”.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		

Table 459. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

26.6.9 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 460. FIFO status register (FIFOSTAT - offset 0xE04) bit description

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

26.6.10 FIFO trigger settings register

This register allows selecting when FIFO-level related interrupts occur.

Table 461. FIFO trigger settings register (FIFOTRIG - offset 0xE08) bit description

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The TX FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET,. This field is not used for DMA requests (see DMATX in Section 26.6.8 “FIFO Configuration register”).	0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An trigger will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. The RX FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET.This field is not used for DMA requests (see DMARX in Section 26.6.8 “FIFO Configuration register”).	0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An trigger will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
10:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 “Hardware Wake-up control register”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 “Hardware Wake-up control register”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO has 8 entries (has become full).	0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.11 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Table 462. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10) bit description

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.12 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 463. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14) bit description

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.13 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 26.6.9 and Section 26.6.10 for details.

Table 464. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18) bit description

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0
1	RXERR	RX FIFO error.	0
2	TXLVL	Transmit FIFO level interrupt.	0
3	RXLVL	Receive FIFO level interrupt.	0
4	PERINT	Peripheral interrupt.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.14 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO.

FIFOWR provides the possibility of altering some SPI controls at the same time as sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically.These added fields are described for bits 16 through 27 below.

Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified. Halfword writes to just the control bits (offset 0xE22) and does not push anything into the FIFO. A 0 written to the upper halfword will not modify the control settings. Non-zero writes to it will modify all the control bits. This is a write only register. Do not read-modify-write the register.

Byte, halfword or word writes to FIFOWR will push the data and control bits into the FIFO. Word writes with the upper halfword of 0, byte writes or halfword writes to FIFOWR will push the data and the current control bits, into the FIFO. Word writes with a non-zero upper halfword will modify the control bits before pushing them onto the stack.

To set-up a slave SPI for receive only, the control bit settings must be pushed into the write FIFO to become active. Therefore, at least one write to the FIFOWR data bits must be done to make the control bits active.

Table 465. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Value	Description	Reset value
15:0	TXDATA		Transmit data to the FIFO.	-
16	TXSSEL0_N		Transmit Slave Select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register.	-
		0	SSEL0 asserted.	
		1	SSEL0 not asserted.	

Table 465. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Value	Description	Reset value
17	TXSSEL1_N		Transmit Slave Select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register.	-
		0	SSEL1 asserted.	
		1	SSEL1 not asserted.	
18	TXSSEL2_N		Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register.	-
		0	SSEL2 asserted.	
		1	SSEL2 not asserted.	
19	TXSSEL3_N		Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register.	-
		0	SSEL3 asserted.	
		1	SSEL3 not asserted.	
20	EOT		End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the TRANSFER_DELAY value in the DLY register.	-
		0	SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data.	
		1	SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.	
21	EOF		End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits.	-
		0	Data not EOF. This piece of data transmitted is not treated as the end of a frame.	
		1	Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.	
22	RXIGNORE		Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA.	-
		0	Read received data. Received data must be read first and then the RxData should be written to allow transmission to progress for non-DMA cases. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received.	
		1	Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.	

Table 465. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Value	Description	Reset value
23	-	-	Reserved. Read value is undefined, only zero should be written.	-
27:24	LEN		Data Length. Specifies the data length from 4 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0-2 = Reserved 0x3 = Data transfer is 4 bits in length. 0x4 = Data transfer is 5 bits in length. ... 0xF = Data transfer is 16 bits in length.	-
31:28	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.15 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO.

Table 466. FIFO read data register (FIFORD - offset 0xE30) bit description

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave Select for receive. This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
17	RXSSEL1_N	Slave Select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
18	RXSSEL2_N	Slave Select for receive. This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
19	RXSSEL3_N	Slave Select for receive. This field allows the state of the SSEL3 pin to be saved along with received data. The value will reflect the SSEL3 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
20	SOT	Start of Transfer flag. This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bits.	-
31:21	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.16 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 467. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave Select for receive.	-
17	RXSSEL1_N	Slave Select for receive.	-
18	RXSSEL2_N	Slave Select for receive.	-
19	RXSSEL3_N	Slave Select for receive.	-
20	SOT	Start of Transfer flag.	-
31:21	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.17 Module identification register

The ID register identifies the type and revision of the SPI module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 468. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE020

26.7 Functional description

26.7.1 AHB bus access

With the exception of the FIFOWR register, the bus interface to the SPI registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the SPI function for those registers.

The FIFOWR register also supports byte and halfword (data only) writes in order to allow writing FIFO data without affecting the SPI control fields above bit 15 (see Section 26.6.14 "FIFO write data register").

26.7.2 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in Table 469 and shown in Figure 58. CPOL and CPHA are configured by bits in the CFG register (Section 26.6.1).

Table 469. SPI mode summary

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0 with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1 with SCK inverted.	high	falling	rising

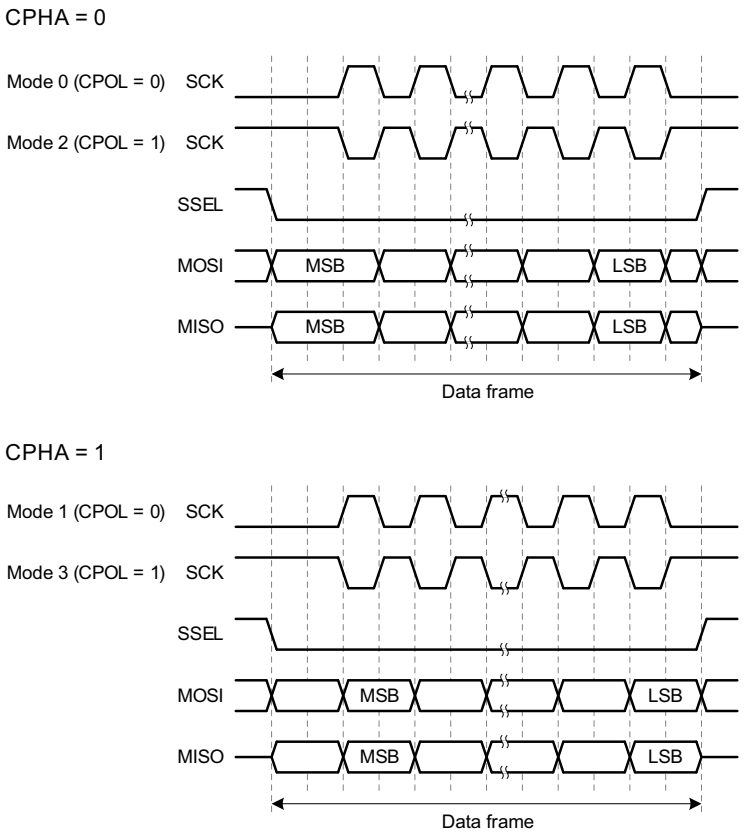


Fig 58. Basic SPI operating modes

26.7.3 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins
- Post_delay: delay at the end of a data frame before SSEL is deasserted
- Frame_delay: delay between data frames when SSEL is not deasserted
- Transfer_delay: minimum duration of SSEL in the deasserted state between transfers

26.7.3.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 59. The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the deassertion of SSEL.

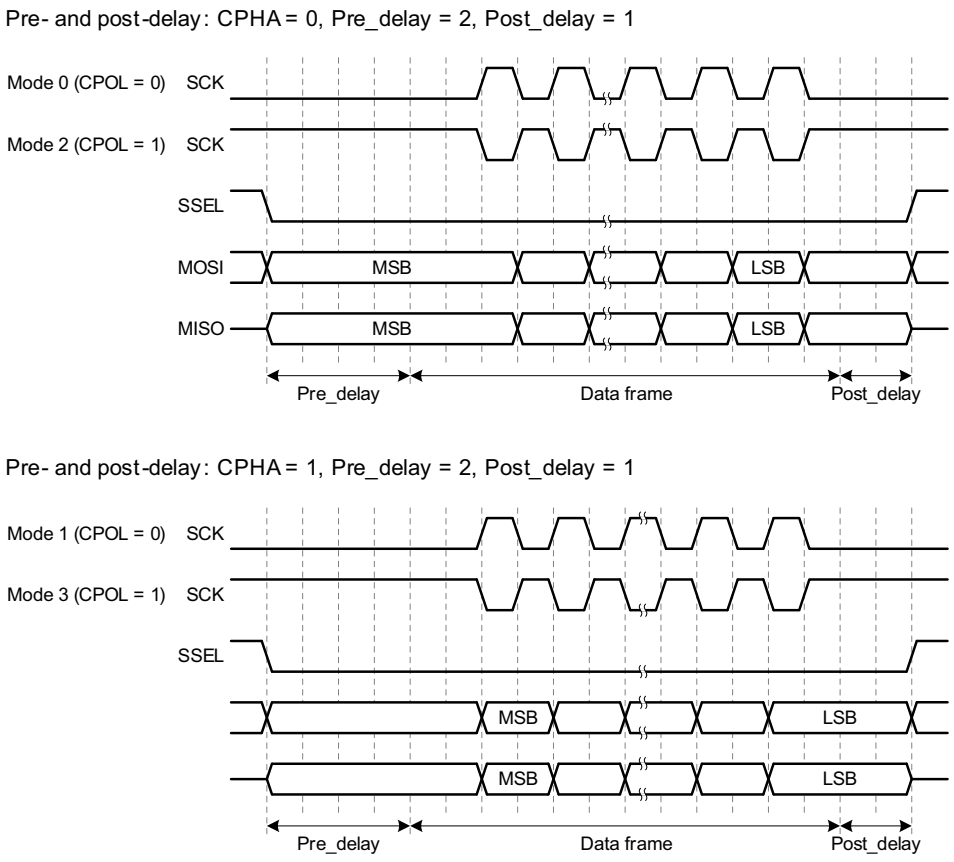


Fig 59. Pre_delay and Post_delay

26.7.3.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the EOF bit = 1. Frame_delay is illustrated by the examples in Figure 60. Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See Section 26.7.7 for more information.

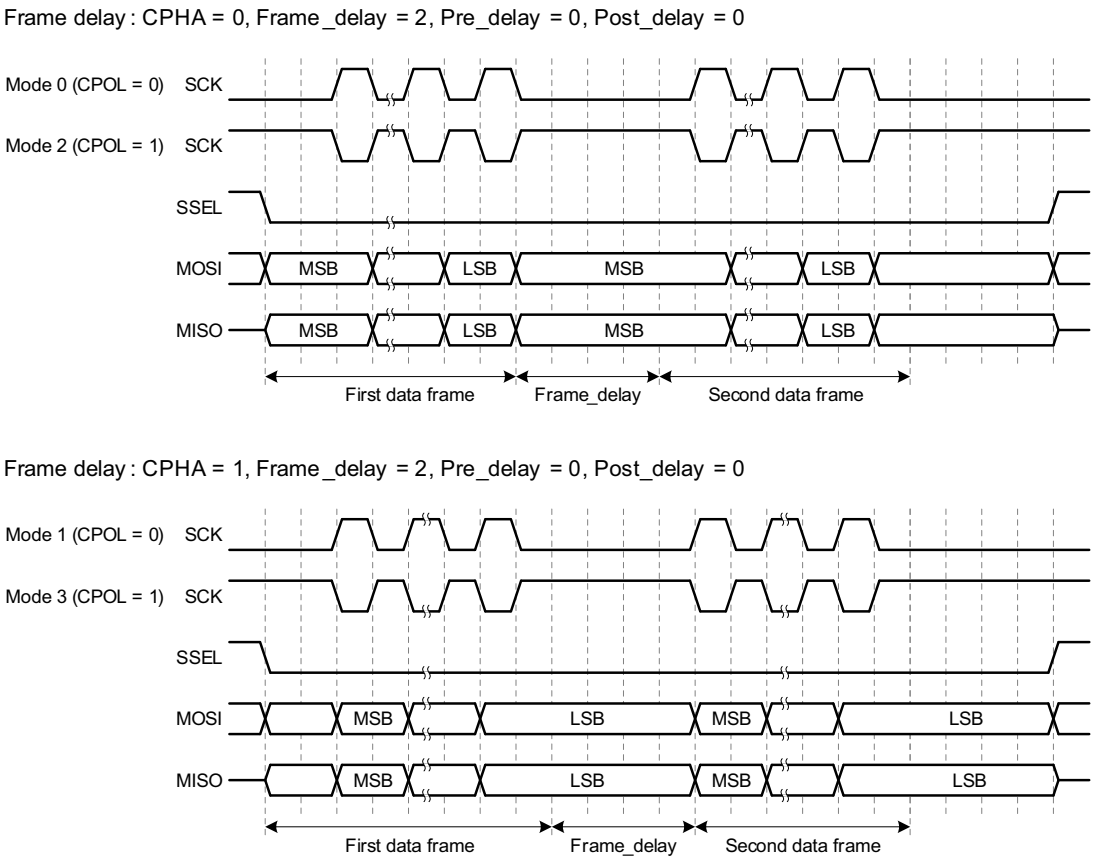


Fig 60. Frame_delay

26.7.3.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 61.

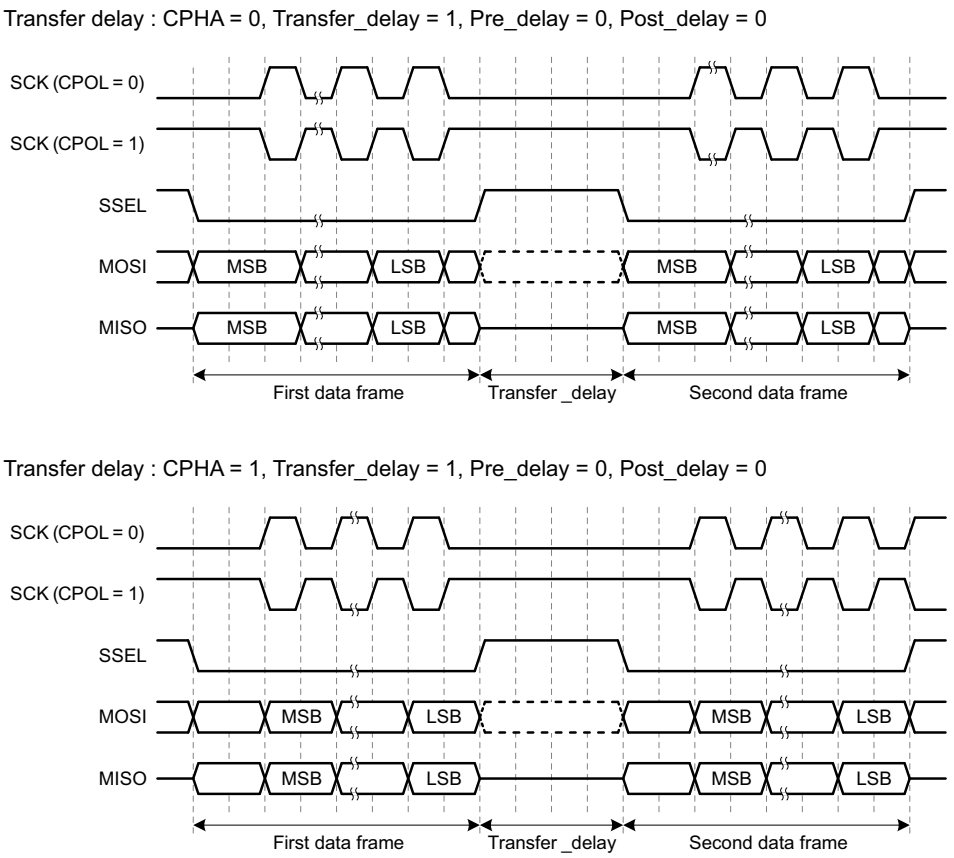


Fig 61. Transfer_delay

26.7.4 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock and selection of the clock divider value in DIV. See Figure 7 “Clock generation”.

26.7.4.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected FCLK, or at lower integer divide rates. The SPI rate will be = FCLK of Flexcomm n / DIVVAL.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

26.7.5 Slave select

The SPI block provides for four Slave Select inputs in slave mode or outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, **any** asserted SSEL that is connected to a pin will activate the SPI. In master mode, all SSELs that are connected to a pin will be output as defined in the SPI registers. In the latter case, the SSELs could potentially be decoded externally in order to address more than four slave devices. Note that at least one SSEL is asserted when data is transferred in master mode.

In master mode, Slave Selects come from the TXSSEL bits in the FIFOWR register. In slave mode, the state of all four SSELs is saved along with received data in the RXSSEL_N field of the FIFORD register.

26.7.6 DMA operation

A DMA request is provided for each SPI direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request.

The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.

The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

26.7.6.1 DMA master mode End-Of-Transfer

When using polled or interrupt mode to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is simple. The EOT bit of the FIFOWR control bits would be set just before or along with the writing of the last data to be sent.

When using the DMA in master mode, the end-of-transfer status (drive SSEL inactive) can be generated in the following ways:

1. Using DMA interrupt and a second DMA transfer:

   To use only 8 or 16 bit wide DMA transfers for all the data, a second DMA transfer can be used to terminate the transfer (drive SSEL inactive).

   The transfer would be started by setting the control bits and then initiating the DMA transfer of all but the last byte/halfword of data. The DMA completion interrupt function must modify the control bits to set EOT and then set-up DMA to send the last data.

2. Using DMA and SPI interrupts (or background SPI status polling):

   To use only one 8 or 16 bit wide DMA transfer for all the data, two interrupts would be required to properly terminate the transfer (drive SSEL inactive).

   The SPI Tx DMA completion interrupt function sets the TXLVL field in the SPI FIFOTRIG register to 0 and sets the TXLVL interrupt enable bit in the FIFOINTENSET register.

   The interrupt function handling the SPI TXLVL would set the SPI STAT register "END TRANSFER" bit, to force termination after all data output is complete.

3. Using DMA linked descriptor:

   The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/halfword. These data transfers can be done as 8 or 16 bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the EOT bit.

   Note: The DMA interrupt function cannot set the SPI Status register (STAT) END TRANSFER control bit. This may terminate the transfer while the FIFO still has data to send.

4. Using 32 bit wide DMA:

   Write both data and control bits to FIFOWR for all data. The control bits for the last entry would include the setting of the EOT bit. This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

26.7.7 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits or 2 groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be deasserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

26.7.8 Data stalls

A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

A stall for Master receive can happen when a FIFO overflow (see RXERR in the FIFOSTAT register) would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This stall happens one clock edge earlier than the transmitter stall.

In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected in the STAT register by the Stalled status flag, which indicates the current SPI status. The transmitter will be stalled until data is read from the receive FIFO. Use the RXIGNORE control bit setting to avoid the need to read the received data.

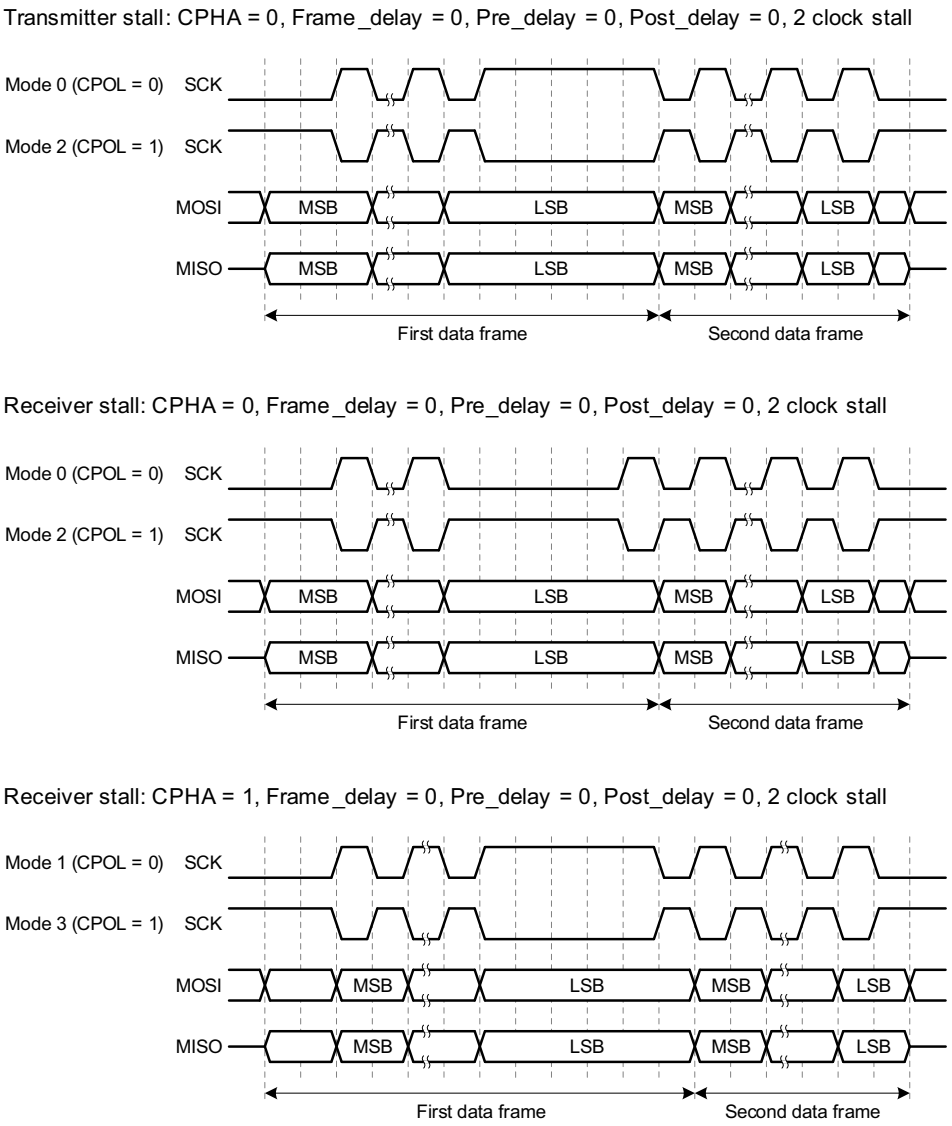


Fig 62. Examples of data stalls

27.1 How to read this chapter

I2C-bus functions are available on all LPC546xx devices as a selectable function in each Flexcomm Interface. Up to 10 Flexcomm Interfaces are available.

27.2 Features

- Independent Master, Slave, and Monitor functions.
- Bus speeds supported:
  - Standard mode, up to 100 kbits/s.
  - Fast-mode, up to 400 kbits/s.
  - Fast-mode Plus, up to 1 Mbits/s (on specific I2C pins).
  - High speed mode, 3.4 Mbits/s as a Slave only (on specific I2C pins).
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I2C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I2C bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management Bus (SMBus).
- Separate DMA requests for Master, Slave, and Monitor functions.
- No chip clocks are required in order to receive and compare an address as a Slave, so this event can wake up the device from deep-sleep mode.
- Automatic modes optionally allow less software overhead for some use cases.

27.3 Pin description

The I2C pins are fixed-pin functions and enabled through IOCON. See the IOCON settings in Table 471 and in Section 10.5.2.

Table 470. I2C-bus pin description

Function	Type	Pin name used in data sheet	Description
SCL	I/O	FCn_TXD_SCL_MISO, FCn_TXD_SCL_MISO_WS, or FCn_RTS_SCL_SSEL1	I2C serial clock.
SDA	I/O	FCn_RXD_SDA_MOSI, FCn_RXD_SDA_MOSI_DATA, or FCn_CTS_SDA_SSEL0	I2C serial data.

Table 471. Suggested I2C pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	OD: Set to 1 for simulated open-drain output.	Same as type D.	I2CFILTER: 0 for Fast / Standard mode I2C. 1 for Fast Mode Plus or High Speed slave.
10	SLEW: Set to 0.	Not used, set to 0.	I2CDRIVE: 0 for Fast / Standard mode I2C. 1 for Fast Mode Plus or High Speed slave.
9	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
7	INVERT: Set to 0.	Same as type D.	Same as type D.
6	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 0.
5:4	MODE: Set to 0 (pull-down/pull-up resistor disabled).	Same as type D.	Not used, set to 0.
3:0	FUNC: The function will be “SCL” or “SDA”.	Same as type D.	FUNC: The function will be “SCL” or “SDA”.
General comment	A reasonable choice for I2C at or below 400 kHz.	Same as type D.	Recommended for I2C operation above 400 kHz.

27.4 Basic configuration

Configure the I2C and related clocks as follows:

- If needed, use the PRESETCTRL1 or PRESETCTRL2 register (see Table 130) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 24.7.1). Note that any selection that has been made will be cleared if the Flexcomm Interface itself is reset via the PRESETCTRL1 or PRESETCTRL2 register.
- Configure the I2C for the desired functions:
  - In the AHBCLKCTRL1 or AHBCLKCTRL2 register (see Table 140), set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC (see Table 88).
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 10 “LPC546xx I/O pin configuration (IOCON)”.
  - Configure the I2C clock and data rate. This includes the CLKDIV register for both master and slave modes, and MSTTIME for master mode. Also see Section 27.6.6 and Section 27.7.2.

    Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.

Remark: While the I2C function is incorporated into the Flexcomm Interface, it does not make use of the Flexcomm Interface FIFO.

27.4.1 I2C transmit/receive in master mode

In this example, Flexcomm Interface 1 is configured as an I2C master. The master sends 8 bits to the slave and then receives 8 bits from the slave.

If specialized I2C pins are used (PIO0_23 through PIO0_26), the pins should be configured as required for the I2C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. If these or standard pins are used, they should be configured as described in Section 10.5.2.

The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTL register.

Configure the I2C bit rate:

- Select a source for the Flexcomm Interface 1 clock that will allow for the desired I2C-bus rate. Divide the clock as needed, see Table 484.
- Further divide the source clock if needed using the CLKDIV register (Section 27.6.6).
- Set the SCL high and low times to complete the bus rate setup. See Section 27.6.9.

27.4.1.1 Master write to slave

Configure Flexcomm Interface 1 as I2C interface, see Chapter 24 “LPC546xx Flexcomm Interface serial communication”.

Configure the I2C as a master: set the MSTEN bit to 1 in the CFG register. See Table 477.

Write data to the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 0 to the Master data register MSTDAT. See Table 488.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 486. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCONT bit to 1 in the Master control register. See Table 486. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 486.

Table 472. Code example

Master write to slave

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTDAT = 0xdd; // send data
I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

27.4.1.2 Master read from slave

Configure Flexcomm Interface 1 as I2C interface, see Chapter 24 “LPC546xx Flexcomm Interface serial communication”.

Configure the I2C as a master: set the MSTEN bit to 1 in the CFG register. See Table 477.

Read data from the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 1 to the Master data register MSTDAT. See Table 488.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 486. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
   - The slave sends 8 bit of data.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the MSTDAT register.
5. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 486.

Table 473. Code example

Master read from slave

```
// Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = I2C->MSTDAT; // read data
if(data != 0xdd) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

27.4.2 I2C receive/transmit in slave mode

In this example, Flexcomm Interface 1 is configured as an I2C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The SCL and SDA functions must be enabled on pins PIO0_22 and PIO0_23 through IOCON. See Section 10.5.2.

The pins should be configured as required for the I2C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. See Section 10.5.2.

The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data to be sent to the master are available in the SLVDAT register. After sending and receiving data, continue to the next step of the transmission protocol by writing to the SLVCTL register.

27.4.2.1 Slave read from master

Configure Flexcomm Interface 1 as I2C interface, see Chapter 24 “LPC546xx Flexcomm Interface serial communication”.

Configure the I2C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 477.
- Write the slave address x to the address 0 match register. See Table 491.

Read data from the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. Acknowledge (“ack”) the address by setting SLVCONTINUE = 1 in the slave control register. See Table 489.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the SLVDAT register. See Table 490.
5. Acknowledge (“ack”) the data by setting SLVCONTINUE = 1 in the slave control register. See Table 489.

Table 474. Code example

Slave read from master

```
//Slave read 1 byte from master. Address 0x23. Polling mode.
uint8_t data;
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
I2C->CFG;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = I2C->SLVDAT; // read data
if(data != 0xdd) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data
```

27.4.2.2 Slave write to master

Configure Flexcomm Interface 1 as I2C interface, see Chapter 24 “LPC546xx Flexcomm Interface serial communication”.

Configure the I2C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 477.
- Write the slave address x to the address 0 match register. See Table 491.

Write data to the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. ACK the address by setting SLVCONTINUE = 1 in the slave control register. See Table 489.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to SLVDAT register. See Table 490.
5. Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See Table 489.

Table 475. Code example

Slave write to master

```
//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
I2C->CFG;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();
I2C->SLVDAT = 0xdd; // write data
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction
```

27.4.3 Configure the I2C for wake-up

In sleep mode, any activity on the I2C-bus that triggers an I2C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the Flexcomm Interface clock remains active in sleep mode, the I2C can wake up the part independently of whether the I2C interface is configured in master or slave mode.

In deep-sleep mode, the I2C clock is turned off as are all peripheral clocks. However, if the I2C is configured in slave mode and an external master on the I2C-bus provides the clock signal, the I2C interface can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the I2C interface INTENCLR register, can then wake up the core.

27.4.3.1 Wake-up from sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
  - Master pending
  - Change to idle state
  - Start/stop error
  - Slave pending
  - Address match (in slave mode)
  - Data available/ready

27.4.3.2 Wake-up from deep-sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C interrupt in the STARTER1 register in the SYSCON block to create the interrupt signal asynchronously while the core and the peripheral are not clocked. See Table 223 “Start enable register 1 (STARTER1, main syscon: offset 0x684) bit description”.
- Configure the I2C in slave mode.
- Enable the I2C the interrupt in the INTENCLR register which configures the interrupt as wake-up event. Examples are the following events:
  - Slave deselect
  - Slave pending (wait for read, write, or ACK)
  - Address match
  - Data available/ready for the Monitor function

27.5 General description

The architecture of the I2C-bus interface is shown in Figure 63.

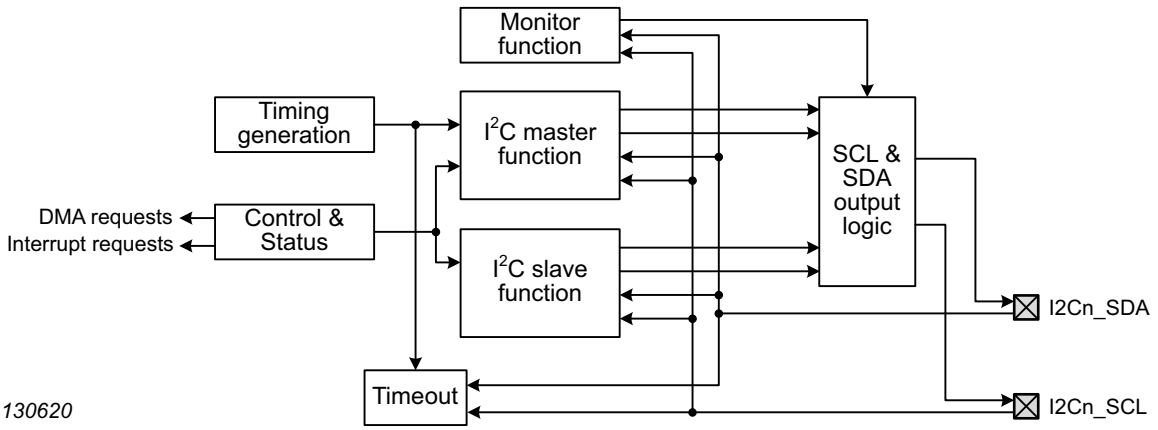


Fig 63. I2C block diagram

27.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 476. I2C register overview

Name	Access	Offset	Description	Reset value	Section
Shared I2C registers:					
CFG	R/W	0x800	Configuration for shared functions.	0	27.6.1
STAT	R/W	0x804	Status register for Master, Slave, and Monitor functions.	0x0801	27.6.2
INTENSET	R/W	0x808	Interrupt enable set and read.	0	27.6.3
INTENCLR	WO	0x80C	Interrupt enable clear.	NA	27.6.4
TIMEOUT	R/W	0x810	Time-out value.	0xFFFF	27.6.5
CLKDIV	R/W	0x814	Clock pre-divider for the entire I2C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function.	0	27.6.6
INTSTAT	RO	0x818	Interrupt status register for Master, Slave, and Monitor functions.	0	27.6.7
Master function registers:					
MSTCTL	R/W	0x820	Master control.	0	27.6.8
MSTTIME	R/W	0x824	Master timing configuration.	0x77	27.6.9
MSTDAT	R/W	0x828	Combined Master receiver and transmitter data.	NA	27.6.10
Slave function registers:					
SLVCTL	R/W	0x840	Slave control.	0	27.6.11
SLVDAT	R/W	0x844	Combined Slave receiver and transmitter data.	NA	27.6.12
SLVADR0	R/W	0x848	Slave address 0.	0x01	27.6.13
SLVADR1	R/W	0x84C	Slave address 1.	0x01	27.6.14
SLVADR2	R/W	0x850	Slave address 2.	0x01	27.6.14
SLVADR3	R/W	0x854	Slave address 3.	0x01	27.6.14
SLVQUAL0	R/W	0x858	Slave qualification for address 0.	0	27.6.15
Monitor function registers:					
MONRXDAT	RO	0x880	Monitor receiver data.	0	27.6.16
ID register:					
ID	RO	0xFFC	I2C module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2C is selected.	0xE030 0000	27.6.17

27.6.1 I2C Configuration register

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Table 477. I2C Configuration register (CFG, offset 0x800) bit description

Bit	Symbol	Value	Description	Reset Value
0	MSTEN		Master enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset.	0
		0	Disabled. The I2C Master function is disabled.	
		1	Enabled. The I2C Master function is enabled.	
1	SLVEN		Slave enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset.	0
		0	Disabled. The I2C slave function is disabled.	
		1	Enabled. The I2C slave function is enabled.	
2	MONEN		Monitor enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset.	0
		0	Disabled. The I2C Monitor function is disabled.	
		1	Enabled. The I2C Monitor function is enabled.	
3	TIMEOUTEN		I2C bus time-out enable. When disabled, the time-out function is internally reset.	0
		0	Disabled. Time-out function is disabled.	
		1	Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.	
4	MONCLKSTR		Monitor function clock stretching.	0
		0	Disabled. The monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical.	
		1	Enabled. The Monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the Monitor function.	
5	HSCAPABLE		High-speed mode capable enable. Since high speed mode alters the way I2C pins drive and filter, as well as the timing for certain I2C signalling, enabling high-speed mode applies to all functions: Master, Slave, and Monitor.	0
		0	Standard or Fast modes. The I2C interface will support Standard-mode, Fast-mode, and Fast-mode Plus, to the extent that the pin electronics support these modes. Any changes that need to be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I2C pin,	
		1	High-speed. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I2C interface will support High-speed mode to the extent that the pin electronics support these modes. See Section 27.7.2.2 for more information.	
31:6	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.2 I2C Status register

The STAT register provides status flags and state information about all of the functions of the I2C interface. Access to bits in this register varies. RO = read-only, W1C = write 1 to clear.

Details of the master and slave states described in the MSTSTATE and SLVSTATE bits in this register are listed in Table 479 and Table 480.

Table 478. I2C Status register (STAT, offset 0x804) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	MSTPENDING		Master Pending. Indicates that the Master is waiting to continue communication on the I2C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt.	1	RO
		0	In progress. Communication is in progress and the Master function is busy and cannot currently accept a command.		
		1	Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.		
3:1	MSTSTATE		Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved. See Table 479 for details of state values and appropriate responses.	0	RO
		0x0	Idle. The Master function is available to be used for a new transaction.		
		0x1	Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave.		
		0x2	Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave.		
		0x3	NACK Address. Slave NACKed address.		
		0x4	NACK Data. Slave NACKed transmitted data.		
4	MSTARBLOSS		Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE.	0	W1C
		0	No Arbitration Loss has occurred.		
		1	Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle.		
5	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

Table 478. I2C Status register (STAT, offset 0x804) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
6	MSTSTSTPERR		Master Start/Stop Error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE.	0	W1C
		0	No Start/Stop Error has occurred.		
		1	The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the I2C specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that the bus has not stalled.		
7	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
8	SLVPENDING		Slave Pending. Indicates that the Slave function is waiting to continue communication on the I2C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the I2C interface is in HSCAPABLE mode. See Section 27.7.2.2.2. When the I2C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I2C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched.	0	RO
		0	In progress. The Slave function does not currently need service.		
		1	Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.		
10:9	SLVSTATE		Slave State code. Each value of this field indicates a specific required service for the Slave function. All other values are reserved. See Table 480 for state values and actions. **Remark:** note that the occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes.	0	RO
		0x0	Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware.		
		0x1	Slave receive. Received data is available (Slave Receiver mode).		
		0x2	Slave transmit. Data can be transmitted (Slave Transmitter mode).		
11	SLVNOTSTR		Slave Not Stretching. Indicates when the slave function is stretching the I2C clock. This is needed in order to gracefully invoke deep-sleep mode during slave operation. This read-only flag reflects the slave function status in real time.	1	RO
		0	Stretching. The slave function is currently stretching the I2C bus clock. deep-sleep mode cannot be entered at this time.		
		1	Not stretching. The slave function is not currently stretching the I2C bus clock. deep-sleep mode could be entered at this time.		

Table 478. I2C Status register (STAT, offset 0x804) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
13:12	SLVIDX		Slave address match Index. This field is valid when the I2C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here.	0	RO
		0x0	Address 0. Slave address 0 was matched.		
		0x1	Address 1. Slave address 1 was matched.		
		0x2	Address 2. Slave address 2 was matched.		
		0x3	Address 3. Slave address 3 was matched.		
14	SLVSEL		Slave selected flag. SLVSEL is set after an address match when software tells the Slave function to acknowledge the address, or when the address has been automatically acknowledged. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, when there is a Stop detected on the bus, when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data.	0	RO
		0	Not selected. The Slave function is not currently selected.		
		1	Selected. The Slave function is currently selected.		
15	SLVDESEL		Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit.	0	W1C
		0	Not deselected. The Slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag.		
		1	Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.		
16	MONRDY		Monitor Ready. This flag is cleared when the MONRXDAT register is read.	0	RO
		0	No data. The Monitor function does not currently have data available.		
		1	Data waiting. The Monitor function has data waiting to be read.		
17	MONOV		Monitor Overflow flag.	0	W1C
		0	No overrun. Monitor data has not overrun.		
		1	Overrun. A Monitor data overrun has occurred. This can only happen when Monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.		
18	MONACTIVE		Monitor Active flag. Indicates when the Monitor function considers the I2C bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop.	0	RO
		0	Inactive. The Monitor function considers the I2C bus to be inactive.		
		1	Active. The Monitor function considers the I2C bus to be active.		

Table 478. I2C Status register (STAT, offset 0x804) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
19	MONIDLE		Monitor Idle flag. This flag is set when the Monitor function sees the I2C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit.	0	W1C
		0	Not idle. The I2C bus is not idle, or this flag has been cleared by software.		
		1	Idle. The I2C bus has gone idle at least once since the last time this flag was cleared by software.		
23:20	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
24	EVENTTIMEOUT		Event Time-out Interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I2C-bus is idle.	0	W1C
		0	No time-out. I2C bus events have not caused a time-out.		
		1	Event time-out. The time between I2C bus events has been longer than the time specified by the TIMEOUT register.		
25	SCLTIMEOUT		SCL Time-out Interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit.	0	W1C
		0	No time-out. SCL low time has not caused a time-out.		
		1	Time-out. SCL low time has caused a time-out.		
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

Table 479. Master function state codes (MSTSTATE)

MST STATE	Description	Actions	DMA allowed
0x0	**Idle.** The Master function is available to be used for a new transaction.	Send a Start or disable MSTPENDING interrupt if the Master function is not needed currently.	No
0x1	**Received data is available (Master Receiver mode).** Address plus Read was previously sent and Acknowledged by slave.	Read data and either continue, send a Stop, or send a Repeated Start.	Yes
0x2	**Data can be transmitted (Master Transmitter mode).** Address plus Write was previously sent and Acknowledged by slave.	Send data and continue, or send a Stop or Repeated Start.	Yes
0x3	**Slave NACKed address.**	Send a Stop or Repeated Start.	No
0x4	**Slave NACKed transmitted data.**	Send a Stop or Repeated Start.	No

Table 480. Slave function state codes (SLVSTATE)

SLVSTATE		Description	Actions	DMA allowed
0	SLVST_ADDR	**Address plus R/W received.** At least one of the 4 slave addresses has been matched by hardware.	Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 27.7.4 regarding 10-bit addressing.	No
1	SLVST_RX	**Received data is available (Slave Receiver mode).**	Read data, reply with an ACK or a NACK.	Yes
2	SLVST_TX	**Data can be transmitted (Slave Transmitter mode).**	Send data. Note that when the Master NACKs dat transmitted by the slave, the slave becomes de-selected.	Yes

27.6.3 Interrupt Enable Set and read register

The INTENSET register controls which I2C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register (Table 478), if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 481. Interrupt Enable Set and read register (INTENSET, offset 0x808) bit description

Bit	Symbol	Value	Description	Reset value
0	MSTPENDINGEN		Master Pending interrupt Enable.	0
		0	Disabled. The MstPending interrupt is disabled.	
		1	Enabled. The MstPending interrupt is enabled.	
3:1	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	MSTARBLOSSEN		Master Arbitration Loss interrupt Enable.	0
		0	Disabled. The MstArbLoss interrupt is disabled.	
		1	Enabled. The MstArbLoss interrupt is enabled.	
5	-	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERREN		Master Start/Stop Error interrupt Enable.	0
		0	Disabled. The MstStStpErr interrupt is disabled.	
		1	Enabled. The MstStStpErr interrupt is enabled.	
7	-		Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDINGEN		Slave Pending interrupt Enable.	0
		0	Disabled. The SlvPending interrupt is disabled.	
		1	Enabled. The SlvPending interrupt is enabled.	
10:9	-	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTREN		Slave Not Stretching interrupt Enable.	0
		0	Disabled. The SlvNotStr interrupt is disabled.	
		1	Enabled. The SlvNotStr interrupt is enabled.	
14:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESELEN		Slave Deselect interrupt Enable.	0
		0	Disabled. The SlvDeSel interrupt is disabled.	
		1	Enabled. The SlvDeSel interrupt is enabled.	
16	MONRDYEN		Monitor data Ready interrupt Enable.	0
		0	Disabled. The MonRdy interrupt is disabled.	
		1	Enabled. The MonRdy interrupt is enabled.	
17	MONOVEN		Monitor Overrun interrupt Enable.	0
		0	Disabled. The MonOv interrupt is disabled.	
		1	Enabled. The MonOv interrupt is enabled.	
18	-	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLEEN		Monitor Idle interrupt Enable.	0
		0	Disabled. The MonIdle interrupt is disabled.	
		1	Enabled. The MonIdle interrupt is enabled.	
23:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 481. Interrupt Enable Set and read register (INTENSET, offset 0x808) bit description

Bit	Symbol	Value	Description	Reset value
24	EVENTTIMEOUTEN		Event time-out interrupt Enable.	0
		0	Disabled. The Event time-out interrupt is disabled.	
		1	Enabled. The Event time-out interrupt is enabled.	
25	SCLTIMEOUTEN		SCL time-out interrupt Enable.	0
		0	Disabled. The SCL time-out interrupt is disabled.	
		1	Enabled. The SCL time-out interrupt is enabled.	
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.4 Interrupt Enable Clear register

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register.

Bits that do not correspond to defined bits in INTENSET are reserved and only zeroes should be written to them.

Table 482. Interrupt Enable Clear register (INTENCLR, offset 0x80C) bit description

Bit	Symbol	Description	Reset value
0	MSTPENDINGCLR	Master Pending interrupt clear. Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented.	0
3:1	-	Reserved. Read value is undefined, only zero should be written.	-
4	MSTARBLOSSCLR	Master Arbitration Loss interrupt clear.	0
5	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERRCLR	Master Start/Stop Error interrupt clear.	0
7	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDINGCLR	Slave Pending interrupt clear.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTRCLR	Slave Not Stretching interrupt clear.	0
14:12	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESELCLR	Slave Deselect interrupt clear.	0
16	MONRDYCLR	Monitor data Ready interrupt clear.	0
17	MONOVCLR	Monitor Overrun interrupt clear.	0
18	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLECLR	Monitor Idle interrupt clear.	0
23:20	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUTCLR	Event time-out interrupt clear.	0
25	SCLTIMEOUTCLR	SCL time-out interrupt clear.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.5 Time-out value register

The TIMEOUT register allows setting an upper limit to certain I2C bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can elect to use either of them.

1. EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the EVENTTIMEOUTEN bit in the INTENSET register.
2. SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the SCLTIMEOUTEN bit in the INTENSET register. The SCLTIMEOUT can be used with the SMBus.

Also see Section 27.7.3 “Time-out”.

Table 483. Time-out value register (TIMEOUT, offset 0x810) bit description

Bit	Symbol	Description	Reset value
3:0	TOMIN	Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I2C function clocks and also a time-out resolution of 16 I2C function clocks.	0xF
15:4	TO	Time-out time value. Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I2C function clock. 0x001 = A time-out will occur after 32 counts of the I2C function clock. ... 0xFFF = A time-out will occur after 65,536 counts of the I2C function clock.	0xFFF
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.6 Clock Divider register

The CLKDIV register divides down the Flexcomm Interface clock (FCLK) to produce the I2C function clock that is used to time various aspects of the I2C interface. The I2C function clock is used for some internal operations in the I2C interface and to generate the timing required by the I2C bus specification, some of which are user configured in the MSTTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

See Section 27.7.2.1 “Rate calculations” for details on bus rate setup.

Table 484. I2C Clock Divider register (CLKDIV, offset 0x814) bit description

Bit	Symbol	Description	Reset value
15:0	DIVVAL	This field controls how the Flexcomm Interface clock (FCLK) is used by the I2C functions that need an internal clock in order to operate. See Section 27.7.2.1 “Rate calculations”. 0x0000 = I2C clock divider provides FCLK divided by 1. 0x0001 = I2C clock divider provides FCLK divided by 2. 0x0002 = I2C clock divider provides FCLK divided by 3. ... 0xFFFF = I2C clock divider provides FCLK divided by 65,536.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.7 Interrupt Status register

The INTSTAT register provides register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 478 for detailed descriptions of the interrupt flags.

Table 485. I2C Interrupt Status register (INTSTAT, offset 0x818) bit description

Bit	Symbol	Description	Reset value
0	MSTPENDING	Master Pending.	1
3:1	-	Reserved.	-
4	MSTARBLOSS	Master Arbitration Loss flag.	0
5	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERR	Master Start/Stop Error flag.	0
7	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDING	Slave Pending.	0
10:9	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTR	Slave Not Stretching status.	1
14:12	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESEL	Slave Deselected flag.	0
16	MONRDY	Monitor Ready.	0
17	MONOV	Monitor Overflow flag.	0
18	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLE	Monitor Idle flag.	0
23:20	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUT	Event time-out Interrupt flag.	0
25	SCLTIMEOUT	SCL time-out Interrupt flag.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.8 Master Control register

The MSTCTL register contains bits that control various functions of the I2C Master interface. Only write to this register when the master is pending (MSTPENDING = 1 in the STAT register, Table 478).

Software should always write a complete value to MSTCTL, and not OR new control bits into the register as is possible in other registers such as CFG. This is due to the fact that MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I2C Start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes. In this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

Remark: When in the idle or slave NACKed states (see Table 479), set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 486. Master Control register (MSTCTL, offset 0x820) bit description

Bit	Symbol	Value	Description	Reset value
0	MSTCONTINUE		Master Continue. This bit is write-only.	0
		0	No effect.	
		1	Continue. Informs the Master function to continue to the next operation. This must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	
1	MSTSTART		Master Start control. This bit is write-only.	0
		0	No effect.	
		1	Start. A Start will be generated on the I2C bus at the next allowed time.	
2	MSTSTOP		Master Stop control. This bit is write-only.	0
		0	No effect.	
		1	Stop. A Stop will be generated on the I2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (Master Receiver mode).	
3	MSTDMA		Master DMA enable. Data operations of the I2C can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the I2C is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when Automatic Operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop.This bit is read/write.	0
		0	Disable. No DMA requests are generated for master operation.	
		1	Enable. A DMA request is generated for I2C master data operations. When this I2C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.9 Master Time register

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

The I2C clock pre-divider is described in Table 484.

Table 487. Master Time register (MSTTIME, offset 0x824) bit description

Bit	Symbol	Value	Description	Reset value
2:0	MSTSCLLOW		Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I2C bus specification. I2C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW.	7
		0x0	SCL low multiplier = 2. See Section 27.7.2.1 “Rate calculations”	
		0x1	SCL low multiplier = 3. See Section 27.7.2.1 “Rate calculations”	
		0x2	SCL low multiplier = 4. See Section 27.7.2.1 “Rate calculations”	
		0x3	SCL low multiplier = 5. See Section 27.7.2.1 “Rate calculations”	
		0x4	SCL low multiplier = 6. See Section 27.7.2.1 “Rate calculations”	
		0x5	SCL low multiplier = 7. See Section 27.7.2.1 “Rate calculations”	
		0x6	SCL low multiplier = 8. See Section 27.7.2.1 “Rate calculations”	
		0x7	SCL low multiplier = 9. See Section 27.7.2.1 “Rate calculations”	
3	-	-	Reserved.	-
6:4	MSTSCLHIGH		Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I2C bus specification. I2C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH.	7
		0x0	SCL high multiplier = 2. See Section 27.7.2.1 “Rate calculations”.	
		0x1	SCL high multiplier = 3. See Section 27.7.2.1 “Rate calculations”.	
		0x2	SCL high multiplier = 4. See Section 27.7.2.1 “Rate calculations”	
		0x3	SCL high multiplier = 5. See Section 27.7.2.1 “Rate calculations”	
		0x4	SCL high multiplier = 6. See Section 27.7.2.1 “Rate calculations”	
		0x5	SCL high multiplier = 7. See Section 27.7.2.1 “Rate calculations”	
		0x6	SCL high multiplier = 8. See Section 27.7.2.1 “Rate calculations”	
		0x7	SCL high multiplier = 9. See Section 27.7.2.1 “Rate calculations”	
31:7	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.10 Master Data register

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Table 488. Master Data register (MSTDAT, offset 0x828) bit description

Bit	Symbol	Description	Reset value
7:0	DATA	Master function data register. Read: read the most recently received data for the Master function. Write: transmit data using the Master function.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.11 Slave Control register

The SLVCTL register contains bits that control various functions of the I2C Slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register, Table 478).

Refer to Section 27.7.8 “Automatic operation” for details of the AUTOACK, AUTOMATCHREAD, and related settings.

Remark: When in the slave address state (slave state 0, see Table 480), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 489. Slave Control register (SLVCTL, offset 0x840) bit description

Bit	Symbol	Value	Description	Reset Value
0	SLVCONTINUE		Slave Continue.	0
		0	No effect.	
		1	Continue. Informs the Slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING = 1.	
1	SLVNACK		Slave NACK.	0
		0	No effect.	
		1	NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode).	
2	-	-	Reserved. Read value is undefined, only zero should be written.	-
3	SLVDMA		Slave DMA enable.	0
		0	Disabled. No DMA requests are issued for Slave mode operation.	
		1	Enabled. DMA requests are issued for I2C slave data transmission and reception.	
7:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 489. Slave Control register (SLVCTL, offset 0x840) bit description

Bit	Symbol	Value	Description	Reset Value
8	AUTOACK		Automatic Acknowledge.When this bit is set, it will cause an I2C header which matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. If this bit is clear and a header matches SLVADR0, the behavior is controlled by AUTONACK in the SLVADR0 register: allowing NACK or interrupt.	0
		0	Normal, non-automatic operation. If AUTONACK = 0, an SlvPending interrupt is generated when a matching address is received. If AUTONACK = 1, received addresses are NACKed (ignored).	
		1	A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. The ACK will clear this bit. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, the behavior will depend on the AUTONACK bit in the SLVADR0 register: if AUTONACK is set, then it will be Nacked; else if AUTONACK is clear, then a SlvPending interrupt is generated.	
9	AUTOMATCHREAD		When AUTOACK is set, this bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Since DMA needs to be configured to match the transfer direction, the direction needs to be specified. This bit allows a direction to be chosen for the next operation.	0
		0	The expected next operation in Automatic Mode is an I2C write.	
		1	The expected next operation in Automatic Mode is an I2C read.	
31:10	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.12 Slave Data register

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Table 490. Slave Data register (SLVDAT, offset 0x844) bit description

Bit	Symbol	Description	Reset value
7:0	DATA	Slave function data register. Read: read the most recently received data for the Slave function. Write: transmit data using the Slave function.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.13 Slave Address 0 register

The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I2C slave hardware.

The I2C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Refer to Section 27.7.8 “Automatic operation” for details of AUTONACK and related settings.

Table 491. Slave Address 0 register (SLVADR[0], offset 0x848) bit description

Bit	Symbol	Value	Description	Reset value
0	SADISABLE0		Slave Address 0 Disable.	1
		0	Enabled. Slave Address 0 is enabled.	
		1	Ignored Slave Address 0 is ignored.	
7:1	SLVADR0		Slave Address. Seven bit slave address that is compared to received addresses if enabled. The compare can be affected by the setting of the SLVQUAL0 register.	0
14:8	-	-	Reserved. Read value is undefined, only zero should be written.	-
15	AUTONACK		Automatic NACK operation. Used in conjunction with AUTOACK and AUTOMATCHREAD, allows software to ignore I2C traffic while handling previous I2C data or other operations.	0
		0	Normal operation, matching I2C addresses are not ignored.	
		1	Automatic-only mode. All incoming addresses are ignored (NACKed), unless AUTOACK is set, it matches SLVADR0, and AUTOMATCHREAD matches the direction.	
31:16	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.14 Slave Address 1, 2, and 3 registers

These slave address registers provide for three additional addresses that can be automatically recognized by the I2C slave hardware.

Table 492. Slave Address registers (SLVADR[1:3], offset [0x84C:0x854]) bit description

Bit	Symbol	Value	Description	Reset value
0	SADISABLE		Slave Address n Disable.	1
		0	Enabled. Slave Address n is enabled.	
		1	Ignored Slave Address n is ignored.	
7:1	SLVADR		Slave Address. Seven bit slave address that is compared to received addresses if enabled.	0
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.15 Slave address Qualifier 0 register

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

Table 493. Slave address Qualifier 0 register (SLVQUAL0, offset 0x858) bit description

Bit	Symbol	Value	Description	Reset Value
0	QUALMODE0		Qualify mode for slave address 0.	0
		0	Mask. The SLVQUAL0 field is used as a logical mask for matching address 0.	
		1	Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.	
7:1	SLVQUAL0		Slave address Qualifier for address 0. A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] ≤ received address ≤ SLVQUAL0[7:1]).	0
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.16 Monitor data register

The read-only MONRXDAT register provides information about events on the I2C bus, primarily to facilitate debugging of the I2C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the MONEN bit in the CFG register. Monitor mode can be configured to stretch the I2C clock if data is not read from the MONRXDAT register in time to prevent it, via the MONCLKSTR bit in the CFG register. This can help ensure that nothing is missed but can cause the Monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I2C bus.

Details of clock stretching are different in HS mode, see Section 27.7.2.2.2.

Table 494. Monitor data register (MONRXDAT, offset 0x880) bit description

Bit	Symbol	Value	Description	Reset value
7:0	MONRXDAT		Monitor function Receiver Data. This reflects every data byte that passes on the I2C pins.	0
8	MONSTART		Monitor Received Start.	0
		0	No start detected. The Monitor function has not detected a Start event on the I2C bus.	
		1	Start detected. The Monitor function has detected a Start event on the I2C bus.	
9	MONRESTART		Monitor Received Repeated Start.	0
		0	No repeated start detected. The Monitor function has not detected a Repeated Start event on the I2C bus.	
		1	Repeated start detected. The Monitor function has detected a Repeated Start event on the I2C bus.	
10	MONNACK		Monitor Received NACK.	0
		0	Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave receiver.	
		1	Not acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

27.6.17 Module identification register

The ID register identifies the type and revision of the I2C module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 495. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x00
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE030

27.7 Functional description

27.7.1 AHB bus access

The bus interface to the I2C registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I2C function.

27.7.2 Bus rates and timing considerations

Due to the nature of the I2C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc.

In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I2C traffic (i.e. when it is the only master on the bus, or during arbitration between masters).

In addition, I2C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure it actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line.

Rate calculations give a base frequency that represents the fastest that the I2C bus could operate if nothing slows it down.

27.7.2.1 Rate calculations

Master timing

SCL high time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL high multiplier (see Table 484 and Table 487).

SCL low time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL low multiplier (see Table 484 and Table 487).

Nominal SCL rate =
Flexcomm Interface function clock rate / (SCL high time + SCL low time)

Remark: DIVVAL must be ≥ 1.

Remark: For 400 KHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 2 MHz. Table 496 shows the recommended settings for 400 KHz clock rate.

Table 496. Settings for 400 KHz clock rate

Input clock to I2C	DIVVAL for CLKDIV register	MSTSCLHIGH for MSTTIME register	MSTSCLLOW for MSTTIME register
30 MHz	14	0	1
24 MHz	14	0	0
24 MHz	11	0	1
12 MHz	5	0	1

Slave timing

Most aspects of slave operation are controlled by SCL received from the I2C bus master. However, if the slave function stretches SCL to allow for software response, it must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting one clock time of CLKDIV at that point.

If CLKDIV is already configured for master operation, that is sufficient. If only the slave function is used, CLKDIV should be configured such that one clock time is greater than the tSU;DAT value noted in the I2C bus specification for the I2C mode that is being used.

27.7.2.2 Bus rate support

The I2C interface can support 4 modes from the I2C bus specification:

- Standard-mode (SM, rate up to 100 kbits/s)
- Fast-mode (FM, rate up to 400 kbits/s)
- Fast-mode Plus (FM+, rate up to 1 Mbits/s)
- High-speed mode (HS, rate up to 3.4 Mbits/s)

See Ref. 3 “UM10204” for details of I2C modes and other details.

The I2C interface supports Standard-mode, Fast-mode, and Fast-mode Plus with the same software sequence, which also supports SMBus. High-speed mode is intrinsically incompatible with SMBus due to conflicting requirements and limitations for clock stretching, and therefore requires a slightly different software sequence.

27.7.2.2.1 High-speed mode support

High-speed mode requires different pin filtering, somewhat different timing, and a different drive strength on SCL for the master function. The changes needed for the handling of the acknowledge bit mean that SMBus cannot be supported when the I2C is configured to be HS capable. This limitation is intrinsic to the SMBus and High-speed I2C specifications.

Because of the timing of changes to pin drive strength and filtering, the I2C interface is designed to directly control those pad characteristics when configured to be HS capable. The I2C also recognizes HS master codes and responds to programmed addresses when HS capable.

For software consistency, the changes required for handling of acknowledge and address recognition, and which affect when interrupts occur, are always in effect when the I2C is configured to be HS capable. This means that software does not need to know if a particular transfer is actually in HS mode or not.

27.7.2.2.2 Clock stretching

The I2C interface automatically stretches the clock when it does not have sufficient information on how to proceed, i.e. software has not supplied data and/or instructions to generate a start or stop. In principle, at least, I2C can allow the clock to be stretched by any bus participant at any time that SCL is low, in SM, FM, and MF+ modes.

In practice, the I2C interface described here may stretch SCL at the following times, in SM, FM, and MF+ modes:

- As a Slave:

- after an address is received that complies with at least one slave address (before the address is acknowledged)
- as a slave receiver, after each data byte received (software then acknowledges the data)
- as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master

- As a master:

  after each

  - address is sent and the acknowledge bit has been received
  - as a master receiver, after each after each data byte is received (software then acknowledges the data)
  - as a master transmitter, after each data byte is sent and the matching acknowledge bit has been received from the slave

In HS mode:

- As a Slave (only slave functions in HS mode are supported on this device)
  - as a slave receiver, after each data byte is received and automatically acknowledged
  - as a slave transmitter, after each after each data byte is sent and the matching acknowledge is received from the master

In each case, the relevant pending flag (MSTPENDING or SLVPENDING) is set at the point where clock stretching occurs.

27.7.3 Time-out

A time-out feature on an I2C interface can be used to detect a “stuck” bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I2C interface and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

In the first type of time-out, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the time-out check. These events include Start, Stop, and all changes on the I2C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I2C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I2C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I2C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated only when the I2C bus is considered busy, i.e. when there has been a Start condition more recently than a Stop condition.

27.7.4 Ten-bit addressing

Ten-bit addressing is accomplished by the I2C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I2C frame simply continues with data after the 2 address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I2C address.

For the Master function, the I2C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The slave address qualifier feature (see Section 27.6.15) can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one. In the case of Slave Receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The Slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode, the slave function responds to the initial address in the same fashion as for Slave Receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

27.7.5 Clocking and power considerations

The Master function of the I2C always requires a peripheral clock to be running in order to operate. The Slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to deep-sleep mode can be entered, and the device will wake up when the I2C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

27.7.6 Interrupt handling

The I2C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

27.7.7 DMA

DMA with the I2C is done only for data transfer, DMA cannot handle control of the I2C. Once DMA is transferring data, I2C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

Generally, data transfers can be handled by DMA for Master mode after an address is sent and acknowledged by a slave, and for Slave mode after software has acknowledged an address. In either mode, software is always involved in the address portion of a

message. In master and slave modes, data receive and transmit data can be transferred by the DMA. The DMA supports three DMA requests: data transfer in master mode, slave mode, and Monitor mode.

DMA may be used in connection with Automatic Operation in order to minimize software overhead time for I2C handling.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop and an interrupt to be generated. A Repeated Start sensed on the bus will similarly cause DMA to stop and an interrupt to be generated.

The Monitor function may be used with DMA if a channel is available See Section 15.5.1.1.1 for how DMA channels are used with the Monitor function.

27.7.7.1 DMA as a Master transmitter

A basic sequence for a Master transmitter:

- Software sets up DMA to transmit a message.
- Software causes a slave address with write command to be sent and checks that the address was acknowledged.
- Software turns on DMA mode in the I2C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

27.7.7.2 DMA as a Master receiver

A basic sequence for a Master receiver:

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and checks that the address was acknowledged.
- Software starts DMA.
- DMA completes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

27.7.7.3 DMA as a Slave transmitter

A basic sequence for a Slave transmitter:

- Software acknowledges an I2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA completes.

27.7.7.4 DMA as a Slave receiver

A basic sequence for a Slave receiver:

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA completes.
- Software sets SLVNACK prior to receiving the final data byte.
- Software receives the final data byte.

27.7.8 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I2C slave functions with some limitations. They are intended to be used primarily in conjunction with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register, and the AUTONACK bit in the SLVADR0 register. Table 27 shows how these controls may be used. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Automatic operation cases

Conditions:			Response:		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I2C bus**	**Description**
0	0	x	Yes	None	Normal, non-automatic operation.
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.
x	1	Yes	No	ACK	Automatic slave DMA: expected read/write case. When the automatic Ack is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	x	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

28.1 How to read this chapter

I2S functionality is available on all LPC546xx devices. I2S is a function that is implemented in selected Flexcomm Interfaces. In the LPC546xx, the I2S function is included in Flexcomm Interface 6 and Flexcomm Interface 7. Each of these Flexcomm Interfaces implements four I2S channel pairs.

28.2 Features

The I2S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select/frame trigger signal, providing single or dual (mono or stereo) audio data transfer as well as other configurations.

The I2S interface within one Flexcomm Interface provides at least one channel pair that can be configured as a master or a slave. Other channel pairs, if present, always operate as slaves. All of the channel pairs within one Flexcomm Interface share one set of I2S signals, and are configured together for either transmit or receive operation, using the same mode, same data configuration and frame configuration. All such channel pairs can participate in a time division multiplexing (TDM) arrangement. For cases requiring an MCLK input and/or output, this is handled outside of the I2S block in the system level clocking scheme.

- A Flexcomm Interface may implement one or more I2S channel pairs. The first channel pair can be either a master or a slave, and the rest of the channel pairs are always slaves. All channel pairs are configured together for either transmit or receive and other shared attributes. The number of channel pairs is defined for each Flexcomm Interface and may be from 0 to 4.
- Configurable data size for all channels within one Flexcomm Interface, from 4 bits to 32 bits. Each channel pair can also be configured independently to act as a single channel (mono as opposed to stereo operation).
- All channel pairs within one Flexcomm Interface share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
- Data for all I2S traffic within one Flexcomm Interface uses the Flexcomm Interface FIFO. The FIFO depth is 8 entries.
- Left justified and right justified data modes.
- DMA support using FIFO level triggering.
- TDM (Time Division Multiplexing) with a several stereo slots and/or mono slots is supported. Each channel pair can act as any data slot. Multiple channel pairs can participate as different slots on one TDM data line.
- The bit clock and WS can be selectively inverted.
- Sampling frequencies supported depends on the specific device configuration and applications constraints (e.g. system clock frequency, PLL availability, etc.) but generally supports standard audio data rates.

28.3 Basic configuration

Initial configuration of the I2S peripheral is accomplished as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 register (Section 7.5.20).
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 8. Also see Section 7.5.37.

   Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.
3. If needed, use the PRESETCTRL1 register (Table 130) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
4. Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 24.7.1).
5. Pins: Make sure that the IOCON block is enabled in the AHBCLKCTRL0 register (Section 7.5.19). Select I2S pins and pin modes through the relevant IOCON registers (Chapter 10 “LPC546xx I/O pin configuration (IOCON)”).
6. I2S rate: For master operation, the I2S rate is determined by the clock selected in step 2 above, optionally modified using the DIV register (Table 503). Slave functions typically use the incoming I2S clock directly.
7. Interrupts: To enable I2S channel pair interrupts, see (FIFOINENSET in Section 28.7.8, FIFOINTENCLR in Section 28.7.9, and FIFOINTSTAT in Section 28.7.10). The related Flexcomm Interface interrupt must be enabled in the NVIC using the appropriate Interrupt Set Enable register (see Chapter 6 “LPC546xx Nested Vectored Interrupt Controller (NVIC)”).
8. DMA: I2S channel pair master and slave functions can operated with the system DMA controller (see Chapter 15 “LPC546xx DMA controller”), and must be enabled in the FIFOCFG register (Section 28.7.5).

28.4 Architecture

The overall architecture of an example I2S subsystem is shown in Figure 64.

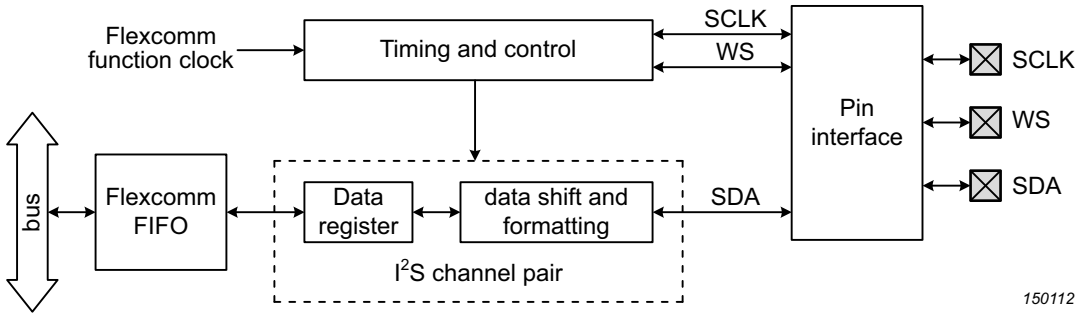


Fig 64. I2S block diagram

28.5 Terminology

Table 497. List of some terminology used in this document

Term	Description
Channel	Essentially, one piece of information on a single SDA line. In classic I2S, there is a single set of stereo data, which is 2 channels (left and right). In TDM modes, there may be many channels on a single SDA line.
Channel Pair	Two channels of data can be carried on one wire in classic I2S: left and right. On a microcontroller, this is typically what is implemented in a single instance of an I2S interface.
Classic I2S	This term is used in this document in reference to the original I2S bus specification from Philips Semiconductors. That specification defines 2 channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by 1 clock after WS transitions. The many variations of I2S that may be found have descended from this original specification.
DSP mode	DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. WS may be a single SCK pulse, or a single data slot long pulse, in addition to a 50% duty cycle pulse. May be used in conjunction with TDM mode.
MCLK	Master Clock. In some I2S systems, this is provided as a multiple of the sample rate (fs), higher than the bit rate, such as 256 fs. Devices could potentially use this clock to construct a bit clock, or for internal operations such as data filtering.
SCK	Serial Clock. Sometimes referred to as BCK. This is a bit clock for data on the SDA line.
SDA	Serial Data. A single SDA provides one data stream, which may have many formats.
Slot	One data position in an I2S stream, typically each with the same slot length. For classic I2S, there is only one slot for stereo data. In a TDM mode, there can be several slots. In MONO mode, each slot is defined as one piece of data, rather than both left and right data.
TDM mode	TDM mode uses multiple data slots in order to put more channels of data into a single stream. May be used in conjunction with DSP mode or I2S mode.
WS	Word Select. Sometimes called LRCLK. Distinguishes left versus right data in most single stereo formats. Used as a frame delimiter in DSP and TDM modes.

28.6 Pin description

Remark: When the I2S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. For that reason, those signals must in fact be connected to a device pin, via IOCON selection, in order for the I2S to operate.

Table 498. I2S Pin Description

Pin	Type	Name used in Pin Configuration chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for I2Sn. Clock signal used to synchronize the transfer of data on the SDA pin. It is normally driven by the master and received by one or more slaves. **Remark:** When the primary I2S channel pair of a Flexcomm Interface is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the I2S to work properly.
WS	I/O	FCn_TXD_SCL_MISO_WS	Word Select for I2Sn. Synchronizing signal for the beginning of each data frame and, in some modes, left vs. right channel data. It is normally driven by the master and received by one or more slaves. **Remark:** When the primary I2S channel pair of a Flexcomm Interface is configured as a master, such that WS is an output, it must actually be connected to a pin in order for the I2S to work properly.
SDA	I/O	FCn_RXD_SDA_MOSI_DATA	Serial Data for a single data stream used by one or more I2S channel pairs of I2Sn. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	MCLK	Master Clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm Interface in order to locally generate SCK and/or WS. This clock is not created inside the I2S block. If MCLK is supported as an input to the device, it can be routed to the I2S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I2S block and used to operate its functions.

28.7 Register description

The registers shown in Table 499 apply if the I2S function is selected in a Flexcomm Interface that supports I2S. The primary channel pair uses registers as shown under the row heading “Registers for the primary channel pair and shared registers”, followed by FIFO related registers. Registers for any additional channel pairs are shown under the row heading “Registers for secondary channel pairs:”.

The reset value reflects the value of defined bits only, and does not include reserved bits.

Table 499. Register overview for the I2S function of one Flexcomm Interface

Name	Access	Offset [1]	Description	Reset value	Section
Registers for the primary channel pair and shared registers:					
CFG1	R/W	0xC00	Configuration register 1 for the primary channel pair.	0	28.7.1
CFG2	R/W	0xC04	Configuration register 2 for the primary channel pair.	0	28.7.2
STAT	RO/W1C	0xC08	Status register for the primary channel pair.	0	28.7.3
DIV	R/W	0xC1C	Clock divider, used by all channel pairs.	0	28.7.4
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable.	0	28.7.5
FIFOSTAT	R/W	0xE04	FIFO status.	0x18	28.7.6
FIFOTRIG	R/W	0xE08	FIFO trigger settings for interrupt and DMA request.	0	28.7.7
FIFOINTENSET	R/W1C	0xE10	FIFO interrupt enable set (enable) and read.	0	28.7.8
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read.	0	28.7.9
FIFOINTSTAT	RO	0xE18	FIFO interrupt status.	0	28.7.10
FIFOWR	WO	0xE20	FIFO write data.	-	28.7.11
FIFOWR48H	WO	0xE24	FIFO write data for upper data bits. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	28.7.12
FIFORD	RO	0xE30	FIFO read data.	-	28.7.13
FIFORD48H	RO	0xE34	FIFO read data for upper data bits. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	28.7.14
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	28.7.15
FIFORD48HNOPOP	RO	0xE44	FIFO data read for upper data bits with no FIFO pop. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	28.7.16
Registers for secondary channel pairs:					
P1CFG1	R/W	0xC20	Configuration register 1 for channel pair 1.	0	28.7.17
P1CFG2	R/W	0xC24	Configuration register 2 for channel pair 1.	0	28.7.17
P1STAT	RO/W1C	0xC28	Status register for channel pair 1.	0	28.7.18
P2CFG1	R/W	0xC40	Configuration register 1 for channel pair 2.	0	28.7.17
P2CFG2	R/W	0xC44	Configuration register 2 for channel pair 2.	0	28.7.17
P2STAT	RO/W1C	0xC48	Status register for channel pair 2.	0	28.7.18
P3CFG1	R/W	0xC60	Configuration register 1 for channel pair 3.	0	28.7.17
P3CFG2	R/W	0xC64	Configuration register 2 for channel pair 3.	0	28.7.17

Table 499. Register overview for the I2S function of one Flexcomm Interface

Name	Access	Offset [1]	Description	Reset value	Section
P3STAT	RO/W1C	0x68	Status register for channel pair 3.	0	28.7.18
ID register:					
ID	RO	0xFFC	I2S Module identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2S is the selected function.	0xE090 0000	28.7.20

[1] Offset is within the related Flexcomm Interface address space.

28.7.1 Configuration register 1

The CFG1 register contains mode settings, most of which apply to all I2S channel pairs within one Flexcomm Interface. A few settings apply only to the primary channel pair, as noted.

Table 500. Configuration register 1 (CFG1 - offset 0xC00) bit description

Bit	Symbol	Value	Description	Reset value
0	MAINENABLE		Main enable for I2S function in this Flexcomm Interface	0
		0	All I2S channel pairs in this Flexcomm Interface are disabled and the internal state machines, counters, and flags are reset. No other channel pairs can be enabled.	
		1	This I2S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRENABLE bits.	
1	DATAPAUSE		Data flow pause. Allows pausing data flow between the I2S serializer/deserializer and the FIFO. This could be done in order to change streams, or while restarting after a data underflow or overflow. When paused, FIFO operations can be done without corrupting data that is in the process of being sent or received. Once a data pause has been requested, the interface may need to complete sending data that was in progress before interrupting the flow of data. Software must check that the pause is actually in effect before taking action. This is done by monitoring the DATAPAUSED flag in the STAT register. When DATAPAUSE is cleared, data transfer will resume at the beginning of the next frame.	0
		0	Normal operation, or resuming normal operation at the next frame if the I2S has already been paused.	
		1	A pause in the data flow is being requested. It is in effect when DATAPAUSED in STAT = 1.	
3:2	PAIRCOUNT		Provides the number of I2S channel pairs in this Flexcomm Interface This is a read-only field whose value may be different in other Flexcomm Interfaces. 00 = there is one I2S channel pair in this Flexcomm Interface. 01 = there are two I2S channel pairs in this Flexcomm Interface. 10 = there are three I2S channel pairs in this Flexcomm Interface. 11 = there are four I2S channel pairs in this Flexcomm Interface.	0x3
5:4	MSTSLVCFG		Master / slave configuration selection, determining how SCK and WS are used by all channel pairs in this Flexcomm Interface.	0
		0x0	Normal slave mode, the default mode. SCK and WS are received from a master and used to transmit or receive data.	
		0x1	WS synchronized master. WS is received from another master and used to synchronize the generation of SCK, when divided from the Flexcomm Interface function clock.	
		0x2	Master using an existing SCK. SCK is received and used directly to generate WS, as well as transmitting or receiving data.	
		0x3	Normal master mode. SCK and WS are generated so they can be sent to one or more slave devices.	

Table 500. Configuration register 1 (CFG1 - offset 0xC00) bit description

Bit	Symbol	Value	Description	Reset value
7:6	MODE		Selects the basic I2S operating mode. Other configurations modify this to obtain all supported cases. See Section 28.8.2 “Formats and modes” for examples.	0
		0x0	I2S mode a.k.a. “classic” mode. WS has a 50% duty cycle, with (for each enabled channel pair) one piece of left channel data occurring during the first phase, and one pieces of right channel data occurring during the second phase. In this mode, the data region begins one clock after the leading WS edge for the frame. **Remark:** For a 50% WS duty cycle, FRAMELEN must define an even number of I2S clocks for the frame. If FRAMELEN defines an odd number of clocks per frame, the extra clock will occur on the right.	
		0x1	DSP mode where WS has a 50% duty cycle. See remark for mode 0.	
		0x2	DSP mode where WS has a one clock long pulse at the beginning of each data frame.	
		0x3	DSP mode where WS has a one data slot long pulse at the beginning of each data frame.	
8	RIGHTLOW		Right channel data is in the Low portion of FIFO data. Essentially, this swaps left and right channel data as it is transferred to or from the FIFO. This bit is not used if the data width is greater than 24 bits or if PDMDATA = 1. Note that if the ONECHANNEL field (bit 10 of this register) = 1, the one channel to be used is the nominally the left channel. POSITION can still place that data in the frame where right channel data is normally located. **Remark:** If all enabled channel pairs have ONECHANNEL = 1, then RIGHTLOW = 1 is not allowed.	0
		0	The right channel is taken from the high part of the FIFO data. For example, when data is 16 bits, FIFO bits 31:16 are used for the right channel.	
		1	The right channel is taken from the low part of the FIFO data. For example, when data is 16 bits, FIFO bits 15:0 are used for the right channel.	
9	LEFTJUST		Left Justify data.	0
		0	Data is transferred between the FIFO and the I2S serializer/deserializer right justified, i.e. starting from bit 0 and continuing to the position defined by DATALEN. This would correspond to right justified data in the stream on the data bus.	
		1	Data is transferred between the FIFO and the I2S serializer/deserializer left justified, i.e. starting from the MSB of the FIFO entry and continuing for the number of bits defined by DATALEN. This would correspond to left justified data in the stream on the data bus.	
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to the first I2S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0
		0	I2S data for this channel pair is treated as left and right channels.	
		1	I2S data for this channel pair is treated as a single channel, functionally the left channel for this pair. **Remark:** In mode 0 only, the right side of the frame begins at POSITION = 0x100. This is because mode 0 makes a clear distinction between the left and right sides of the frame. When ONECHANNEL = 1, the single channel of data may be placed on the right by setting POSITION to 0x100 + the data position within the right side (e.g. 0x108 would place data starting at the 8th clock after the middle of the frame). In other modes, data for the single channel of data is placed at the clock defined by POSITION.	

Table 500. Configuration register 1 (CFG1 - offset 0xC00) bit description

Bit	Symbol	Value	Description	Reset value
11	PDMDATA		PDM Data selection. This bit controls the data source for I2S transmit, and cannot be set in Rx mode. **Remark:** This bit only has an effect if the device the Flexcomm Interface resides in includes a DMIC subsystem. For the LPC546xx, this bit applies only to Flexcomm Interface 7.	-
		0	Normal operation, data is transferred to or from the Flexcomm Interface FIFO.	
		1	The data source is the DMIC subsystem. When PDMDATA = 1, only the primary channel pair can be used in this Flexcomm Interface. If ONECHANNEL = 1, only the PDM left data is used. **Remark:** The WS rate must match the Fs (sample rate) of the DMIC decimator. A rate mismatch will at some point cause the I2S to overrun or underrun.	
12	SCK_POL		SCK polarity.	0
		0	Data is launched on SCK falling edges and sampled on SCK rising edges (standard for I2S).	
		1	Data is launched on SCK rising edges and sampled on SCK falling edges.	
13	WS_POL		WS polarity.	0
		0	Data frames begin at a falling edge of WS (standard for classic I2S).	
		1	WS is inverted, resulting in a data frame beginning at a rising edge of WS (standard for most “non-classic” variations of I2S).	
15:14	-		Reserved. Read value is undefined, only zero should be written.	-
20:16	DATALEN		Data Length, minus 1 encoded, defines the number of data bits to be transmitted or received for all I2S channel pairs in this Flexcomm Interface. Note that data is only driven to or received from SDA for the number of bits defined by DATALEN. DATALEN is also used in these ways by the I2S: 1. Determines the size of data transfers between the FIFO and the I2S serializer/deserializer. See Section 28.8.4 “FIFO buffer configurations and usage” 2. In mode 1, 2, and 3, determines the location of right data following left data in the frame. 3. In mode 3 (where WS has a one data slot long pulse at the beginning of each data frame) determines the duration of the WS pulse. Values: 0x00 to 0x02 = not supported 0x03 = data is 4 bits in length 0x04 = data is 5 bits in length ... 0x1F = data is 32 bits in length	0
31:21	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.2 Configuration register 2

The CFG2 register contains bits that control various aspects of data configuration.

Table 501. Configuration register 2 (CFG2 - offset 0xC04) bit description

Bit	Symbol	Description	Reset value
8:0	FRAMELEN	Frame Length, minus 1 encoded, defines the number of clocks and data bits in the frames that this channel pair participates in. See Section 28.8.2.1 “Frame format”. 0x000 to 0x002 = not supported 0x003 = frame is 4 bits in total length 0x004 = frame is 5 bits in total length ... 0x1FF = frame is 512 bits in total length **Remark:** If FRAMELEN is an defines an odd length frame (e.g. 33 clocks) in mode 0 or 1, the extra clock appears in the right half. **Remark:** When MODE = 3, FRAMELEN must be larger than DATALEN in order for the WS pulse to be generated correctly.	0
15:9	-	Reserved. Read value is undefined, only zero should be written.	-
24:16	POSITION	Data Position. Defines the location within the frame of the data for this channel pair. POSITION + DATALEN must be less than FRAMELEN. See Section 28.8.2.1 “Frame format”. **Remark:** When MODE = 0, POSITION defines the location of data in both the left phase and right phase, starting one clock after the WS edge. In other modes, POSITION defines the location of data within the entire frame. ONECHANNEL = 1 while MODE = 0 is a special case, see the description of ONECHANNEL. **Remark:** The combination of DATALEN and the POSITION fields of all channel pairs must be made such that the channels do not overlap within the frame. 0x000 = data begins at bit position 0 (the first bit position) within the frame or WS phase. 0x001 = data begins at bit position 1 within the frame or WS phase. 0x002 = data begins at bit position 2 within the frame or WS phase. ...	0
31:25	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.3 Status register

The STAT register provides status flags for the I2S function, and does not include FIFO status. Note that the FIFO status register supplies peripheral interrupt notification and would be the status register normally observed first for an interrupt service. Some information in this register is read-only, some flags can be cleared by writing a 1 to them, details can be found in Table 502.

Table 502. Status register (STAT - offset 0xC08) bit description

Bit	Symbol	Value	Description	Reset value	Type
0	BUSY		Busy status for the primary channel pair. Other BUSY flags may be found in the STAT register for each channel pair.	0	RO
		0	The transmitter/receiver for channel pair is currently idle.		
		1	The transmitter/receiver for channel pair is currently processing data.		
1	SLVFRMERR		Slave Frame Error flag. This applies when at least one channel pair is operating as a slave. An error indicates that the incoming WS signal did not transition as expected due to a mismatch between FRAMELEN and the actual incoming I2S stream.	0	W1C
		0	No error has been recorded.		
		1	An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.		
2	LR		Left/Right indication. This flag is considered to be a debugging aid and is not expected to be used by an I2S driver. Valid when one channel pair is busy. Indicates left or right data being processed for the currently busy channel pair.	-	RO
		0	Left channel.		
		1	Right channel.		
3	DATAPAUSED		Data Paused status flag. Applies to all I2S channels	0	RO
		0	Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the DATAPAUSE control bit in the CFG1 register.		
		1	A data pause has been requested and is now in force.		
31:4	-		Reserved. Read value is undefined, only zero should be written.	-	-

28.7.4 Clock Divider register

The DIV register controls how the Flexcomm Interface function clock is used. See Section 28.8.3 “Data rates” for more details.

Remark: DIV must be set to 0 if SCK is used as an input clock for the I2S function, which is the case when the MSTSLVCFG field in the CFG1 register = 0 or 2.

Table 503. Clock Divider register (DIV - offset 0xC1C) bit description

Bit	Symbol	Description	Reset value
11:0	DIV	This field controls how this I2S block uses the Flexcomm Interface function clock. 0x000 = The Flexcomm Interface function clock is used directly. 0x001 = The Flexcomm Interface function clock is divided by 2. 0x002 = The Flexcomm Interface function clock is divided by 3. ... 0xFFF = The Flexcomm Interface function clock is divided by 4,096.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.5 FIFO Configuration register

This register configures FIFO usage. A peripheral must be selected within the Flexcomm Interface prior to configuring the FIFO.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time. Also note that the FIFO for the selected I2S data direction **must** be enabled because the FIFO is the only means for accessing I2S data.

Table 504. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
2	TXI2SE0		Transmit I2S empty 0.Determines the value sent by the I2S in transmit mode if the TX FIFO becomes empty. This value is sent repeatedly until the I2S is paused, the error is cleared, new data is provided, and the I2S is un-paused.	0	R/W
		0	If the TX FIFO becomes empty, the last value is sent. This setting may be used when the data length is 24 bits or less, or when MONO = 1 for this channel pair.		
		1	If the TX FIFO becomes empty, 0 is sent. Use if the data length is greater than 24 bits or if zero fill is preferred.		
3	PACK48		Packing format for 48-bit data. This relates to how data is entered into or taken from the FIFO by software or DMA.	0	R/W
		0	48-bit I2S FIFO entries are handled as all 24-bit values.		
		1	48-bit I2S FIFO entries are handled as alternating 32-bit and 16-bit values.		
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0, 0x1 = not applicable to I2S. 0x2 = FIFO is configured as 8 entries of 32 bits, each corresponding to 2 16-bit data values for left and right channels. This setting occurs when the I2S DATALEN is less than 16 bits, or from 25 to 32 bits. 0x3 = FIFO is configured as 8 entries of 48 bits, each corresponding to either 2 16-bit data values for left and right channels. This setting occurs when the I2S DATALEN is from 17 to 24 bits.	-	RO
11:6	-		Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request DMA for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request DMA for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		

Table 504. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See Section 7.5.96 “Hardware Wake-up control register”.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 7.5.96 “Hardware Wake-up control register”.	0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX		Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX		Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-		Reserved. Read value is undefined, only zero should be written.	-	-

28.7.6 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 505. FIFO status register (FIFOSTAT - offset 0xE04) bit description

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	-
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

28.7.7 FIFO trigger settings register

This register allows selecting when FIFO-level related interrupts occur.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 506. FIFO trigger settings register (FIFOTRIG - offset 0xE08) bit description

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
10:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 “Hardware Wake-up control register”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 “Hardware Wake-up control register”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO increases to 8 entries (has become full).	0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.8 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 507. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10) bit description

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.9 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 508. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14) bit description

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.10 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 28.7.6 and Section 28.7.7 for details.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 509. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18) bit description

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0
1	RXERR	RX FIFO error.	0
2	TXLVL	Transmit FIFO level interrupt.	0
3	RXLVL	Receive FIFO level interrupt.	0
4	PERINT	Peripheral interrupt.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.11 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 28.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 510. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Description	Reset value
31:0	TXDATA	Transmit data to the FIFO. The number of bits used depends on configuration details.	-

28.7.12 FIFO write data for upper data bits

The FIFOWR48H register is used under certain conditions to write values to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 28.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 511. FIFO write data for upper data bits (FIFOWR48H - offset 0xE24) bit description

Bit	Symbol	Description	Reset value
23:0	TXDATA	Transmit data to the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.13 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO. Details of how FIFORD and FIFORD48H are used can be found in Section 28.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 512. FIFO read data register (FIFORD - offset 0xE30) bit description

Bit	Symbol	Description	Reset value
31:0	RXDATA	Received data from the FIFO. The number of bits used depends on configuration details.	-

28.7.14 FIFO read data for upper data bits

The FIFORD48H register is used under certain conditions to read values from the FIFO. Details of how FIFORD and FIFORD48H are used can be found in Section 28.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 513. FIFO read data for upper data bits (FIFORD48H - offset 0xE34) bit description

Bit	Symbol	Description	Reset value
23:0	RXDATA	Received data from the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.15 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 514. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset value
31:0	RXDATA	Received data from the FIFO.	-

28.7.16 FIFO data read for upper data bits with no FIFO pop

This register acts in exactly the same way as FIFORD48H, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 515. FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP - offset 0xE44) bit description

Bit	Symbol	Description	Reset value
23:0	RXDATA	Received data from the FIFO.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.17 Configuration register 1 for channel pairs 1, 2, and 3

The P1CFG1, P2CFG1, and P3CFG1 registers contain mode settings that apply to channel pairs other than the first pair, within the same Flexcomm Interface.

Table 516. Configuration register 1 for channel pairs 1, 2, and 3 (P1CFG1 - offset 0xC20; P2CFG1 - offset 0xC40; P3CFG1 - offset 0xC60) bit description

Bit	Symbol	Value	Description	Reset value
0	PAIRENABLE		Enable for this channel pair.	0
		0	This I2S channel pair is disabled.	
		1	This I2S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRNABLE bits.	
9:1	-	-	Reserved. Read value is undefined, only zero should be written.	-
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to this I2S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0
		0	I2S data for this channel pair is treated as left and right channels.	
		1	I2S data for this channel pair is treated as a single channel, functionally the left channel for this pair.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.18 Configuration register 2 for channel pairs 1, 2, and 3

These registers contain the frame position for channel pairs beyond the main pair.

Table 517. Configuration register 2 channel pairs 1, 2, 3 (P1CFG2 - offset 0xC24; P2CFG2 - offset 0xC44; P1CFG2 - offset 0xC64)

Bit	Symbol	Description	Reset value
15:0	-	Reserved. Read value is undefined, only zero should be written.	-
24:16	POSITION	Data Position. Defines the location within the frame of the data for this channel pair. See details in the description of POSITION for the primary channel pair.	0
31:25	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.19 Status registers for channel pairs 1, 2, and 3

These read-only registers provide status flags for additional channel pairs beyond the primary channel pair.

Table 518. Status registers for channel pairs 1, 2, and 3 (P1STAT - offset 0xC28; P2STAT - offset 0xC48; P3STAT - offset 0xC68) bit description

Bit	Symbol	Value	Description	Reset value
0	BUSY		Busy status for this channel pair.	0
		0	The transmitter/receiver for this channel pair is currently idle.	
		1	The transmitter/receiver for this channel pair is currently processing data.	
1	SLVFRMERR		Save Frame Error flag. This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details.	0
2	LR		Left/Right indication. This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details.	0
3	DATAPAUSED		Data Paused status flag. This flag is shared by the STAT and PnSTAT registers.	0
		0	Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the DATAPAUSE control bit in the CFG1 register.	
		1	A data pause has been requested and is now in force.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.7.20 Module identification register

The ID register identifies the type and revision of the module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 519. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x00
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE090

28.8 Functional description

28.8.1 AHB bus access

The bus interface to the I2S registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I2S function.

28.8.2 Formats and modes

The format of data frames and WS is determined by several fields in the CFG1 and CFG2 registers, described in Sections 28.7.1 and 28.7.2 respectively. CFG1 and CFG2 together control the formatting of the data and the format of the frame in which the data is contained.

28.8.2.1 Frame format

The overall frame format is defined by fields in the CFG1 and CFG2 registers. The frame includes data related to the primary channel pair and any other channel pairs implemented by this I2S. These fields plus the position of data for each channel pair, as determined by the POSITION field in CFG2, define the main features of the frame.

- MODE: 2-bit field in CFG1 that defines the overall character of the frame.
- FRAMELEN: 9-bit field in CFG2, defines the length of the data frame this I2S participates in. This field is Minus 1 encoded: the value 63 means 64 clocks and bit positions in each frame.
- DATALEN: 5-bit field in CFG1, defines the number of data bits that are used by the transmitter or receiver. This field is minus 1 encoded: the value 15 means 16 data bits. For each channel pair, data is only driven to or received from SDA for the number of bits defined by DATALEN.

  DATALEN is also used in these ways:

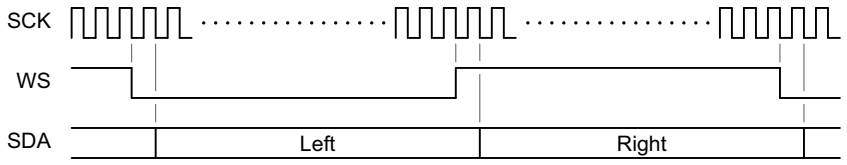
  1) Determines the size of data transfers between the FIFO and the I2S serializer/deserializer.

  2) When MODE = 0x1, 0x2, or 0x3 (i.e. not 0x0), determines the position of Right data following Left data within the frame.

  3) When MODE = 0x3, determines the duration of the WS pulse.

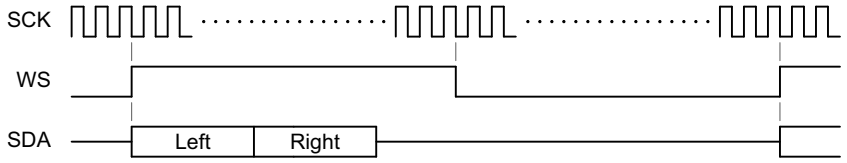
28.8.2.2 Example frame configurations

A sampling of frame slot formats are shown in the following figures. This is not an exhaustive set of possibilities, but shows the various frame formatting concepts. Note that slot identifications are illustrative only, data positions are flexible and there are no predefined slots for the hardware.



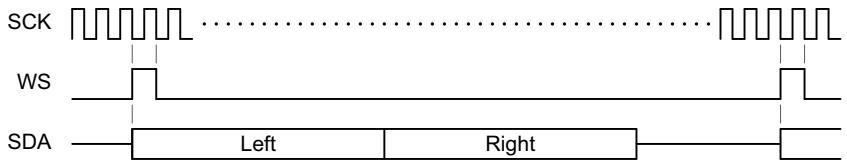
MODE = 0; POSITION = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

Fig 65. Classic I2S mode



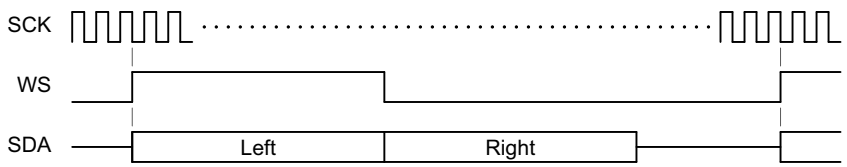
MODE = 1; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 66. DSP mode with 50% WS



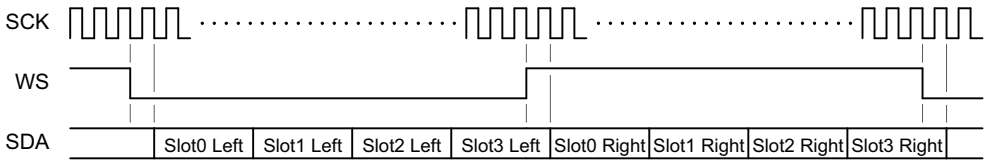
MODE = 2; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 67. DSP mode with 1 SCK pulsed WS



MODE = 3; POSITION = 0; SCK_POL = 0; WS_POL = 1; MONO = 0

Fig 68. DSP mode with 1 slot pulsed WS

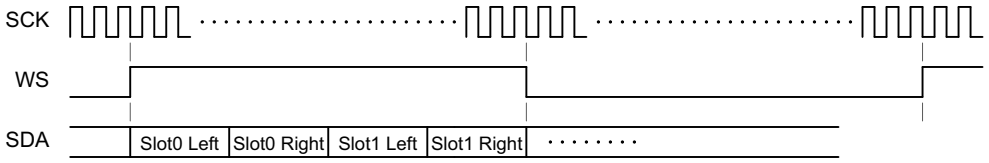


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

POSITION = bit position of the first used data bit for a slot (within the data for each WS phase).

One Left /Right slot is used by one I2S channel pair. This example shows 4 data slots.

Fig 69. TDM in classic I2S mode

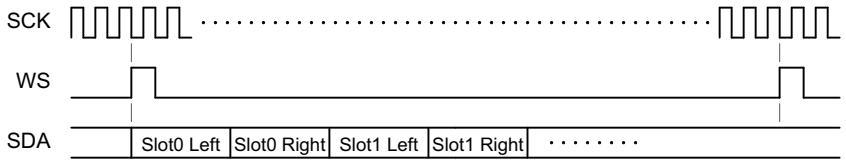


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 70. TDM and DSP modes with 50% WS

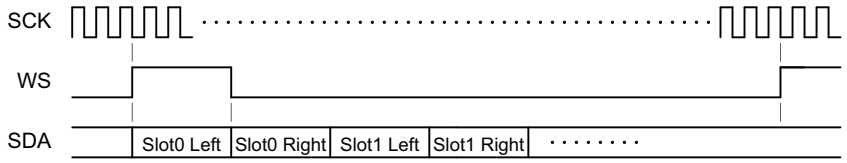


MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 71. TDM and DSP modes with 1 SCK pulsed WS

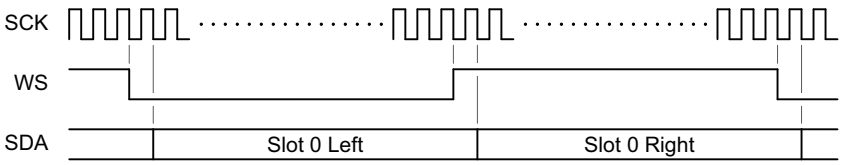


MODE = 3; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 72. TDM and DSP modes with 1 slot pulsed WS

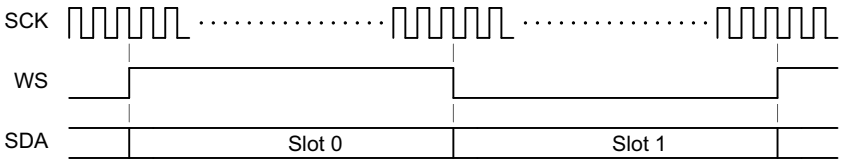


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 1.

POSITiON = bit position of the first used data bit for slot 0 Left, bit position within the second half + 0x100 for Slot 0 Right.

One slot would be used by one I2S.

Fig 73. I2S mode, mono

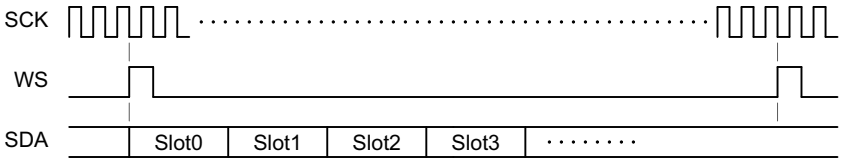


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 1

POSITiON = bit position of the first used data bit for each slot.

One slot would be used by one I2S.

Fig 74. DSP mode, mono



MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 1.

POSITION = bit position of the first used data bit for each slot.

One slot would be used by one I2S.

Fig 75. TDM and DSP modes, mono, with WS pulsed for one SCK time

28.8.2.3 I2S signal polarities

Figure 76 shows examples of SCK and WS polarities and how they relate to data positions.

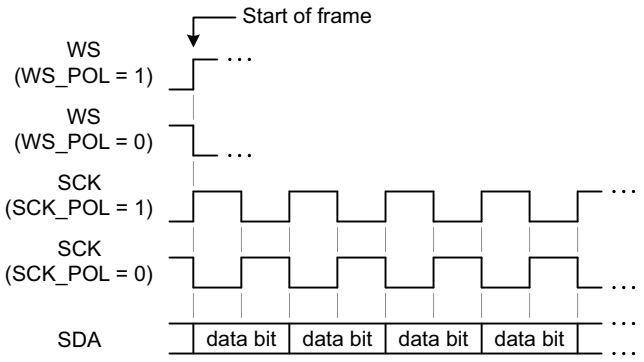


Fig 76. Data at the start of a frame, shown with both SCK and WS polarities

28.8.3 Data rates

28.8.3.1 Rate support

The actual I2S clock rates, sample rates, etc. that can be supported depend on the clocking that is available to run the interface. As a slave, the interface will be receiving SCK from a master. In that case, there is an upper limit to how fast the interface can operate (this will be specified in the interface AC characteristics in a specific device data sheet) and a limit to how much data and be transferred across clock domains and handled by the CPU.

In general, the I2S can support:

- Standard sample rates such as 16, 22.05, 32, 44.1, 48, and 96 kHz, and others.
- External MCLK inputs up to approximately 25 MHz (256 fs of a 96 kHz sample rate) and more. Refer to a specific device data sheet for details.

28.8.3.2 Rate calculations

For operation as a master, the frequency need as the clock input of the I2S is generally an integer multiple of:

- Frame/sample rate * number of bits/clocks in a data frame

If this is a multiple of the desired frequency, the I2S function divider can be used to produce the desired frequency.

Example 1

This I2S channel pair is being used to transfer stereo audio data with 32 bit data slots and a 96 kHz sample rate.

Setup: the sample rate is 96 kHz, the frame is configured for two 32-bit data slots (32-bit stereo). The function clock divider output rate would be 96,000 * (2 * 32) = 6.144 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 24.576 MHz (256 fs of the 96 kHz sample rate), the divider needs to divide by 4 (DIV = 3) to obtain the target divider output rate of 6.144 MHz.

Example 2

This I2S channel pair is being used to supply one 16-bit data slot in a 4 slot frame with a frame rate of 50 kHz.

Setup: the sample rate is 50 kHz, the frame is configured four 16-bit data slots, The function clock divider output rate would be 50,000 * (4 * 16) = 3.2 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 16 MHz, the divider needs to divide by 5 (DIV = 4) to obtain the target divider output rate of 3.2 MHz.

28.8.4 FIFO buffer configurations and usage

The Flexcomm Interface supports several possibilities of data packing/unpacking depending on the size of data being handled.

Some details of FIFO usage are determined by the value of the I2S DATALEN field in the CFG1 register, and some other configuration bits as follows:

- If DATALEN specifies a number of data bits from 4 to 16:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO will be a pair of left and right values, which fit into a 32-bit word. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred, nominally the left.
- If DATALEN specifies a number of data bits from 17 to 24:
  - The FIFO will be configured as 48 bits wide and 8 entries deep.
  - Data transfer between the bus and the FIFO depends the PACK48 configuration bit and whether or not DMA is enabled. When DMA is enabled, all transfers are done with FIFOWR or FIFORD. When DMA is not enabled, transfers will alternate between FIFOWR or FIFORD and FIFOWR48H or FIFORD48H, depending on the data direction selected for the I2S function. In all cases, the 2 transfers will constitute a pair of left and right values. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If PACK48 = 0, each of the two transfers both define 17 to 24 bits of data. If PACK48 = 1, the first transfer provides 32 bits of data, the second provides the remainder need to complete the paired data as defined.
  - If a channel pair is configured with ONECHANNEL = 1, then only the left value is transferred using the FIFOWR or FIFORD register.
- If DATALEN specifies a number of data bits from 25 to 32:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO will be a single value, starting with left, then right.

– If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred.

28.8.5 DMA

The Flexcomm Interface can generate DMA requests based on FIFO levels. Data transfers for any channel can be handled by DMA once the I2S clocking and has been configured, that channel has been configured, DMA has been configured, and the I2S bus is running. DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm Interface I2S may be found in the FIFOCFG register (Section 28.7.5) bits DMATX, DMARX, WAKETX, WAKERX, and PACK48, and in the FIFOTRIG register (Section 28.7.7) bits TXLVLENA, RXLVLENA, and fields TXLVL and RXLVL.

28.8.6 Clocking and power considerations

The master function of the I2S requires the Flexcomm Interface function clock to be running in order to operate. The slave function can operate using external clocks, and can wake up the CPU when data is needed or available.

29.1 How to read this chapter

The DMIC subsystem, including the digital microphone interface (DMIC) and hardware voice activity detector (HWVAD), is available on all LPC546xx devices

29.2 Features

- DMIC (dual/stereo digital microphone interface)
  - PDM (Pulse-Density Modulation) data input for left and/or right channels on 1 or 2 buses.
  - Flexible decimation.
  - 16 entry FIFO for each channel.
  - DC blocking or unaltered DC bias can be selected.
  - Data can be transferred using DMA from deep-sleep mode without waking up the CPU, then automatically returning to deep-sleep mode.
  - Data can be streamed directly to I2S on Flexcomm Interface 7.
- HWVAD (Hardware-based voice activity detector):
  - Optimized for PCM signals with 16 kHz sampling frequency.
  - Configurable detection levels.
  - Noise envelope estimator register output for further software analysis.

29.3 Basic configuration

The DMIC is configured as follows:

- Clock:
  - Enable the clock source that will be used, if it is not already running (most oscillators may be turned off when not needed in order to save power).
  - Select the clock source that will be used in the DMICCLKSEL register. See Section 7.5.40.
  - Set up the clock divider (DMICCLKDIV) that follows the clock source selection mux to obtain the desired clock rate. See Section 7.5.41.
  - Enable clock to the peripheral in the AHBCLKCTRL1 register. See Section 7.5.20.
- Reset: The peripheral may be specifically reset using the PRESETCTRL1 register, but must be removed form the reset state before continuing. See Section 7.5.10.
- Pins: Configure pins that will be used for this peripheral in the IOCON register block. See Chapter 10.
- Interrupts: If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 6.
- Wake-up: Enable interrupts for waking up from deep-sleep mode, enable the interrupts in the STARTER0 register.

- PDM internal setup:
  - Enable DMIC PDM channels via the EN_CH0/1 bits in the CHANEN register. See Section 29.6.1.
  - Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers. See Section 29.6.2.
  - If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 6.
  - If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. See Chapter 15. DMA must also be enabled via the DMAEN bit in the FIFOCTRL register. See Section 29.6.6.
  - Set up other functional configurations and controls for this peripheral as needed.
- HWVAD internal setup:
  - The HWVAD is active when the DMIC interface is active.
  - Reset the filters with a 1 pulse of bit RSTT in register HWVADRSTT.
  - Wait for a few milliseconds to let the filter converge.
  - Enable the HWVAD interrupt with the appropriate NVIC bit. See Chapter 6.
  - Start the HWVAD process by toggling bit ST10 in register HWVADST10 from 0 to 1 and back. This also clears the interrupt flag inside the HWVAD block.
  - In the HWVAD interrupt service routine take appropriate action.
  - Restart the HWVAD with step 6. A precedent reset of the filters in step 3 is optional.

29.4 Pin description

Table 520 gives a brief summary of each of the PDM pins used by the DMIC subsystem.

Table 520. DMIC subsystem PDM pin description

Pin	Type	Description
PDM0_CLK	O	Clock output to digital microphone on PDM interface 0.
PDM0_DATA	I	Data input from digital microphone on PDM interface 0.
PDM1_CLK	O	Clock output to digital microphone on PDM interface 1.
PDM1_DATA	I	Data input from digital microphone on PDM interface 1. Also PDM clock input in bypass mode.
I^2S related pins on Flexcomm Interface 7	I/O	Used as configured for Flexcomm Interface 7 when configured as an I^2S transmitter, and when this function is selected by 7.

Recommended IOCON settings are shown in Table 521. See Chapter 10 for definitions of pin types.

Table 521: Suggested PDM pin settings for the audio input

IOCON bit(s)	Type D pin	Type A pin
10	OD: Set to 0.	Same as type D.
9	SLEW: Generally set to 0.	Not used, set to 0
8	FILTEROFF: Generally set to 1.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.
6	INVERT: Set to 0.	Same as type D.
5	Not used, set to 0.	Same as type D.
4:3	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.
General comment	A good choice for PDM data.	A reasonable choice for PDM data.

The PDM interface provides options to support 2 single-channel microphones or a single stereo microphone. The general connections are shown in Figure 77. Specific use examples are shown in Figure 78 through Figure 80.

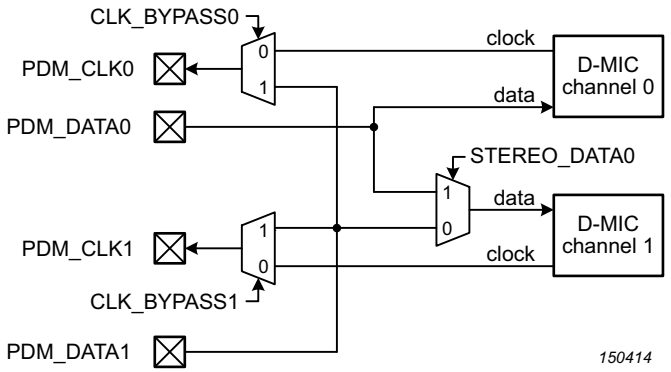
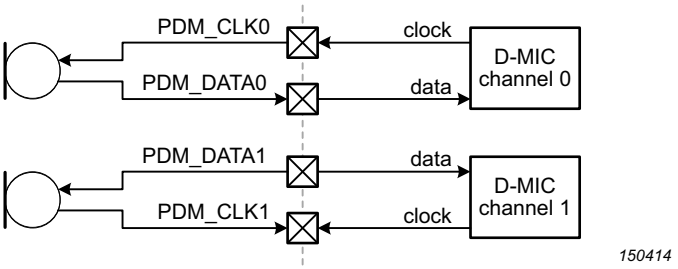
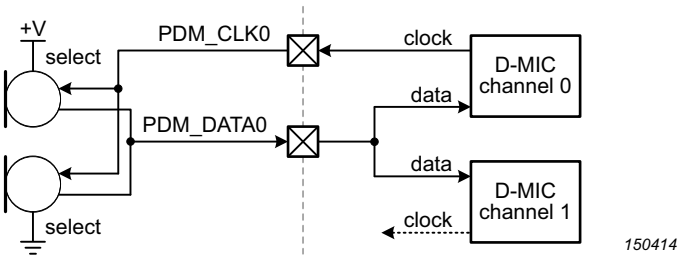


Fig 77. DMIC subsystem pin multiplexing



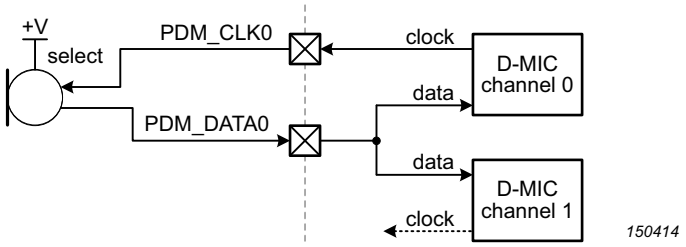
CLK_BYPASS0 = 0; CLK_BYPASS1 = 0; DUAL_DATA = 0

Fig 78. Typical connection to two independent microphones



CLK_BYPASS0 = 0; CLK_BYPASS1 = 0; DUAL_DATA = 1

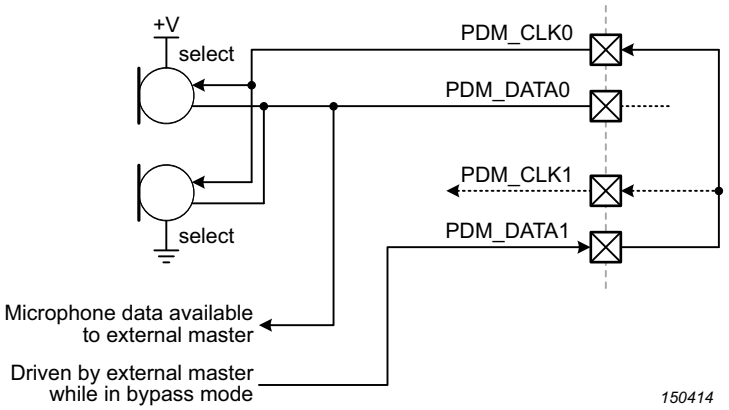
Fig 79. Typical connection to two microphones sharing a data line



CLK_BYPASS0 = 0; CLK_BYPASS1 = 0; DUAL_DATA = 1

Fig 80. Typical connection to a stereo microphone

The PDM interface also provides the possibility of an external codec or other PDM master to take over the PDM interface on this device. An example of this using dual microphones sharing one data line is shown in Figure 81.



CLK_BYPASS0 = 1

Fig 81. Bypass mode with an external device taking over microphone access

29.5 General description

The hardware voice activity detector (HWVAD) implements a wave envelope detector and a floor noise envelope detector. It provides an interrupt when the delta between the two detectors is larger than a predefined value. The input signal for the HWVAD can come from DMIC channel 0.

The basic detection of a voice activity can be the starting point for a more sophisticated task like for example voice recognition. As the DMA for the DMIC subsystem, the HWVAD can be active during deep-sleep mode and therefore provide lowest power operation, compared with a software based implementation.

The DMIC receives PDM data, typically from one or two digital microphones, and produces a data stream that can be ready by the CPU or DMA, or can be sent directly to the I2S of Flexcomm Interface 7.

Detailed descriptions of both blocks can be found in Section 29.7 “Functional description”.

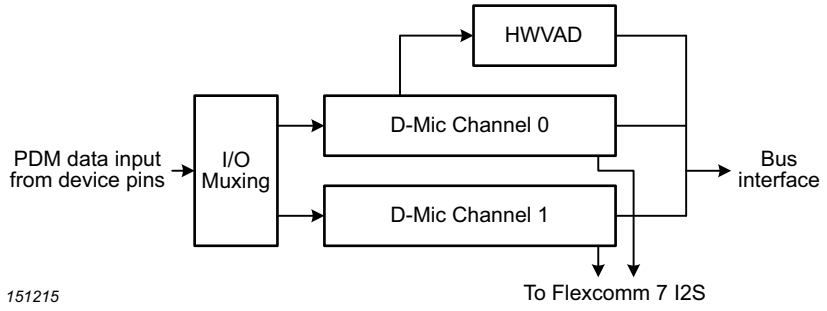


Fig 82. DMIC subsystem block diagram

29.6 Register description

Table 522. Register overview: DMIC subsystem (base address 0x4009 0000)

Name	Access	Offset	Description	Reset value	Section
Registers for DMIC channel 0:					
OSR0	R/W	0x000	CIC filter decimation rate (oversample rate).	0x0	29.6.1
DIVHFCLK0	R/W	0x004	DMIC clock divider.	0x0	29.6.2
PREAC2FSCOEF0	R/W	0x008	Pre-emphasis filter coefficient for 2 FS.	0x0	29.6.3
PREAC4FSCOEF0	R/W	0x00C	Pre-emphasis filter coefficient for 4 FS.	0x0	29.6.4
GAINSHIFT0	R/W	0x010	Decimator output gain adjustment.	0x0	29.6.5
FIFOCTRL0	R/W	0x080	FIFO control.	0x0	29.6.6
FIFOSTAT0	R/W	0x084	FIFO status.	0x0	29.6.7
FIFODATA0	R/W	0x088	FIFO data.	NA	29.6.8
PDMSRCCFG0	R/W	0x08C	Configures details of the PDM hardware interface.	0x0	29.6.9
DCCTRL0	R/W	0x090	DC filter control.	0x0	29.6.10
Registers for DMIC channel 1:					
OSR1	R/W	0x100	Oversample rate.	0x0	29.6.1
DIVHFCLK1	R/W	0x104	DMIC clock.	0x0	29.6.2
PREAC2FSCOEF1	R/W	0x108	Pre-emphasis filter coefficient for 2 FS.	0x0	29.6.3
PREAC4FSCOEF1	R/W	0x10C	Pre-emphasis filter coefficient for 4 FS.	0x0	29.6.4
GAINSHIFT1	R/W	0x110	Decimator gain shift.	0x0	29.6.5
FIFOCTRL1	R/W	0x180	FIFO control.	0x0	29.6.6
FIFOSTAT1	R/W	0x184	FIFO status.	0x0	29.6.7
FIFODATA1	R/W	0x188	FIFO data.	NA	29.6.8
PDMSRCCFG1	R/W	0x18C	Configures details of the PDM hardware interface.	0x0	29.6.9
DCCTRL1	R/W	0x190	DC filter control.	0x0	29.6.10
DMIC common registers:					
CHANEN	R/W	0xF00	Channel enable.	0x0	29.6.11
IOCFG	R/W	0xF0C	Configures aspects of pin usage of the PDM interface.	0x0	29.6.12
USE2FS	R/W	0xF10	Use decimate by multiple of 2.	0x0	29.6.13
HWVAD registers:					
HWVADGAIN	R/W	0xF80	Input gain register.	0x5	29.6.14
HWVADHPFS	R/W	0xF84	Filter control register.	0x1	29.6.15
HWVADST10	R/W	0xF88	Control register.	0x0	29.6.16
HWVADRSTT	R/W	0xF8C	Filter reset register.	0x0	29.6.17
HWVADTHGN	R/W	0xF90	Noise estimator gain register.	0x0	29.6.18
HWVADTHGS	R/W	0xF94	Signal estimator gain register.	0x4	29.6.19
HWVADLOWZ	RO	0xF98	Noise envelope estimator register.	0x0	29.6.20
ID register:					
ID	RO	0xFFC	Module ID.	0x2	29.6.21

29.6.1 Oversample rate register

This register selects the oversample rate (CIC decimation rate) for the related input channel.

Table 523. Oversample Rate register (OSR[0:1], offset 0x000 (OSR0) and 0x100 (OSR1)) bit description

Bit	Symbol	Description	Reset value
7:0	OSR	Selects the CIC decimation rate for the related input channel.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.2 DMIC clock register

This register controls the clock pre-divider for the related input channel.

Table 524. DMIC clock register (DIVHFCLK[0:1], offset 0x004 (DIVHFCLK0) to 0x104 (DIVHFCLK1)) bit description

Bit	Symbol	Description	Reset value
3:0	PDMDIV	PDM clock divider value. 0 = divide by 1 1 = divide by 2 2 = divide by 3 3 = divide by 4 4 = divide by 6 5 = divide by 8 6 = divide by 12 7 = divide by 16 8 = divide by 24 9 = divide by 32 10 = divide by 48 11 = divide by 64 12 = divide by 96 13 = divide by 128 others = reserved.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.3 Pre-Emphasis filter coefficient for 2 FS register

This register selects the pre-emphasis filter coefficient for the related input channel when 2 FS mode is used (see Section 29.6.13 “Use 2 FS register”).

Table 525. Pre-emphasis filter coefficient for 2 FS register (PREAC2FSCOEF[0:1], offset 0x008 (PREAC2FSCOEF0) and 0x108 (PREAC2FSCOEF1)) bit description

Bit	Symbol	Description	Reset value
1:0	COMP	Pre-emphasis filer coefficient for 2 FS mode. See Figure 83. 0 = Compensation = 0 1 = Compensation = -0.16 2 = Compensation = -0.15 3 = Compensation = -0.13	0x0
31:2	-	Reserved. Read value is undefined, only zero should be written.	-

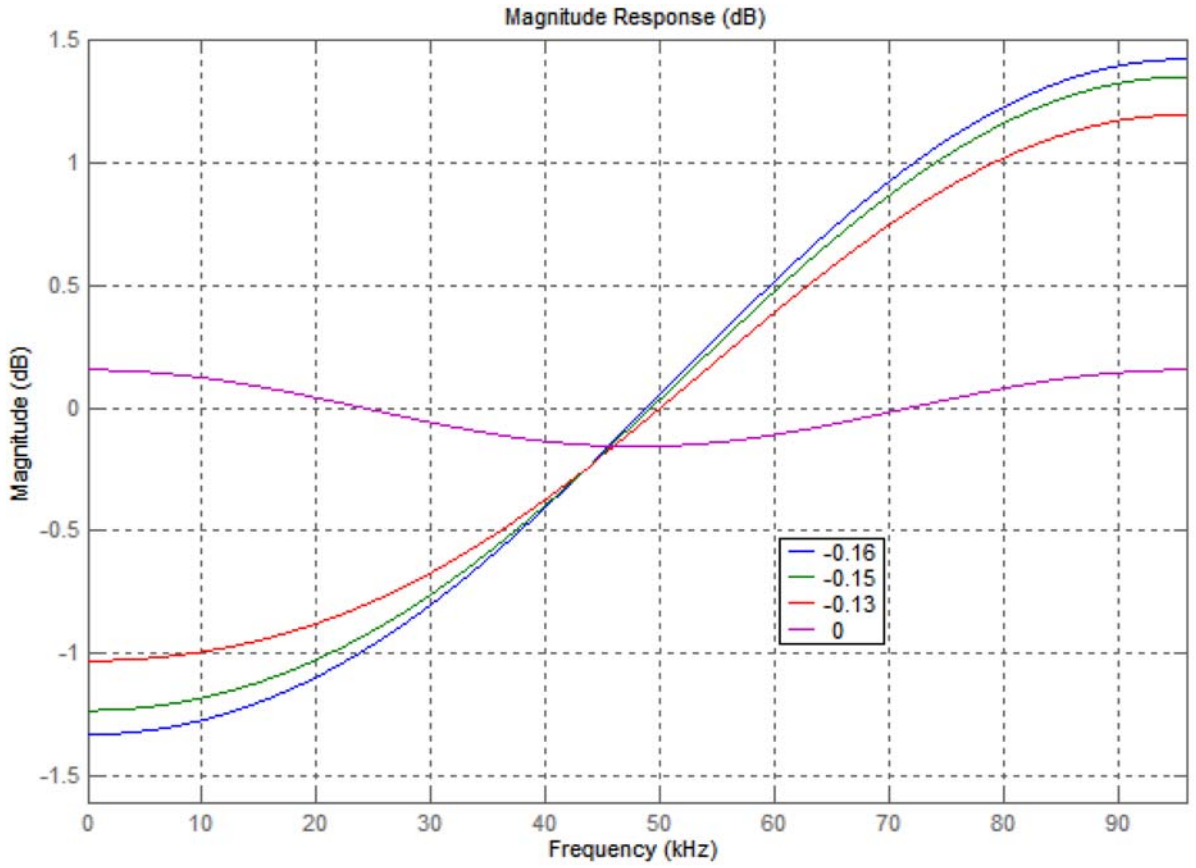


Fig 83. Pre-emphasis filter quantized response at 96 kHz

29.6.4 Pre-emphasis filter coefficient for 4 FS register

This register selects the pre-emphasis filter coefficient for the related input channel when 4 FS mode is used (see Section 29.6.13 “Use 2 FS register”).

Table 526. Pre-emphasis filter coefficient for 4 FS register (PREAC4FSCOEF[0:1], offset 0x00C (PREAC4FSCOEF0) and 0x10C (PREAC4FSCOEF1)) bit description

Bit	Symbol	Description	Reset value
1:0	COMP	Pre-emphasis filer coefficient for 4 FS mode. See Figure 83. 0 = Compensation = 0 1 = Compensation = -0.16 2 = Compensation = -0.15 3 = Compensation = -0.13	0x0
31:2	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.5 Decimator Gain Shift register

This register adjust the gain of the 4FS PCM data from the input filter.

Table 527. Decimator gain shift register (GAINSHFT[0:1], offset 0x010 (GAINSHFT0) and 0x110 (GAINSHFT1)) bit description

Bit	Symbol	Description	Reset value
5:0	GAIN	Gain control, as a positive or negative (two's complement) number of bits to shift.	0x0
31:6	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.6 FIFO control register

This register configures FIFO usage.

Table 528. FIFO control register (FIFOCTRL[0:1], offset 0x080 (FIFOCTRL0) and 0x180 (FIFOCTRL1)) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		FIFO enable.	0x0
		0	FIFO is not enabled. Enabling a DMIC channel with the FIFO disabled could be useful while data is being streamed to the I2S, or in order to avoid a filter settling delay when a channel is re-enabled after a period when the data was not needed.	
		1	FIFO is enabled. The FIFO must be enabled in order for the CPU or DMA to read data from the DMIC via the FIFODATA register.	
1	RESETN		FIFO reset.	0x0
		0	Reset the FIFO. This bit must be cleared before resuming operation.	
		1	Normal operation.	
2	INTEN		Interrupt enable.	0x0
		0	FIFO level interrupts are not enabled.	
		1	FIFO level interrupts are enabled.	
3	DMAEN		DMA enable.	0x0
		0	DMA requests are not enabled.	
		1	DMA requests based on FIFO level are enabled.	
15:4	-	-	Reserved. Read value is undefined, only zero should be written.	-
20:16	TRIGLVL	-	FIFO trigger level. Selects the data trigger level for interrupt or DMA operation. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 7.5.96 "Hardware Wake-up control register". 0 = trigger when the FIFO has received one entry (is no longer empty). 1 = trigger when the FIFO has received two entries. ... 15 = trigger when the FIFO has received 16 entries (has become full).	0x0
31:21	-	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.7 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 529. FIFO status register (FIFOSTAT[0:1], offset 0x084 (FIFOSTAT0) and 0x184 (FIFOSTAT1)) bit description

Bit	Symbol	Description	Reset value
0	INT	Interrupt flag. Asserted when FIFO data reaches the level specified in the FIFOCTRL register. Writing a one to this bit clears the flag. **Remark:** note that the bus clock to the DMIC subsystem must be running in order for an interrupt to occur.	0
1	OVERRUN	Overrun flag. Indicates that a FIFO overflow has occurred at some point. Writing a one to this bit clears the flag. This flag does not cause an interrupt.	0
2	UNDERRUN	Underrun flag. Indicates that a FIFO underflow has occurred at some point. Writing a one to this bit clears the flag.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.8 FIFO data register

The FIFODATA register is used to read values that have been received by the via the PDM stream.

Table 530. FIFO data register (FIFODATA[0:1], offset 0x088 (FIFODATA0) and 0x188 (FIFODATA1)) bit description

Bit	Symbol	Description	Reset value
23:0	DATA	Data from the top of the input filter FIFO.	NA
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.9 PDM source configuration register

This register configure how the PDM source signals are interpreted.

Table 531. PDM source configuration register (PDMSRCCFG[0:1], offset 0x08C (PDMSRCCFG0) to 0x18C (PDMSRCCFG1)) bit description

Bit	Symbol	Value	Description	Reset value
0	PHY_FALL	0	PDM_DATA is sampled into the decimator on the rising edge of PDM_CLK.	0x0
		1	PDM_DATA is sampled into the decimator on the falling edge of PDM_CLK.	
1	PHY_HALF	0	Standard half rate sampling. The clock to the DMIC is sent at the same rate as the decimator is providing.	0x0
		1	Use half rate sampling. The PDM clock to DMIC is divided by 2. Each PDM data is sampled twice into the decimator. The purpose of this mode is to allow slower sampling rate in quiet periods of listening for a trigger. Allowing the decimator to maintain the same decimation rate between the higher quality, higher PDM clock rate and the lower quality lower PDM clock rate means that the user can quickly switch to higher quality without re-configuring the decimator, and thus avoiding long filter settling times, when switching to higher quality (higher freq PDM clock) for recognition.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.10 DC control register

This register controls the DC filter. The frequencies noted for DCPOLE in the table assume a PCM output frequency of 16 MHz. If the actual PCM output frequency is 8 MHz, for example, the noted frequencies would be divided by 2.

Table 532. DC control register (DCCTRL[0:1], offset 0x090 (DCCTRL0) and 0x190 (DCCTRL1)) bit description

Bit	Symbol	Value	Description	Reset value
1:0	DCPOLE		DC block filter.	0x0
		0	Flat response, no filter.	
		1	155 Hz.	
		2	78 Hz.	
		3	39 Hz.	
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
7:4	DCGAIN	-	Fine gain adjustment in the form of a number of bits to downshift.	0x0
8	SATURATEAT16BIT		Selects 16-bit saturation.	0x0
		0	Results roll over if out range and do not saturate.	
		1	If the result overflows, it saturates at 0x7FFF for positive overflow and 0x8000 for negative overflow.	
31:9	-	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.11 Channel enable register

This register allows enabling either or both PDM channels.

Table 533. Channel enable register (CHANEN, offset 0xF00) bit description

Bit	Symbol	Description	Reset value
0	EN_CH0	Enable channel 0. When 1, PDM channel 0 is enabled.	0x0
1	EN_CH1	Enable channel 1. When 1, PDM channel 1 is enabled.	0x0
31:2	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.12 I/O configuration register

This register configures the use of the PDM pins.

Table 534. I/O configuration register (IOCFG, offset 0xF0C) bit description

Bit	Symbol	Description	Reset value
0	CLK_BYPASS0	Bypass CLK0. When 1, PDM_DATA1 becomes the clock for PDM channel 0. This provides for the possibility of an external codec taking over the PDM bus. See Figure 77 through Figure 81.	0x0
1	CLK_BYPASS1	Bypass CLK1. When 1, PDM_DATA1 becomes the clock for PDM channel 1. This provides for the possibility of an external codec taking over the PDM bus. See Figure 77 through Figure 81.	0x0
2	STEREO_DATA0	Stereo PDM select. When 1, PDM_DATA0 is routed to both PDM channels in a configuration that supports a single stereo digital microphone.See Figure 77 through Figure 81.	0x0
31:3	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.13 Use 2 FS register

This register allows selecting 2FS output rather than 1FS output.

Table 535. Use 2FS register (USE2FS, offset 0xF10) bit description

Bit	Symbol	Value	Description	Reset value
0	USE2FS	0	Use 1FS output for PCM data.	0x0
		1	Use 2FS output for PCM data.	
31:1	-	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.14 HWVAD input gain register

This register controls the input gain of the HWVAD.

Table 536. HWVAD input gain register (HWVADGAIN, offset 0xF80) bit description

Bit	Symbol	Description	Reset value
3:0	INPUTGAIN	Shift value for input bits	0x05
		0x00: -10 bits	
		0x01: -8 bits	
		0x02: -6 bits	
		0x03: -4 bits	
		0x04: -2 bits	
		0x05: 0 bits (default)	
		0x06: +2 bits	
		0x07: +4 bits	
		0x08: +6 bits	
		0x09: +8 bits	
		0x0A: +10 bits	
		0x0B: +12 bits	
		0x0C: +14 bits	
		0x0D to 0x0F: Reserved.	
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.15 HWVAD filter control register

This filter setting parameter can be used to optimize performance for different background noise situations. In order to find the best setting, software needs to perform a rough spectral analysis of the audio signal.

Rule of thumb: If the amount of low-frequency content in the background noise is small, then HPFS=0x2, else 0x1.

Table 537. HWVAD filter control register (HWVADHPFS, offset 0xF84) bit description

Bit	Symbol	Value	Description	Reset value
1:0	HPFS	0x0	First filter by-pass.	0x1
		0x1	High pass filter with -3dB cut-off at 1750Hz.	
		0x2	High pass filter with -3dB cut-off at 215Hz.	
		0x3	Reserved.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.16 HWVAD control register

This register controls the operation of the filter block and resets the internal interrupt flag. Once the HWVAD triggered an interrupt, a short 1 pulse on bit ST10 clears the interrupt.

Keeping the bit on 1 level for some time also has a special function for filter convergence, see more information in Section 29.7.1.

Table 538. HWVAD control register (HWVADST10, offset 0xF88) bit description

Bit	Symbol	Value	Description	Reset value
0	ST10	0	Normal operation, waiting for HWVAD trigger event (stage 0).	0x0
		1	Reset internal interrupt flag by writing a 1 pulse.	
31:1	-	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.17 HWVAD filter reset register

Setting bit RSTT to 1 causes a synchronous reset of all filters inside the HVWAD. The RSTT bit must be cleared in order to allow HWVAD operation. See more information in Section 29.7.1.

Table 539. HWVAD filter reset register (HWVADRSTT, offset 0xF8C) bit description

Bit	Symbol	Value	Description	Reset value
0	RSTT		HWVAD filter reset. Writing a 1, then writing a 0 resets all filter values	0x0
		0	Filters are not held in reset.	
		1	Holds the filters in reset	
31:1	-		Reserved. Read value is undefined, only zero should be written.	-

29.6.18 HWVAD noise estimator gain register

Gain value for the noise estimator value. This parameter is used in the following calculation (implemented in hardware):

```
if z8 * (THGS+1) > z7 * (THGN+1)
     HWVAD_RESULT = 1;
else
     HWVAD_RESULT = 0;
```

Table 540. HWVAD noise estimator gain register (HWVADTHGN, offset 0xF90) bit description

Bit	Symbol	Description	Reset value
3:0	THGN	Gain value for the noise estimator.	0x0
		0 to 14: 0 corresponds to a gain of 1.	
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.19 HWVAD signal estimator gain register

Gain value for the signal estimator value. This parameter is used in the following calculation (implemented in hardware):

```
if z8 * (THGS+1) > z7 * (THGN+1)
     HWVAD_RESULT = 1;
else
     HWVAD_RESULT = 0;
```

Table 541. HWVAD signal estimator gain register (HWVADTHGS, offset 0xF94) bit description

Bit	Symbol	Description	Reset value
3:0	THGS	Gain value for the signal estimator.	0x4
		0 to 14: 0 corresponds to a gain of 1.	
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.20 HWVAD noise envelope estimator register

This register contains 2 bytes of the output of filter stage z7. It can be used as an indication for the noise floor and must be evaluated by software. See more information in Section 29.7.1.

Note: For power saving reasons this register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read can be assured to not be in the middle of a transition.

Table 542. HWVAD noise envelope estimator register (HWVADLOWZ, offset 0xF98) bit description

Bit	Symbol	Description	Reset value
15:0	LOWZ	Noise envelope estimator value.	0x0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

29.6.21 Module Identification register

The ID register identifies the type and revision of the module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 543. Module Identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
31:0	ID	Indicates module ID and the number of channels in this DMIC interface.	0x2

29.7 Functional description

29.7.1 HWVAD

The hardware voice activity detector (HWVAD) analyses the PCM data from DMIC channel 0 by means of a filter block. Both the noise floor and the signal wave are examined and result in separate filter outputs. The HWVAD interrupt is issued when a specific delta between the signal and the noise result is detected

Gain levels for the input signal as well as for the signal and noise filter outputs can be set independently from each other, in order to adapt the HWVAD to different acoustic situations.

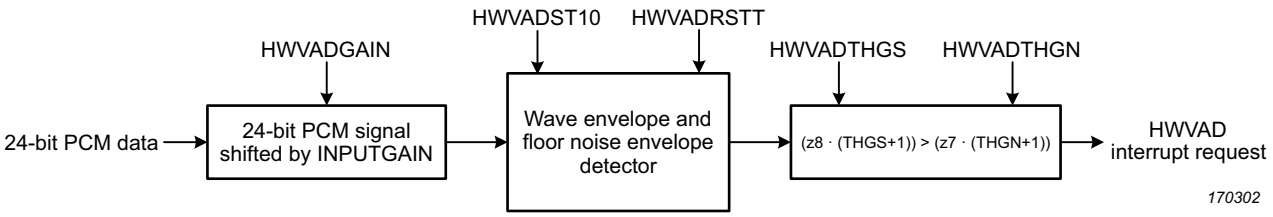
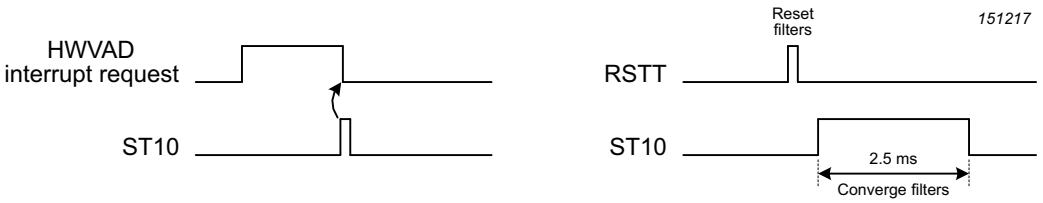


Fig 84. HWVAD block diagram

Because of the non-uniqueness of the input signal, which includes normally noise and voice with various frequency components and different volume, there is no one-and-only operation mode for the HWVAD. The few parameters as well as the chronology can play an important role for a good performance.

29.7.1.1 Basic operations

There are some basic operations for the HWVAD, which can be combined differently in order to achieve different behavior.



1) The internal HWVAD interrupt flag is reset with a short pulse on ST10
2) A pulse on RSTT resets all detection filters
3) Keeping ST10 high for a period causes a special filter convergence. The 2.5ms period is valid for 800kHz DMIC sample rate. For 1MHz sample rate the period is 2ms.

Fig 85. Use of ST10 and RSTT controls (in the HWVADST10 and HWVADRSTT registers)

With bit ST10 the HWVAD can be prepared for an interrupt. The internal flag is reset with the rising edge of ST10 and the HWVAD waits for the next event. In case the application involves some post-processing after a HWVAD event (outside of the interrupt service routine), the flag should only be cleared at the end of this processing. The interrupt status on NVIC level is not affected by this bit setting

With bit RSTT in register HWVADRSTT all filters can be reset. After this reset the HWVAD filters need to converge, so for the first few milliseconds the result is not reliable. The HWVAD interrupt should be masked on NVIV level during this time frame. The wait period depends on the sample rate of the incoming data, at 1MHz DMIC sample rate the filters need about 2ms to converge, for 800KHz the period is 2.5ms.

If it makes sense to reset the filters before starting into a new detection process depends on the use case. For a voice application the filters can adapt continuously to the background environment, between the voice events there is normally enough time to let the filters converge to a changed background noise situation.

Keeping ST10 on high level during the convergence period enables a special mode. If the filters should adapt to a current background noise floor (without voice), then this can be done during this period. With ST10 returning to low level, the filter calculation is then based on a different filter pre-setting. This could be an advantage in special type of applications, where the signal is not continuously delivered to the HWVAD. In a DMIC system with continuous sampling, this convergence period is not required, bit ST10 is just used to clear the interrupt flag.

A complete setup sequence for standard operation looks like this:

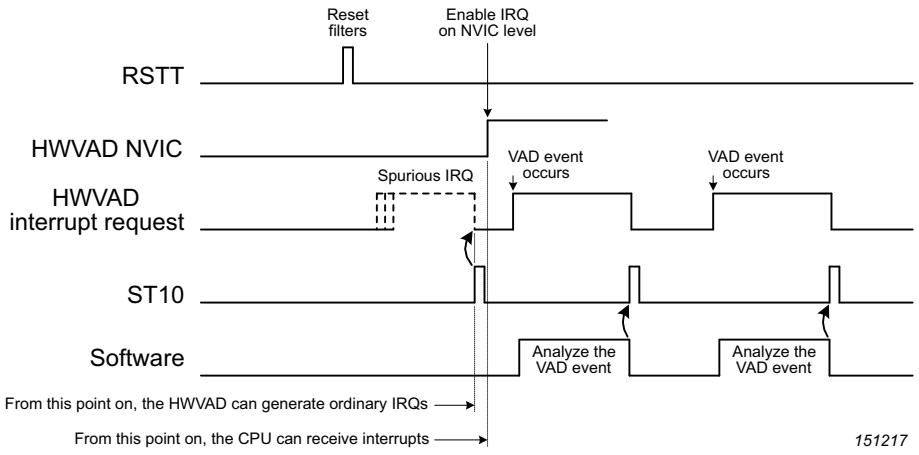


Fig 86. Complete HWVAD setup

1. Reset filters with bit RSTT and provide some time to let the filter converge to the signal conditions.
2. Pulse bit ST10 to clear any spurious interrupts which were generated during bad filter conditions.
3. Enable HWVAD IRQ on NVIC level.
4. Process the VAD event in case of an interrupt, when finished clear the interrupt flag with a high pulse of ST10.

29.7.1.2 Extended operation

There are a few parameters which can be set to influence the behavior of the HWVAD. There is also an intermediate filter result value available, which can be used for proprietary software-based analysis.

29.7.1.2.1 Input gain setting

The 16-bit PCM input signal can be shifted left or right with the gain setting in the register HWVADGAIN. This increases or decreases the volume of the input signal for the HWVAD processing. Note that the reset value 0x05 equals a gain factor of 1, the signal is not shifted in either direction.

29.7.1.2.2 Filter result gain setting

The output values for the final equation can also have a gain factor.

[z8 * (THGS+1)] > [z7 * (THGN+1)]

These gain factors determine the proportion between the results of the signal and the noise filters. The values depend on the audio signal and noise environment, the reset values THGN = 0 and THGS = 4 are more suitable for a low noise environment. For noisy environment the gain for THGN and THGS needs to be increased. In a typical voice recognition application THGN = 3 and THGS = 6 is a good starting point.

29.7.1.2.3 High pass filter setting

The setting in register HWVADHPFS can be used to adapt the filters to different background noise situations. In order to find the best setting, software could perform a rough spectral analysis of the audio signal.

For a background with more low-frequency content, HPFS should be set to 0x1. This is the standard use case. For environments where the low-frequency content is small, the filter can be set to 0x2.

29.7.1.2.4 Noise floor evaluation

The register HWVADLOWZ contains 2 bytes of the output of filter stage z7, which computes the noise floor. The characteristic of the filter block for voice applications is best for a value of 500 … 1000 in LOWZ. Software can tune the input gain to get the LOWZ value into this region.

Note: For power saving reasons this register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read can be assured to not be in the middle of a transition.

29.7.2 DMIC

The DMIC interface receives PDM data from one or two digital microphones and processes it to produce 24-bit PCM data. This data can be read by the CPU or DMA, and/or can be sent to the I2S interface for output. Many aspects of DMIC operation can be controlled. A block diagram of one DMIC channel is shown in Figure 87.

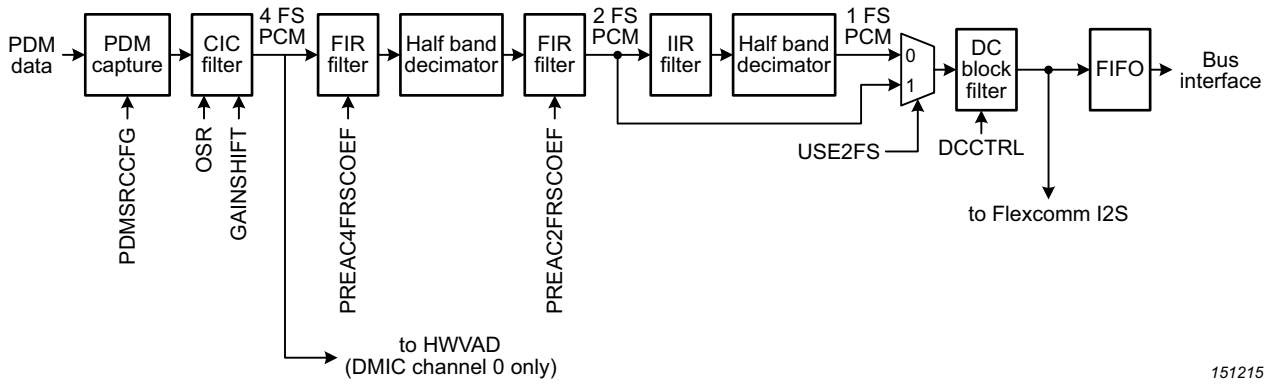


Fig 87. DMIC channel block diagram

29.7.2.1 Clocking and DMIC data rates

The DMIC interface operation is determined by 3 clock domains:

- DMIC interface base clock: supply clock for the peripheral block
- DMIC clock: sample clock for the digital microphone
- PCM sample rate: sample rate of the PCM data resulting from the PDM to PCM conversion

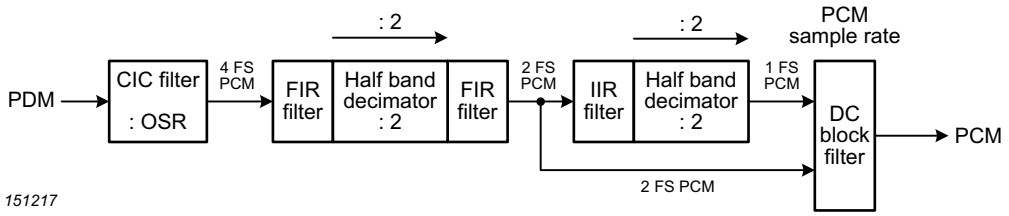


Fig 88. DMIC interface clock domains

The source for the base clock can be set in register DMICCLKSEL in the SYSCON block (see Section 7.5.40). Note that all of these clock sources may be divided by a factor of up to 256 by the DMIC clock divider, controlled by DMICCLKDIV (Section 7.5.58).

Table 544. Table 1.Base clock sources for DMIC interface peripheral

Source	Range	DMIC interface base clock	Note
FRO	12 MHz	≤ 12 MHz	
FRO	48 MHz or 96 MHz	≤ 24 MHz	
System PLL	1.2 MHz … 150 MHz	≤ 24.576 MHz	PLL can produce up to 150 MHz, but main clock is limited to 100 MHz
MCLK input	≤ 100 MHz	≤ 24.576 MHz	
Main clock	≤ 100 MHz	≤ 24.576 MHz	
Watchdog oscillator	6 kHz … 1.5 MHz	≤ 1.5 MHz	Inaccurate clock, but very low power

For the DMIC clock, the base clock divider values can be set in registers DIVHFCLK[0:1].

However, for power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example using register DMICCLKDIV in the SYSCON block).

The DMIC peripheral block is designed to run at a DMIC clock speed no faster than 6.144 MHz and with an input frequency no faster than 4 * 6.144 MHz = 24.576 MHz. With regards to power consumption the lowest possible frequency should be selected. This frequency very much depends on the application requirements. For a simple voice activity detection a sample rate of 200 kHz for the DMIC might be sufficient, for a good quality voice tag recognition the DMIC should be clocked at least with 800 kHz. Depending on the current operating mode of the application, the clocks can be set dynamically from one sample rate to the other.

For a glitch free reduction of the DMIC clock rate by factor 2 the DMIC interface contains dedicated circuitry. By setting bit PHY_HALF in registers PDMSRCCFG[0:1] the DMIC clock is divided to half the frequency internally used for the filters. This enables an on-the-fly switching of the DMIC clock without affecting the operation of the filters. As long as the sample quality on half of the frequency is good enough for the application, for example in listening mode only, this helps to decrease the power consumption of the external digital microphone.

If the PDM interface operates during deep-sleep mode (always listening), then the presence of the clock source in this mode must be taken into account as well. For example, the PLL output is not present during deep-sleep mode, but the 12 MHz FRO is there.

In general, other clocks such as the 48 MHz FRO or the watchdog oscillator are available in deep-sleep mode. It depends on the use case whether a faster or slower clock provides any advantage to the system. At high PDM data rates, for example at 6 MHz, a 48 MHz clock will shorten the “awake” periods compared to 12 MHz operation. A trade-off between the sleep and the active periods, and the internal voltage required at the chosen clock rate, that determine which of the clocks perform better in terms of average power consumption.

The watchdog oscillator low power operation can help to drive the power consumption down in simple voice detection setups. By running the DMIC interface on the slow watchdog oscillator frequency, the HWVAD feature can provide a first audio detection trigger signal to the system. Hereafter the sample rate as well as the processor performance is increased in order to run more sophisticated voice detection and/or voice recognition algorithms.

29.7.2.2 PDM to PCM conversion

The filter block for PDM to PCM conversion consists of four stages. It begins with a CIC filter (Cascaded-Integrator Comb filter) filter which is an optimized finite impulse response (FIR) filter combined with a decimator.The CIC filter converts the PDM stream from the digital microphone into PCM data with a given oversampling rate, set in registers OSR[0] and OSR[1] for each of the two channels. The second block performs a decimation by 2 and compensates for a roll-off at the upper limit of the audio band. The third block decimates the signal again by half, resulting in a PCM signal with the desired sample rate. A final DC filter removes any unwanted DC component in the audio signal.

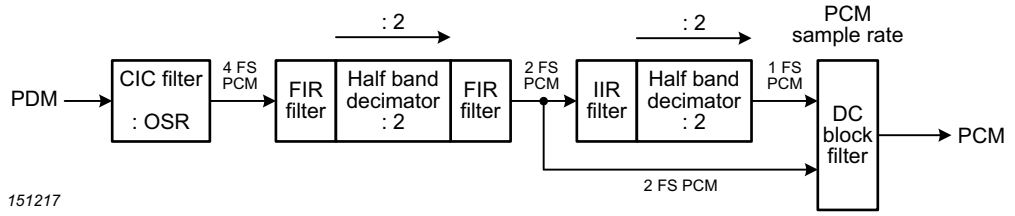


Fig 89. Principle structure of the PDM to PCM conversion

To achieve lower power consumption, the DC filter can be supplied with the 2FS instead of the 1FS signal, bypassing the second half band decimator filter. This reduces the required DMIC base clock by a factor of 2. This is done by setting the USE2FS bit in the USE2FS register.

The PDM to PCM conversion block is designed for providing best results for PCM output signals with a 16 kHz sample rate, covering the enhanced 8 kHz speech band widely used in communication systems. However, other sample rates can be realized as well.

The final relation between the DMIC clock rate and the PCM audio sample rate is:

Table 545. DMIC input and output clock rates

2 FS mode	1 FS mode
PCM Sample rate = DMIC clock rate / (2 * OSR)	PCM Sample rate = DMIC clock rate / (4 * OSR)

Example: DMIC clock = 800 kHz, OSR = 25, 2 FS used

The 800 kHz DMIC data is downsampled by 25 times to 32 kHz. With the following half-band filter the final PCM sample rate is 16 kHz.

29.7.2.3 FIFO and DMA operation

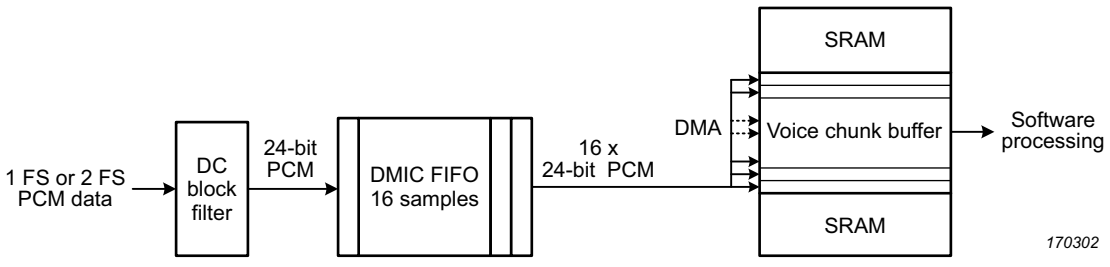


Fig 90. DMIC FIFO and DMA

The 24-bit wide FIFO of the DMIC interface consists of 16 entries for each of the two channels. The trigger level for the FIFO can be set in register FIFOCTRL[0:1] individually for each channel.

The trigger level interrupt for the DMIC interface needs to be enabled on NVIC level and with bit INTEN in register FIFOCTRL[0:1]. Bit DMAEN enables DMA operation. With each FIFO trigger level event the DMA performs a copy of the data from the FIFO into SRAM.

This data batching works without contribution of the core. When reaching the defined chunk buffer size, the DMA issues an interrupt to the ARM core for further processing of the data.

This also works when the device is in deep-sleep mode, as the FIFO event is able to wake up the required part of the hardware. After the DMA finished the job, the device will return into deep-sleep or deep-sleep 2. The two DMIC channel DMA requests are connected to the DMA request input #16 and #17, see table 170 (UM draft v0.6).

Since each DMIC channel provides a separate DMA request, the most obvious configuration of DMA is to have left and right data in separate memory buffers. However, it is possible to configure the DMA controller to interleave left and right data if that is preferable in the application. To do this, the DMA is set up with a data size of halfword, but the next address written to is a word address distance away. The two descriptors would be started on consecutive halfwords. Data is delivered by the DMIC as left channel followed by right channel for each PCM stereo sample.

If more history data is required for a software algorithm, another DMA request can be set up, which copies the current chunk into a larger ring buffer structure. For algorithms like voice detection or voice recognition this is key, in order to converge the software filters to the current background noise situation.

For operation without DMA the dedicated DMIC FIFO interrupt can be enabled in order to inform the ARM core about the FIFO status.

Example:

- PCM output sample rate is 16 kHz
- The 32-bytes FIFO gets full every 1ms
- The DMA copies every 1ms the 32-bytes content of the DMIC FIFO to SRAM
- The DMA is configured to move 512 bytes (= 256 PCM samples) from the DMIC FIFO to SRAM before issuing a DMA interrupt
- Every 16 ms the 256 PCM samples are processed by the ARM core

29.7.2.4 PCM data output on I2S interface

The converted data can be put on the I2S interface on Flexcomm Interface 7, providing TX data for an external CODEC or host system. The sample rate of the PCM signal and the I2S data rate must match to avoid overrun or underrun.

The I2S interface can be set for this mode with the PDMDATA bit in the CFG1 register the I2S of Flexcomm Interface 7.

29.7.2.5 Usage of the DMIC interface in power save modes

The DMIC interface can batch the serial PDM stream from a digital microphone in deep-sleep mode. This requires an appropriate base clock which is active in the respective power saving mode. The best fit for this base clock is the 12 MHz FRO, which provides a good trade-off between power consumption and performance. For lower power operation the watchdog oscillator can be used, taking into account that the clock is relatively inaccurate and the DMIC sample rate is rather low. At any time an external low power clock, connected to pin MCLK, can be used.

In combination with the HWVAD this provides lowest power consumption in listening mode. Except for the short periods with DMA activity, the MCU can remain in deep-sleep mode until the wave envelope detector of the HWVAD identifies an energy change event and issues an interrupt. With the DMA set to larger transfer sizes (maximum is 1024 transfers), there is quite some history data available for any type of software-based analysis of the data causing the HWVAD event.

This also enables the system to realize different strategies for dealing with a HWVAD event. A concrete analysis of the data could for example just be started when the HWVAD detected events over a longer time frame. This would avoid that the ARM core gets active on spurious noise. In case the decision has been taken to take the next step in data analysis, the history buffer still contains the complete PCM data sampled since the first event, nothing got lost. In average the system can stay longer in power save mode if spurious events can be filtered out.

30.1 How to read this chapter

The SD/MMC card interface is available on all LPC546xx devices.

30.2 Features

The SD/MMC card interface supports the following features:

- Secure Digital memory protocol commands.
- Secure Digital I/O protocol commands.
- Multimedia Card protocol commands.
- CE-ATA digital protocol commands.
- Command Completion signal and interrupt to processor.
- Completion Signal disable feature.
- One SD or MMC (4.4), CE-ATA (1.1), or eMMC (4.4) device.
- CRC 2.0 generation and error detection.
- SDIO interrupts in 1-bit and 4-bit modes.
- SDIO suspend and resume operation.
- SDIO read wait.
- Block size of 1 to 65,535 bytes
- FIFO over-run and under-run prevention by stopping card clock.
- Little-endian mode of AHB operation.
- Internal (bus mastering) DMA.
- Two FIFOs, TX and RX FIFO (FIFO depth = 32 and FIFO data width = 32 bits).

30.3 Basic configuration

The SD/MMC interface is configured as follows:

- Clock:Section 30.9.2.3
  - Enable the clock source that will be used, if it is not already running (most oscillators may be turned off when not needed in order to save power).
  - Select the clock source that will be used in the SDIOCLKSEL register. See Section 7.5.43.
  - When using phase delay, the input clock selected in the SDIOCLKSEL register must be 2x of the SDIO clock. See Section 30.9.2.3, SDIOCLKCTRL register. For clocking and timing guidelines, see Section 30.9.
  - Set up the clock divider (SDIOCLKDIV) that follows the clock source selection mux to obtain the desired clock rate. See Section 7.5.63.

    Remark: The SDIO function clock to the interface can be up to 50 MHZ.

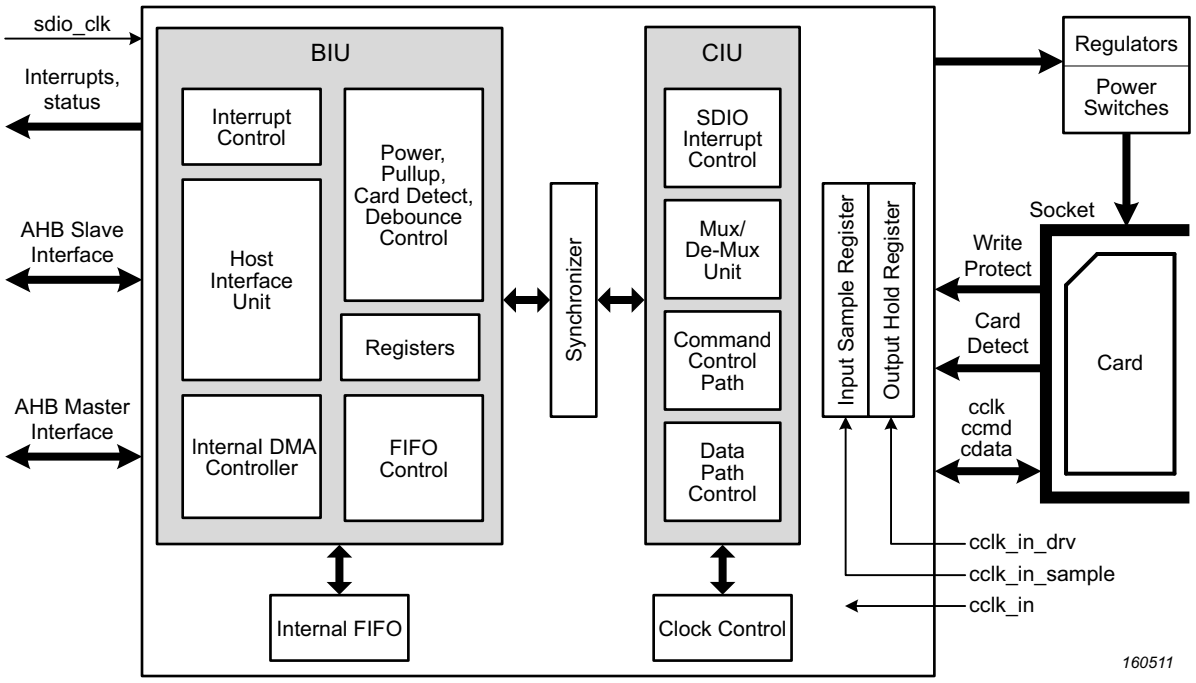
- Enable clock to the peripheral in the AHBCLKCTRL2 register. See Section 7.5.21.

- Reset: The peripheral may be specifically reset using the PRESETCTRL1 register, but must be removed form the reset state before continuing. See Section 7.5.10.
- Pins: Configure pins that will be used for this peripheral in the IOCON register block. See Chapter 10 for IOCON details, and Section 30.5 for recommended IOCON settings for the SDIO.
- Interrupts: If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 6.
- Peripheral internal setup:
  - Configure the SDIO/SDMMC as needed for the application.
- The delay values on the sample and drive inputs and outputs can be adjusted using the SDIOCLKCTRL register in the SYSCON block. See Section 7.5.76.

30.4 General description

The SD/MMC controller interface consists of the following main functional blocks:

- Bus Interface Unit (BIU) - Provides AHB and DMA interfaces for register and data read/writes.
- Card Interface Unit (CIU) - Handles the card protocols and provides clock management.
- Internal MCI DMA controller: AHB bus mastering DMA controller



Note: The Card Detect and Write Protect signals are from the SD/MMC card socket, not the SD/MMC card itself.

Fig 91. SD/MMC block diagram

30.5 Pin description

Table 546. SD/MMC pin description

Pin function	Type	Description
SD_CLK	O	SD/SDIO/MMC clock.
SD_CARD_DET_N	I	SDIO card detect for single slot. A 0 represents the presence of a card.
SD_WR_PRT	I	SDIO card write protect. A 1 represents write is protected.
SD_CMD	I/O	Command input/output.
SD_D[7:0]	I/O	Data input/output for data lines DAT[7:0].
SD_VOLT[2:0]	O	SD/MMC bus voltage select output 2:0. SD/MMC General Purpose Output pins on pins SD_VOLT0, SD_VOLT1, and SD_VOLT2. These pins can be used to control an optional external regulator for the SD/MMC slot. If an external regulator is used to control the SD/MMC slot, voltage level translation will be needed between the IO pads and the SD/MMC slot.
SD_POW_EN	O	SD/SDIO/MMC slot power enable.
SD_BACKEND_PWR	O	Back-end power supply for embedded device. Controls back-end power supply for one embedded device; this bit does not control the VDDH of the host controller. A register bit enables software programming. The value on this register controls switching on and off of power to embedded device.
SD_CARD_INT_N	I	Card interrupt line. This pin is used to indicate a card interrupt, which is sampled even when the clock to the card is switched off. Connected to the eSDIO card interrupt line; it is defined only for eSDIO.

Table 547: Suggested pin settings for SD_CLK, SD_CMD, SD_Dn

IOCON bit(s)	Type D pin	Type A pin
11	OD: Set to 0 unless open-drain output is desired.	Same as type D.
10	SLEW: Set to 1.	Not used, set to 0
9	FILTEROFF: Set to 1.	Same as type D.
8	DIGIMODE: Set to 1.	Same as type D.
7	INVERT: Set to 0.	Same as type D.
6	Not used, set to 0.	Same as type D.
5:4	MODE: set to 0.	Same as type D.
3:0	FUNC: Must select the correct function for this peripheral.	Same as type D.
General comment	A good choice for SDIO functions.	A potential choice, performance may be reduced by absence of the SLEW function

30.6 Register description

Figure 92 shows the memory map of the SDIO peripheral.

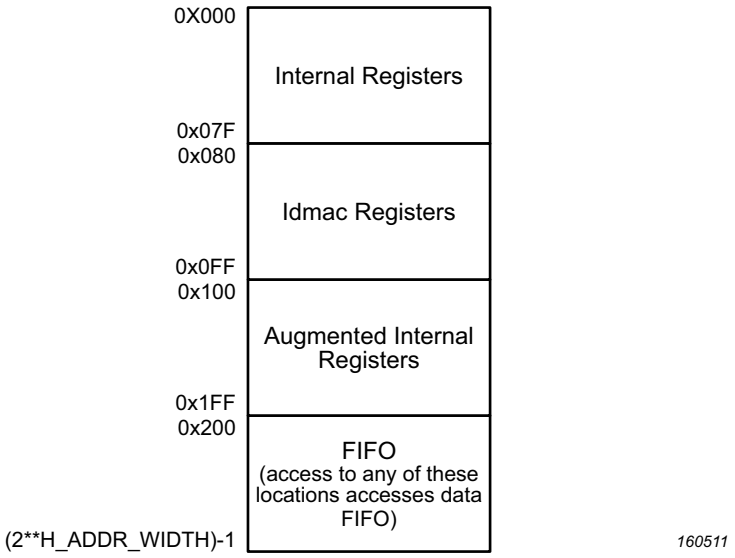


Fig 92. SDIO memory map

Table 548. Register overview: SDMMC (base address: 0x4009 B000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control.	0	30.6.1
PWREN	R/W	0x004	Power enable.	0	30.6.2
CLKDIV	R/W	0x008	Clock divider.	0	30.6.3
CLKENA	R/W	0x010	Clock enable.	0	30.6.4
TMOUT	R/W	0x014	Time-out.	0xFFFF FF40	30.6.5
CTYPE	R/W	0x018	Card type.	0	30.6.6
BLKSIZ	R/W	0x01C	Block size.	0x200	30.6.7
BYTCNT	R/W	0x020	Byte count.	0x200	30.6.8
INTMASK	R/W	0x024	Interrupt mask.	0	30.6.9
CMDARG	R/W	0x028	Command argument.	0	30.6.10
CMD	R/W	0x02C	Command.	0	30.6.11
RESP0	R	0x030	Response 0.	0	30.6.12
RESP1	R	0x034	Response 1.	0	30.6.13
RESP2	R	0x038	Response 2.	0	30.6.14
RESP3	R	0x03C	Response 3.	0	30.6.15
MINTSTS	R	0x040	Masked interrupt status.	0	30.6.16
RINTSTS	R/W	0x044	Raw interrupt status.	0	30.6.17
STATUS	R	0x048	Status.	0x406	30.6.18
FIFOTH	R/W	0x04C	FIFO threshold watermark.	0x0F80 0000	30.6.19
CDETECT	R	0x050	Card detect.	0	30.6.20

Table 548. Register overview: SDMMC (base address: 0x4009 B000)

Name	Access	Offset	Description	Reset value	Section
WRTPRT	R	0x054	Write protect.	0	30.6.21
TCBCNT	R	0x05C	Transferred CIU card byte count.	0	30.6.22
TBBCNT	R	0x060	Transferred host to BIU-FIFO byte count.	0	30.6.23
DEBNCE	R/W	0x064	Debounce count.	0xFFFFFF	30.6.24
RST_N	R/W	0x078	Hardware reset.	0x1	30.6.25
BMOD	R/W	0x080	Bus mode.	0	30.6.26
PLDMND	W	0x084	Poll demand.	0	30.6.27
DBADDR	R/W	0x088	Descriptor list base address.	0	30.6.28
IDSTS	R/W	0x08C	Internal DMAC status.	0	30.6.29
IDINTEN	R/W	0x090	Internal DMAC interrupt enable.	0	30.6.30
DSCADDR	R	0x094	Current host descriptor address.	0	30.6.31
BUFADDR	R	0x098	Current buffer descriptor address.	0	30.6.32
CARDTHRCTL	R/W	0x100	Card threshold control. Controls whether the host controller initiates transfers depending on the FIFO level.	0	30.6.33
BACKENDPWR	R/W	0x104	Power control.	0	30.6.34
DATA	R/W	≥ 0x200	Data FIFO read/write; if address is equal or greater than 0x100, then FIFO is selected as long as device is selected. Address 0x100 and above are mapped to the data FIFO. More than one address is mapped to the data FIFO so that the FIFO can be accessed using bursts.	-	-

30.6.1 Control register

Table 549. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	CONTROLLER_RESET		Controller reset. To reset controller, software should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets: - BIU/CIU interface - CIU and state machines - ABORT_READ_DATA, SEND_IRQ_RESPONSE, and READ_WAIT bits of Control register - START_CMD bit of Command register Does not affect any registers or DMA interface, or FIFO. or host interrupts.	0
		0	No change.	
		1	Reset. Reset SD/MMC controller.	
1	FIFO_RESET		Fifo reset. To reset FIFO, software should set bit to 1. This bit is auto-cleared after completion of reset operation. auto-cleared after two AHB clocks.	0
		0	No change.	
		1	Reset. Reset to data FIFO To reset FIFO pointers	

Table 549. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
2	DMA_RESET		DMA reset. To reset DMA interface, software should set bit to 1. This bit is auto-cleared after two AHB clocks.	0
		0	No change.	
		1	Reset. Reset internal DMA interface control logic.	
3	-		Reserved	-
4	INT_ENABLE		Global interrupt enable/disable bit. The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.	0
		0	Disable interrupts.	
		1	Enable interrupts.	
5	-		Reserved. Always write this bit as 0.	0
6	READ_WAIT		Read/wait. For sending read-wait to SDIO cards.	0
		0	Clear read wait.	
		1	Assert read wait.	
7	SEND_IRQ_RESPONSE		Send irq response. This bit automatically clears once response is sent. To wait for MMC card interrupts, the host issues CMD40, and the SD/MMC controller waits for an interrupt response from the MMC card. In the meantime, if the host wants the SD/MMC interface to exit waiting for interrupt state, it can set this bit, at which time the SD/MMC interface command state-machine sends a CMD40 response on the bus and returns to idle state.	0
		0	No change.	
		1	Send auto IRQ response.	
8	ABORT_READ_DATA		Abort read data. Used in SDIO card suspend sequence.	0
		0	No change.	
		1	Abort. After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. This bit automatically clears once data state machine resets to idle. Used in SDIO card suspend sequence.	
9	SEND_CCSD		Send ccsd. When set, the SD/MMC controller sends CCSD to the CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device. Once the CCSD pattern is sent to device, the SD/MMC interface automatically clears SEND_CCSD bit. It also sets Command Done (CD) bit in RINTSTS register and generates interrupt to host if Command Done interrupt is not masked. NOTE: Once SEND_CCSD bit is set, it takes two card clock cycles to drive the CCSD on the CMD line. Due to this, during the boundary conditions it may happen that CCSD is sent to the CE-ATA device, even if the device signalled CCS.	0
		0	Clear bit if the SD/MMC controller does not reset the bit.	
		1	Send Command Completion Signal Disable (CCSD) to CE-ATA device.	

Table 549. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
10	SEND_AUTO_STOP_CCSD		Send auto stop ccsd. NOTE: Always set SEND_AUTO_STOP_CCSD and SEND_CCSD bits together; SEND_AUTO_STOP_CCSD should not be set independent of SEND_CCSD. When set, the SD/MMC interface automatically sends internallygenerated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in RINTSTS is set and generates interrupt to host if Auto Command Done interrupt is not masked. After sending the CCSD, the SD/MMC interface automatically clears SEND_AUTO_STOP_CCSD bit.	0
		0	Clear this bit if the SD/MMC controller does not reset the bit.	
		1	Send internally generated STOP after sending CCSD to CE-ATA device.	
11	CEATA_DEVICE_INTERRUPT_STATUS		CEATA device interrupt status. Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled (nIEN = 1). If the host enables CE-ATA device interrupt, then software should set this bit.	0
		0	Disabled. Interrupts not enabled in CE-ATA device (nIEN = 1 in ATA control register)	
		1	Enabled. Interrupts are enabled in CE-ATA device (nIEN = 0 in ATA control register)	
15:12	-	-	Reserved	-
16	CARD_VOLTAGE_A0		Controls the state of the SD_VOLT0 pin. SD/MMC card voltage control is not implemented.	0
17	CARD_VOLTAGE_A1		Controls the state of the SD_VOLT1 pin. SD/MMC card voltage control is not implemented.	0
18	CARD_VOLTAGE_A2		Controls the state of the SD_VOLT2 pin. SD/MMC card voltage control is not implemented.	0
24:19	-	-	Reserved	-
25	USE_INTERNAL_DMAC		SD/MMC DMA use.	0
		0	Host. The host performs data transfers through the slave interface	
		1	DMA. Internal DMA used for data transfer	
31:26	-	-	Reserved	-

30.6.2 Power Enable register

Table 550. Power Enable register (PWREN, offset 0x004) bit description

Bit	Symbol	Description	Reset value
0	POWER_ENABLE	Power on/off switch for card; once power is turned on, software should wait for regulator/switch ramp-up time before trying to initialize card. 0 - power off 1 - power on Optional feature: port can be used as general-purpose output on the SD_POW pin.	0
31:1	-	Reserved	-

30.6.3 Clock Divider register

Table 551. Clock Divider register (CLKDIV, offset 0x008) bit description

Bit	Symbol	Description	Reset value
7:0	CLK_DIVIDER0	Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2 * 0 = 0 (no division, bypass), value of 1 means divide by 2 * 1 = 2, value of “FF” means divide by 2 * 255 = 510, and so on.	0
31:8	-	Reserved	-

30.6.4 Clock Enable register

Table 552. Clock Enable register (CLKENA, offset 0x010) bit description

Bit	Symbol	Description	Reset value
0	CCLK_ENABLE	Clock-enable control for SD card clock. One MMC card clock supported. 0 - Clock disabled 1 - Clock enabled	0
15:1	-	Reserved	-
16	CCLK_LOW_POWER	Low-power control for SD card clock. One MMC card clock supported. 0 - Non-low-power mode 1 - Low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).	0
31:17	-	Reserved	-

30.6.5 Time-out register

Table 553. Time-out register (TMOUT, offset 0x014) bit description

Bit	Symbol	Description	Reset value
7:0	RESPONSE_TIMEOUT	Response time-out value. Value is in number of card output clocks - cclk_out.	0x40
31:8	DATA_TIMEOUT	Value for card Data Read time-out; same value also used for Data Starvation by Host time-out. Value is in number of card output clocks - cclk_out of selected card. Starvation by Host time-out. Value is in number of card output clocks - cclk_out of selected card.	0xFFFFFF

30.6.6 Card Type register

Table 554. Card Type register (CTYPE, offset 0x018) bit description

Bit	Symbol	Description	Reset value
0	CARD_WIDTH0	Indicates if card is 1-bit or 4-bit: 0 - 1-bit mode 1 - 4-bit mode 1 and 4-bit modes only work when 8-bit mode in CARD_WIDTH1 is not enabled (bit 16 in this register is set to 0).	0
15:1	-	Reserved	-
16	CARD_WIDTH1	Indicates if card is 8-bit: 0 - Non 8-bit mode 1 - 8-bit mode.	0
31:17	-	Reserved	-

30.6.7 Block Size register

Table 555. Block Size register (BLKSIZ, offset 0x01C) bit description

Bit	Symbol	Description	Reset value
15:0	BLOCK_SIZE	Block size	0x200
31:16	-	Reserved	-

30.6.8 Byte Count register

Table 556. Byte Count register (BYTCNT, offset 0x020) bit description

Bit	Symbol	Description	Reset value
31:0	BYTE_COUNT	Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.	0x200

30.6.9 Interrupt Mask register

Table 557. Interrupt Mask register (INTMASK, offset 0x024) bit description

Bit	Symbol	Description	Reset value
0	CDET	Card detect. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
1	RE	Response error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
2	CDONE	Command done. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
3	DTO	Data transfer over. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
4	TXDR	Transmit FIFO data request. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
5	RXDR	Receive FIFO data request. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
6	RCRC	Response CRC error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
7	DCRC	Data CRC error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
8	RTO	Response time-out. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
9	DRTO	Data read time-out. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
10	HTO	Data starvation-by-host time-out (HTO). Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
11	FRUN	FIFO underrun/overrun error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
12	HLE	Hardware locked write error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
13	SBE	Start-bit error. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0

Table 557. Interrupt Mask register (INTMASK, offset 0x024) bit description

Bit	Symbol	Description	Reset value
14	ACD	Auto command done. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
15	EBE	End-bit error (read)/Write no CRC. Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.	0
16	SDIO_INT_MASK	Mask SDIO interrupt. When masked, SDIO interrupt detection for card is disabled. A 0 masks an interrupt, and 1 enables an interrupt. In MMC-Ver3.3-only mode, this bit is always 0.	0
31:17	-	Reserved	-

30.6.10 Command Argument register

Table 558. Command Argument register (CMDARG, offset 0x028) bit description

Bit	Symbol	Description	Reset value
31:0	CMD_ARG	Value indicates command argument to be passed to card.	0

30.6.11 Command register

Table 559. Command register (CMD, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
5:0	CMD_INDEX	-	Command index	0
6	RESPONSE_EXPECT		Response expect	0
		0	None. No response expected from card	
		1	Expected. Response expected from card	
7	RESPONSE_ LENGTH		Response length	0
		0	Short. Short response expected from card	
		1	Long. Long response expected from card	
8	CHECK_RESPONSE_CRC		Check response CRC. Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.	0
		0	Do not check response CRC	
		1	Check response CRC	
9	DATA_EXPECTED		Data expected	0
		0	None. No data transfer expected (read/write)	
		1	Data. Data transfer expected (read/write)	
10	READ_WRITE		read/write. Don't care if no data expected from card.	0
		0	Read from card	
		1	Write to card	
11	TRANSFER_MODE		Transfer mode. Don't care if no data expected.	0
		0	Block data transfer command	
		1	Stream data transfer command	

Table 559. Command register (CMD, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
12	SEND_AUTO_STOP		Send auto stop. When set, the SD/MMC interface sends stop command to SD_MMC_CEATA cards at end of data transfer. Refer to Table 583 to determine: - when SEND_AUTO_STOP bit should be set, since some data transfers do not need explicit stop commands - open-ended transfers that software should explicitly send to stop command Additionally, when “resume” is sent to resume - suspended memory access of SD-Combo card - bit should be set correctly if suspended data transfer needs SEND_AUTO_STOP. Don't care if no data expected from card.	0
		0	No stop command sent at end of data transfer	
		1	Send stop command at end of data transfer	
13	WAIT_PRVDATA_COMPLETE		Wait prvdata complete. The WAIT_PRVDATA_COMPLETE = 0 option typically used to query status of card during data transfer or to stop current data transfer.	0
		0	Send. Send command at once, even if previous data transfer has not completed.	
		1	Wait. Wait for previous data transfer completion before sending command.	
14	STOP_ABORT_CMD		Stop abort command. When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = DISABLE_BOOT.	0
		0	Disabled. Neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0.	
		1	Enabled. Stop or abort command intended to stop current data transfer in progress.	
15	SEND_INITIALIZATION		Send initialization. After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).	0
		0	No. Do not send initialization sequence (80 clocks of 1) before sending this command.	
		1	Send. Send initialization sequence before sending this command.	
20:16	-		Reserved. Always write as 0.	0

Table 559. Command register (CMD, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
21	UPDATE_CLOCK_REGISTERS_ONLY		Update clock registers only. Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when UPDATE_CLOCK_REGISTERS_ONLY = 0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card(s). When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC_CEATA cards.	0
		0	Normal. Normal command sequence	
		1	No. Do not send commands, just update clock register value into card clock domain	
22	READ_CEATA_DEVICE		Read ceata device. Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data time-out indication while performing CE-ATA read transfers. Maximum value of I/O transmission delay can be no less than 10 seconds.The SD/MMC interface should not indicate read data time-out while waiting for data from CE-ATA device.	0
		0	No read. Host is not performing read access (RW_REG or RW_BLK) towards CE-ATA device.	
		1	Read. Host is performing read access (RW_REG or RW_BLK) towards CE-ATA device.	
23	CCS_EXPECTED		CCS expected. If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. The SD/MMC controller sets the Data Transfer Over (DTO) bit in the RINTSTS register and generates an interrupt to the host if the Data Transfer Over interrupt is not masked.	0
		0	Disabled. Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register), or command does not expect CCS from device.	
		1	Enabled. Interrupts are enabled in CE-ATA device (nIEN = 0), and RW_BLK command expects command completion signal from CE-ATA device.	
24	ENABLE_BOOT	-	Enable Boot - this bit should be set only for mandatory boot mode. When Software sets this bit along with START_CMD, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set DISABLE_BOOT and ENABLE_BOOT together.	0
25	EXPECT_BOOT_ACK	-	Expect Boot Acknowledge. When Software sets this bit along with ENABLE_BOOT, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.	0
26	DISABLE_BOOT	-	Disable Boot. When software sets this bit along with START_CMD, CIU terminates the boot operation. Do NOT set DISABLE_BOOT and ENABLE_BOOT together.	0

Table 559. Command register (CMD, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
27	BOOT_MODE		Boot Mode	0
		0	Mandatory Boot operation	
		1	Alternate Boot operation	
28	VOLT_SWITCH		Voltage switch bit	0
		0	Disabled. No voltage switching	
		1	Enabled. Voltage switching enabled; must be set for CMD11 only	
29	USE_HOLD_REG		Use Hold Register.	0
		0	CMD and DATA sent to card bypassing HOLD register.	
		1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 7.5.76.	
30	-	-	Reserved	-
31	START_CMD	-	Start command. Once command is taken by CIU, this bit is cleared. When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register. Once command is sent and response is received from SD_MMC_CEATA cards, Command Done bit is set in the raw interrupt register.	0

30.6.12 Response register 0

Table 560. Response register 0 (RESP0, offset 0x030) bit description

Bit	Symbol	Description	Reset value
31:0	RESPONSE0	Bit[31:0] of response	0

30.6.13 Response register 1

Table 561. Response register 1 (RESP1, offset 0x034) bit description

Bit	Symbol	Description	Reset value
31:0	RESPONSE1	Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always “short” for them. For information on when CIU sends auto-stop commands, refer to Section 30.7.2 “Auto-Stop”.	0

30.6.14 Response register 2

Table 562. Response register 2 (RESP2, offset 0x038) bit description

Bit	Symbol	Description	Reset value
31:0	RESPONSE2	Bit[95:64] of long response	0

30.6.15 Response register 3

Table 563. Response register 3 (RESP3, offset 0x03C) bit description

Bit	Symbol	Description	Reset value
31:0	RESPONSE3	Bit[127:96] of long response	0

30.6.16 Masked Interrupt Status register

Table 564. Masked Interrupt Status register (MINTSTS, offset 0x040) bit description

Bit	Symbol	Description	Reset value
0	CDET	Card detect. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
1	RE	Response error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
2	CDONE	Command done. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
3	DTO	Data transfer over. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
4	TXDR	Transmit FIFO data request. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
5	RXDR	Receive FIFO data request. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
6	RCRC	Response CRC error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
7	DCRC	Data CRC error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
8	RTO	Response time-out. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
9	DRTO	Data read time-out. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
10	HTO	Data starvation-by-host time-out (HTO). Interrupt enabled if corresponding bit in interrupt mask register is set.	0
11	FRUN	FIFO underrun/overrun error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
12	HLE	Hardware locked write error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
13	SBE	Start-bit error. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
14	ACD	Auto command done. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
15	EBE	End-bit error (read)/write no CRC. Interrupt enabled if corresponding bit in interrupt mask register is set.	0
16	SDIO_INTERRUPT	Interrupt from SDIO card. SDIO interrupt for card enabled if corresponding SDIO_INT_MASK bit is set in Interrupt Mask register (INTMASK). Mask bit 1 enables interrupt; 0 masks interrupt. 0 - No SDIO interrupt from card 1 - SDIO interrupt from card In MMC-Ver3.3-only mode, this bit is always 0.	-
31:17	-	Reserved	-

30.6.17 Raw Interrupt Status register

Table 565. Raw Interrupt Status register (RINTSTS, offset 0x044) bit description

Bit	Symbol	Description	Reset value
0	CDET	Card detect. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
1	RE	Response error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
2	CDONE	Command done. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
3	DTO	Data transfer over. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
4	TXDR	Transmit FIFO data request. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
5	RXDR	Receive FIFO data request. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
6	RCRC	Response CRC error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
7	DCRC	Data CRC error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
8	RTO_BAR	Response time-out (RTO)/Boot Ack Received (BAR). Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
9	DRTO_BDS	Data read time-out (DRTO)/Boot Data Start (BDS). Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
10	HTO	Data starvation-by-host time-out (HTO). Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
11	FRUN	FIFO underrun/overrun error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
12	HLE	Hardware locked write error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
13	SBE	Start-bit error. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
14	ACD	Auto command done. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
15	EBE	End-bit error (read)/write no CRC. Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status.	0
16	SDIO_INTERRUPT	Interrupt from SDIO card. Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 0 - No SDIO interrupt from card 1 - SDIO interrupt from card In MMC-Ver3.3-only mode, bits always 0. Bits are logged regardless of interrupt-mask status.	0
31:17	-	Reserved.	-

30.6.18 Status register

Table 566. Status register (STATUS, offset 0x048) bit description

Bit	Symbol	Description	Reset value
0	FIFO_RX_WATERMARK	FIFO reached Receive watermark level; not qualified with data transfer.	0
1	FIFO_TX_WATERMARK	FIFO reached Transmit watermark level; not qualified with data transfer.	1
2	FIFO_EMPTY	FIFO is empty status.	1
3	FIFO_FULL	FIFO is full status.	0
7:4	CMDFSMSTATES	Command FSM states: 0 - Idle 1 - Send init sequence 2 - Tx cmd start bit 3 - Tx cmd tx bit 4 - Tx cmd index + arg 5 - Tx cmd crc7 6 - Tx cmd end bit 7 - Rx resp start bit 8 - Rx resp IRQ response 9 - Rx resp tx bit 10 - Rx resp cmd idx 11 - Rx resp data 12 - Rx resp crc7 13 - Rx resp end bit 14 - Cmd path wait NCC 15 - Wait; CMD-to-response turnaround NOTE: The command FSM state is represented using 19 bits. The STATUS register(7:4) has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the STATUS(7:4) register. The three states that are not represented in the STATUS register(7:4) are: - Bit 16 - Wait for CCS - Bit 17 - Send CCSD - Bit 18 - Boot Mode Due to this, while command FSM is in “Wait for CCS state” or “Send CCSD” or “Boot Mode”, the STATUS register indicates status as 0 for the bit field 7:4.	0
8	DATA_3_STATUS	Raw selected card_data[3]; checks whether card is present. 0 - card not present 1 - card present	0
9	DATA_BUSY	Inverted version of raw selected card_data[0]. 0 - card data not busy 1 - card data busy	0
10	DATA_STATE_MC_BUSY	Data transmit or receive state-machine is busy.	1
16:11	RESPONSE_INDEX	Index of previous response, including any auto-stop sent by core.	0
29:17	FIFO_COUNT	FIFO count - Number of filled locations in FIFO.	0
30	DMA_ACK	DMA acknowledge signal state.	0
31	DMA_REQ	DMA request signal state.	0

30.6.19 FIFO Threshold Watermark register

Table 567. FIFO Threshold Watermark register (FIFOTH, offset 0x04C) bit description

Bit	Symbol	Value	Description	Reset value
11:0	TX_WMARK	-	FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If Interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated, regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. 12 bits - 1 bit less than FIFO-count of STATUS register, which is 13 bits. Limitation: TX_WMARK >= 1; Recommended value: TX_WMARK = 16; (means less than or equal to FIFO_DEPTH/2).	0
15:12	-	-	Reserved	-
27:16	RX_WMARK	-	FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. 12 bits - 1 bit less than FIFO-count of STATUS register, which is 13 bits. Limitation: RX_WMARK less than FIFO_DEPTH-2 Recommended: RX_WMARK = 15; (means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS time-out.	0x1F

Table 567. FIFO Threshold Watermark register (FIFOTH, offset 0x04C) bit description

Bit	Symbol	Value	Description	Reset value
30:28	DMA_MTS	-	Burst size of multiple transaction; should be programmed same as DW-DMA controller multiple-transaction-size SRC/DEST_MSIZE.The units for transfers is the H_DATA_WIDTH parameter. A single transfer (dw_dma_single assertion in case of Non DW DMA interface) would be signalled based on this value. Value should be sub-multiple of (RX_WMARK + 1) and (32 - TX_WMARK). For example, if FIFO_DEPTH = 16, FDATA_WIDTH = H_DATA_WIDTH Allowed combinations for MSize and TX_WMARK are: MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMARK = 8 MSize = 4, TX_WMARK = 4 MSize = 4, TX_WMARK = 12 MSize = 8, TX_WMARK = 8 MSize = 8, TX_WMARK = 4 Allowed combinations for MSize and RX_WMARK are: MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMARK = 3 MSize = 4, RX_WMARK = 7 MSize = 4, RX_WMARK = 11 MSize = 8, RX_WMARK = 7 MSize = 8, RX_WMARK = 11 Recommended: MSize = 8, TX_WMARK = 8, RX_WMARK = 7	0
		0x0	1 transfer	
		0x1	4 transfers	
		0x2	8 transfers	
		0x3	16 transfers	
		0x4	32 transfers	
		0x5	64 transfers	
		0x6	128 transfers	
		0x7	256 transfers	
31	-	-	Reserved	-

30.6.20 Card Detect register

Table 568. Card Detect register (CDETECT, offset 0x050) bit description

Bit	Symbol	Description	Reset value
0	CARD_DETECT	Card detect. 0 represents presence of card.	0
31:1	-	Reserved	-

30.6.21 Write Protect register

Table 569. Write Protect register (WRTPRT, offset 0x054) bit description

Bit	Symbol	Description	Reset value
0	WRITE_PROTECT	Write protect. 1 represents write protection.	0
31:1	-	Reserved	-

30.6.22 Transferred CIU Card Byte Count register

Table 570. Transferred CIU Card Byte Count register (TCBCNT, offset 0x05C) bit description

Bit	Symbol	Description	Reset value
31:0	TRANS_CARD_BYTE_COUNT	Number of bytes transferred by CIU unit to card. Register should be read only after data transfer completes; during data transfer, register returns 0.	0

30.6.23 Transferred Host to BIU-FIFO Byte Count register

Table 571. Transferred Host to BIU-FIFO Byte Count register (TBBCNT, offset 0x060) bit description

Bit	Symbol	Description	Reset value
31:0	TRANS_FIFO_BYTE_COUNT	Number of bytes transferred between Host/DMA memory and BIU FIFO.	0

30.6.24 Debounce Count register

Table 572. Debounce Count register (DEBNCE, offset 0x064) bit description

Bit	Symbol	Description	Reset value
23:0	DEBOUNCE_COUNT	Number of host clocks (SD_CLK) used by debounce filter logic for card detect; typical debounce time is 5-25 ms.	0xFFFFFF
31:24	-	Reserved	-

30.6.25 Hardware Reset

Table 573. Hardware Reset (RST_N, offset 0x078) bit description

Bit	Symbol	Description	Reset value
0	CARD_RESET	Hardware reset. 1 - Active mode 0 - Reset Toggles state on SD_RST pin. This bit causes the card to enter pre-idle state, which requires it to be re-initialized.	1
31:1	-	Reserved	-

30.6.26 Bus Mode register

Table 574. Bus Mode register (BMOD, offset 0x080) bit description

Bit	Symbol	Value	Description	Reset value
0	SWR	-	Software Reset. When set, the DMA Controller resets all its internal registers. SWR is read/write. It is automatically cleared after 1 clock cycle.	0
1	FB	-	Fixed Burst. Controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations. FB is read/write.	0
6:2	DSL	-	Descriptor Skip Length. Specifies the number of HWord/Word/Dword to skip between two unchained descriptors. This is applicable only for dual buffer structure. DSL is read/write.	0
7	DE	-	SD/MMC DMA Enable. When set, the SD/MMC DMA is enabled. DE is read/write.	0

Table 574. Bus Mode register (BMOD, offset 0x080) bit description

Bit	Symbol	Value	Description	Reset value
10:8	PBL	-	Programmable Burst Length. These bits indicate the maximum number of beats to be performed in one SD/MMC DMA transaction. The SD/MMC DMA will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. The permissible values are 1, 4, 8, 16, 32, 64, 128 and 256. This value is the mirror of MSIZE of FIFOTH register. In order to change this value, write the required value to FIFOTH register. This is an encode value as follows.Transfer unit is 32 bit. PBL is a read-only value.	0
		0x0	1 transfer	
		0x1	4 transfers	
		0x2	8 transfers	
		0x3	16 transfers	
		0x4	32 transfers	
		0x5	64 transfers	
		0x6	128 transfers	
		0x7	256 transfers	
31:11	-	-	Reserved	-

30.6.27 Poll Demand register

Table 575. Poll Demand register (PLDMND, offset 0x084) bit description

Bit	Symbol	Description	Reset value
31:0	PD	Poll Demand. If the OWN bit of a descriptor is not set, the FSM goes to the Suspend state. The host needs to write any value into this register for the SD/MMC DMA state machine to resume normal descriptor fetch operation. This is a write only register. PD bit is write-only.	-

30.6.28 Descriptor List Base Address register

Table 576. Descriptor List Base Address register (DBADDR, offset 0x088) bit description

Bit	Symbol	Description	Reset value
31:0	SDL	Start of Descriptor List. Contains the base address of the First Descriptor. The LSB bits [1:0] are ignored and taken as all-zero by the SD/MMC DMA internally. Hence these LSB bits are read-only.	0

30.6.29 Internal DMAC Status register

Table 577. Internal DMAC Status register (IDSTS, offset 0x08C) bit description

Bit	Symbol	Description	Reset value
0	TI	Transmit interrupt. Indicates that data transmission is finished for a descriptor. Writing a 1 clears this bit.	0
1	RI	Receive interrupt. Indicates the completion of data reception for a descriptor. Writing a 1 clears this bit.	0
2	FBE	Fatal bus error interrupt. Indicates that a Bus Error occurred (IDSTS[12:10]). When this bit is set, the DMA disables all its bus accesses. Writing a 1 clears this bit.	0
3	-	Reserved	-
4	DU	Descriptor unavailable interrupt. This bit is set when the descriptor is unavailable due to OWN bit = 0 (DES0[31] =0). Writing a 1 clears this bit.	0

Table 577. Internal DMAC Status register (IDSTS, offset 0x08C) bit description

Bit	Symbol	Description	Reset value
5	CES	Card error summary. Indicates the status of the transaction to/from the card; also present in RINTSTS. Indicates the logical OR of the following bits: EBE - End Bit Error RTO - Response Time-out/Boot Ack Time-out RCRC - Response CRC SBE - Start Bit Error DRTO - Data Read Time-out/BDS time-out DCRC - Data CRC for Receive RE - Response Error Writing a 1 clears this bit.	0
7:6	-	Reserved	-
8	NIS	Normal interrupt summary. Logical OR of the following: IDSTS[0] - Transmit Interrupt IDSTS[1] - Receive Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes NIS to be set is cleared. Writing a 1 clears this bit.	0
9	AIS	Abnormal interrupt summary. Logical OR of the following: IDSTS[2] - Fatal Bus Interrupt IDSTS[4] - DU bit Interrupt IDSTS[5] - Card Error Summary Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. Writing a 1 clears this bit.	0
12:10	EB	Error bits. Indicates the type of error that caused a Bus Error. Valid only with Fatal Bus Error bit (IDSTS[2]) set. This field does not generate an interrupt. 001 - Host Abort received during transmission 010 - Host Abort received during reception Others: Reserved. EB is read-only.	0
16:13	FSM	DMAC state machine present state. 0 - DMA_IDLE 1 - DMA_SUSPEND 2 - DESC_RD 3 - DESC_CHK 4 - DMA_RD_REQ_WAIT 5 - DMA_WR_REQ_WAIT 6 - DMA_RD 7 - DMA_WR 8 - DESC_CLOSE This field is read-only.	0
31:17	-	Reserved	-

30.6.30 Internal DMAC Interrupt Enable register

Table 578. Internal DMAC Interrupt Enable register (IDINTEN, offset 0x090) bit description

Bit	Symbol	Description	Reset value
0	TI	Transmit interrupt enable. When set with Normal Interrupt Summary Enable, Transmit Interrupt is enabled. When reset, Transmit Interrupt is disabled.	0
1	RI	Receive interrupt enable. When set with Normal Interrupt Summary Enable, Receive Interrupt is enabled. When reset, Receive Interrupt is disabled.	0
2	FBE	Fatal bus error enable. When set with Abnormal Interrupt Summary Enable, the Fatal Bus Error Interrupt is enabled. When reset, Fatal Bus Error Enable Interrupt is disabled.	0
3	-	Reserved	-
4	DU	Descriptor unavailable interrupt. When set along with Abnormal Interrupt Summary Enable, the DU interrupt is enabled.	0

Table 578. Internal DMAC Interrupt Enable register (IDINTEN, offset 0x090) bit description

Bit	Symbol	Description	Reset value
5	CES	Card error summary interrupt enable. When set, it enables the Card Interrupt summary.	0
7:6	-	Reserved	-
8	NIS	Normal interrupt summary enable. When set, a normal interrupt is enabled. When reset, a normal interrupt is disabled. This bit enables the following bits: IDINTEN[0] - Transmit Interrupt IDINTEN[1] - Receive Interrupt	0
9	AIS	Abnormal interrupt summary enable. When set, an abnormal interrupt is enabled. This bit enables the following bits: IDINTEN[2] - Fatal Bus Error Interrupt IDINTEN[4] - DU Interrupt IDINTEN[5] - Card Error Summary Interrupt	0
31:10	-	Reserved	-

30.6.31 Current Host Descriptor Address register

Table 579. Current Host Descriptor Address register (DSCADDR, offset 0x094) bit description

Bit	Symbol	Description	Reset value
31:0	HDA	Host descriptor address pointer. Cleared on reset. Pointer updated by IDMAC during operation. This register points to the start address of the current descriptor read by the SD/MMC DMA.	0

30.6.32 Current Buffer Descriptor Address register

Table 580. Current Buffer Descriptor Address register (BUFADDR, offset 0x098) bit description

Bit	Symbol	Description	Reset value
31:0	HBA	Host buffer address pointer. Cleared on Reset. Pointer updated by SD/MMC DMA during operation. This register points to the current Data Buffer Address being accessed by the SD/MMC DMA.	0

30.6.33 Card Threshold Control register

Note: This register is applicable when CARDRDTHREN is set to '1'. See Section 30.8.2.16.

Table 581. Card Threshold Control register (CARDTHRCTL, offset 0x100) bit description

Bit	Symbol	Value	Description	Reset value
0	CARDRDTHREN		Card read threshold enable.	0
		0	Card read threshold disabled.	
		1	Card Read Threshold enabled. Host Controller initiates Read Transfer only if Card Threshold amount of space is available in receive FIFO. For more information, refer to Section 30.8.2.16 “Card read threshold”.	
1	BSYCLRINTEN		Busy clear interrupt enable. Note: The application can disable this feature if it does not want to wait for a Busy Clear Interrupt. For example, in a multi-card scenario, the application can switch to the other card without waiting for a busy to be completed. In such cases, the application can use the polling method to determine the status of busy. By default this feature is disabled and backward-compatible to the legacy drivers where polling is used.	0
		0	Busy clear interrupt disabled.	
		1	Busy clear interrupt enabled.	

Table 581. Card Threshold Control register (CARDTHRCTL, offset 0x100) bit description

Bit	Symbol	Value	Description	Reset value
15:2	-	-	Reserved	-
23:16	CARDTHRESHOLD	-	Card threshold size. Sets the read and/or write threshold within the 32-entry FIFOs. This field is applicable when CARDRDTHREN is set to '1'.	0
31:24	-	-	Reserved	-

30.6.34 Back-end power register

Controls back-end power to the card application. Also see Section 30.8.2.17.

Table 582. Back-end power register (BACK_END_POWER, offset 0x104) bit description

Bit	Symbol	Value	Description	Reset value
0	BACKENDPWR		Back-end Power control for card application.	0
		0	Back-end power off to card application.	
		1	Back-end power supplied to card application.	
31:1	-	-	Reserved	-

30.7 Functional description

30.7.1 Power/pull-up control and card detection unit

Signal pull-up resistors can be enabled for the SD pins in IOCON by enabling the pull-up for the pads. The approximate pull-up value for a pin is about 50 kOhm. For designs that need to support legacy MMC cards in open-drain mode, an external pull-up controlled with a general purpose output and FET will be needed for the CMD line.

Slot power can be controlled with the SD_POW pin and the SD_VOLT[2:0] pins. It is recommended that the slot power regulator is enabled and disabled via the SD_POW pin, which can be directly controlled with bit 0 of the Power enable register (PWREN).

Use of the SD_VOLT[2:0] pins is optional and not needed in a design with a single power supply sourcing the card slot.

The card detection signal is debounced based on the number of blocks specified in the Debounce Count register (DEBNCE). When this signal is connected to the card detect pin of the card slot, then CDETECT register's bit 0 state will be filtered by the number of debounce cycles specified in DEBNCE. This guarantees that interrupt related to the card detect signal are debounced before occurring.

30.7.2 Auto-Stop

The auto-stop command helps to send an exact number of data bytes using a stream read or write for the MMC, and a multiple-block read or write for SD memory transfer for SD cards. The module internally generates a stop command and is loaded in the command path when the SEND_AUTO_STOP bit is set in the Command register (CMD).

The software should set the SEND_AUTO_STOP bit according to details listed in the table below:

Table 583. SEND_AUTO_STOP bit

Card Type	Transfer Type	Byte Count	SEND_AUTO_STOP bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	>0	No	Byte count = 0 is illegal
MMC	Single-block write	>0	No	Byte count = 0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	>0	Yes [1]	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	>0	Yes [1]	Pre-defined multiple block
SDMEM	Single-block read	>0	No	Byte count = 0 is illegal
SDMEM	Single-block write	>0	No	Byte count = 0 is illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	>0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block

Table 583. SEND_AUTO_STOP bit

Card Type	Transfer Type	Byte Count	SEND_AUTO_STOP bit set	Comments
SDMEM	Multiple-block write	>0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	>0	No	Byte count = 0 is illegal
SDIO	Single-block write	>0	No	Byte count = 0 is illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block
SDIO	Multiple-block read	>0	No	Pre-defined multiple block
SDIO	Multiple-block write	0	No	Open-ended multiple block
SDIO	Multiple-block write	>0	No	Pre-defined multiple block

[1] The condition under which the transfer mode is set to block transfer and byte_count is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If byte_count = n × block_size (n = 2, 3, …), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the CPU software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue CMD18/CMD25 commands without setting the SEND_AUTO_STOP bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the SEND_AUTO_STOP bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop command after the programmed byte count.

The following list conditions for the auto-stop command.

- Stream read for MMC card with byte count greater than 0 - The Module generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 - The Module generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.
- Multiple-block read memory for SD card with byte count greater than 0 - If the block size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.
- Multiple-block write memory for SD card with byte count greater than 0 - If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.
- Precaution for CPU software during auto-stop - Whenever an auto-stop command is issued, the CPU software should not issue a new command to the Module until the auto-stop is sent by the Module and the data transfer is complete. If the CPU issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the

CPU wants to terminate the data transfer before the data transfer is complete, it can issue a stop or abort command, in which case the Module does not generate an auto-stop command.

30.8 Programming the SD/MMC

30.8.1 Software/hardware restrictions

Only one data transfer command should be issued at one time. For CE-ATA devices, if CE-ATA device interrupts are enabled (nIEN = 0), only one RW_MULTIPLE_BLOCK command (RW_BLK) should be issued; no other commands (including a new RW_BLK) should be issued before the Data Transfer. Over status is set for the outstanding RW_BLK.

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

To avoid glitches in the card clock outputs (cclk_out), the software should use the following steps when changing the card clock frequency:

1. Update the Clock Enable register (CLKENA) to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:
   - START_CMD bit
   - "update clock registers only" bits
   - "wait_previous data complete" bit

   Wait for the CIU to take the command by polling for 0 on the START_CMD bit.
2. Set the START_CMD bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.
3. Set START_CMD to update the Clock Enable register (CLKENA) in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the Data Transfer Over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX_WMARK interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. In DMA mode while reading from a card, the DTO interrupt occurs only after all the FIFO data is flushed to memory by the DMA Interface unit.

While writing to a card in DMA mode, if an undefined-length transfer is selected by setting the Byte Count register (BYTCNT) to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a CONTROLLER_RESET command by setting control register (CTRL) bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the CPU. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly
- DMA data read is ignored
- Write data is unknown (x)

Additionally, if DMA_RESET is also issued, any pending DMA transfer is abruptly terminated. The DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.

One data-transfer requirement between the FIFO and CPU is that the number of transfers should be a multiple of the FIFO data width (F_DATA_WIDTH), which is 32. So if you want to write only 15 bytes to an SD/MMC/CE-ATA card (BYTCNT), the CPU should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers, if DMA mode is enabled. The software can still program the Byte Count register (BYTCNT) to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the CPU should still read all 16 bytes from the FIFO.

It is recommended that you do not change the FIFO threshold register in the middle of data transfers.

30.8.2 Programming sequence

30.8.2.1 Initialization

Once the power and clocks are stable, reset_n should be asserted (active-low) for at least two clocks of clk or cclk_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

1. After power on reset, configure the SD/MMC pin functions via IOCON, see Chapter 10 “LPC546xx I/O pin configuration (IOCON)”.
2. Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register (INTMASK). Set the global INT_ENABLE bit of the control register (CTRL). It is recommended that you write 0xFFFF_FFFF to the Raw Interrupt register in order to clear any pending interrupts before setting the INT_ENABLE bit.
3. Enumerate card stack. Each card is enumerated according to card type; for details, refer to Section 30.8.2.2 “Enumerated card stack”. For enumeration, you should restrict the clock frequency to 400 kHz in accordance with SD_MMC/CE-ATA standards.
4. Changing clock. The cards operate at a maximum of 25 MHz (at maximum of 50 MHz in high-speed mode and speed detection using CMD6).
5. Set other IP parameters, which normally do not need to be changed with every command, with a typical value such as time-out values in cclk_out according to SD_MMC/CE-ATA specifications.

ResponseTimeOut = 0x40

DataTimeOut = highest of one of the following:

- (10 × ((TAAC × Fop) + (100 × NSAC))
- CPU FIFO read/write latency from FIFO empty/full

FIFO threshold value in bytes in the FIFOTH register. Typically, the threshold value can be set to half the FIFO depth (= 32/2); that is:

- RX_WMARK = (FIFO_DEPTH/2) - 1;
- TX_WMARK = FIFO_DEPTH/2

6. If the software decides to handle the interrupts provided by the IP core, you should create another thread to handle interrupts.

30.8.2.2 Enumerated card stack

The card stack does the following:

- Enumerates all connected cards
- Sets the RCA for the connected cards
- Reads card-specific information
- Stores card-specific information locally

Enumerate_Card_Stack - Enumerates the card connected on the module. The card can be of the type MMC, CE-ATA, SD, or SDIO. All types of SDIO cards are supported; that is, SDIO_IO_ONLY, SDIO_MEM_ONLY, and SDIO_COMBO cards. The enumeration sequence includes the following steps:

1. Check if the card is connected.
2. Clear the bits in the card_type register. Clear the register bit for a 1-bit, 4-bit, or 8-bit bus width.
3. Identify the card type; that is, SD, MMC, or SDIO.
   - Send CMD5 first. If a response is received, then the card is SDIO
   - If not, send ACMD41; if a response is received, then the card is SD.
   - Otherwise, the card is an MMC or CE-ATA
4. Enumerate the card according to the card type.

   Use a clock source with a frequency = Fod (that is, 400 kHz) and use the following enumeration command sequence:
   - SD card - Send CMD0, ACMD41, CMD2, CMD3.
   - SDHC card - send CMD0, SDCMD8, ACMD41, CMD2, CMD3
   - SDIO - Send CMD5; if the function count is valid, CMD3. For the SDIO memory section, follow the same commands as for the SD card.
   - MMC - Send CMD0, CMD1, CMD2, CMD3
5. Identify the MMC/CE-ATA device.
   - Selecting ATA mode for a CE-ATA device.
   - CPU should query the byte 504 (S_CMD_SET) of EXT_CSD register by sending CMD8. If bit 4 is set to 1, then the device supports ATA mode.

- If ATA mode is supported, the CPU should select the ATA mode by setting the ATA bit (bit 4) of the EXT_CSD register slice 191(CMD_SET) to activate the ATA command set for use. The CPU selects the command set using the SWITCH (CMD6) command.
- The current mode selected is shown in byte 191 of the EXT_CSD register.

  If the device does not support ATA mode, then the device can be an MMC device or a CE-ATA v1.0 device.
- Send RW_REG; if a response is received and the response data contains CE-ATA signature, the device is a CE-ATA device.
- Otherwise the device is an MMC card.

6. You can change the card clock frequency after enumeration.

30.8.2.3 Clock programming

The clock programming has to be done in the CGU. The cclk_in has to be equal to the cclk_out. Therefore the registers that support this have to be:

- CLKDIV = 0x0 (bypass of clock divider).
- CLKSRC = 0x0
- CLKENA = 0x0 or 0x1. This register enables or disables clock for the card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The Module loads each of these registers only when the START_CMD bit and the Update_clk_regs_only bit in the CMD register are set. When a command is successfully loaded, the Module clears this bit, unless the Module already has another command in the queue, at which point it gives an HLE (Hardware Locked Error); for details on HLEs, refer to Section 30.8.2.19 “Error handling”.

Software should look for the START_CMD and the Update_clk_regs_only bits, and should also set the WAIT_PRVDATA_COMPLETE bit to ensure that clock parameters do not change during data transfer. Note that even though START_CMD is set for updating clock registers, the Module does not raise a command_done signal upon command completion.

30.8.2.4 No-Data command with or without response sequence

To send any non-data command, the software needs to program the CMD register and the CMDARG register with appropriate parameters. Using these two registers, the Module forms the command and sends it to the command bus. The Module reflects the errors in the command response through the error bits of the RINTSTS register.

When a response is received - either erroneous or valid - the Module sets the command_done bit in the RINTSTS register. A short response is copied in Response register 0 (RESP0), while a long response is copied to all four response registers. The RESPONSE3 register bit 31 represents the MSB, and the RESPONSE0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

1. Program the Command register with the appropriate command argument parameter.
2. Program the Command register with the settings in Table 584.

3. Wait for command acceptance by CPU. The following happens when the command is loaded into the Module:
   - Module accepts the command for execution and clears the START_CMD bit in the CMD register, unless one command is in process, at which point the Module can load and keep the second command in the buffer.
   - If the Module is unable to load the command - that is, a command is already in progress, a second command is in the buffer, and a third command is attempted - then it generates an HLE (hardware-locked error).
   - Check if there is an HLE.
   - Wait for command execution to complete. After receiving either a response from a card or response time-out, the Module sets the command_done bit in the RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.
   - Check if response_timeout error, response_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the RINTSTS register. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers.

Software should not modify clock parameters while a command is being executed.

Table 584. CMD register settings for No-Data command

Name	Value	Comment
START_CMD	1	
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	0	No data command.
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	Can be 1 for commands to stop data transfer, such as CMD12
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	Can be 1 for R2 (long) response
RESPONSE_EXPECT	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in the Syscon block. See Section 7.5.76.
WAIT_PRVDATA_COMPLETE	1	Before sending command on command line, CPU should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

30.8.2.5 Data transfer commands

Data transfer commands transfer data between the memory card and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively.

For the data transfer commands, it is important that the same bus width that is programmed in the card should be set in the Card Type register (CTYPE).

The Module generates an interrupt for different conditions during data transfer, which are reflected in the RINTSTS register as:

1. Data_Transfer_Over (bit 3) - When data transfer is over or terminated. If there is a response time-out error, then the Module does not attempt any data transfer and the "Data Transfer Over" bit is never set.
2. Transmit_FIFO_Data_request (bit 4) - FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO.
3. Receive_FIFO_Data_request (bit 5) - FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
4. Data starvation by CPU time-out (bit 10) - FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the Module cannot continue with data transfer. The clock to the card has been stopped.
5. Data read time-out error (bit 9) - Card has not sent data within the time-out period.
6. Data CRC error (bit 7) - CRC error occurred during data reception.
7. Start bit error (bit 13) - Start bit was not received during data reception.
8. End bit error (bit 15) - End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

   Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response time-out, then no data transfer occurred.

30.8.2.6 Single-block or multiple-block read

Steps involved in a single-block or multiple-block read are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register. The Module expects data from the card in blocks of size BLKSIZ each.
3. Program the CMDARG register with the data address of the beginning of a data read. Program the Command register with the parameters listed in Table 585. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

   After writing to the CMD register, the Module starts executing the command; when the command is sent to the bus, the command_done interrupt is generated.

4. Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
5. Software should look for Receive_FIFO_Data_request and/or data starvation by CPU time-out conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
6. When a Data_Transfer_Over interrupt is received, the software should read the remaining data from the FIFO.

Table 585. CMD register settings for single-block or multiple-block read

Name	Value	Comment
START_CMD	1	
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	1	
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	Can be 1 for commands to stop data transfer, such as CMD12
SEND_AUTO_STOP	0/1	-
TRANSFER_MODE	0	Block transfer
READ_WRITE	0	Read from card
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	Can be 1 for R2 (long) response
RESPONSE_EXPECT	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 7.5.76.
WAIT_PRVDATA_COMPLETE	1	Before sending command on command line, CPU should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

30.8.2.7 Single-block or multiple-block write

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register; the Module sends data in blocks of size BLKSIZ each.
3. Program CMDARG register with the data address to which data should be written.
4. Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.

5. Program the Command register with the parameters listed in Table 586. For SD and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

   After writing to the CMD register, Module starts executing a command; when the command is sent to the bus, a command_done interrupt is generated.
6. Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending the STOP command.
7. Software should look for Transmit_FIFO_Data_request and/or time-out conditions from data starvation by the CPU. In both cases, the software should write data into the FIFO.
8. When a Data_Transfer_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the Module should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto_command_done interrupt - bit 14 of the RINTSTS register. A response to AUTO_STOP is stored in RESP1.

Table 586. CMD register settings for single-block or multiple-block write

Name	Value	Comments
START_CMD	1	
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	1	
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	Can be 1 for commands to stop data transfer, such as CMD12
SEND_AUTO_STOP	0/1	-
TRANSFER_MODE	0	Block transfer
READ_WRITE	1	Write to card
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	Can be 1 for R2 (long) response
RESPONSE_EXPECT	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 7.5.76.
WAIT_PRVDATA_COMPLETE	1	Before sending command on command line, CPU should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.

30.8.2.8 Stream read

A stream read is like the block read mentioned in "Single-Block or Multiple-Block Read", except for the following bits in the Command register (CMD):

TRANSFER_MODE = 1; //Stream transfer

CMD_INDEX = CMD20;

A stream transfer is allowed for only a single-bit bus width.

30.8.2.9 Stream write

A stream write is exactly like the block write mentioned in "Single-Block or Multiple-Block Write", except for the following bits in the Command register (CMD):

- TRANSFER_MODE = 1;//Stream transfer
- CMD_INDEX = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a transfer, the Module sends the STOP command. Completion of this AUTO_STOP command is reflected by the Auto_command_done interrupt. A response to an AUTO_STOP is stored in the RESP1 register.

A stream transfer is allowed for only a single-bit bus width.

30.8.2.10 Packed commands

In order to reduce overhead, read and write commands can be packed in groups of commands—either all read or all write—that transfer the data for all commands in the group in one transfer on the bus.

Packed commands can be of two types:

- Packed Write: CMD23 → CMD25
- Packed Read: CMD23 → CMD25 → CMD23 → CMD18

Packed commands are put in packets by the application software and are transparent to the core. For more information on packed commands, refer to the eMMC specification.

30.8.2.11 Sending Stop or Abort in middle of transfer

The STOP command can terminate a data transfer between a memory card and the Module, while the ABORT command can terminate an I/O data transfer for only the SDIO_IOONLY and SDIO_COMBO cards.

- Send STOP command - Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer. For information on sending this command, refer to Section 30.8.2.4 “No-Data command with or without response sequence”.

  You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (STOP_ABORT_CMD) to 1. If stop STOP_ABORT_CMD is not set to 1, the user stopped a data transfer. Reset bit 13 of the Command register (WAIT_PRVDATA_COMPLETE) to 0 in order to make the Module send the command at once, even though there is a data transfer in progress.

- Send ABORT command - Can be used with only an SDIO_IOONLY or SDIO_COMBO card. To abort the function that is transferring data, program the function number in ASx bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

  This is a non-data command. For information on sending this command, refer to Section 30.8.2.4 “No-Data command with or without response sequence”.

  Program the CMDARG register with the appropriate command argument parameters listed in Table 587.
- Program the Command register using the command index as CMD52. Similar to the STOPcommand, set bit 14 of the Command register (STOP_ABORT_CMD) to 1, which must be done in order to inform the Module that the user aborted the data transfer. Reset bit 13 (WAIT_PRVDATA_COMPLETE) of the Command register to 0 in order to make the Module send the command at once, even though a data transfer is in progress.
- Wait for command_transfer_over.
- Check response (R5) for errors.

- During an open-ended card write operation, if the card clock is stopped because the FIFO is empty, the software must first fill the data into the FIFO and start the card clock before issuing a stop/abort command to the card.

Table 587. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1
30-28	Function number	0, for CCCR access
27	RAW flag	1, if needed to read after write
26	Don’t care	-
25-9	Register address	0x06
8	Don’t care	-
7-0	Write data	Function number to be aborted

30.8.2.12 Suspend or Resume sequence

In an SDIO card, the data transfer between an I/O function and the Module can be temporarily halted using the SUSPEND command; this may be required in order to perform a high-priority data transfer with another function. When desired, the data transfer can be resumed using the RESUME command.

The following functions can be implemented by programming the appropriate bits in the CCCR register (Function 0) of the SDIO card. To read from or write to the CCCR register, use the CMD52 command.

1. SUSPEND data transfer - Non-data command.
   - Check if the SDIO card supports the SUSPEND/RESUME protocol; this can be done through the SBS bit in the CCCR register of the card.

     Check if the data transfer for the required function number is in process; the function number that is currently active is reflected in bits 0-3 of the CCCR register. Note that if the BS bit is 1, then only the function number given by the FSx bits is valid.

To suspend the transfer, set BR (bit 2) of the CCCR register.

Poll for clear status of bits BR (bit 1) and BS (bit 0) of the CCCR. The BS (Bus Status) bit is 1 when the currently-selected function is using the data bus; the BR (Bus Release) bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function has been suspended.

During a read-data transfer, the Module can be waiting for the data from the card. If the data transfer is a read from a card, then the Module must be informed after the successful completion of the SUSPEND command. The Module then resets the data state machine and comes out of the wait state. To accomplish this, set ABORT_READ_DATA (bit 8) in the control register (CTRL).

Wait for data completion. Get pending bytes to transfer by reading the TCBCNT register.

2. RESUME data transfer - This is a data command.
   - Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.

     If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.

     Check that a function to be resumed is ready for data transfer; this can be confirmed by reading the RFx flag in CCCR. If RF = 1, then the function is ready for data transfer.

     To resume transfer, use CMD52 to write the function number at FSx bits (0-3) in the CCCR register. Form the command argument for CMD52 and write it in CMDARG; bit values are listed in Table 588.
   - Write the block size in the BLKSIZ register; data will be transferred in units of this block size.

     Write the byte count in the BYTCNT register. This is the total size of the data; that is, the remaining bytes to be transferred. It is the responsibility of the software to handle the data.

     Program Command register; similar to a block transfer. For details, refer to Section 30.8.2.6 “Single-block or multiple-block read” and Section 30.8.2.7 “Single-block or multiple-block write”.

     When the Command register is programmed, the command is sent and the function resumes data transfer. Read the DF flag (Resume Data Flag). If it is 1, then the function has data for the transfer and will begin a data transfer as soon as the function or memory is resumed. If it is 0, then the function has no data for the transfer.

     If the DF flag is 0, then in case of a read, the Module waits for data. After the data time-out period, it gives a data time-out error.

Table 588. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1
30-28	Function number	0, for CCCR access
27	RAW flag	1, read after write
26	Don’t care	-

Table 588. Parameters for CMDARG register

Bits	Contents	Value
25-9	Register address	0x0D
8	Don't care	-
7-0	Write data	Function number to be aborted

30.8.2.13 Read_Wait Sequence

Read_wait is used with only the SDIO card and can temporarily stall the data transfer-either from function or memory-and allow the CPU to send commands to any function within the SDIO device. The CPU can stall this transfer for as long as required. The Module provides the facility to signal this stall transfer to the card. The steps for doing this are:

1. Check if the card supports the read_wait facility; read SRW (bit 2) of the CCCR register. If this bit is 1, then all functions in the card support the read_wait facility. Use CMD52 to read this bit.
2. If the card supports the read_wait signal, then assert it by setting the READ_WAIT (bit 6) in the CTRL register.
3. Clear the READ_WAIT bit in the CTRL register.

30.8.2.14 CE-ATA Data transfer commands

This section describes the CE-ATA data transfer commands. For information on the basic settings and interrupts generated for different conditions, refer to Section 30.8.2.5 "Data transfer commands".

30.8.2.14.1 Reset and device recovery

Before starting CE-ATA operations, the CPU should perform an MMC reset and initialization procedure. The CPU and device should negotiate the MMC TRAN state (defined by the MultiMedia Card System Specification) before the device enters the MMC TRAN state. The CPU should follow the existing MMC Card enumeration procedure in order to negotiate the MMC

TRAN state. After completing normal MMC reset and initialization procedures, the CPU should query the initial ATA Task File values using RW_REG/CMD39.

By default, the MMC block size is 512 bytes-indicated by bits 1:0 of the srcControl register inside the CE-ATA device. The CPU can negotiate the use of a 1KB or 4KB MMC block size. The device indicates MMC block sizes that it can support through the srcCapabilities register; the CPU reads this register in order to negotiate the MMC block size. Negotiation is complete when the CPU controller writes the MMC block size into the srcControl register bits 1:0 of the device.

30.8.2.14.2 ATA task file transfer

ATA task file registers are mapped to addresses 0x00h-0x10h in the MMC register space. RW_REG is used to issue the ATA command, and the ATA task file is transmitted in a single RW_REG MMC command sequence.

The CPU software stack should write the task file image to the FIFO before setting the CMDARG and CMD registers. The CPU processor then sets the address and byte count in the CMDARG-offset 0x28 in the BIU register space-before setting the CMD (offset 0x2C) register bits.

For RW_REG, there is no command completion signal from the CE-ATA device

ATA Task File Transfer Using RW_MULTIPLE_REGISTER (RW_REG)

This command involves data transfer between the CE-ATA device and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Steps involved in an ATA Task file transfer (read or write) are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register; the Module expects a single block transfer.
3. Program the CMDARG register with the beginning register address.

You should program the CMDARG, CMD, BLKSIZ, and BYTCNT registers according to the following tables.

- Program the Command Argument (CMDARG) register as shown below.

Table 589. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1 (write) or 0 (read)
30-24	Reserved	0
23:18	Starting register address for read/write; Dword aligned	0
17:16	Register address; Dword aligned	0
15-8	Reserved; bits cleared to 0 by CPU	0
7:2	Number of bytes to read/write; integral number of Dwords	16
1:0	Byte count in integral number of Dwords	0

- Program the Command (CMD) register as shown below.

Table 590. CMD register settings

Name	Value	Comment
START_CMD	1	
CSS_EXPECT	0	Command Completion Signal is not expected
READ_CEATA_DEVICE	0/1	1 – If RW_BLK or RW_REG read
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	1	
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	
SEND_AUTO_STOP	0	
TRANSFER_MODE	0	Block transfer
READ_WRITE	0/1	0 - read from card 1 - Write to card

Table 590. CMD register settings

Name	Value	Comment
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	
RESPONSE_EXPECT	1	
User-selectable		
USE_HOLD_REG	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register. Hold settings applied through the SDIOCLKCTRL register in Syscon. See Section 7.5.76.
WAIT_PRVDATA_COMPLETE	1	0 – Sends command immediately 1 – Sends command after previous data transfer over
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC

- Program the block size (BLKSIZ) register as shown below.

Table 591. BLKSIZ register

Bits	Value	Comment
31:16	0	Reserved bits as zeroes (0)
15:0	16	For accessing entire task file (16, 8-bit registers); block size of 16 bytes

- Program the Byte Count (BYTCNT) register as shown below.

Table 592. BYTCNT register

Bits	Value	Comment
31:0	16	For accessing entire task file(16, 8 bit registers); byte count value of 16 is used with the block size set to 16

30.8.2.14.3 ATA payload transfer using RW_MULTIPLE_BLOCK (RW_BLK)

This command involves data transfer between the CE-ATA device and the Module. To send a data command, the Module needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Steps involved in an ATA payload transfer (read or write) are:

1. Write the data size in bytes in the BYTCNT register.
2. Write the block size in bytes in the BLKSIZ register. The Module expects a single/multiple block transfer.
3. Program the CMDARG register to indicate the Data Unit Count.

   You should program the CMDARG, CMD, BLKSIZ, and BYTCNT registers according to the following tables.

   – Program the Command Argument (CMDARG) register as shown below.

Table 593. Parameters for CMDARG register

Bits	Contents	Value
31	R/W flag	1 (write) or 0 (read)
30-24	Reserved	0

Table 593. Parameters for CMDARG register

Bits	Contents	Value
23:16	Reserved	0
15:8	Data Count Unit [15:8]	Data count
1:0	Data Count Unit [7:0]	Data count

- Program the Command (CMD) register as shown below.

Table 594. CMD register settings

Name	Value	Comment
START_CMD	1	-
CSS_EXPECT	1	Command Completion Signal is expected; set for RW_BLK if interrupts are enabled in CE-ATA device, nIEN = 0
READ_CEATA_DEVICE	0/1	1 – If RW_BLK or RW_REG read
UPDATE_CLOCK_REGISTERS_ONLY	0	No clock parameters update command
DATA_EXPECTED	1	
SEND_INITIALIZATION	0	Can be 1, but only for card reset commands, such as CMD0
STOP_ABORT_CMD	0	
SEND_AUTO_STOP	0	
TRANSFER_MODE	0	Block transfer
READ_WRITE	0/1	0 - read from card 1 - Write to card
CMD_INDEX	Command index	
RESPONSE_LENGTH	0	
RESPONSE_EXPECT	1	
User-selectable		
WAIT_PRVDATA_COMPLETE	1	0 – Sends command immediately 1 – Sends command after previous data transfer over
CHECK_RESPONSE_CRC	1	0 – Do not check response CRC 1 – Check response CRC

- Program the block size (BLKSIZ) register as shown below.

Table 595. BLKSIZ register

Bits	Value	Comment
31:16	0	Reserved bits as zeroes (0)
15:0	512, 1024, 4096	MMC block size can be 512, 1024, or 4096 bytes as negotiated by CPU

- Program the Byte Count (BYTCNT) register as shown below.

Table 596. BYTCNT register

Bits	Value	Comment
31:0	N × block_size	byte_count should be integral multiple of block size; for ATA media access commands, byte count should be multiple of 4KB. (N × block_size = X × 4KB, where N and X are integers)

30.8.2.14.4 Sending command completion signal disable

While waiting for the Command Completion Signal (CCS) for an outstanding RW_BLK, the CPU can send a Command Completion Signal Disable (CCSD).

- Send CCSD - Module sends CCSD to the CE-ATA device if the SEND_CCSD bit is set in the CTRL register; this bit is set only after a response is received for the RW_BLK.
- Send internal Stop command - Send internally generated STOP (CMD12) command after sending the CCSD pattern. If SEND_AUTO_STOP_CCSD bit is also set when the controller is programmed to send the CCSD pattern, the Module sends the internally generated STOP command on the CMD line. After sending the STOP command, the Module sets the Auto Command Done bit in the RINTSTS register.

30.8.2.14.5 Recovery after command completion signal time-out

If time-out happened while waiting for Command Completion Signal (CCS), the CPU needs to send Command Completion Signal Disable (CCSD) followed by a STOP command to abort the pending ATA command. The CPU can program the Module to send internally generated STOP command after sending the CCSD pattern

- Send CCSD - Set the SEND_CCSD bit in the CTRL register.
- Reset bit 13 of the Command register (WAIT_PRVDATA_COMPLETE) to 0 in order to make the Module send the command at once, even though there is a data transfer in progress.
- Send internal STOP command - Set SEND_AUTO_STOP_CCSD bit in the CTRL register, which programs the CPU controller to send the internally generated STOP command. After sending the STOP command, the Module sets the Auto Command Done bit in the RINTSTS register.

30.8.2.14.6 Reduced ATA command set

It is necessary for the CE-ATA device to support the reduced ATA command subset. The following details discuss this reduced command set.

- IDENTIFY DEVICE - Returns 512-byte data structure to the CPU that describes device-specific information and capabilities. The CPU issues the IDENTIFY DEVICE command only if the MMC block size is set to 512 bytes; any other MMC block size has indeterminate results.

The CPU issues RW_REG for the ATA command, and the data is retrieved through RW_BLK.

The CPU controller uses the following settings while sending RW_REG for the IDENTIFY DEVICE ATA command. The following lists the primary bit

- CMD register setting - DATA_EXPECTED field set to 0
- CMDARG register settings:
  - Bit [31] set to 0
  - Bits [7:2] set to 128.
- Task file settings:
  - Command field of the ATA task file set to ECh
  - Reserved fields of the task file cleared to 0
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16

  The CPU controller uses the following settings for data retrieval (RW_BLK):\

- CMD register settings:
  - ccs_expect set to 1
  - DATA_EXPECTED set to 1
- CMDARG register settings:
  - Bit [31] set to 0 (Read operation)

    Data Count set to 1 (16'h0001)
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 512 IDENTIFY DEVICE can be aborted as a result of the CPU issued CMD12.
  - READ DMA EXT - Reads a number of logical blocks of data from the device using the Data-In data transfer protocol. The CPU uses RW_REG to issue the ATA command and RW_BLK for the data transfer.
  - WRITE DMA EXT - Writes a number of logical blocks of data to the device using the Data-Out data transfer protocol. The CPU uses RW_REG to issue the ATA command and RW_BLK for the data transfer.
  - STANDBY IMMEDIATE - No data transfer (RW_BLK) is expected for this ATA command, which causes the device to immediately enter the most aggressive power management mode that still retains internal device context.
- CMD register setting - DATA_EXPECTED field set to 0

  CMDARG register settings:
  - Bit [31] set to 1
  - Bits [7:2] set to 4
- Task file settings:
  - Command field of the ATA task file set to E0h
  - Reserved fields of the task file cleared to 0
- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16
  - FLUSH CACHE EXT - No data transfer (RW_BLK) is expected for this ATA command. For devices that buffer/cache written data, the FLUSH CACHE EXT command ensures that buffered data is written to the device media. For devices that do not buffer written data, FLUSH CACHE EXT returns a success status. The CPU issues RW_REG for the ATA command, and the status is retrieved through CMD39/RW_REG; there can be error status for this ATA command, in which case fields other than the status field of the ATA task file are valid.
- The CPU uses the following settings while sending the RW_REG for STANDBY IMMEDIATE ATA command:
- CMD register setting - DATA_EXPECTED field set to 0
- CMDARG register settings:
- Bit [31] set to 1
- Bits [7:2] set to 4
- Task file settings:
- Command field of the ATA task file set to EAh
- Reserved fields of the task file cleared to 0

- BLKSIZ register bits [15:0] and BYTCNT register - Set to 16

30.8.2.15 Controller/DMA/FIFO reset usage

Communication with the card involves the following:

- Controller - Controls all functions of the Module.
- FIFO - Holds data to be sent or received.
- DMA - If DMA transfer mode is enabled, then transfers data between system memory and the FIFO.
- Controller reset - Resets the controller by setting the CONTROLLER_RESET bit (bit 0) in the CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset - Resets the FIFO by setting the fifo_reset bit (bit 1) in the CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

  DMA reset - Resets the internal DMA controller logic by setting the DMA_RESET bit (bit 2) in the CTRL register, which abruptly terminates any DMA transfer in process. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

The following are recommended methods for issuing reset commands:

- Non-DMA transfer mode - Simultaneously sets CONTROLLER_RESET and FIFO_RESET; clears the RAWINTS register using another write in order to clear any resultant interrupt.
- Generic DMA mode - Simultaneously sets CONTROLLER_RESET, FIFO_RESET, and DMA_RESET; clears the RAWINTS register by using another write in order to clear any resultant interrupt. If a "graceful" completion of the DMA is required, then it is recommended to poll the STATUS register to see whether the dma request is 0 before resetting the DMA interface control and issuing an additional FIFO reset.
- In DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the RAWINTS register caused by the DMA transfers after the FIFO was reset.

30.8.2.16 Card read threshold

When an application needs to perform a Single or Multiple Block Read command, the application must program the CARDTHRCTL register with the appropriate Card Threshold size (CARDTHRESHOLD) and set the Card Read Threshold Enable (CARDRDTHREN) bit to 1'b1. This additional programming ensures that the Host controller sends a Read Command only if there is space equal to the CardRDThreshold available in the Rx FIFO. This in turn ensures that the card clock is not stopped in the middle a block of data being transmitted from the card. The Card Threshold can be set to the block size of the transfer, which ensures that there is a minimum of one block size of space in the RxFIFO before the controller enables the card clock.

The Card Read Threshold is required when the Round Trip Delay is greater than 0.5 cclk_in period as listed in Table 597.

Table 597. Card Read Threshold for Round Trip Delay

Model	Round Trip Delay (Delay_R = Delay_O + tODLY + Delay_I)	Is Stopping of Card Clock Allowed?	Card Read Threshold Required?
SDR25	Delay_R > 0.5cclk_in period Delay_R < 0.5cclk_in period	No Yes	Yes No
SDR12	Delay_R > 0.5cclk_in period Delay_R < 0.5cclk_in period	No Yes	Yes No

30.8.2.16.1 Recommended usage guidelines for card read threshold

1. The CARDTHRCTL register must be programmed before the programming the CMD register for a Data Read command.
2. The CARDTHRCTL register should not be programmed when a data transfer command is in progress.
3. A CardRdThreshold greater than or equal to the block size of the read transfer ensures that the card clock does not stop in between a block of data.
4. If the delay from the output of the card to the first sampling flip-flop in the Host controller is greater than 0.5 UI, then the Card Read Threshold must be enabled and the Card Threshold must be programmed as per guideline #3 to ensure that the Card Clock does not stop in between a block of data.

   UI (Unit Interval) - 1 is equal to 1 card clock period

   If (Delay_O + tODLY + Delay_S) > 0.5 UI, then the Host Controller samples data incorrectly if the card clock stops in between a block of data.
5. CardRdThreshold is greater than or equal to BlockSize (Recommended)

   The Card Read Threshold Size (CardRdThreshold) must be programmed to at the least 1*BlockSize of the multi block transfer to ensure that the Card Clock does not stop in between a block of data due to the RxFIFO becoming full during the read transfer.
6. CardRdThreshold is less than BlockSize

   If the CardReadThreshold is programmed to less than the BlockSize of the transfer, then the Host Controller System must ensure that the receive FIFO never becomes full and overflows during the read transfer; this can cause the card clock from the SDMMC/SDIO interface to stop. The SDMMC/SDIO interface is not be able to guarantee that the Card Clock does not stop during a read transfer.

   Remark: If the CardRdThreshold, RX_WMARK, and MSIZE are programmed incorrectly, then the Card Clock may stop indefinitely and no interrupts will be generated from the Host Controller.

30.8.2.16.2 Card read threshold programming sequence

Most cards such as SDHC or SDXC typically support block sizes that are specified in the card or are fixed to 512 bytes. For SDIO cards—standard capacity SD cards that support READ_BL_PARTIAL = 1 and MMC cards—the block size is variable and can be chosen by the application.

In order to use the Card Read Threshold feature effectively and to guarantee that the Card Clock does not stop because of a FIFO Full condition in the middle of a block of data being read from the Card, follow these steps:

1. Choose Block Size

   The block size must be based on the following:

   – Rule 1 – DWORD-aligned Block Size

     The block size requested by the application from the card for the read transfer card must be DWORD-aligned.

2. Enable Card Read Threshold feature

   – Rule 2 – Block Size ≤ Total Fifo Depth

     CardRdThreshold can be enabled only if the block size for the given transfer is less than the total depth of the FIFO.

     BlkSize ≤ FifoDepth

     Where:

     BlkSize = (block size in bytes) * 8/F_DATA_WIDTH; that is, the number of the block size in terms of FIFO locations

     FifoDepth = total number of FIFO Locations

   Remark: Note To use the Card Read Threshold for different block sizes when selecting the FIFO depth during configuration of the Host Controller, the selected FIFO depth must be greater than or equal to the maximum block size that the Host Controller is required to support. Typically, the largest block size to support is 512 bytes; the corresponding FIFO depth would 128 locations in the 32-bit-wide FIFO. If you choose a FIFO depth that is two or more times the maximum block size that the Host Controller is required to support, there will be greater performance and flexibility in choosing the MSIZE and RX_WMARK for the best throughput.

3. Choose CardRdThreshold

   – If BlkSize ≥ ½ FifoDepth, choose CardRdThreshold such that CardRdThreshold ≤ BlkSize in bytes

   – If BlkSize < ½ FifoDepth, choose CardRdThreshold such that CardRdThreshold = BlkSize in bytes

   Remark: If the Host Controller is operating in Internal DMA Mode, or if the application does not use burst transfers to read data out of the FIFO while operating in External DMA mode or Slave mode, then use steps 4 and 5 below, and skip steps 6, 7, and 8. If the application uses burst transfers to read data out of the FIFO while operating in External DMA mode or in Slave mode, then skip steps 4 and 5, and instead follow steps 6, 7, and 8 below.

4. Choose DW_DMA_Mutiple_Transaction_Size

   The possible values for the DW_DMA_Mutiple_Transaction_Size (MSIZE) are 1, 4, 8, 16, 32, 64, 128, and 256 transfers. Choose the value of MSIZE from the above transfer values so that MSIZE is a multiple of BlkSize.

   BlkSize% (MSIZE) = 0

   Special Cases:

   – When MSIZE = 1 transfer
     The MSIZE is equal to 1 if the block size chosen in Step 1 is not a multiple of the FIFO Width (in bytes).
     If MSIZE = 1 is not acceptable and a higher burst size is desired—that is, a higher MSIZE— then go back to Step 1 and recalculate the block size.

– Internal DMA (IDMAC) mode
The size of the data buffer (BuffSize in bytes) for each descriptor must be a multiple of MSIZE * H_DATA_WIDTH / 8. For example, BuffSize = n * MSIZE * H_DATA_WIDTH / 8, where n = 1, 2, 3…

5. Choose RX Watermark

– If MSIZE = 1, then the RX_WMARK = 1 or RX_WMARK = BlkSize -1

Remark: Note For slave mode when the RX_WMARK is reached, the application must read only RX_WMARK number of locations from the FIFO.

Additionally, for all DMA modes the RX Watermark (RX_WMARK) chosen must be a multiple of the chosen MSIZE.

RX_WMARK = MSIZE *n, where n = 1, 2, 3 …

Remark: If the Host Controller is operating in Internal DMA Mode, or if the application does not use burst transfers to read data out of the FIFO while operating in External DMA mode or Slave mode, then use steps 4 and 5 above, and skip steps 6, 7, and 8. If the application uses burst transfers to read data out of the FIFO while operating in External DMA mode or in Slave mode, then skip steps 4 and 5 above, and instead follow steps 6, 7, and 8 below.

6. Choose Burst Size

a. In order to determine burst transfers to drain data from the FIFO in external DMA or slave mode, use the following:
BurstSize = number_of_beats * transfer_size in bytes per beat

b. Choose the number of beats and the transfer size per beat so that BurstSize is a sub-multiple of BlkSize:
BlkSize % (BurstSize) = 0
Where:
BlkSize = Block Size in bytes * 8 /F_DATA_WIDTH
If H_DATA_WIDTH = 16 then:
(BlkSize * 2) % (BurstSize) = 0

7. Choose RX Watermark

– Use the following to determine RX Watermark (RX_WMARK):
RX_WMARK = (BurstSize / F_DATA_WIDTH * 8) - 1

Remark: Note For slave mode when the RX_WMARK is reached, the application must read only RX_WMARK number of locations from the FIFO.

8. Program MSIZE

The possible values for the DW_DMA_Mutiple_Transaction_Size (MSIZE) are 1, 4, 8, 16, 32, 64, 128 or 256 transfers (refer FIFOTH[30:28]).

a. Choose the following to determine the value of MSIZE:
MSIZE > (BurstSize / H_DATA_WIDTH * 8)

30.8.2.16.3 Example card read threshold programming when BLKSIZE > 1/2 FIFO depth

Given:
H_DATA_WIDTH = 32
F_DATA_WIDTH = 32
Total FIFO Depth:
FifoDepth = 32 locations = 32 * F_DATA_WIDTH/8 = 128 bytes

The following shows how to program the CardRdThreshold when BlkSize is greater than ½ FifoDepth.

1. Choose a DWORD-aligned BlkSize less than FifoDepth.

   If Block Size is 192 bytes, then BlkSize = 192 * 8/F_DATA_WIDTH = 48 FIFO locations
2. For DMA modes, choose MSIZE such that the BlkSize is a multiple of MSIZE.

   Possible MSIZE values 1, 4, 8 and 16, such that (48% MSIZE) = 0.

   Choose MSIZE = 8.
3. Choose the RX_WMARK = MSIZE -1.

   For example, RX_WMARK ≤ 8 - 1 = 7 FIFO locations.
4. Since BlkSize > ½ FifoDepth, choose CardRdThreshold = Block Size.

   CardRdThreshold = 192 bytes

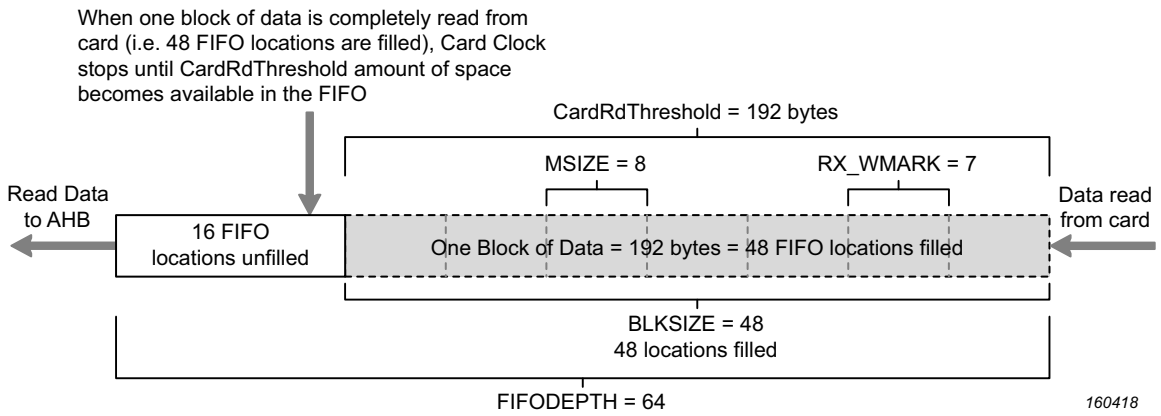


Fig 93. FIFO contents when BlkSize > ½ FifoDepth

30.8.2.16.4 Example card read threshold programming when BLKSIZE < 1/2 FIFO depth

Given:
H_DATA_WIDTH = 32
F_DATA_WIDTH = 32
Total FIFO Depth:
FifoDepth = 32 locations = 32 * F_DATA_WIDTH/8 = 128 bytes

The following shows how to program the CardRdThreshold when BlkSize is less than ½ FifoDepth.

1. Choose a DWORD-aligned BlkSize less than FifoDepth.

   If Block Size is 64 bytes, then BlkSize = 64 * 8/F_DATA_WIDTH = 64 * 8/32 = 16 FIFO locations
2. For DMA modes, choose MSIZE such that the BlkSize is a multiple of MSIZE.

   Possible MSIZE values are 1, 4, 8, and 16 so that (16%MSIZE) = 0.

   Choose MSIZE = 16.
3. Choose the RX_WMARK = MSIZE - 1.

For example, RX_WMARK = 16 - 1 = 15 FIFO locations.

4. Since BlkSize < ½ FifoDepth, choose CardRdThreshold = Block Size.
   CardRdThreshold = 64 bytes.

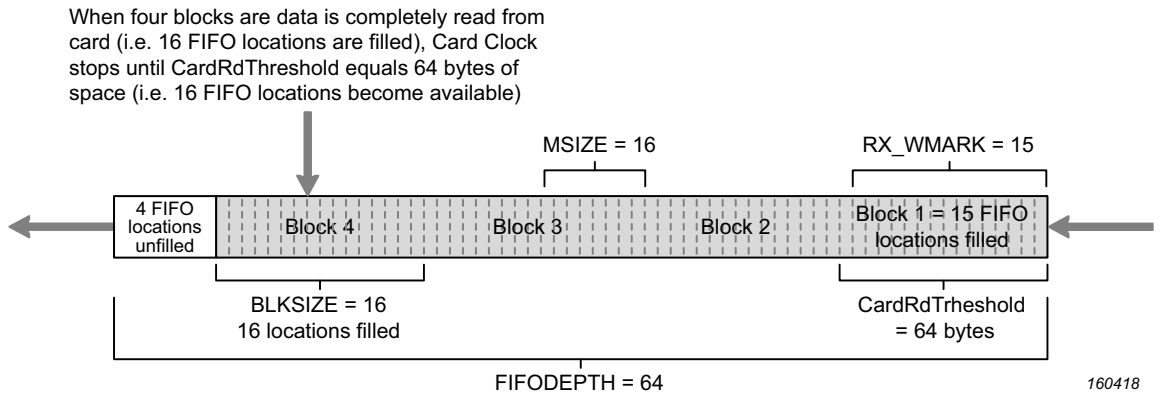


Fig 94. FIFO contents when BlkSize < ½ FifoDepth

30.8.2.17 Back-end power

The host controller needs to set the bit to enable the power sent to the back-end function of the card.

30.8.2.18 Master power control

Whether a card requires greater than 720 mW of power or not can be determined by reading the SMPC (Support Master Power Control) and the EMPC (Enable Master Power Control) registers in the card. Figure 95 illustrates the CCCR register.

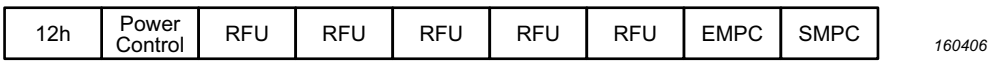


Fig 95. Card common control registers (CCCR)

The chart in Figure 96 illustrates a basic flow that the device driver follows in order to detect the power requirements of the card.

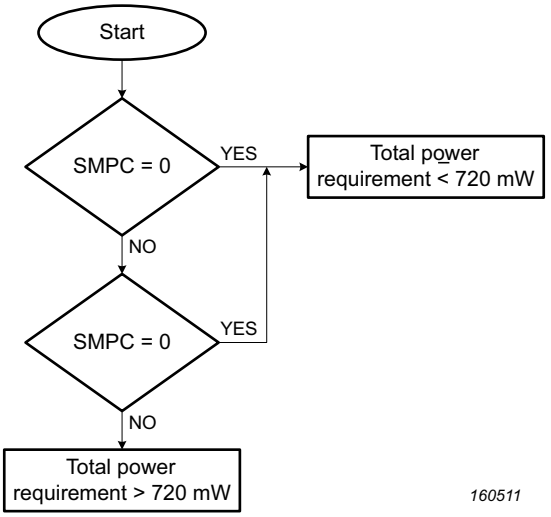


Fig 96. Flowchart for card power requirements

1. The driver reads the SMPC register.
2. If the value is 0, then the total power requirement is less than or equal to 720 mW.
3. If the value is 1, then the driver reads the EMPC register.
   a. If the value is 0 then the total power requirement is less than or equal to 720 mW.
   b. If the value is 1, then the total power requirement is greater than 720 mW.

Once the power requirement has been determined to be more than 720 mW, the driver may tune the power to match the card requirements by tuning each function in the card.

Figure 97 illustrates the two mechanisms for power tuning to get the correct power levels of each function.

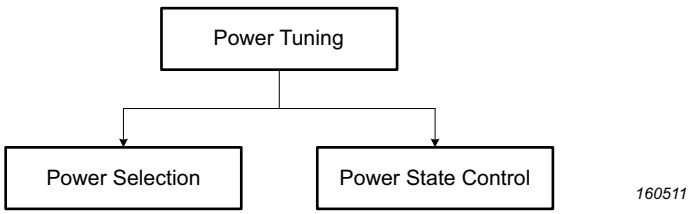


Fig 97. Different power tuning mechanisms

In addition to the CCCR, each supported I/O function has an FBR register.

Table 598. FBR registers for power tuning

Addr: Field	Type	Description
SPS	R	Support Power Selection • SPS = 0: Has no Power Selection; EPS is zero. • SPS = 1: Has two power modes selected by EPS. This bit is effective when EMPC = 1 in CCCR and PS[3:0] = 0.
EPS	R/W	Enable Power Selection Indicates if function has Power Selection. • EPS = 0 (default): Operates in Higher Current Mode. Maximum current is given in TPLFE_HP_MAX_PWR_3.3V. • EPS = 1: Operates in Lower Current Mode. Maximum current is given in TPLFE_LP_MAX_PWR_3.3V.. This bit is reset when IOEx = 0 and is effective when EMPC = 1 in CCR and PS[3:0] = 0.
PSx	R/W	Power State PS[3:0] If PS[3:0] is set to 0, TPL_CODE_CISTPL_FUNCE (22h) extension 01h is used and card power is controlled by EMPC and EPS (SDIO Ver 2.0-compatible). Power State control is defined by SDIO Ver 3.0 and is effective when EMPC is set to 1 and PS[3:0] is set to greater than 0. In this case, a list of card-supported power states is determined by TPL_CODE_CISTPL_FUNCE (22h) extension 02h (Power State Tuple). Host driver finds affordable power in Tuple and n (>0) is set to this field. Total current of card is sum of selected current of each function.

The chart in Figure 98 defines a basic flow that the device driver follows in order to detect and program the power requirements of each function in the card.

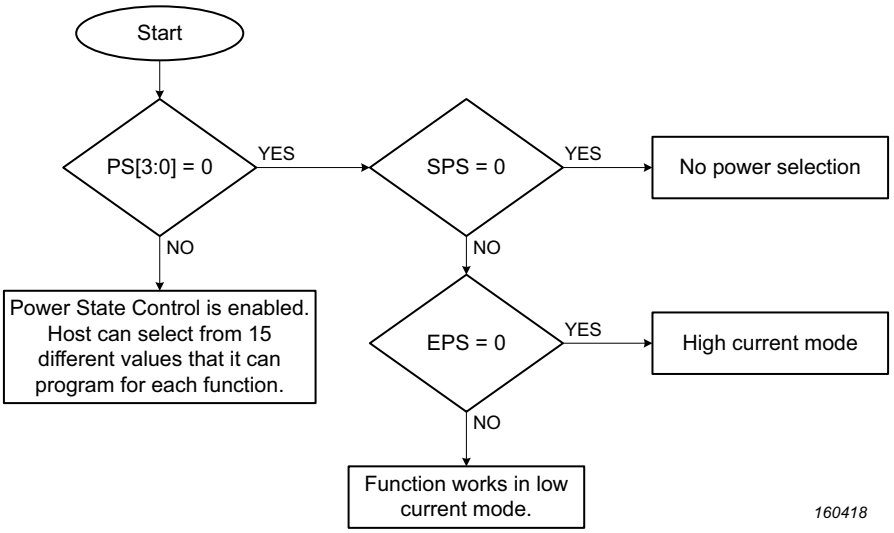


Fig 98. Flowchart for detect and program power requirements of functions

The driver reads the PS[3:0] register.

1. If the value of PS[3:0] is 1, the driver can program a value in the PS[3:0] register based on its affordable power in the tuples and program a number greater than 0.
2. If the value of PS [3:0] is 0, the driver reads the SPS register.

- If the value of SPS is 0, the function does not support power selection.
- If the value of SPS is 1, the driver reads the EPS register.
  If the value of EPS is 0, the function is set to High Current Mode.
  If the value of EPS is 1, the function is set to Low Current Mode.

Remark: Note Testing this feature may not be of much value to the verification team, but will help in building the driver.

30.8.2.19 Error handling

The Module implements error checking; errors are reflected in the RAWINTS register and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (INT_ENABLE in the CTRL register is 0), and all the interrupts are masked (bits 0-31 of the INTMASK register; default is 0). Error handling:

- Response and data time-out errors - For response time-out, software can retry the command. For data time-out, the Module has not received the data start bit - either for the first block or the intermediate block - within the time-out period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the TCBCNT later, the software can decide how many bytes remain to be copied.
- Response errors - Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors - Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.
- Hardware locked error - Set when the Module cannot load a command issued by software. When software sets the START_CMD bit in the CMD register, the Module tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
- FIFO underrun/overrun error - If the FIFO is full and software tries to write data in the FIFO, then an overrun error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read
- The FIFO_EMPTY or FIFO_FULL bits in the STATUS register.
- Data starvation by CPU time-out - Raised when the Module is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated time-out period. Under this condition and when a read transfer is in process, the software
- Should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command - If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response time-out is expected from the Module. The ATA layer is notified that an MMC transport layer error occurred.

- Write operation - Any MMC Transport layer error known to the device causes an outstanding ATA command to be terminated. The ERR bits are set in the ATA status registers and the appropriate error code is sent to the ATA Error register.
- If nIEN = 0, then the Command Completion Signal (CCS) is sent to the CPU.

If device interrupts are not enabled (nIEN = 1), then the device completes the entire Data Unit Count if the CPU controller does not abort the ongoing transfer.

During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register. It then continues further data transmission until all the bytes are transmitted.

- Read operation - If MMC transport layer errors are detected by the CPU controller, the CPU completes the ATA command with an error status.

The CPU controller can issue a Command Completion Signal Disable (CCSD) followed by a STOP TRANSMISSION (CMD12) to abort the read transfer. The CPU can also transfer the entire Data Unit Count bytes without aborting the data transfer.

30.8.2.20 Transmission and reception with internal DMAC (IDMAC)

The general sequence of events for transmit and receive is as follows:

1. Program the required programming in the Bus Mode register (BMOD). If IDMAC enable is disabled during the middle of an IDMAC transfer, it has no effect. It will only take effect for a new data transfer command. Issuing a Software reset will immediately terminate the transfer. It is recommended that the host issue a reset to DMA interface by setting DMA_RESET bit of the CTRL register and then issue a IDMAC software reset. The PBL contents are read-only and a direct reflection of the DW_DMA_Mutiple_Transaction_Size contents in FIFOTH register. The Fixed burst bit has to be programmed appropriately for system performance.
2. In case a Descriptor Unavailable Interrupt is asserted, the host needs to form the descriptor, appropriately set its own bit and then write to Poll Demand register (PLDMND) for the IDMAC to re-fetch the descriptor.
3. It is always appropriate for the host to enable abnormal interrupts since any errors related to the transfer are reported to the host.
4. For handling scenarios like Fatal Bus Error, Abort and FIFO overrun/under-run refer to Section 30.8.3.2.6 “Interrupts”.

For more information, refer to Section 30.8.3.2.4 “Transmission” and Section 30.8.3.2.5 “Reception”.

30.8.3 DMA descriptors

The SD/MMC DMA controller uses the following descriptor structures:

- Dual buffer Structure – The distance between two descriptors is determined by the Skip Length value programmed in the Descriptor Skip Length (DSL) field of the Bus Mode register (BMOD).
- Chain Structure – Each descriptor points to a unique buffer and the next descriptor.

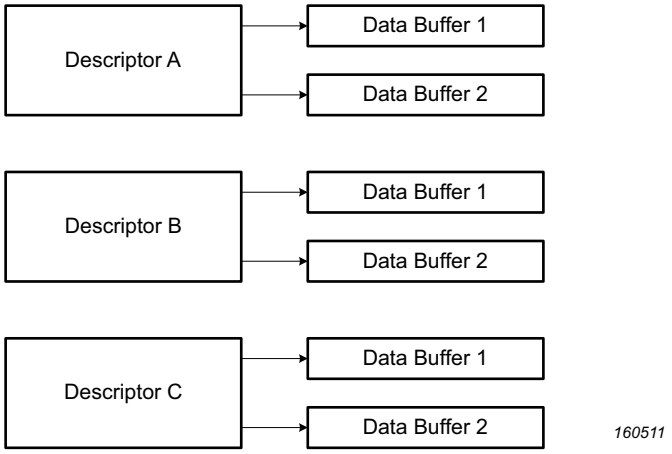


Fig 99. Dual buffer descriptor structure

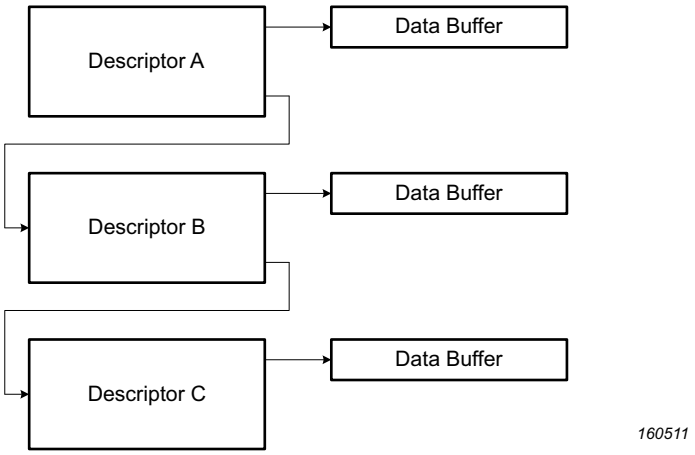


Fig 100. Chain descriptor structure

30.8.3.1 SD/MMC DMA descriptors

30.8.3.1.1 SD/MMC DMA descriptor DESC0

The DES0 descriptor contains control and status information.

Table 599. SD/MMC DMA DESC0 descriptor

Bit	Symbol	Description
0	-	Reserved
1	DIC	Disable Interrupt on Completion When set, this bit will prevent the setting of the TI/RI bit of the IDMAC Status register (IDSTS) for the data that ends in the buffer pointed to by this descriptor.
2	LD	Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the data.

Table 599. SD/MMC DMA DESC0 descriptor

Bit	Symbol	Description
3	FS	First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the data. If the size of the first buffer is 0, next Descriptor contains the beginning of the data.
4	CH	Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, BS2 (DES1[25:13]) should be all zeros.
5	ER	End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The IDMAC returns to the base address of the list, creating a Descriptor Ring. This is meaningful for only a dual buffer descriptor structure.
29:6	-	Reserved
30	CES	Card Error Summary These error bits indicate the status of the transaction to or from the card. These bits are also present in RINTSTS Indicates the logical OR of the following bits: • EBE: End Bit Error • RTO: Response Time-out • RCRC: Response CRC • SBE: Start Bit Error • DRTO: Data Read Time-out • DCRC: Data CRC for Receive • RE: Response Error
31	OWN	When set, this bit indicates that the descriptor is owned by the SD/MMC DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The SD/MMC DMA clears this bit when it completes the data transfer.

30.8.3.1.2 SD/MMC DMA descriptor DESC1

The DES1 descriptor contains the buffer size.

Table 600. SD/MMC DMA DESC1 descriptor

Bit	Symbol	Description
12:0	BS1	Buffer 1 Size Indicates the data buffer byte size, which must be a multiple of 4 bytes. The buffer 1 size must not exceed 4096 bytes (0x1000). If this field is 0, the DMA ignores this buffer and proceeds to the next descriptor in case of a chain structure, or to the next buffer in case of a dual buffer structure. **Remark:** If there is only one descriptor and only one buffer to be programmed, use only the Buffer 1 and not Buffer 2.
25:13	BS2	Buffer 2 Size These bits indicate the second data buffer byte size. The buffer size must be a multiple of 4. The buffer 2 size must not exceed 4096 bytes (0x1000).This field is not valid if DES0[4] is set.
31:26	-	Reserved

30.8.3.1.3 SD/MMC DMA descriptor DESC2

The DES2 descriptor contains the address pointer to the data buffer.

Table 601. SD/MMC DMA DESC2 descriptor

Bit	Symbol	Description
31:0	BAP1	Buffer Address Pointer 1 These bits indicate the physical address of the first data buffer. The SD/MMC DMA ignores DES2 [1:0], corresponding to the bus width of 64/32/16, internally.

30.8.3.1.4 SD/MMC DMA descriptor DESC3

The DES3 descriptor contains the address pointer to the next descriptor if the present descriptor is not the last descriptor in a chained descriptor structure or the second buffer address for a dual buffer structure.

Table 602. SD/MMC DMA DESC3 descriptor

Bit	Symbol	Description
31:0	BAP2	Buffer Address Pointer 2/ Next Descriptor Address These bits indicate the physical address of the second buffer when the dual buffer structure is used. If the Second Address Chained (DES0[4]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present. If this is not the last descriptor, then the Next Descriptor address pointer must be bus-width aligned (DES3[1:0] = 0, internally the LSBs are ignored).

30.8.3.2 Initialization

For the SD/MMC DMA initialization, follow these steps:

1. Write to the Bus Mode register (BMOD) to set the Host bus access parameters.
2. Write to the Interrupt Enable register (IDINTEN) to mask unnecessary interrupt causes.
3. The software driver creates either the Transmit or the Receive descriptor list. Then it writes to Descriptor List Base Address register (DBADDR), providing the IDMAC with the starting address of the list.
4. The SD/MMC DMA engine attempts to acquire descriptors from the descriptor lists.

30.8.3.2.1 Host bus burst access

The SD/MMC DMA attempts to execute fixed-length burst transfers on the AHB Master interface if configured using the FB bit of the IDMAC Bus Mode register. The maximum burst length is indicated and limited by the PBL field. The descriptors are always accessed in the maximum possible burst-size for the 16-bytes to be read: 16*8/bus-width.

The SD/MMC DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in the FIFO or the number of bytes to the end of data, when less than the configured burst-length. The SD/MMC DMA indicates the start address and the number of transfers required to the AHB Master Interface. When the AHB Interface is configured for fixed-length bursts, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise, in no fixed-length bursts, it transfers data using INCR (undefined length) and SINGLE transactions.

30.8.3.2.2 Host data buffer alignment

The transmit and receive data buffers in host memory must be 32-bit aligned.

30.8.3.2.3 Buffer size calculations

The driver knows the amount of data to transmit or receive. For transmitting to the card, the IDMAC transfers the exact number of bytes to the FIFO, indicated by the buffer size field of DES1.

If a descriptor is not marked as last - LS bit of DES0 - then the corresponding buffers of the descriptor are full, and the amount of valid data in a buffer is accurately indicated by its buffer size field. If a descriptor is marked as last, then the buffer cannot be full, as indicated by the buffer size in DES1. The driver is aware of the number of locations that are valid in this case.

30.8.3.2.4 Transmission

The SD/MMC transmission occurs as follows:

1. The Host sets up the Descriptor (DES0-DES3) for transmission and sets the OWN bit (DES0[31]). The Host also prepares the data buffer.
2. The Host programs the write data command in the CMD register in BIU.
3. The Host will also program the required transmit threshold level (TX_WMARK field in FIFOTH register).
4. The SD/MMC DMA determines that a write data transfer needs to be done as a consequence of step 2.
5. The SD/MMC DMA engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the SD/MMC DMA enters suspend state and asserts the Descriptor Unable interrupt in the SD/MMC DMA status register (IDSTS). In such a case, the host needs to release the SD/MMC DMA by writing any value to the Poll Demand register (PLDMND).
6. It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
7. The SD/MMC DMA engine will now wait for a DMA interface request from BIU. This request will be generated based on the programmed transmit threshold value. For the last bytes of data which can’t be accessed using a burst, SINGLE transfers are performed on AHB Master Interface.
8. The SD/MMC DMA fetches the Transmit data from the data buffer in the Host memory and transfers to the FIFO for transmission to card.
9. When data spans across multiple descriptors, the SD/MMC DMA will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
10. When data transmission is complete, status information is updated in SD/MMC DMA status register (IDSTS) by setting Transmit Interrupt, if enabled. Also, the OWN bit is cleared by the SD/MMC DMA by performing a write transaction to DES0.

30.8.3.2.5 Reception

The SD/MMC reception occurs as follows:

1. The Host sets up the Descriptor (DES0-DES3) for reception, sets the OWN (DES0[31]).
2. The Host programs the read data command in the CMD register in BIU.

3. The Host will program the required receive threshold level (RX_WMARK field in FIFOTH register).
4. The SD/MMC DMA determines that a read data transfer needs to be done as a consequence of step 2.
5. The SD/MMC DMA engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the DMA enters suspend state and asserts the Descriptor Unable interrupt in the SD/MMC DMA Status register (IDSTS). In such a case, the host needs to release the SD/MMC DMA by writing any value to the Poll Demand register (PLDMND).
6. It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
7. The SD/MMC DMA engine will now wait for a DMA interface request (dw_dma_req) from BIU. This request will be generated based on the programmed receive threshold value. For the last bytes of data which can't be accessed using a burst, SINGLE transfers are performed on AHB.
8. The SD/MMC DMA fetches the data from the FIFO and transfer to Host memory.
9. When data spans across multiple descriptors, the SD/MMC DMA will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
10. When data reception is complete, status information is updated in SD/MMC DMA Status register (IDSTS) by setting Receive Interrupt, if enabled. Also, the OWN bit is cleared by the SD/MMC DMA by performing a write transaction to DES0.

30.8.3.2.6 Interrupts

Interrupts can be generated as a result of various events. The SD/MMC DMA Status register (IDSTS) contains all the bits that might cause an interrupt. The SD/MMC DMA Interrupt Enable register (IDINTEN) contains an Enable bit for each of the events that can cause an interrupt.

There are two groups of summary interrupts - Normal and Abnormal - as outlined in Status register (IDSTS). Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal is deasserted.

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt (IDSTS[1]) indicates that one or more data was transferred to the Host buffer.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the SD/MMC DMA Status register for the interrupt cause.

Remark: The final interrupt (int) signal from created is a logical OR of the interrupt from BIU and SD/MMC DMA.

30.8.3.2.7 Abort

When the host issues CMD12 when a data transfer on the card data lines is in progress, the FSM closes the present descriptor after completing the transfer of data until a DTO interrupt is asserted. Once an abort command is issued, the DMA performs single burst transfers:

1. When the host issues CMD12 when a data transfer on the card data lines is in progress, the FSM closes the present descriptor after completing the transfer of data until a DTO interrupt is asserted. Once an abort command is issued, the DMAC performs single burst transfers.
2. For a card read, the SD/MMC DMA keeps popping data from FIFO and writes to the host memory until a DTO interrupt is generated. This is required since DTO interrupt is not generated until and unless all the FIFO data is emptied.

Remark: The following scenarios apply for closing the descriptors:

- In case of an FBE, the current descriptor and the remaining unread descriptors are not closed by the SD/MMC DMA.
- In case of a write abort, only the current descriptor during which an abort occurred is closed by the SD/MMC DMA. The remaining unread descriptors are not closed by the IDMAC.
- In case of a read abort, the SD/MMC DMA pops the data out of the FIFO and writes them to the corresponding descriptor data buffers. The remaining unread descriptors are not closed.

30.8.3.2.8 FBE scenarios

An FBE occurs due to an AHB error response on the AHB bus. This is a system error, so the software driver should not perform any further programming to the SD/MMC The only recovery mechanism from such scenarios is to do one of the following:

- Issue a hard reset by asserting the reset_n signal.
- Do a program controller reset by writing to the CTRL[0] register.

30.8.3.2.9 FIFO overflow and underflow

During normal data transfer conditions, FIFO overflow and underflow will not occur. However if there is a programming error, then FIFO overflow/underflow can result. For example, consider the following scenarios.

For transmit: PBL = 4, Tx watermark = 1. For these programming values, if the FIFO has only one location empty, it issues a dw_dma_req to DMA state machine. Due to PBL value = 4, the DMA performs 4 pushes into the FIFO. This will result in a FIFO overflow interrupt.

For receive: PBL=4, Rx watermark = 1. For these programming values, if the FIFO has only one location filled, it issues a dw_dma_req to the DMA state machine. Due to PBL value = 4, the DMA performs 4 pops to the FIFO. This will result in a FIFO underflow interrupt.

The driver should ensure that the number of bytes to be transferred as indicated in the descriptor should be a multiple of 4 bytes. For example, if the BYTCNT = 13, the number of bytes indicated in the descriptor should be 16.

30.8.3.2.10 Programming of PBL and watermark levels

The SD/MMC DMA performs data transfers depending on the programmed PBL and threshold values. Table 603 lists the allowed programming values.

Table 603. PBL and watermark levels

PBL (number of transfers)	Transmit/receive watermark value
1	greater than or equal to 1
4	greater than or equal to 4
8	greater than or equal to 8
16	greater than or equal to 16
32	greater than or equal to 32
64	greater than or equal to 64
128	greater than or equal to 128
256	greater than or equal to 256

30.8.4 Back-end power

Each device needs one bit to control the back-end power supply for an embedded device; this bit does not control the VDDH of the host controller. A BACK_END_POWER register enables software programming for back-end power. The value on this register is output to the back_end_power signal, which can be used to switch power on and off the embedded device.

30.8.5 Master power control

SDIO cards prior to the 1.10 specification required a maximum power of 200 mA * 3.3v = 720 mW, regardless of the number of functions in each card. Newer cards have increased requirements greater than 720 mW.

30.8.6 Dedicated interrupt pin

The interrupt line is defined only for eSDIO devices. This interrupt line can operate even when the card clock is switched off and can be used only during an asynchronous interrupt period.

Figure 101 shows the pins of an eSDIO device.

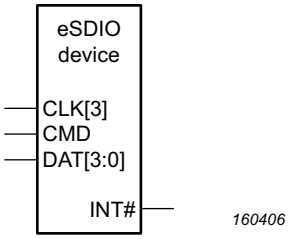


Fig 101. eSDIO device

30.8.7 Card-Detect and Write-Protect mechanism

Figure 102 illustrates how the SDMMC/SDIO interface card detection and write-protect signals are connected. Most of the SD_MMC sockets have card-detect pins. When no card is present, SD_CARD_DETECT_N is 1 due to the pull-up. When the SD_MMC card

is inserted, the card-detect pin is shorted to ground, which makes card_detect_n go to 0. Similarly in SD cards, when the write-protect switch is toward the left, it shorts the SD_WR_PRT port to ground.

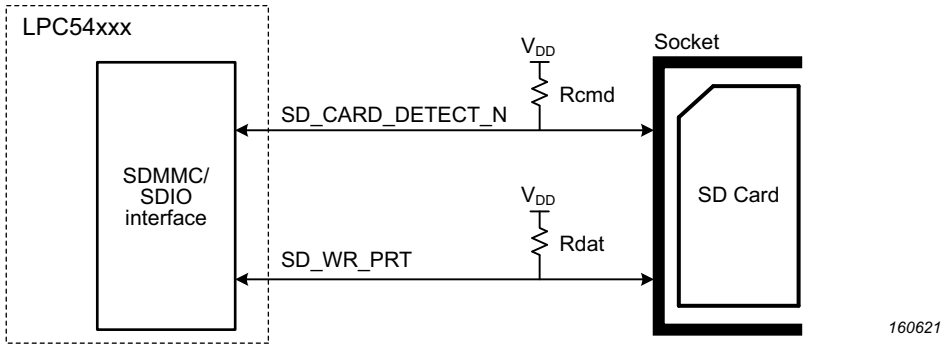


Fig 102. Card-Detect and Write-Protect

30.8.8 Termination requirement

Figure 103 illustrates the SDMMC/SDIO interface termination requirements, which is required to pull up the SD_CMD and SD_Dn lines on the SD_MMC_CEATA bus. The recommended specification for pull-up on the SD_CMD line is 4.7K - 100K for MMC, and 10K - 100K for an SD. The recommended pull-up on the SD_Dn line is 50K - 100K.

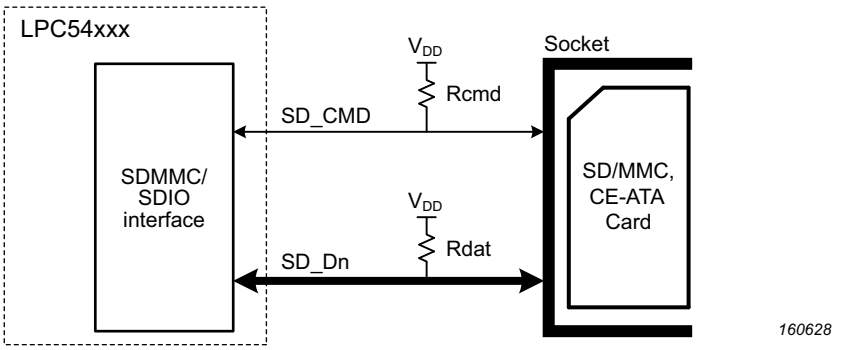


Fig 103. Termination

30.8.8.1 Rcmd and Rod calculation

The SD and MMC card enumeration happens at a very low frequency – 100-400 kHz. During enumeration open-drive mode is used. The pull-up in the command line pulls the bus to 1 when the card drives “z.” MMC interrupt mode also uses the pull-up. During normal data transfer, the card driver switches to push-pull mode.

For example, if enumeration is done at 400 kHz and the total bus capacitance is 20 pf, the pull-up needed during enumeration is:

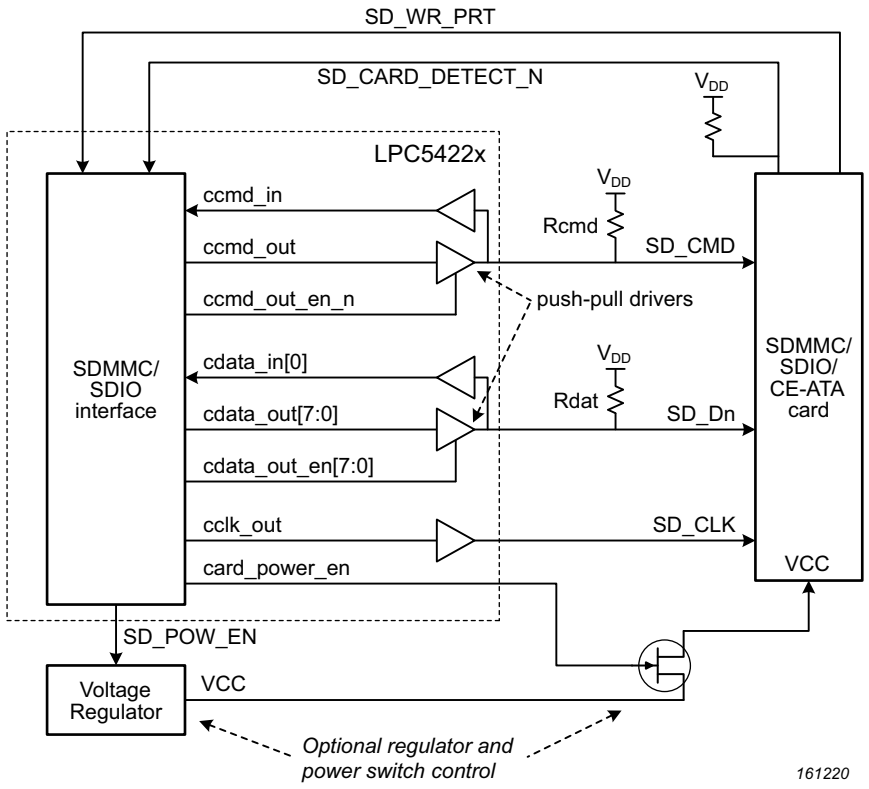
2.2 RC = rise-time = 1/400 kHz
R = 1/(2.2 * C * 100 kHz
= 1/(2.2 x 20 x 10**-12 x 400 x 10**3
= 1/(1.76 x 10**-5
= 56.8K

The Rod and Rcmd should be adjusted in such a way that the effective pull-up is at the maximum 5.68K during enumeration. Since only one card is supported, a fixed Rcmd resister is sufficient and there is no need for an additional Rod pull-up during enumeration. You should also ensure the effective pull-up will not violate the Iol rating of the drivers.

30.8.9 Interfacing to SD memory, SDIO, and MMC card

Figure 104 illustrates the connection between the SDMMC/SDIO interface and SD memory, SDIO, and MMC cards in SD_MMC_CE-ATA mode, which supports all three types of cards in the same controller. The primary differences between the MMC-Ver3.3-only and SD_MMC_CE-ATA modes are:

- In SD_MMC_CE-ATA mode, an SD card can be either a 1-bit data or 4-bit data card; MMC (3.31) cards are always 1-bit only, while MMC (4.0) cards could be either in 1-bit, 4-bit, or 8-bit mode.



Rcmd - typically 4.7 to 100 K
Rdat - typically 50 to 100 K
Rod - VDD / (Rod || Rcmd) < Iol (2 mA)

Fig 104. SD/MMC, SDIO, and MMC Card/CE-ATA single-card connection

30.9 Clocking and timing guidelines

The SDMMC/SDIO interface (also referred to as the host controller) has four input clocks and one output clock.

30.9.1 Clock domains

The SDMMC/SDIO interface has the following clocks:

Table 604. Clocks

Clock name	Input/Output	Edge used within the controller
clk	Input	Rising edge
cclk_in	Input	Rising and falling edges
cclk_in_drv	Input	Rising and falling edges
cclk_in_sample	Input	Rising and falling edges
cclk_out	Output	-

- clk – System/host/AHB clock. BIU block is clocked with this clock; all configuration registers work on this clock.
- cclk_in (rising edge) – Most blocks/registers in CIU are clocked with rising edge of cclk_in clock.
- cclk_in_drv – Phase-shifted/delayed version of cclk_in on which output-related registers work, used to clock outputs to the external card. This clock is used to meet the hold-time requirements across various modes—SDR25, SDR12, and so on. By providing an appropriate phase shift in cclk_in_drv, required hold-time requirements can be achieved.
- cclk_in_sample – Phase-shifted/delayed version of cclk_in used to sample outputs from the card into the core through ccmd_in and cdata_in.
- cclk_out – Generated output clock derived from cclk_in. This becomes SD_CLK in the outside world. There is a large skew between cclk_in and cclk_out, which occurs due to:

  cclk_out_generation_delay + routing_delay_inside_core + cclk_out_PAD_delay

  The card uses cclk_out to sample the outputs from the core and also drives the data and response to the core with respect to cclk_out. Therefore, these signals are guaranteed to meet setup and hold times with respect to cclk_out. However, when these card outputs are sampled by the core, they cannot be directly sampled by cclk_in because of a large skew between cclk_out and cclk_in. A delayed sample clock (cclk_in_sample) is needed in order to match the above mentioned delay/skew. The SDMMC/SDIO interface samples these signals with respect to cclk_in_sample, and the sampled signals are then used inside the core.
- cclk_in (falling edge) – The clock gating signals that gate cclk_out are generated from the falling edge of cclk_in. This is done to avoid clock-gating glitches on cclk_out, which is gated under the following conditions:
  - When a card is idle.
  - When low-power mode is selected.

– During data transfers, in order to avoid data loss when the host does not push or pop data to or from the FIFO at the rate the data is sent or received from the cards; that is, when the FIFO is full and a read from a card occurs, or the FIFO is empty and a write to a card occurs.

30.9.1.1 Relationships between clocks

Figure 105 shows the different clocks in the SDMMC/SDIO interface:

- cclk_in
- cclk_out
- cclk_in_sample
- cclk_in_drv

Remark: Note that the AMBA clock should be at least equal to or greater than one-tenth of cclk_in.

All relevant delays corresponding to the core are indicated—bypass_mode refers to the selection where cclk_in clock divider logic can be bypassed; that is, cclk_out is an undivided version of cclk_in.

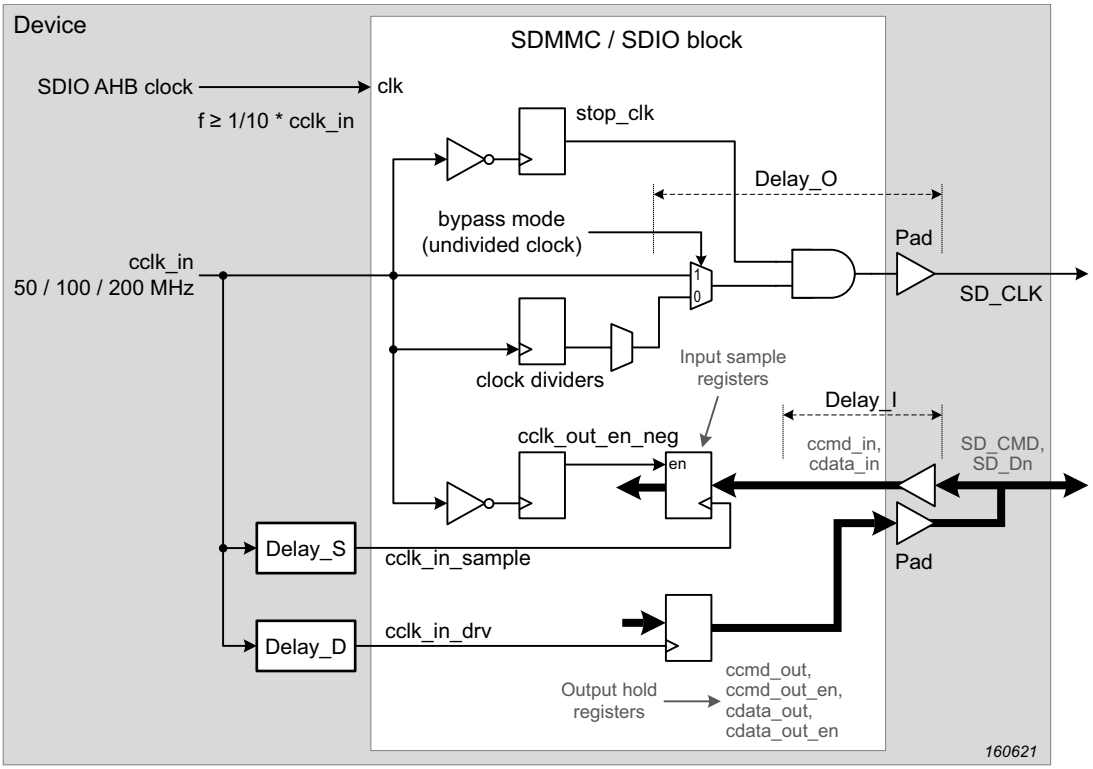


Fig 105. Different clocks and delays

Definitions:

- Delay_O = cclk_in to cclk_out delay (including PAD)
- Delay_I = Input PAD delay + routing delay to input register

- tODLY = cclk_out to card output delay (varies across card manufactures and speed modes)
- Delay_S = Delay by which cclk_in_sample is phase-shifted with regard to cclk_in
- Delay_D = Delay by which cclk_in_drv is phase-shifted with regard to cclk_in
- Delay_R = Delay_O+Delay_I + tODLY = Total turn-around delay

Figure 105 and illustrates relationships for these delays.

Figure 106 shows the relationships between cclk_in, cclk_out, cclk_in_sample, and cclk_in_drv.

Remark: Controller outputs driven to the card are driven on cclk_in_drv when USE_HOLD_REG = 1. SDMMC/SDIO interface sampled_inputs are sampled on cclk_in_sample.

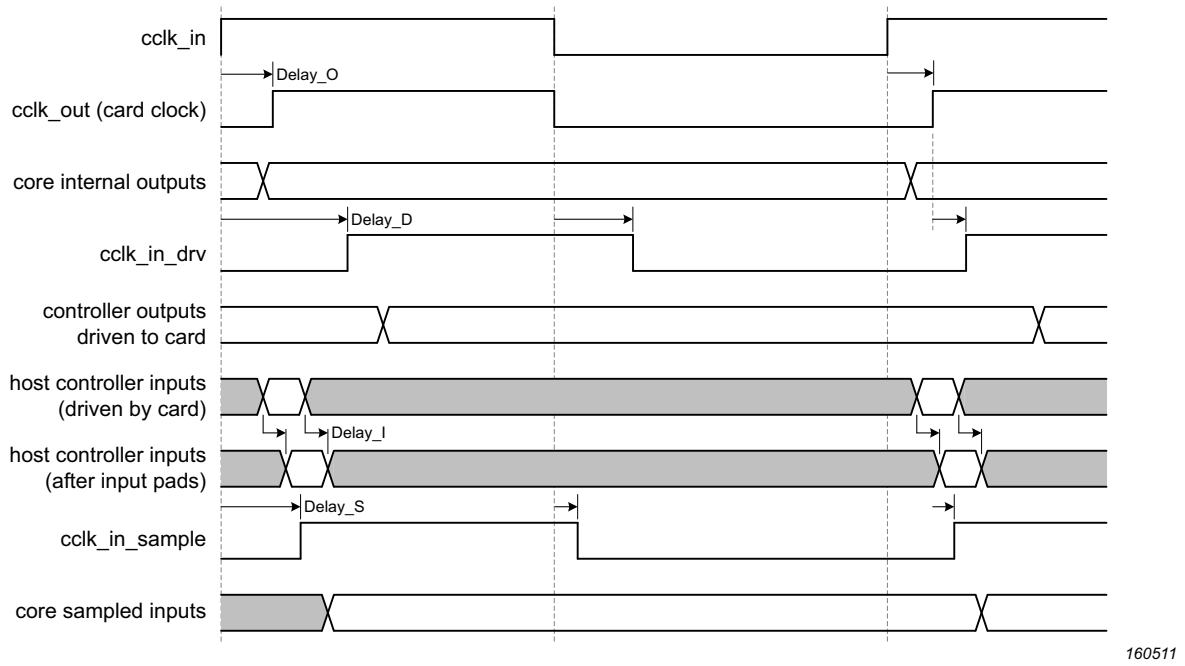


Fig 106. Clock relationship

30.9.2 Clock requirements and recommendations

There are several clock requirements and recommendations when interfacing the SDMMC/SDIO interface with cards. Table 605 lists timing requirements for the various speed modes.

Table 605. Timing requirements

Speed Mode	Max cclk_out Frequency		Min Hold Time	Min Setup Time	Min tODLY [1]	Max tODLY	
	MHz	ns				ns	UI
SDR25 (High Speed mode)	50	20	2.0	6.0	0	14.0	0.7

Table 605. Timing requirements

Speed Mode	Max cclk_out Frequency		Min Hold Time	Min Setup Time	Min tODLY [1]	Max tODLY	
	MHz	ns				ns	UI
SDR12	25	40	5.0	5.0	-	14.0	0.35
Identification Mode	0.4	2500	5.0	5.0	-	50.0	0.02
MMC High Speed (DAT and CMD)	50	20	3.0	3.0	-	13.7	0.685

[1] tODLY = card output delay

For more details, refer to Bus Timing sections of the SD and MMC specifications. The following subsections assume these SD and MMC bus timing requirements, and in order to meet these timing requirements, Delay_S and Delay_D must be implemented in the design.

30.9.2.1 Clock generation recommendations

The following are recommendations for clock generation:.

- The cclk_in_drv and cclk_in_sample clocks are phase-shifted versions of cclk_in. The value of the phase shift should be selectable, based on the speed modes. The phase shift can have a resolution of 90° or 45° with respect to the cclk_in clock period—the higher the resolution, the better it is. All clocks have a 50% duty cycle.

30.9.2.2 Clock phase-shift technique

Figure 107 illustrates a technique for achieving phase shifts on the clocks. The phase shifter provides phase shifts of 0°, 90°, 180°, and 270°.

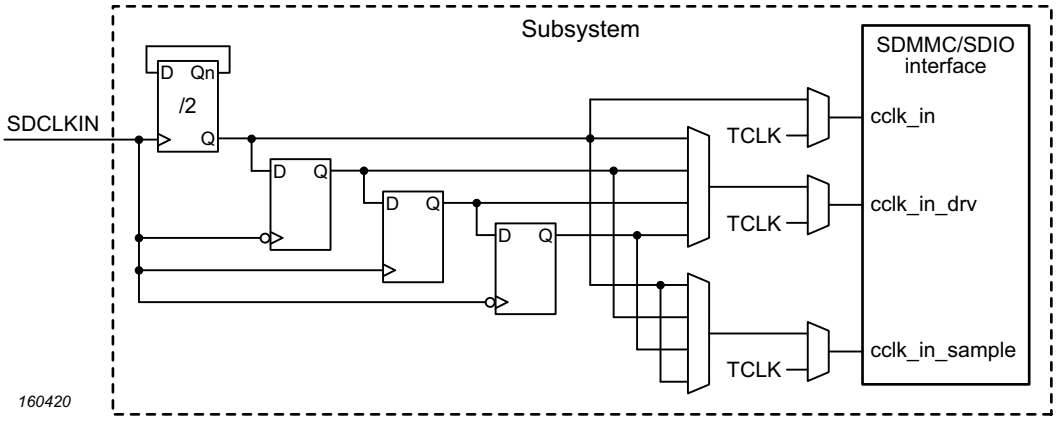


Fig 107. Clock phase-shift technique

Figure 108 shows the timing diagram for phase-shifted clocks.

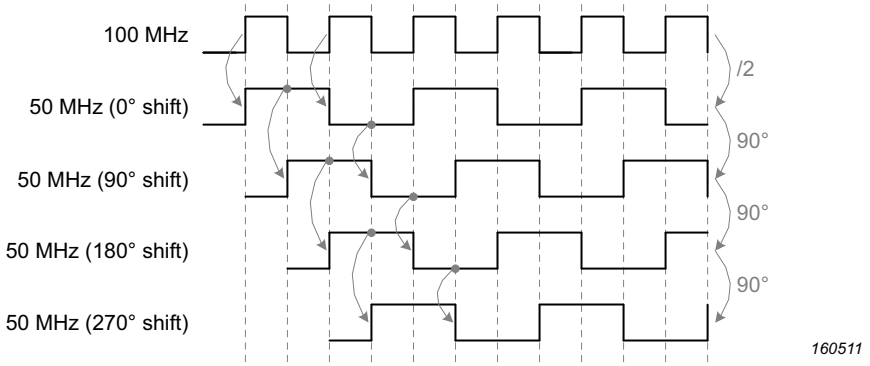


Fig 108. Timing diagram for phase-shifted clocks

30.9.2.3 SDIOCLKCTRL register

The SDIOCLKCTRL is a 32-bit SYSCON register that allows delay the SD/MMC internal input clock for both sampling of input data from the SD card and delay of data output (drive) from the LPC54xxx to the SD card. Sampling and drive delays are unique settings. See Section 7.5.76 for a detailed description of SDIOCLKCTRL.

Sample delay shifts the clocking of input data from the SD card into the SDIO interface.

The delays can be programmed in two ways.

1. The programmable delay is multiples of 250ps up to 31*250ps = 7.75ns. There are separate programmable delay settings for sample and drive delays each with a required enable bit.
2. Delay by phase shifts of 0, 90, 180, 270 of a 2X input clock. Setting the phase active bit means the input clock is 2X the actual SD output clock. If running the interface at 52MHz (26MB/s with a 4-bit interface), use a 104MHz input clock. 90 degree translates to 0.25/104e6 = 2.4ns delay. There are separate phase shift settings for sample and drive delays.

If either the programmable delay or phase shifts are enabled, SD commands must be sent with CMD register bit 29, USE_HOLD_REG set.

Programmable delays and phase shifts are additive when both are employed. If the input clock is 100MHz, programmable delay and phase shift is employed for the sample clock, and programmable delay is set to 10 and phase shift is set to 180 degrees, then the total delay is (10*250e-12) + (0.50/100e6) = 2.5ns + 5ns = 7.5ns.

30.9.3 Stop clock

Alternatively, you can avoid a stop-clock scenario by correctly enabling the Card Read Threshold feature and programming the Card Read Threshold Size—RX_WMARK and MSIZE; for details, refer to “Card Read Threshold Programming Sequence” on page 227.

For this method, it is recommended that the minimum FIFO size should be equal to the largest block size of a transfer that can be supported by the SDMMC/SDIO interface. For example, if the largest block size of the transfer that can be supported is 512 bytes and the FIFO width is 32 bits, the minimum FIFO size is 128 locations. Choosing a FIFO depth that is two or more times the maximum Block Size that the SDMMC/SDIO interface requires gives a greater performance and flexibility when choosing the Burst Size (MSIZE) and Rx Threshold (RX_WMARK). This also helps achieve best throughput.

31.1 How to read this chapter

The LPC546xx devices consist of two Smart Card interfaces (SCI0 and SCI1)

31.2 Features

- ISO 7816-3 compliant Smart Card interface.
- Data size of 8 bits.
- Parity generation and checking: odd and even.
- One or two stop bits.
- 16 byte Receive and Transmit FIFOs.
- Supports DMA for both transmit and receive.

31.3 Basic configuration

The SCI0/1 peripherals are configured using the following registers:

1. Clock: In the AHBCLKCTRL2 register (Section SYSCON), set the SCIx bit to enable the clock to the respective SCI block being used.

   Remark: The SCI blocks are disabled on reset (SCI0 and SCI1 = 0).
2. The main clock is the clock source used for the SCI. Use the SCICLKDIV0 and SCICLKDIV1 registers (See Section) to divide the main clock to reach the required SCI clock frequency if needed.
3. Pins: Select the SCI pins and pin modes through the relevant IOCON registers (See Chapter 10 “LPC546xx I/O pin configuration (IOCON)”.
4. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
5. Configure GPIO pins for SCI control (see section Section 31.5.10.2 “Smart Card connection”).
6. Configure SCI registers.

31.4 Pin description

Table 606. Smart Card interface pin description

Pin	Type	Description
SCI0_IO, SCI1_IO	Input/ Output	Serial Input/Output. Smart Card receive and transmit data.
SCI0_SCLK, SCI1_SCLK	Output	Serial Clock. Smart Card clock output.

31.5 Register description

Each Smart Card interface (SCI0 and SCI1) contains registers as shown in Table 607. The Divisor Latch Access Bit (DLAB) is contained in SCInLCR7 and enables access to the Divisor Latches.

Table 607. Register overview: SCI0/1 (SCI0 base address: 0x4003 6000, SCI1 base address 0x4003 7000)

Name	Access	Offset	Description	Reset value[1]	Section
RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read.	NA	31.5.1
THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here (DLAB = 0).	NA	31.5.2
DLL	R/W	0x000	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider (DLAB = 1).	0x01	31.5.3
DLM	R/W	0x004	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider (DLAB = 1).	0	31.5.3
IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential SCI interrupts (DLAB = 0).	0	31.5.4
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01	31.5.5
FCR	WO	0x008	FIFO Control Register. Controls SCI FIFO usage and modes.	0	31.5.6
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0	31.5.7
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60	31.5.8
SCR	R/W	0x01C	Scratch Pad Register. 8-bit temporary storage for software.	0	31.5.9
OSR	R/W	0x02C	Oversampling register. Controls the degree of oversampling during each bit time.	0xF0	31.5.10
SCICTRL	R/W	0x048	Smart Card Interface control register. Enables and configures the Smart Card Interface.	0	31.5.11

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

31.5.1 SCIn Receiver Buffer Register

The SCInRBR is the top byte of the SCIn Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with 0s.

The Divisor Latch Access Bit (DLAB) in LCR must be 0 in order to access the SCInRBR. The SCInRBR is always read-only.

Since PE, FE and BI bits correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the SCI0LSR register, and then to read a byte from the SCInRBR.

Table 608. SCIn Receiver Buffer Register when DLAB = 0, read only (RBR - address 0x4003 6000 (SCI0), 0x4003 7000 (SCI1)) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The SCIn Receiver Buffer Register contains the oldest received byte in the SCIn Rx FIFO.	Undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

31.5.2 SCIn Transmit Holding Register

The SCInTHR is the top byte of the SCIn TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in SCInLCR must be 0 in order to access the SCInTHR. The SCInTHR is always write-only.

Table 609. SCIn Transmit Holding Register when DLAB = 0, write only (THR - address 0x4003 6000 (SCI0), 0x4003 7000 (SCI1)) bit description

Bit	Symbol	Description	Reset Value
7:0	THR	Writing to the SCIn Transmit Holding Register causes the data to be stored in the SCIn transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA
31:8	-	Reserved. Read value is undefined, only 0 should be written.	NA

31.5.3 SCIn Divisor Latch LSB register

The SCIn Divisor Latch is part of the SCIn Baud Rate Generator and holds the value used, to divide the SCInCLK in order to produce the baud rate clock, which must be 16× the desired baud rate. The SCInDLL and SCInDLM registers together form a 16-bit divisor where SCInDLL contains the lower 8 bits of the divisor and SCInDLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by 0 is not allowed. The Divisor Latch Access Bit (DLAB) in SCInLCR must be one in order to access the SCIn Divisor Latches. See Section 31.5.3 for details on how to select the right value for SCInDLL and SCInDLM.

Table 610. SCIn Divisor Latch LSB register when DLAB = 1 (DLL - address 0x4003 6000 (SCI0), 0x4003 7000 (SCI1)) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The SCIn Divisor Latch LSB Register, along with the SCInDLM register, determines the baud rate of the SCIn.	0x01
31:8	-	Reserved. Read value is undefined, only 0 should be written.	NA

Table 611. SCIn Divisor Latch MSB register when DLAB = 1 (DLM - address 0x4003 6004 (SCI0), 0x4003 7004 (SCI1)) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The SCIn Divisor Latch MSB Register, along with the DLL register, determines the baud rate of the SCIn.	0
31:8	-	Reserved. Read value is undefined, only 0 should be written.	NA

31.5.4 SCIn Interrupt Enable Register

The SCInIER is used to enable the three SCIn interrupt sources.

Table 612. SCIn Interrupt Enable Register when DLAB = 0 (IER - address 0x4003 6004 (SCI0), 0x4003 7004 (SCI1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for SCIn. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THREIE		THRE Interrupt Enable. Enables the THRE interrupt for SCIn. The status of this can be read from SCInLSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXIE		RX Line Status Interrupt Enable. Enables the SCIn RX line status interrupts. The status of this interrupt can be read from SCInLSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
31:3	-		Reserved. Read value is undefined, only 0 should be written.	NA

31.5.5 SCIn Interrupt Identification Register

The SCInIIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an SCInIIR access. If an interrupt occurs during an SCInIIR access, the interrupt is recorded for the next SCInIIR access.

Table 613. SCIn Interrupt Identification Register, read only (IIR - address 0x4003 6008 (SCI0), 0x4003 7008 (SCI1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	INTSTATUS		Interrupt status. Note that SCInIIR[0] is active low. The pending interrupt can be determined by evaluating SCInIIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. SCInIER[3:1] identifies an interrupt corresponding to the SCIn Rx or TX FIFO. All other combinations of SCInIER[3:1] not listed below are reserved (000,100,101,111).	0
		0x1	3 - THRE Interrupt	
		0x2	2a - Receive Data Available (RDA).	
		0x3	1 - Receive Line Status (RLS).	
		0x6	2b - Character Time-out Indicator (CTI).	
5:4	-		Reserved. Read value is undefined, only 0 should be written.	NA
7:6	FIFOENABLE		Copies of SCInFCR[0].	0
31:8	-		Reserved. Read value is undefined, only 0 should be written.	NA

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be 0. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 614. Given the status of SCInIIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The SCInIIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The SCIn RLS interrupt (SCInIIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the SCIn Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The SCIn Rx error condition that set the interrupt can be observed via SCInLSR[4:1]. The interrupt is cleared upon an SCInLSR read.

The SCIn RDA interrupt (SCInIIR[3:1] = 010) shares the second level priority with the CTI interrupt (SCInIIR[3:1] = 110). The RDA is activated when the SCIn Rx FIFO reaches the trigger level defined in SCInFCR[7:6] and is reset when the SCIn Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (SCInIIR[3:1] = 110) is a second level interrupt and is set when the SCIn Rx FIFO contains at least one character and no SCIn Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any SCIn Rx FIFO activity (read or write of SCIn RSR) will clear the interrupt. This interrupt is intended to flush the SCIn RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would

receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 614. SCIn Interrupt Handling

IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	SCInLSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (SCInFCR0=1)	SCInRBR Read[3] or SCIn FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	SCInRBR Read[3]
0010	Third	THRE	THRE[2]	SCInIIR Read (if source of interrupt) or THR write[4]

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 31.5.8 “SCIn Line Status Register”

[3] For details see Section 31.5.1 “SCIn Receiver Buffer Register”

[4] For details see Section 31.5.5 “SCIn Interrupt Identification Register” and Section 31.5.2 “SCIn Transmit Holding Register”

The SCIn THRE interrupt (SCInIIR[3:1] = 001) is a third level interrupt and is activated when the SCIn THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the SCIn THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the SCInTHR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to SCInTHR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the SCIn THR FIFO has held two or more characters at one time and currently, the SCInTHR is empty. The THRE interrupt is reset when a SCInTHR write occurs or a read of the SCInIIR occurs and the THRE is the highest interrupt (SCInIIR[3:1] = 001).

31.5.6 SCIn FIFO Control Register

The write-only SCInFCR controls the operation of the SCIn Rx and TX FIFOs.

Table 615. SCIn FIFO Control Register, write only (FCR - address 0x4003 6008 (SCI0), 0x4003 7008 (SCI1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFOEN		FIFO Enable.	0
		0	SCIn FIFOs are disabled. Must not be used in the application.	
		1	Active high enable for both SCIn Rx and TX FIFOs and SCInFCR[7:1] access. This bit must be set for proper SCI operation. Any transition on this bit will automatically clear the related SCI FIFOs.	
1	RXFIFORES		RX FIFO Reset.	0
		0	No impact on either of SCIn FIFOs.	
		1	Writing a logic 1 to SCInFCR[1] will clear all bytes in SCIn Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset.	0
		0	No impact on either of SCIn FIFOs.	
		1	Writing a logic 1 to SCInFCR[2] will clear all bytes in SCIn TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMAMODE		DMA Mode Select. When the FIFO enable (bit 0 of this register) is set, this bit selects the DMA mode. See Section 31.5.6.1.	0
5:4	-		Reserved. Read value is undefined, only 0 should be written.	NA
7:6	RXTRIGLVL		RX Trigger Level. These two bits determine how many receiver SCIn FIFO characters must be written before an interrupt or DMA request is activated.	0
		0x0	Trigger level 0 (1 character or 0x01).	
		0x1	Trigger level 1 (4 characters or 0x04).	
		0x2	Trigger level 2 (8 characters or 0x08).	
		0x3	Trigger level 3 (14 characters or 0x0E).	
31:8	-		Reserved. Read value is undefined, only 0 should be written.	NA

31.5.6.1 DMA Operation

The user can optionally operate the SCI transmit and/or receive using DMA. The DMA mode is determined by the DMA Mode Select bit in the FCR register. This bit only has an affect when the FIFOs are enabled via the FIFO Enable bit in the FCR register.

SCI receiver DMA

In DMA mode, the receiver DMA request is asserted on the event of the receiver FIFO level becoming equal to or greater than trigger level, or if a character timeout occurs. See the description of the RX Trigger Level above. The receiver DMA request is cleared by the DMA controller.

SCI transmitter DMA

In DMA mode, the transmitter DMA request is asserted on the event of the transmitter FIFO transitioning to not full. The transmitter DMA request is cleared by the DMA controller.

31.5.7 SCIn Line Control Register

The SCInLCR determines the format of the data character that is to be transmitted or received.

Table 616. SCIn Line Control Register (LCR - address 0x4003 600C (SCI0), 0x4003 700C (SCI1)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WLS		Word Length Select.	0
		0x0	5-bit character length	
		0x1	6-bit character length	
		0x2	7-bit character length	
		0x3	8-bit character length	
2	SBS		Stop Bit Select	0
		0	1 stop bit.	
		1	2 stop bits (1.5 if SCInLCR[1:0]=00).	
3	PE		Parity Enable.	0
		0	Disable parity generation and checking.	
		1	Enable parity generation and checking.	
5:4	PS		Parity Select	0
		0x0	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	
		0x1	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		0x2	Forced 1 stick parity.	
		0x3	Forced 0 stick parity.	
6	-		Reserved	-
7	DLAB		Divisor Latch Access Bit	0
		0	Disable access to Divisor Latches.	
		1	Enable access to Divisor Latches.	
31:8	-		Reserved. Read value is undefined, only 0 should be written.	NA

31.5.8 SCIn Line Status Register

The SCInLSR is a read-only register that provides status information on the SCIn TX and RX blocks.

Table 617. SCIn Line Status Register (LSR - address 0x4003 6014 (SCI0), 0x4003 7014 (SCI1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready. SCInLSR[0] is set when the SCInRBR holds an unread character and is cleared when the SCIn RBR FIFO is empty.	0
		0	The SCIn receiver FIFO is empty.	
		1	The SCIn receiver FIFO is not empty.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. An SCInLSR read clears SCInLSR[1]. SCInLSR[1] is set when SCIn RSR has a new character assembled and the SCIn RBR FIFO is full. In this case, the SCIn RBR FIFO will not be overwritten and the character in the SCIn RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	PE		Parity Error. When the parity bit of a received character is in the wrong state, a parity error occurs. An SCInLSR read clears SCInLSR[2]. Time of parity error detection is dependent on SCInFCR[0]. **Note:** A parity error is associated with the character at the top of the SCIn RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	FE		Framing Error. When the stop bit of a received character is a logic 0, a framing error occurs. An SCInLSR read clears SCInLSR[3]. The time of the framing error detection is dependent on SCInFCR[0]. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the SCIn RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	-		Reserved	0
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty SCIn THR and is cleared on a SCInTHR write.	1
		0	SCInTHR contains valid data.	
		1	SCInTHR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both SCInTHR and SCInTSR are empty; TEMT is cleared when either the SCInTSR or the SCInTHR contain valid data.	1
		0	SCInTHR and/or the SCInTSR contains valid data.	
		1	SCInTHR and the SCInTSR are empty.	
7	RXFE		Error in RX FIFO. SCInLSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the SCInRBR. This bit is cleared when the SCInLSR register is read and there are no subsequent errors in the SCIn FIFO.	0
		0	SCInRBR contains no SCIn RX errors or SCInFCR[0]=0.	
		1	SCIn RBR contains at least one SCIn RX error.	
31:8	-		Reserved. The value read from a reserved bit is not defined.	NA

31.5.9 SCIn Scratch Pad Register

The SCInSCR has no effect on the SCIn operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the SCInSCR has occurred.

Table 618. SCIn Scratch Pad Register (SCR - address 0x4003 601C (SCI0), 0x4003 701C (SCI1)) bit description

Bit	Symbol	Description	Reset Value
7:0	PAD	A readable, writable byte.	0
31:8	-	Reserved. Read value is undefined, only 0 should be written.	NA

31.5.10 SCIn Oversampling Register

In most applications, the Smart Card interface samples received data 16 times in each nominal bit time, and sends bits that are 16 input clocks wide. This register allows software to control the ratio between the input clock and bit clock. This is required for Smart Card mode, and provides an alternative to fractional division for other modes.

Table 619. SCIn Oversampling Register (OSR - address 0x4003 602C (SCI0) address 0x4003 702C (SCI1)) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only 0 should be written.	NA
3:1	OSFRAC	Fractional part of the oversampling ratio, in units of 1/8th of an input clock period. (001 = 0.125, …, 111 = 0.875)	0
7:4	OSINT	Integer part of the oversampling ratio, minus 1. The reset values equate to the normal operating mode of 16 input clocks per bit time.	0xF
14:8	FDINT	These bits act as a more-significant extension of the OSint field, allowing an oversampling ratio up to 2048 as required by ISO7816-3. In Smart Card mode, bits 14:4 should initially be set to 371, yielding an oversampling ratio of 372.	0
31:15	-	Reserved. Read value is undefined, only 0 should be written.	NA

Example: For a baud rate of 3.25Mbps with a 24 MHz clock frequency, the ideal oversampling ratio is 24/3.25 or 7.3846. Setting OSInt to 0110 for 7 clocks/bit and OSFrac to 011 for 0.375 clocks/bit, results in an oversampling ratio of 7.375.

OSInt is extended by FDInt. This extends the possible oversampling to 2048, as required to support ISO 7816-3. Note that this value can be exceeded when D<0. The initial value of OSInt and FDInt should be programmed as "00101110011" (372 minus one).

31.5.10.1 SCIn Control register

This register enables the Smart Card interface and configures to be used in ISO7816-3 compliant asynchronous Smart Card applications.

Table 620. SCIn Smart Card Interface Control register (SCICTRL - address 0x4003 6048 (SCI0), address 0x4003 7048 (SCI1)) bit description

Bit	Symbol	Value	Description	Reset value
0	SCIEN		Smart Card Interface Enable.	0
		0	Smart Card interface disabled.	
		1	Asynchronous half duplex Smart Card interface is enabled.	

Table 620. SCIn Smart Card Interface Control register (SCICTRL - address 0x4003 6048 (SCI0), address 0x4003 7048 (SCI1)) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
1	NACKDIS		NACK response disable. Only applicable in T=0.	0
		0	A NACK response is enabled.	
		1	A NACK response is inhibited.	
2	PROTSEL		Protocol selection as defined in the ISO7816-3 standard.	0
		0	T = 0	
		1	T = 1	
4:3	-	-	Reserved	
7:5	TXRETRY		Maximum number of retransmissions in case of a negative acknowledge (protocol T=0). When the retry counter is exceeded, the USART will be locked until the FIFO is cleared. A TX error interrupt is generated when enabled.	0x0
15:8	GUARDTIME		Extra guard time. See Figure 110 “Smart Card T = 0 waveform”. No extra guard time (0x0) results in a standard guard time as defined in ISO 7816-3, depending on the protocol type. A guard time of 0xFF indicates a minimal guard time as defined for the selected protocol.	0x0
31:16	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

31.5.10.2 Smart Card connection

When the SCIEN bit in the SCInSCICTRL register is set as described above, the Smart Card provides bidirectional serial data on the SCIn_IO pin. If the SCIn_CLK function is enabled in the I/O Configuration block, a serial clock is output on the pin: use of such a clock is optional for Smart Cards. Software must use timers to implement character and block waiting times (no hardware support via trigger signals is provided). GPIO pins can be used to control the Smart Card reset and power pins. Any power supplied to the card must be externally switched as card power supply requirements often exceed source currents possible on the LPC546xx. As the specific application may accommodate any of the available ISO 7816 class A, B, or C power requirements, be aware of the logic level tolerances and requirements when communicating or powering cards that use different power rails than the LPC546xx.

Figure 109 shows a typical asynchronous Smart Card application.

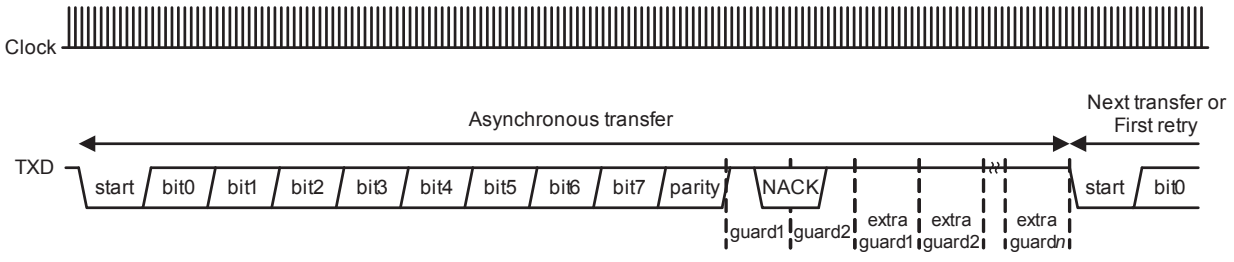


Fig 109. Typical Smart Card application

31.5.11 Smart Card set-up procedure

A T = 0 protocol transfer consists of 8-bits of data, an even parity bit, and two guard bits that allow for the receiver of the particular transfer to flag parity errors through the NACK response (see Figure 110). Extra guard bits may be added according to card requirements. If no NACK is sent (provided the interface accepts them in SCICTRL), the next byte may be transmitted immediately after the last guard bit. If the NACK is sent, the transmitter will retry sending the byte until successfully received or until the SCICTRL retry limit has been met.

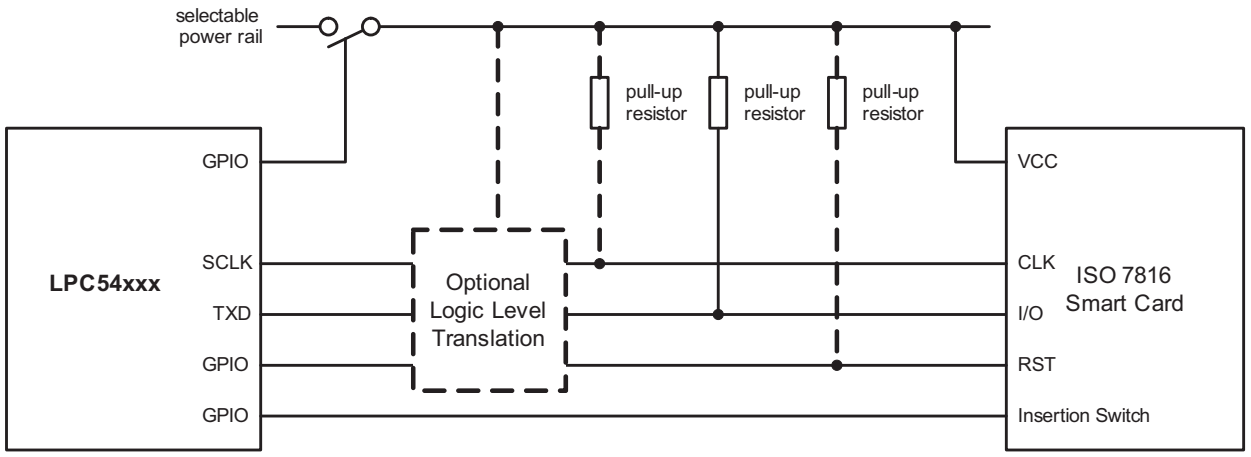


Fig 110. Smart Card T = 0 waveform

The Smart Card must be set up with the following considerations:

- If necessary, program PRESETCTRL2 (Table 131) so that the Smart Card is not continuously reset. Program PRESETCTRL2 so that the Smart Card is not in reset.
- Program AHBCLKCTRL2 (Table 141) to enable the Smart Card clock.
- Program IOCON register to enable SCIn_IO function. Program IOCON to attach three GPIO pins as power and reset outputs, and as Smart Card insert input.
- Program IOCON register to select the SCIn_SCLK function.
- Program SCInCLKDIV (Table 169) for an initial frequency of 3.58 MHz.
- Program the OSR (Section 31.5.10) for 372x oversampling.
- Program DLM and DLL (Section 31.5.3) to produce a 9600 bit per second communication speed.
- Program the LCR (Section 31.5.7) for 8-bit characters, parity enabled, even parity.
- Program the GPIO signals associated with the Smart Card so that (in this order):
  a. Reset is HIGH.
  b. VCC is LOW.
- Program SCICTRL (Section 31.5.10.1) to enable the Smart Card with the desired options.
- Set up one or more timer(s) to provide timing as needed for ISO 7816 startup.
- Poll GPIO attached to Smart Card insert signal.

- Power Smart Card (VCC) via GPIO attached to Smart Card power.
- Wait for clock to settle.
- Reset LOW GPIO attached to Smart Card reset.
- Reset HIGH GPIO attached to Smart Card reset.
- Capture Answer-to-Reset response from Smart Card.

32.1 How to read this chapter

The SPI flash interface is available on all LPC546xx devices.

32.2 Features

- Quad SPI Flash Interface (SPIFI) interface to external flash.
- Transfer rates of up to SPIFI_CLK/2 bytes per second.
- Code in the serial flash memory can be executed as if it was in the CPU's internal memory space. This is accomplished by mapping the external flash memory directly into the CPU memory space.
- Supports 1-, 2-, and 4-bit bidirectional serial protocols.
- Half-duplex protocol compatible with various vendors and devices.

32.3 Basic configuration

Initial configuration of the SPIFI peripheral is accomplished as follows:

1. Power: In the AHBCLKCTRL0 register (see Section 7.5.19), set bit SPIFI.

   Remark: On reset, the SPIFI is disabled.
2. SPIFI clock: set up the SPIFI clock using the SPIFICLKSEL and SPIFICLKDIV registers (see Section 7.5.33 and Section 7.5.53).
3. Pins: Select SPIFI pins and pin modes through the relevant IOCON registers (see Chapter 10 "LPC546xx I/O pin configuration (IOCON)").

32.4 General description

The SPI Flash Interface (SPIFI) allows low-cost serial flash memories to be connected to the CPU with little performance penalty compared to parallel flash devices with higher pin count.

Many serial flash devices use a half-duplex command-driven SPI protocol for device setup and initialization. Quad devices then use a half-duplex, command-driven 4-bit protocol for normal operation. Different serial flash vendors and devices accept or require different commands and command formats. SPIFI provides sufficient flexibility to be compatible with common flash devices, and includes extensions to help insure compatibility with future devices.

Serial flash devices respond to commands sent by software or automatically sent by the SPIFI when software reads either of the two read-only serial flash regions in the memory map (see Table 621).

Table 621. SPIFI flash memory map

Memory	Address
SPIFI data	0x1000 0000 to 0x17FF FFFF **Remark:** This is the address space allocated to the SPIFI for direct execution. The area allocated allows a maximum of 128 MB of SPI flash to be mapped into the CPU memory space. In practice, the usable space is limited to the size of the connected device.

32.5 Pin description

Table 622. SPIFI Pin description

Pin function	Type	Description
SPIFI_SCK	O	Serial clock for the flash memory, switched only during active bits on the MOSI/IO0, MISO/IO1, and IO3:2 lines.
SPIFI_CSN	O	Chip select for the flash memory, driven low while a command is in progress, and high between commands. In the typical case of one serial slave, this signal can be connected directly to the device. If more than one serial slave is connected, software and off-chip hardware should use general-purpose I/O signals in combination with this signal to generate the chip selects for the various slaves.
SPIFI_IO0 / SPIFI_MOSI	I/O	This is an output except in quad/dual input data fields. After a quad/dual input data field, it becomes an output again one serial clock period after CS goes high.
SPIFI__IO1 / SPIFI_MISO	I/O	This is an output in quad/dual opcode, address, intermediate, and output data fields, and an input in SPI mode and in quad/dual input data fields. After an input data field in quad/dual mode, it becomes an output again one serial clock period after CS goes high.
SPIFI_IO[3:2]	I/O	These are outputs in quad opcode, address, intermediate, and output data fields, and inputs in quad input data fields. If the flash memory does not have quad capability, these pins can be assigned to GPIO or other functions.

32.6 Supported devices

Multiple QSPI devices from various vendors can be used with the SPIFI interface.

32.7 SPIFI hardware

The SPIFI has a base address for the registers and a base address for the memory area in which the serial Flash connected to the SPIFI can be read.

The first operation with the serial Flash is Read JEDEC ID, which is implemented by most serial Flash devices. Depending on the device identity code returned by the serial Flash in this operation, device-specific commands are used for further operation. Programming and other operations on the serial Flash can be performed using the register interface.

32.8 Register description

The SPIFI register interface supports word accesses.

Table 623. Register overview: SPIFI (base address 0x4008 0000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	SPIFI control register	0x400F FFFF	32.8.1
CMD	R/W	0x004	SPIFI command register	0x0	32.8.2
ADDR	R/W	0x008	SPIFI address register	0x0	32.8.3
IDATA	R/W	0x00C	SPIFI intermediate data register	0x0	32.8.4
CLIMIT	R/W	0x010	SPIFI limit register	0x0800 0000	32.8.5
DATA	R/W	0x014	SPIFI data register	0x0	32.8.6
MCMD	R/W	0x018	SPIFI memory command register	0x0	32.8.7
STAT	R/W	0x01C	SPIFI status register	0x0200 0000	32.8.8

32.8.1 SPIFI control register

The SPIFI control register controls the overall operation of the SPIFI and should be written before any commands are initiated.

Table 624. SPIFI control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
15:0	TIMEOUT	-	This field contains the number of serial clock periods without the processor reading data in memory mode, which will cause the SPIFI hardware to terminate the command by driving the CS pin high and negating the CMD bit in the Status register. (This allows the flash memory to enter a lower-power state.) If the processor reads data from the flash region after a time-out, the command in the Memory Command Register is issued again.	0xFFFF
19:16	CSHIGH	-	This field controls the minimum $\overline{CS}$ high time, expressed as a number of serial clock periods minus one.	0xF
20	-	-	Reserved.	-

Table 624. SPIFI control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
21	D_PRFTCH_DIS	-	This bit allows conditioning of memory mode prefetches based on the AHB HPROT (instruction/data) access information. A 1 in this register means that the SPIFI will not attempt a speculative prefetch when it encounters data accesses.	0
22	INTEN	-	If this bit is 1 when a command ends, the SPIFI will assert its interrupt request output. See INTRQ in the status register for further details.	0
23	MODE3		SPI Mode 3 select.	0
		0	SCK LOW. The SPIFI drives SCK low after the rising edge at which the last bit of each command is captured, and keeps it low while CS is HIGH.	
		1	SCK HIGH. the SPIFI keeps SCK high after the rising edge for the last bit of each command and while CS is HIGH, and drives it low after it drives CS LOW. (Known serial flash devices can handle either mode, but some devices may require a particular mode for proper operation.) **Remark:** MODE3, RFCLK, and FBCLK should not all be 1, because in this case there is no final falling edge on SCK on which to sample the last data bit of the frame.	
26:24	-	-	Reserved.	-
27	PRFTCH_DIS		Cache prefetching enable. The SPIFI includes an internal cache. A 1 in this bit disables prefetching of cache lines.	0
		0	Enable. Cache prefetching enabled.	
		1	Disable. Disables prefetching of cache lines.	
28	DUAL		Select dual protocol.	0
		0	Quad protocol. This protocol uses IO3:0.	
		1	Dual protocol. This protocol uses IO1:0.	
29	RFCLK		Select active clock edge for input data.	0
		0	Rising edge. Read data is sampled on rising edges on the clock, as in classic SPI operation.	
		1	Falling edge. Read data is sampled on falling edges of the clock, allowing a full serial clock of of time in order to maximize the serial clock frequency. **Remark:** MODE3, RFCLK, and FBCLK should not all be 1, because in this case there is no final falling edge on SCK on which to sample the last data bit of the frame.	
30	FBCLK		Feedback clock select.	1
		0	Internal clock. The SPIFI samples read data using an internal clock.	
		1	Feedback clock. Read data is sampled using a feedback clock from the SCK pin. This allows slightly more time for each received bit. **Remark:** MODE3, RFCLK, and FBCLK should not all be 1, because in this case there is no final falling edge on SCK on which to sample the last data bit of the frame.	
31	DMAEN	-	A 1 in this bit enables the DMA Request output from the SPIFI. Set this bit only when a DMA channel is used to transfer data in peripheral mode. Do not set this bit when a DMA channel is used for memory-to-memory transfers from the SPIFI memory area. DMAEN should only be used in Command mode.	0

32.8.2 SPIFI command register

The Command Register may only be written as a word, but bytes, halfwords, and words may be read from it. It may be written to when the CMD and MCINIT bits in the Status register are 0, and under these circumstances writing initiates the transmission of a new command. For a command that contains an address and/or intermediate data, software should write to the Address and/or Intermediate Data registers, before writing to this register. If the command contains output data, software should write it to the Data Register after writing to this register. If the command contains input data, software can read it from the Data Register after writing to this register.

Table 625. SPIFI command register (CMD, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
13:0	DATALEN	-	Except when the POLL bit in this register is 1, this field controls how many data bytes are in the command. 0 indicates that the command does not contain a data field.	0
14	POLL	-	This bit should be written as 1 only with an opcode that a) contains an input data field, and b) causes the serial flash device to return byte status repetitively (e.g., a Read Status command). When this bit is 1, the SPIFI hardware continues to read bytes until the test specified by the DATALEN field is met. The hardware tests the bit in each status byte selected by DATALEN bits 2:0, until a bit is found that is equal to DATALEN bit 3. When the test succeeds, the SPIFI captures the byte that meets this test so that it can be read from the Data Register, and terminates the command by raising $\overline{CS}$. The end-of-command interrupt can be enabled to inform software when this occurs	0
15	DOUT		If the DATALEN field is not zero, this bit controls the direction of the data:	0
		0	Input from serial flash.	
		1	Output to serial flash.	
18:16	INTLEN	-	This field controls how many intermediate bytes precede the data. (Each such byte may require 8 or 2 SCK cycles, depending on whether the intermediate field is in serial, 2-bit, or 4-bit format.) Intermediate bytes are output by the SPIFI, and include post-address control information, dummy and delay bytes. See the description of the Intermediate Data register for the contents of such bytes.	0
20:19	FIELDFORM		This field controls how the fields of the command are sent.	0
		0x0	All serial. All fields of the command are serial.	
		0x1	Quad/dual data. Data field is quad/dual, other fields are serial.	
		0x2	Serial opcode. Opcode field is serial. Other fields are quad/dual.	
		0x3	All quad/dual. All fields of the command are in quad/dual format.	
23:21	FRAMEFORM		This field controls the opcode and address fields.	0
		0x0	Reserved.	
		0x1	Opcode. Opcode only, no address.	
		0x2	Opcode one byte. Opcode, least significant byte of address.	
		0x3	Opcode two bytes. Opcode, two least significant bytes of address.	
		0x4	Opcode three bytes. Opcode, three least significant bytes of address.	
		0x5	Opcode four bytes. Opcode, 4 bytes of address.	
		0x6	No opcode three bytes. No opcode, 3 least significant bytes of address.	
		0x7	No opcode four bytes. No opcode, 4 bytes of address.	
31:24	OPCODE	-	The opcode of the command (not used for some FRAMEFORM values).	0

32.8.3 SPIFI address register

Before writing a command that includes an address field to the Command register, software should write the address to this register. The most significant byte of the address is sent first.

Table 626. SPIFI address register (ADDR, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	ADDRESS	Address.	0

32.8.4 SPIFI intermediate data register

Before writing a command to the Command register that requires specific intermediate byte values, software should write the value of the bytes to this register. The least significant byte of this register is sent first. If more than four intermediate bytes are specified in the Command register, zeroes are sent after the 4th byte.

The main use of this register with current serial flash devices is to select the no-opcode mode (continuous read, code execution) using the byte value 0xA5, and cancelling this mode using 0xFF.

Many devices that require dummy (delay) bytes don't care about their contents, in which case this register need not be written.

Table 627. SPIFI intermediate data register (IDATA, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
31:0	IDATA	Value of intermediate bytes.	0

32.8.5 SPIFI cache limit register

The SPIFI hardware includes caching of previously-accessed data to improve performance. Software can write an address within the device to this register, to prevent such caching at and above that address. After Reset this register contains the allocated size of the SPIFI memory area, so that all possible accesses are below that value and are thus cacheable.

Table 628. SPIFI cache limit register (CLIMIT, offset 0x010) bit description

Bit	Symbol	Description	Reset value
31:0	CLIMIT	Zero-based upper limit of cacheable memory	0x0800 0000

32.8.6 SPIFI data register

After initiating a command that includes a data output field by writing to the Command Register, software should write output data to this register. Store Byte instructions provide one data byte, Store Halfword instructions provide two bytes, and Store Word instructions provide 4 bytes of output data. Store commands are waited if the FIFO is too full to accept the number of bytes being stored. For Store Halfword and Store Word, the least significant byte is sent first.

After initiating a command that includes a data input field by writing to the Command Register, software should read input data from this register. Load Byte instructions deliver one data byte to software, Load Halfword instructions deliver two bytes, and Load Word

instructions deliver 4 bytes of input data. Load commands are waited if a command is in progress and the FIFO does not contain the number of bytes being loaded. For Load Halfword and Load Word commands, the least significant byte is received first.

DATALEN bytes should be read from or written to this register. If such a (read or write) command needs to be terminated before that time, software should write a 1 to the RESET bit in the Status register to accomplish this. If software attempts to read or write more data than was specified in DATALEN, a Data Abort exception will occur.

In polling mode (see the POLL bit in the SPIFI command register), one byte must be read from this register because the poll mechanism writes the matching byte.

This register is not used for commands initiated by reading the flash address range in the memory map. In DMA transfers in peripheral to-or-from-memory mode, the address of this register should be used as the peripheral address.

Table 629. SPIFI Data register (DATA, offset 0x014) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	Input or output data	0

32.8.7 SPIFI memory command register

Before accessing the flash area of the memory map, software should set up the device. After optionally writing to the Intermediate Data register, software should write a word to this register to define the command that is used to read data. Thereafter data can be read from the flash memory area, either directly or by means of a DMA channel.

Writing to this register will be ignored when a command is in progress or while data has yet to be written or read from the FIFO for a command issued. Use the MCINIT bit of the Status register to verify that the hardware is in Memory mode. A successful write to this register sets the SPIFI into Memory mode. The content of this register is identical to that of the Command Register, except for the DATALEN (not used), POLL, DOUT, and FRAMEFORM bits.

Table 630. SPIFI memory command register (MCMD, offset 0x018) bit description

Bit	Symbol	Value	Description	Reset value
13:0	-	-	Reserved.	0
14	POLL	-	This bit should be written as 0.	0
15	DOUT	-	This bit should be written as 0.	0
18:16	INTLEN	-	This field controls how many intermediate bytes precede the data. (Each such byte may require 8 or 2 SCK cycles, depending on whether the intermediate field is in serial, 2-bit, or 4-bit format.) Intermediate bytes are output by the SPIFI, and include post-address control information, dummy and delay bytes. See the description of the Intermediate Data register for the contents of such bytes.	0
20:19	FIELDFORM		This field controls how the fields of the command are sent.	0
		0x0	All serial. All fields of the command are serial.	
		0x1	Quad/dual data. Data field is quad/dual, other fields are serial.	
		0x2	Serial opcode. Opcode field is serial. Other fields are quad/dual.	
		0x3	All quad/dual. All fields of the command are in quad/dual format.	

Table 630. SPIFI memory command register (MCMD, offset 0x018) bit description

Bit	Symbol	Value	Description	Reset value
23:21	FRAMEFORM		This field controls the opcode and address fields.	0
		0x0	Reserved.	
		0x1	Opcode. Opcode only, no address.	
		0x2	Opcode one byte. Opcode, least-significant byte of address.	
		0x3	Opcode two bytes. Opcode, 2 least-significant bytes of address.	
		0x4	Opcode three bytes. Opcode, 3 least-significant bytes of address.	
		0x5	Opcode four bytes. Opcode, 4 bytes of address.	
		0x6	No opcode three bytes. No opcode, 3 least-significant bytes of address.	
		0x7	No opcode, 4 bytes of address.	
31:24	OPCODE	-	The opcode of the command (not used for some FRAMEFORM values).	0

32.8.8 SPIFI status register

This register indicates the state of the SPIFI.

Table 631. SPIFI status register (STAT, offset 0x01C) bit description

Bit	Symbol	Description	Reset value
0	MCINIT	This bit is set when software successfully writes the Memory Command register, and is cleared by Reset or by writing a 1 to the RESET bit in this register.	0
1	CMD	This bit is 1 when the Command register is written. It is cleared by a hardware reset, a write to the RESET bit in this register, or the deassertion of $\overline{CS}$ which indicates that the command has completed communication with the SPI Flash.	0
3:2	-	Reserved	0
4	RESET	Write a 1 to this bit to abort a current command or memory mode. This bit is cleared when the hardware is ready for a new command to be written to the Command register.	
5	INTRQ	This bit reflects the SPIFI interrupt request. Write a 1 to this bit to clear it. This bit is set when a CMD was previously 1 and has been cleared due to the deassertion of $\overline{CS}$.	0
31:6	-	Reserved	0

32.9 Functional description

32.9.1 Data transfer

Serial SPI uses the signals SPIFI_SCK, SPIFI_CSN, SPIFI_MISO, and SPIFI_MOSI, while quad mode adds the two IO signals SPIFI_IO[3:2].

The SPIFI implements basic, dual, and quad SPI in half-duplex mode, in which the SPIFI always sends a command to a serial flash memory at the start of each frame. (A frame is the sequence of bytes transmitted during one period with $\overline{CS}$ LOW.) In general, commands start with an opcode byte although some serial flashes allow a no-opcode mode in which commands start with the address to be read. In write commands, the SPIFI sends all of the data in the frame, while in read commands, the SPIFI sends the command, and then the serial flash sends data to the SPIFI.

Classic SPI includes four modes (mode 0 to mode 3), of which the SPIFI and most serial flashes implement modes 0 and 3. In mode 0, the SCK line is LOW between frames while in mode 3 it is HIGH. In mode 0, the SPIFI drives the first data bits from the time that it drives $\overline{CS}$ LOW, and drives the rest of the data on falling edges of SCK. In mode 3, the SPIFI drives SCK LOW one-half clock period after it drives $\overline{CS}$ LOW, and drives data on the falling edge of SCK. In either mode the serial flash samples the data on the rising edges of SCK.

The same scheme (transmitter changes data on falling edges of SCK, receiver samples data on rising edges) is maintained for the entire frame, including read data sent by the serial flash to the SPIFI.

The SPI protocol avoids all issues of set-up and hold times between the clock and data lines by using half of the SCK period to transmit the data. For high clock speeds, it is necessary to sample read data using a feedback clock. The FBCLK bit enables the feedback clock from the SCK pad sampling method. This provides the best possible timing margin for both read and write data under the opposite-edge scheme.

But maximizing clock frequency is of such importance that further improvement is sometimes needed, by means of using the whole serial clock period to transmit data. This choice is enabled for read data by setting the RFCLK bit. When this bit is 1, the SPIFI samples data on the falling edge of the serial clock that follows the rising edge which is normally used. RFCLK and FBCLK and MODE3 should not all be 1 because in this case there would be no falling edge of the feedback clock to capture the last bit of a frame.

Consult the data sheet of the serial flash device to be used for the formats of the commands that it supports. Figure 111 shows commands consisting of an opcode field only, sent in SPI and quad modes. All fields are multiples of 8 bits long. Bytes are sent with the most significant bit first in SPI mode, and the most significant 4 bits first in quad mode.

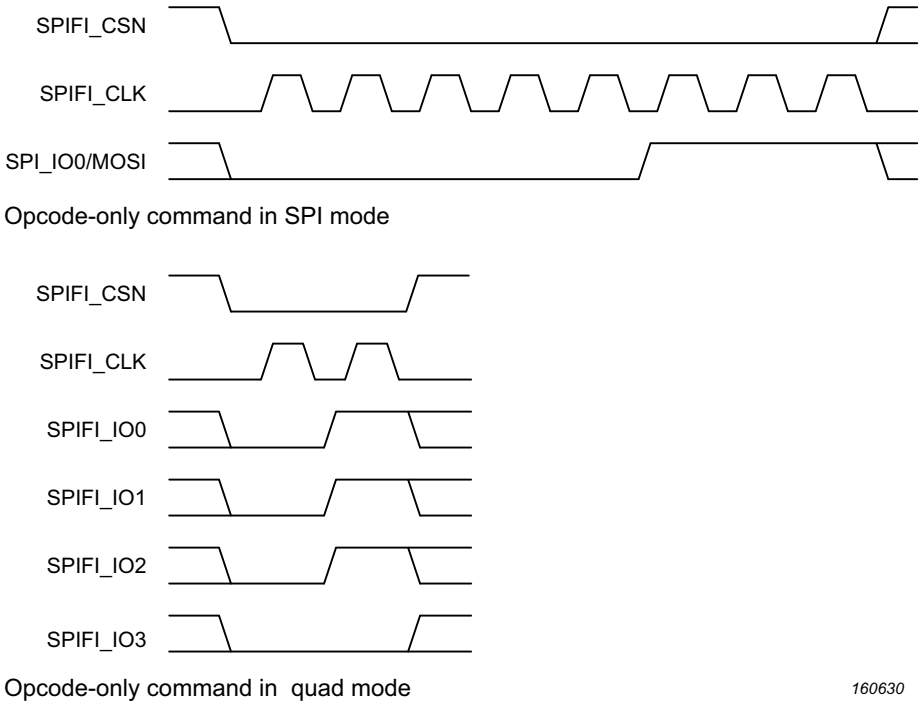


Fig 111. Opcode only commands

Figure 112 shows a command that reads 1 byte from the slave in SPI mode and a command that reads 3 bytes from the slave with the opcode and input data fields both in quad mode.

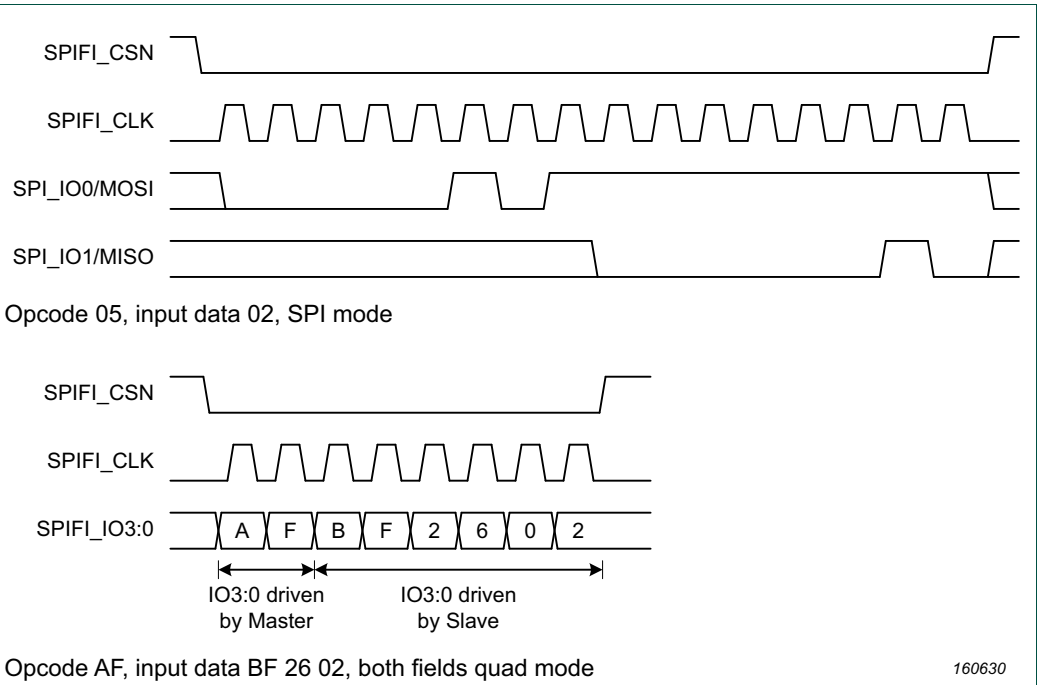


Fig 112. Read commands

In quad mode, the IO3:0 lines are driven by the SPIFI in opcode, address, intermediate and output data fields, and driven by the flash memory in input data fields. In address fields the more significant bytes are sent first.

32.9.2 Software requirements and capabilities

During device set-up, software should initialize the external serial flash device using those commands that place it in its highest-performance mode. When this sequence is complete, software should write the command that will be issued in response to a read from the serial flash region of the memory map, to the Memory Command Register. If software attempts to read the flash region after Reset, power-up, or writing the Command Register without writing the Memory Command Register thereafter, the SPIFI responds with an Abort error.

After writing the Memory Command Register, the contents of the flash would appear to the system as memory mapped. This enables data access or execution from the serial flash over AHB.

SPIFI has two operational modes:

1. Memory Mode - whereby the contents of the FLASH are memory mapped in the chip.
2. Command Mode - whereby the user can manually construct command sequences for the flash.

SPIFI cannot switch over from Memory Mode to Command mode and vice versa without writing 1 to the RESET bit in the SPIFI Status Register and polling until it is cleared by hardware to ensure that the current mode has been aborted.

Because the SPIFI is an AHB device, software or a DMA channel can read bytes, halfwords, or words from the flash region.

Reads from the flash region are delayed by deasserting HREADY when necessary, until the requested bytes are available to be read.

In Memory mode, SPIFI prefetches sequential addresses in order to improve performance.

If no AHB accesses have taken place for a period specified by the time-out (TO) field in the Control register, the SPIFI will deassert $\overline{CS}$. Once a new access occurs that requires a new fetch of data, the SPIFI will reassert $\overline{CS}$ and send a new command to fetch the required data. This is done in order to save power in the SPI flash device.

If software reads or writes more data from the Data Register than was configured in the DATALEN field of the Command register or reads or writes when no command was issued, the SPIFI hardware issues an abort exception.

When the serial flash needs to be programmed or erased, software should not write to the flash region of the address map. Instead, it should write the appropriate sequence of commands to the Command, Address, and Data registers. When an actual erase or program operation is under way in the serial flash device, software should write a Read Status command (with the POLL bit set) to the Command register. Thereafter:

- If INTEN in the Control register is 1, the SPIFI will interrupt the processor when the erase or write operation (and thus the Read Status command) completes.

- If not, software can continually or periodically read the Status register until it indicates that the Read Status command is complete.

When erasing or programming completes, software can do further programming or erasing, or return to normal (memory mode) operation.

32.9.3 Peripheral mode DMA operation

The SPIFI inserts wait states when necessary during read and write operations by the core to maintain synchronization between core accesses and serial data transfer with the serial flash. This mechanism is all that is needed for load and store accesses and for memory-to-memory transfers by a DMA channel.

The peripheral mode is a mode that supports DMA transfers in which the SPIFI acts as a peripheral and drives a request signal to the DMA channel to control data transfer. This mode does not necessarily move data faster than memory-to-memory operation, but it may be advantageous in systems in which software controls dynamic transfer of code and/or data between the serial flash and RAM on an as-needed basis. The advantage is that clock cycles are not lost to wait states, and thus the overall operation of the AHB is more efficient.

The DMA controller should be programmed to present word operations at the fixed address of the Data Register to have a burst size of one transfer. The SPIFI drives the DMA request to the DMA controller.

To use this mode, software should write the Command register to start the command and program a DMA channel as described above to transfer data between the Data register and RAM. The SPIFI asserts the DMA request when:

- DMAEN in the Control register is 1.
- MCINIT is 0.
- There are at least 4 bytes in the FIFO for a read operation, or at least 4 empty byte locations in the FIFO for a write/program operation.

33.1 How to read this chapter

This chapter describes the external memory controller for devices that support external memory. EMC configurations vary with different packages for devices that support external memory, see Table 632.

Table 632. EMC configuration for 180-pin

Device package	Data bus widths supported	Pins available	Dynamic memory configuration registers[1][2]	Static memory configuration registers[1][3]	External memory connections
180-pin	16-bit, 8-bit	EMC_A[20:0] EMC_D[15:0] EMC_OE EMC_WE EMC_BLS1:0 EMC_CS3:0 EMC_DYCS3:0 EMC_CAS EMC_RAS EMC_CLK1:0 EMC_FBCK EMC_CKE3:0 EMC_DQM1:0	EMCDynamicConfig1/0 EMCDynamicRasCas1/0	EMCStaticConfig1/0 EMCStaticWaitWen1/0 EMCStaticWaitOen1/0 EMCStaticWaitRd1/0 EMCStaticWaitPage1/0 EMCStaticWaitWr1/0 EMCStaticWaitTurn1/0	Section 33.14.2 Section 33.14.3

[1] In addition to the registers that are common to all EMC operations: EMCControl and EMCConfig.

[2] In addition to the registers that are common to all EMC dynamic chip selects: EMCDynamicControl, EMCDynamicRefresh, EMCDynamicReadConfig, EMCDynamicRP, EMCDynamicRAS, EMCDynamicSREX, EMCDynamicAPR, EMCDynamicDAL, EMCDynamicWR, EMCDynamicRC, EMCDynamicRFC, EMCDynamicXSR, EMCDynamicRRD, and EMCDynamicMRD

[3] In addition to the EMCStaticExtendedWait register which applies to all static chip selects.

Table 633. EMC configuration for 208-pin

Device package	Data bus widths supported	Pins available	Dynamic memory configuration registers[1][2]	Static memory configuration registers[1][3]	External memory connections
208-pin	32-bit, 16-bit, 8-bit	EMC_A[25:0] EMC_D[31:0] EMC_OE EMC_WE EMC_BLS3:0 EMC_CS3:0 EMC_DYCS3:0 EMC_CAS EMC_RAS EMC_CLK1:0 EMC_FBCK EMC_CKE3:0 EMC_DQM3:0	EMCDynamicConfi3/2/1/0 EMCDynamicRasCas3/2/1/0	EMCStaticConfig3/2/1/0 EMCStaticWaitWen3/2/1/0 EMCStaticWaitOen3/2/1/0 EMCStaticWaitRd3/2/1/0 EMCStaticWaitPage3/2/1/0 EMCStaticWaitWr3/2/1/0 EMCStaticWaitTurn3/2/1/0	Section 33.14.1 Section 33.14.2 Section 33.14.3

[1] In addition to the registers that are common to all EMC operations: EMCControl and EMCConfig.

[2] In addition to the registers that are common to all EMC dynamic chip selects: EMCDynamicControl, EMCDynamicRefresh, EMCDynamicReadConfig, EMCDynamicRP, EMCDynamicRAS, EMCDynamicSREX, EMCDynamicAPR, EMCDynamicDAL, EMCDynamicWR, EMCDynamicRC, EMCDynamicRFC, EMCDynamicXSR, EMCDynamicRRD, and EMCDynamicMRD

[3] In addition to the EMCStaticExtendedWait register which applies to all static chip selects.

33.2 Introduction

The External Memory Controller (EMC) is an ARM PrimeCell™ MultiPort Memory Controller peripheral offering support for asynchronous static memory devices such as RAM, ROM, and Flash, in addition to dynamic memories such as Single Data Rate SDRAM. The EMC is an Advanced Microcontroller Bus Architecture (AMBA) compliant peripheral.

33.3 Features

- Static chip selects each support up to 64 MB of data. By enabling the address shift mode, static chip select 0 can support up to 256 MB, and static chip select 1 can support up to 128 MB (see Section 7.5.71 “EMC system control register”.
- Dynamic chip selects each support up to 256 MB of data.
- Dynamic memory interface support including Single Data Rate SDRAM.
- Asynchronous static memory device support including RAM, ROM, and Flash, with or without asynchronous page mode.
- Low transaction latency.
- Read and write buffers to reduce latency and to improve performance.
- 8-bit, 16-bit, and 32-bit wide static memory support.
- 16-bit and 32-bit wide chip select SDRAM memory support.
- Static memory features include:
  - Asynchronous page mode read
  - Programmable wait states
  - Bus turnaround delay
  - Output enable and write enable delays
  - Extended wait
- Four chip selects for synchronous memory and four chip selects for static memory devices.
- Power-saving modes dynamically control CKE and CLKOUT to SDRAMs.
- Dynamic memory self-refresh mode controlled by software.
- Controller supports 2 kbit, 4 kbit, and 8 kbit row address synchronous memory parts. That is typical 512 Mbit, 256 Mbit, and 128 Mbit parts, with 4, 8, 16, or 32 data bits per device.
- Separate reset domains allow the for auto-refresh through a chip reset if desired.
- Programmable delay elements allow fine-tuning EMC timing.

Note: Synchronous static memory devices (synchronous burst mode) are not supported.

33.4 Basic configuration

The EMC is configured using the following registers:

1. Clock: In the AHBCLKCTRL2 register (Section 7.5.21), set bit PCEMC.
2. The EMC subsystem contains an AHB clock divider and throttling mechanism to allow all the AHB slaves (that is, the memory and control) to operate on a integer divided version of AHB clock. This allows the users to be able to run the EMC at any division from the main clock (including odd and even). The lower rate is intended to be used primarily when the CPU is running faster than the external bus can support. Clock selection for the EMC is described in Section Section 7.5.62.
3. Pins: Select EMC pins and pin modes through the relevant IOCON registers (Section 10.4.2).

4. Configuration: See Table 637 to Table 640. Also see additional EMC configurations in Section 7.5.71 “EMC system control register”. In particular make sure that the address shift mode is configured correctly for the application hardware.
5. MPU: Default memory space permissions for the CPU do not allow program execution from the address range that includes the dynamic memory chip selects. These permissions can be changed by programming the MPU (see the ARM Cortex-M4 User Guide referred to in Section 52.1 for details of MPU operation.
6. To set the EMC delay clock see the EMC delay clock register in the system control block (see Section 7.5.72).
7. To calibrate the EMC clock, see Section 7.5.73.

33.5 EMC functional description

Figure 113 shows a block diagram of the EMC.

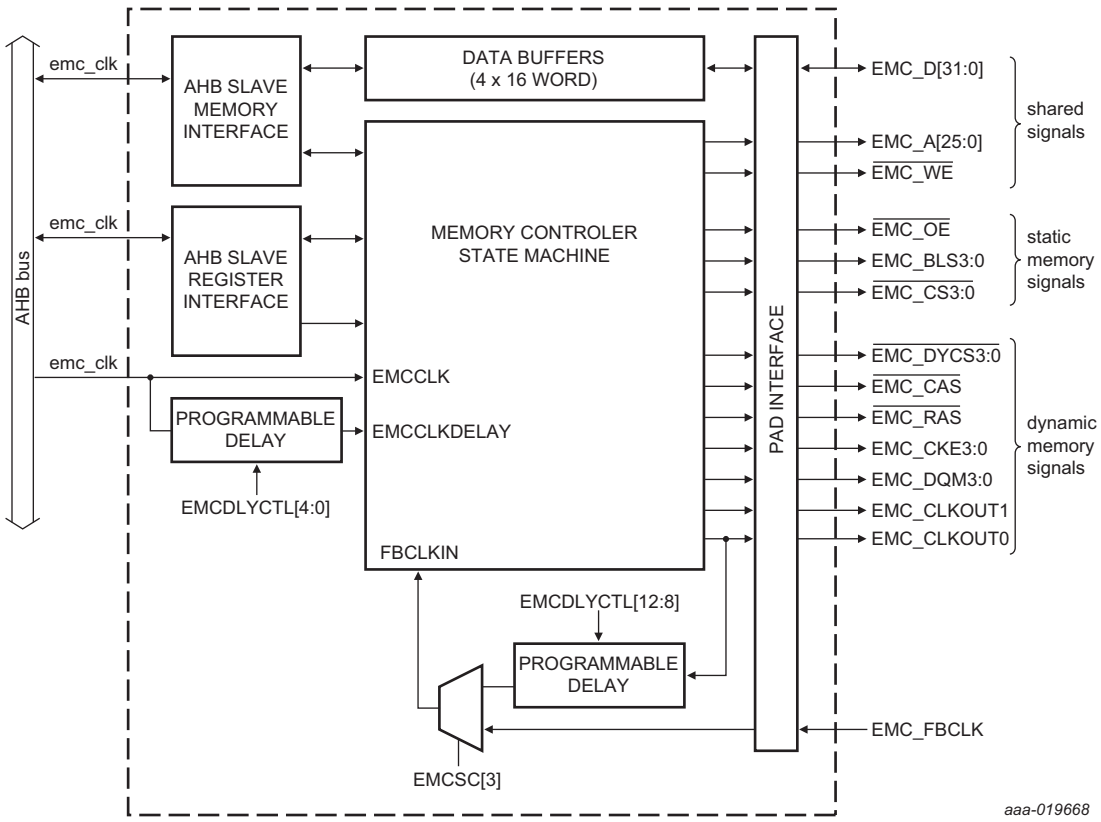


Fig 113. EMC block diagram

The functions of the EMC blocks are described in the following sections:

- AHB slave register interface.
- AHB slave memory interfaces.
- Data buffers.
- Memory controller state machine.
- Pad interface.

Note: For 32 bit wide chip selects, data is transferred to and from dynamic memory in SDRAM bursts of four. For 16 bit wide chip selects, SDRAM bursts of eight are used.

33.5.1 AHB slave register interface

The AHB slave register interface block enables the registers of the EMC to be programmed. This module also contains most of the registers and performs the majority of the register address decoding.

To eliminate the possibility of endianness problems, all data transfers to and from the registers of the EMC must be 32 bits wide.

Note: If an access is attempted with a size other than a word (32 bits), it causes an ERROR response to the AHB bus and the transfer is terminated.

33.5.2 AHB slave memory interface

The AHB slave memory interface allows access to external memories.

33.5.2.1 Memory transaction endianness

The endianness of the data transfers to and from the external memories is determined by the Endian mode (N) bit in the EMCConfig register.

Note: The memory controller must be idle (see the busy field of the EMCStatus Register) before endianness is changed, so that the data is transferred correctly.

33.5.2.2 Memory transaction size

Memory transactions can be 8, 16, or 32 bits wide. Any access attempted with a size greater than a word (32 bits) causes an ERROR response to the AHB bus and the transfer is terminated.

33.5.2.3 Write protected memory areas

Write transactions to write-protected memory areas generate an ERROR response to the AHB bus and the transfer is terminated.

33.5.3 Pad interface

The pad interface uses a feedback clock, FBCLKIN, from the CLKOUT0 output of the EMC to resynchronize SDRAM read data from the off-chip to on-chip domains.

33.5.4 Data buffers

The AHB interface reads and writes via buffers to improve memory bandwidth and reduce transaction latency. The EMC contains four 16-word buffers. The buffers can be used as read buffers, write buffers, or a combination of both. The buffers are allocated automatically.

The buffers must be disabled during SDRAM initialization. The buffers must be enabled during normal operation.

The buffers can be enabled or disabled for static memory using the EMCStaticConfig Registers.

33.5.4.1 Write buffers

Write buffers are used to:

- Merge write transactions so that the number of external transactions are minimized.

  Buffer data until the EMC can complete the write transaction, improving AHB write latency.

  Convert all dynamic memory write transactions into quadword bursts on the external memory interface. This enhances transfer efficiency for dynamic memory.
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Write buffer operation:

- If the buffers are enabled, an AHB write operation writes into the Least Recently Used (LRU) buffer, if empty.

  If the LRU buffer is not empty, the contents of the buffer are flushed to memory to make space for the AHB write data.
- If a buffer contains write data it is marked as dirty, and its contents are written to memory before the buffer can be reallocated.

The write buffers are flushed whenever:

- The memory controller state machine is not busy performing accesses to external memory.

  The memory controller state machine is not busy performing accesses to external memory, and an AHB interface is writing to a different buffer.

Note: For dynamic memory, the smallest buffer flush is a quadword of data. For static memory, the smallest buffer flush is a byte of data.

33.5.4.2 Read buffers

Read buffers are used to:

- Buffer read requests from memory. Future read requests that hit the buffer read the data from the buffer rather than memory, reducing transaction latency.

  Convert all read transactions into quadword bursts on the external memory interface. This enhances transfer efficiency for dynamic memory.
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Read buffer operation:

- If the buffers are enabled and the read data is contained in one of the buffers, the read data is provided directly from the buffer.
- If the read data is not contained in a buffer, the LRU buffer is selected. If the buffer is dirty (contains write data), the write data is flushed to memory. When an empty buffer is available the read command is posted to the memory.

A buffer filled by performing a read from memory is marked as not-dirty (not containing write data) and its contents are not flushed back to the memory controller unless a subsequent AHB transfer performs a write that hits the buffer.

33.5.5 Memory controller state machine

The memory controller state machine comprises a static memory controller and a dynamic memory controller.

33.5.6 Timing control with programmable delay elements

Programmable delay elements are provided to allow fine-tuning the timing of various aspects of EMC operation in connection with SDRAM memory.

- For the command delayed operating mode, a programmable delay is provided to control delay of all command outputs.
- A programmable delay is provided to control the time at which input data from SDRAM memory is sampled.
- The feedback clock can be used to sample SDRAM data

The locations of the programmable delays are shown in the EMC overall block diagram (Figure 113). See descriptions of the EMCDLYCTL and EMCCAL registers for more information.

33.6 Low-power operation

In many systems, the contents of the memory system have to be maintained during low-power sleep modes. The EMC provides a mechanism to place the dynamic memories into self-refresh mode.

Self-refresh mode can be entered by software by setting the SREFREQ bit in the EMCDynamicControl Register and polling the SREFACK bit in the EMCStatus Register.

Any transactions to memory that are generated while the memory controller is in self-refresh mode are rejected and an error response is generated to the AHB bus. Clearing the SREFREQ bit in the EMCDynamicControl Register returns the memory to normal operation. See the memory data sheet for refresh requirements.

Note: The static memory can be accessed as normal when the SDRAM memory is in self-refresh mode.

33.6.1 Low-power SDRAM deep-sleep mode

The EMC supports JEDEC low-power SDRAM deep-sleep mode. Deep-sleep mode can be entered by setting the deep-sleep mode (DP) bit, the dynamic memory clock enable bit (CE), and the dynamic clock control bit (CS) in the EMCDynamicControl register. The device is then put into a low-power mode where the device is powered down and no longer refreshed. All data in the memory is lost.

33.6.2 Low-power SDRAM partial array refresh

The EMC supports JEDEC low-power SDRAM partial array refresh. Partial array refresh can be programmed by initializing the SDRAM memory device appropriately. When the memory device is put into self-refresh mode only the memory banks specified are refreshed. The memory banks that are not refreshed lose their data contents.

33.7 Memory bank select

Eight independently-configurable memory chip selects are supported:

- Pins EMC_CS3 to EMC_CS0 are used to select static memory devices.
- Pins EMC_DYCS3 to EMC_DYCS0 are used to select dynamic memory devices.

Static memory chip select ranges are each 64 megabytes in size, while dynamic memory chip selects cover a range of 256 megabytes each. Table 634 shows the address ranges of the chip selects.

Table 634. Memory bank selection

Chip select pin	Address range	Memory type	Size of range
EMC_CS0	0x8000 0000 - 0x83FF FFFF	Static	64 MB
EMC_CS1	0x8800 0000 - 0x93FF FFFF	Static	64 MB
EMC_CS2	0x9000 0000 - 0x9BFF FFFF	Static	64 MB
EMC_CS3	0x9800 0000 - 0x9FFF FFFF	Static	64 MB
EMC_DYCS0	0xA000 0000 - 0xA3FF FFFF	Dynamic	256 MB
EMC_DYCS1	0xA800 0000 - 0xABFF FFFF	Dynamic	256 MB
EMC_DYCS2	0xB000 0000 - 0xB3FF FFFF	Dynamic	256 MB
EMC_DYCS3	0xB800 0000 - 0xBBFF FFFF	Dynamic	256 MB

33.8 EMC Reset

The EMC receives two reset signals. One is Power-On Reset (POR), asserted when chip power is applied, and when a brown-out condition is detected (see the System Control Block chapter for details of Brown-Out Detect). The other reset is from the external Reset pin and the Watchdog Timer.

A configuration bit in the SCS register, called EMCRD, allows control of how the EMC is reset (see Section 7.5.71 “EMC system control register”). The default configuration (EMCRD = 0) is that both EMC resets are asserted when any type of reset event occurs. In this mode, all registers and functions of the EMC are initialized upon any reset condition.

If EMCRD is set to 1, many portions of the EMC are only reset by a power-on or brown-out event, in order to allow the EMC to retain its state through a warm reset (external reset or watchdog reset). If the EMC is configured correctly, auto-refresh can be maintained through a warm reset.

33.9 Address shift mode

The EMC supports an optional address shift mode for static memories that can simplify board design and potentially increase external memory addressing range in some cases. The latter cases are described in footnotes of Table 3 “Memory usage and details” in the Memory Map chapter of this manual.

Address shift mode is controlled by a configuration bit in the SCS register, called EMC Shift Control (see Section 7.5.71 “EMC system control register”).

When the address shift mode is not activated (the EMC Shift Control bit in the EMCSYSCTRL register = 1), static memory addresses are output as byte addresses. This means that for memories wider than a byte, one or two address lines are not used, and that address connections to memory devices must be shifted in the board design. For example, if a 32-bit wide memory system is connected, the lowest line address of the memory device(s) would be connected to EMC address line 2, skipping bits 0 and 1.

When the address shift mode is activated (the EMC Shift Control bit in the EMCSYSCTRL register = 0), static memory addresses are shifted to match the lowest address bit needed for bus width. In this case, the lowest address line of the memory device(s) is always to EMC address line 0.

33.10 Memory mapped I/O and burst disable

By default, the EMC uses buffering to obtain better external memory access performance. However, in the case of memory mapped I/O devices, the read-ahead operations that occur due to the buffering can cause issues with some such devices. This could be from a change of status in one register caused by reading another register, or could simply cause an unplanned read of a data FIFO when another register in the device is read intentionally.

To prevent this issue, the use of buffering to read ahead of actual CPU memory read requests can be disabled. The configuration bit that controls this function is called EMC Burst Control, and is found in the EMCSYSCTRL register (see Section 7.5.71 “EMC system control register”).

33.11 Using the EMC with SDRAM

33.11.1 Mode register setup

When using the EMC with SDRAM, the SDRAM devices must be configured appropriately for the EMC. This includes setting up the SDRAMs for a 128-bit sequential burst. The burst configuration is done through a mode register in the SDRAM memory. Figure 114 shows the layout for a JEDEC standard SDRAM mode register.

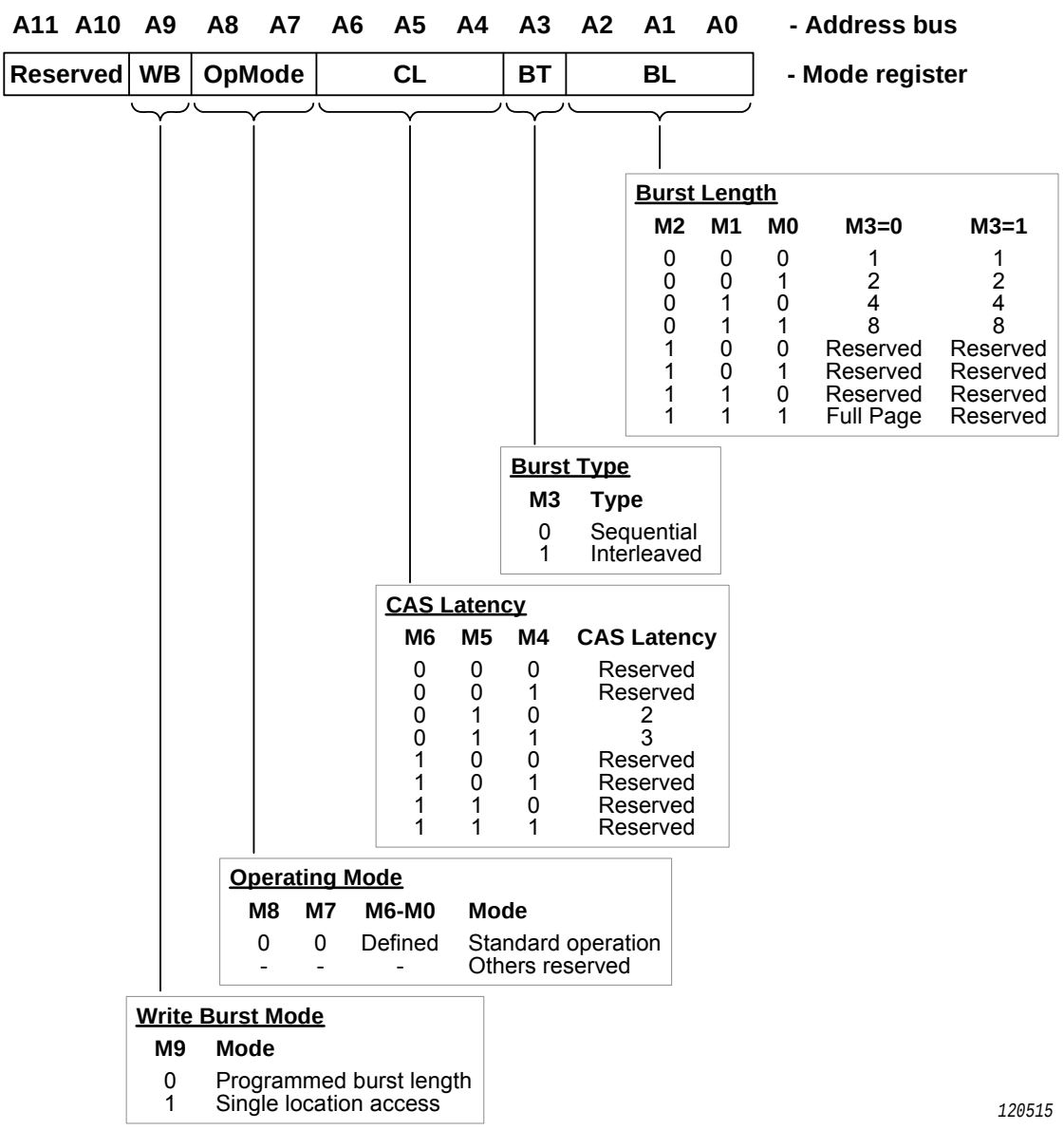


Fig 114. SDRAM mode register

The mode register is loaded by first sending the “Set Mode” Command to the SDRAM using the DYNAMICCONTROL register’s SDRAM Initialization bits to send a MODE command, and then reading the SDRAM at an address that is partially formed from the new mode register value. The actual value loaded into the mode register is taken by the SDRAM from the address lines of the EMC while they are sending the row address during the read.

Example

To determine the address to read from to load the mode register, the portion of the EMC address bits that map to the row address must be identified. In this example, we will use:

- Single 8M by 16-bit external SDRAM chip in Row, Bank, Column mode on CS0.
- CAS latency of 2.

Since the EMC uses bursts of 8 for a 16-bit external memory, we need to load the mode register with a burst length of 8 (8 x 16 bits memory width = 128 bits). The mode register configuration needed is 0x023. To load the mode register, we need to do a read from the address constructed as follows:

Information needed:

- Base address for Dynamic Chip Select 0, found in Table 3. For this device, the address is 0xA000 0000.
- Mode register value, based on information from both the SDRAM data sheet, as in Figure 114, and the EMC. In this example, the value will be 0x23. This represents a programmed burst length, CAS latency of 2, sequential burst type, and a burst length of 8, as described in Section 33.5.
- Bank bits and column bits, look up in Table 656. In this example, it is 4 banks and 9 column bits.
- Bus width, defined in this example to be 16 bits.
- Determine the shift value OFFSET to shift the mode register content by. This shift value depends on the SDRAM device organization and it is calculated as:

  OFFSET = number of columns + (data bus width/16) + bank select bits (Row, Bank, Column mode).

  OFFSET = number of columns + (data bus width/16) (Bank, Row, Column mode).

  Remark: The "data bus width/16" is 1 for a 16-bit bus and 2 for a 32-bit bus. This is the number of bits needed to indicate the number of banks. Most SDRAM devices use 2 bank select bits for four banks.
- Select the SDRAM memory mapped address DYCSX.
- The SDRAM read address is ADDRESS = DYCSX + (MODE << OFFSET).

The Mode register value calculation is:

Base address + (mode register value << (bank bits + column bits + bus width/16)

The shift operation aligns the mode register value with the row address bits.

In this example:

0xA000 0000 + (0x23 << (2 + 9 + 1)) = 0xA000 0000 + 0x23000 = 0xA002 3000

33.12 Pin description

Table 635 shows the interface and control signal pins for the EMC.

Table 635. Pad interface and control signal descriptions

Name	Type	Value on POR reset	Value during self-refresh	Description
EMC_A[23:0]	Output	0	Depends on static memory accesses	External memory address output. Used for both static and SDRAM devices. SDRAM memories use only bits [14:0].
EMC_D[31:0]	Input/ Output	Data outputs = 0	Depends on static memory accesses	External memory data lines. These are inputs when data is read from external memory and outputs when data is written to external memory.
$\overline{\text{EMC\_OE}}$	Output	1	Depends on static memory accesses	Low active output enable for static memory devices.
EMC_BLS3:0	Output	0xF	Depends on static memory accesses	Low active byte lane selects. Used for static memory devices.
$\overline{\text{EMC\_WE}}$	Output	1	Depends on static memory accesses	Low active write enable. Used for SDRAM and static memories.
$\overline{\text{EMC\_CS3:0}}$	Output	0xF	Depends on static memory accesses	Static memory chip selects. Default active LOW. Used for static memory devices.
$\overline{\text{EMC\_DYCS3:0}}$	Output	0xF	0xF	SDRAM chip selects. Used for SDRAM devices.
$\overline{\text{EMC\_CAS}}$	Output	1	1	Column address strobe. Used for SDRAM devices.
$\overline{\text{EMC\_RAS}}$	Output	1	1	Row address strobe. Used for SDRAM devices.
EMC_CLK1:0	Output	Follows EMCCLK	Follows EMCCLK	SDRAM clocks. Used for SDRAM devices.
EMC_CKE3:0	Output	0xF	0x0	SDRAM clock enables. Used for SDRAM devices. One is allocated for each Chip Select.
EMC_DQM3:0	Output	0xF	0xF	Data mask output to SDRAMs. Used for SDRAM devices and static memories.
EMC_FBCK	Input	0x3	0x3	Feedback clock to sample SDRAM data.

33.13 Register description

This chapter describes the EMC registers and provides details required when programming the microcontroller.

The EMC clock delay control and delay chain calibration control registers are located in the system control block. See Section 7.5.72 and Section 7.5.73.

Table 636. Register overview: EMC (base address 0x4008 1000)

Register Name	Access	Address offset	Description	Reset Value[1]	Section
CONTROL	R/W	0x000	Controls operation of the memory controller.	0x3	33.13.1
STATUS	RO	0x004	Provides EMC status information.	0x5	33.13.2
CONFIG	R/W	0x008	Configures operation of the memory controller	0x0	33.13.3
DYNAMICCONTROL	R/W	0x020	Controls dynamic memory operation.	0x006	33.13.4
DYNAMICREFRESH	R/W	0x024	Configures dynamic memory refresh.	0x0	33.13.5
DYNAMICREADCONFIG	R/W	0x028	Configures dynamic memory read strategy.	0x0	33.13.6
DYNAMICRP	R/W	0x030	Precharge command period.	0x0F	33.13.7
DYNAMICRAS	R/W	0x034	Active to precharge command period.	0xF	33.13.8
DYNAMICSREX	R/W	0x038	Self-refresh exit time.	0xF	33.13.9
DYNAMICAPR	R/W	0x03C	Last-data-out to active command time.	0xF	33.13.10
DYNAMICDAL	R/W	0x040	Data-in to active command time.	0xF	33.13.11
DYNAMICWR	R/W	0x044	Write recovery time.	0xF	33.13.12
DYNAMICRC	R/W	0x048	Selects the active to active command period.	0x1F	33.13.13
DYNAMICRFC	R/W	0x04C	Selects the auto-refresh period.	0x1F	33.13.14
DYNAMICXSR	R/W	0x050	Time for exit self-refresh to active command.	0x1F	33.13.15
DYNAMICRRD	R/W	0x054	Latency for active bank A to active bank B.	0xF	33.13.16
DYNAMICMRD	R/W	0x058	Time for load mode register to active command.	0xF	33.13.17
STATICEXTENDEDWAIT	R/W	0x080	Time for long static memory read and write transfers.	0x0	33.13.18
DYNAMICCONFIG0	R/W	0x100	Configuration information for EMC_DYCS0.	0x0	33.13.19
DYNAMICRASCAS0	R/W	0x104	RAS and CAS latencies for EMC_DYCS0.	0x303	33.13.20
DYNAMICCONFIG1	R/W	0x120	Configuration information for EMC_DYCS1.	0x0	33.13.19
DYNAMICRASCAS1	R/W	0x124	RAS and CAS latencies for EMC_DYCS1.	0x303	33.13.20
DYNAMICCONFIG2	R/W	0x140	Configuration information for EMC_DYCS2.	0x0	33.13.19
DYNAMICRASCAS2	R/W	0x144	RAS and CAS latencies for EMC_DYCS2.	0x303	33.13.20
DYNAMICCONFIG3	R/W	0x160	Configuration information for EMC_DYCS3.	0x0	33.13.19
DYNAMICRASCAS3	R/W	0x164	RAS and CAS latencies for EMC_DYCS3.	0x303	33.13.20
STATICCONFIG0	R/W	0x200	Configuration for EMC_CS0.	0x0	33.13.21
STATICWAITWEN0	R/W	0x204	Delay from EMC_CS0 to write enable.	0x0	33.13.22
STATICWAITOEN0	R/W	0x208	Delay from EMC_CS0 or address change, whichever is later, to output enable.	0x0	33.13.23
STATICWAITRD0	R/W	0x20C	Delay from EMC_CS0 to a read access.	0x1F	33.13.24
STATICWAITPAGE0	R/W	0x210	Delay for asynchronous page mode sequential accesses for EMC_CS0.	0x1F	33.13.25

Table 636. Register overview: EMC (base address 0x4008 1000) *...continued*

Register Name	Access	Address offset	Description	Reset Value[1]	Section
STATICWAITWR0	R/W	0x214	Delay from EMC_CS0 to a write access.	0x1F	33.13.26
STATICWAITTURN0	R/W	0x218	Number of bus turnaround cycles EMC_CS0.	0xF	33.13.27
STATICCONFIG1	R/W	0x220	Memory configuration for EMC_CS1.	0x0	33.13.21
STATICWAITWEN1	R/W	0x224	Delay from EMC_CS1 to write enable.	0x0	33.13.22
STATICWAITOEN1	R/W	0x228	Delay from EMC_CS1 or address change, whichever is later, to output enable.	0x0	33.13.23
STATICWAITRD1	R/W	0x22C	Delay from EMC_CS1 to a read access.	0x1F	33.13.24
STATICWAITPAGE1	R/W	0x230	Delay for asynchronous page mode sequential accesses for EMC_CS1.	0x1F	33.13.25
STATICWAITWR1	R/W	0x234	Delay from EMC_CS1 to a write access.	0x1F	33.13.26
STATICWAITTURN1	R/W	0x238	Bus turnaround cycles for EMC_CS1.	0xF	33.13.27
STATICCONFIG2	R/W	0x240	Memory configuration for EMC_CS2.	0x0	33.13.21
STATICWAITWEN2	R/W	0x244	Delay from EMC_CS2 to write enable.	0x0	33.13.22
STATICWAITOEN2	R/W	0x248	Delay from EMC_CS2 or address change, whichever is later, to output enable.	0x0	33.13.23
STATICWAITRD2	R/W	0x24C	Delay from EMC_CS2 to a read access.	0x1F	33.13.24
STATICWAITPAGE2	R/W	0x250	Delay for asynchronous page mode sequential accesses for EMC_CS2.	0x1F	33.13.25
STATICWAITWR2	R/W	0x254	Delay from EMC_CS2 to a write access.	0x1F	33.13.26
EMCStaticWaitTurn2	R/W	0x258	Bus turnaround cycles for EMC_CS2.	0xF	33.13.27
STATICCONFIG3	R/W	0x260	Memory configuration for EMC_CS3.	0x0	33.13.21
STATICWAITWEN3	R/W	0x264	Delay from EMC_CS3 to write enable.	0x0	33.13.22
STATICWAITOEN3	R/W	0x268	Delay from EMC_CS3 or address change, whichever is later, to output enable.	0x0	33.13.23
STATICWAITRD3	R/W	0x26C	Delay from EMC_CS3 to a read access.	0x1F	33.13.24
STATICWAITPAGE3	R/W	0x270	Delay for asynchronous page mode sequential accesses for EMC_CS3.	0x1F	33.13.25
STATICWAITWR3	R/W	0x274	Delay from EMC_CS3 to a write access.	0x1F	33.13.26
STATICWAITTURN3	R/W	0x278	Bus turnaround cycles for EMC_CS3.	0xF	33.13.27

[1] POR reset value. Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

33.13.1 EMC Control register

The EMCControl register is a read/write register that controls operation of the memory controller. The control bits can be altered during normal operation. Table 637 shows the bit assignments for the EMCControl register.

Table 637. EMC Control register (CONTROL - address 0x4008 1000) bit description

Bit	Symbol	Value	Description	Reset Value
0	E		EMC Enable. Indicates if the EMC is enabled or disabled:	1
		0	Disabled	
		1	Enabled (POR and warm reset value).	
			Disabling the EMC reduces power consumption. When the memory controller is disabled the memory is not refreshed. The memory controller is enabled by setting the enable bit, or by reset. This bit must only be modified when the EMC is in idle state.[1]	
1	M		Address mirror. Indicates normal or reset memory map.On POR, EMC_CS1 is mirrored to both EMC_CS0 and EMC_DYCS0 memory areas. Clearing the M bit enables EMC_CS0 and EMC_DYCS0 memory to be accessed.	1
		0	Normal memory map.	
		1	Reset memory map. Static memory EMC_CS1 is mirrored onto EMC_CS0 and EMC_DYCS0 (POR reset value).	
2	L		Low-power mode. Indicates normal, or low-power mode:	0
		0	Normal mode (warm reset value).	
		1	Low-power mode. Entering low-power mode reduces memory controller power consumption. Dynamic memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit (L), or by POR. This bit must only be modified when the EMC is in idle state.[1]	
31:3	-		Reserved. Read value is undefined, only zero should be written.	-

[1] The external memory cannot be accessed in low-power or disabled state. If a memory access is performed an AHB error response is generated. The EMC registers can be programmed in low-power and/or disabled state.

33.13.2 EMC Status register

The read-only EMCStatus register provides EMC status information.

Table 638. EMC Status register (STATUS - address 0x4008 1004) bit description

Bit	Symbol	Value	Description	Reset Value
0	B		Busy. This bit is used to ensure that the memory controller enters the low-power or disabled mode cleanly by determining if the memory controller is busy or not.	1
		0	EMC is idle (warm reset value).	
		1	EMC is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode (POR reset value).	
1	S		Write buffer status.This bit enables the EMC to enter low-power mode or disabled mode cleanly.	0
		0	Write buffers empty (POR reset value)	
		1	Write buffers contain data.	
2	SA		Self-refresh acknowledge. This bit indicates the operating mode of the EMC.	1
		0	Normal mode	
		1	Self-refresh mode (POR reset value).	
31:3	-		Reserved. The value read from a reserved bit is not defined.	NA

33.13.3 EMC Configuration register

The EMCConfig register configures the operation of the memory controller. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This register is accessed with one wait state.

Table 639. EMC Configuration register (CONFIG - address 0x4008 1008) bit description

Bit	Symbol	Value	Description	Reset Value
0	EM		Endian mode. On power-on reset, the value of the endian bit is 0. All data must be flushed in the EMC before switching between little-endian and big-endian modes.	0
		0	Little-endian mode (POR reset value).	
		1	Big-endian mode.	
7:1	-		Reserved. Read value is undefined, only zero should be written.	NA
8	CLKR		This bit must contain 0 for proper operation of the EMC.	0
31:9	-		Reserved. Read value is undefined, only zero should be written.	NA

33.13.4 Dynamic Memory Control register

The EMCDynamicControl register controls dynamic memory operation. The control bits can be altered during normal operation.

Table 640. Dynamic Control register (DYNAMICCONTROL - address 0x4008 1020) bit description

Bit	Symbol	Value	Description	Reset Value
0	CE		Dynamic memory clock enable.	0
		0	Clock enable of idle devices are deasserted to save power (POR reset value).	
		1	All clock enables are driven HIGH continuously.[1]	
1	CS		Dynamic memory clock control. When clock control is LOW the output clock CLKOUT is stopped when there are no SDRAM transactions. The clock is also stopped during self-refresh mode.	1
		0	CLKOUT stops when all SDRAMs are idle and during self-refresh mode.	
		1	CLKOUT runs continuously (POR reset value).	
2	SR		Self-refresh request, EMCSREFREQ. By writing 1 to this bit self-refresh can be entered under software control. Writing 0 to this bit returns the EMC to normal mode. The self-refresh acknowledge bit in the Status register must be polled to discover the current operating mode of the EMC.[2]	1
		0	Normal mode.	
		1	Enter self-refresh mode (POR reset value).	
4:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
5	MMC		Memory clock control.	0
		0	CLKOUT enabled (POR reset value).	
		1	CLKOUT disabled.[3]	
6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8:7	I		SDRAM initialization.	00
		0x0	Issue SDRAM NORMAL operation command (POR reset value).	
		0x1	Issue SDRAM MODE command.	
		0x2	Issue SDRAM PALL (precharge all) command.	
		0x3	Issue SDRAM NOP (no operation) command)	
13:9	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
31:14	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

[1] Clock enable must be HIGH during SDRAM initialization.

[2] The memory controller exits from power-on reset with the self-refresh bit HIGH. To enter normal functional mode set this bit LOW.

[3] Disabling CLKOUT can be performed if there are no SDRAM memory transactions. When enabled this bit can be used in conjunction with the dynamic memory clock control (CS) field.

Remark: Deep-sleep mode can be entered by setting the deep-sleep mode (DP) bit, the dynamic memory clock enable bit (CE), and the dynamic clock control bit (CS) to one. The device is then put into a low-power mode where the device is powered down and no longer refreshed. All data in the memory is lost.

33.13.5 Dynamic Memory Refresh Timer register

The EMCDynamicRefresh register configures dynamic memory operation. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. However, these control bits can, if necessary, be altered during normal operation. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed. .

Table 641. Dynamic Memory Refresh Timer register (DYNAMICREFRESH - address 0x4008 1024) bit description

Bit	Symbol	Description	Reset value
10:0	REFRESH	Refresh timer. Indicates the multiple of 16 EMCCLKs between SDRAM refresh cycles. 0x0 = Refresh disabled (POR reset value). 0x1 - 0x7FF = n x16 = 16n EMCCLKs between SDRAM refresh cycles. For example: 0x1 = 1 x 16 = 16 EMCCLKs between SDRAM refresh cycles. 0x8 = 8 x 16 = 128 EMCCLKs between SDRAM refresh cycles	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

For example, for the refresh period of 16 µs, and a EMCCLK frequency of 50 MHz, the following value must be programmed into this register:

(16 x 10-6 x 50 x 106) / 16 = 50 or 0x32

If auto-refresh through warm reset is requested (by setting the EMC_Reset_Disable bit), the timing of auto-refresh must be adjusted to allow a sufficient refresh rate when the clock rate is reduced during the wake-up period of a reset cycle. During this period, the EMC (and all other portions of the device that are being clocked) run from the IRC oscillator at 12 MHz. So, 12 MHz must be considered the EMCCLK rate for refresh calculations if auto-refresh through warm reset is requested.

Note: The refresh cycles are evenly distributed. However, there might be slight variations when the auto-refresh command is issued depending on the status of the memory controller.

33.13.6 Dynamic Memory Read Configuration register

The EMCDynamicReadConfig register configures the dynamic memory read strategy. This register must only be modified during system initialization. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects, so a single read strategy must be used for all dynamic memories.

Table 642 shows the bit assignments for the EMCDynamicReadConfig register.

Table 642. Dynamic Memory Read Configuration register (DYNAMICREADCONFIG - address 0x4008 1028) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	RD		Read data strategy.	0x0
		0x0	Reserved.	
		0x1	Command delayed strategy, using EMCCLKDELAY (command delayed, clock out not delayed).	
		0x2	Command delayed strategy plus one clock cycle, using EMCCLKDELAY (command delayed, clock out not delayed).	
		0x3	Command delayed strategy plus two clock cycles, using EMCCLKDELAY (command delayed, clock out not delayed).	
31:2	-		Reserved. Read value is undefined, only zero should be written.	NA

When using command delayed strategy, programmable delays can be used to adjust the timing of the control signals output by the EMC. See Section 33.5.6 and Section 7.5.72.

33.13.7 Dynamic Memory Precharge Command Period register

The EMCDynamicTRP register enables you to program the precharge command period, tRP. This register must only be modified during system initialization. This value is normally found in SDRAM data sheets as tRP. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 643. Dynamic Memory Precharge Command Period register (DYNAMICRP - address 0x4008 1030) bit description

Bit	Symbol	Description	Reset value
3:0	TRP	Precharge command period. 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.8 Dynamic Memory Active to Precharge Command Period register

The EMCDynamicTRAS register enables you to program the active to precharge command period, tRAS. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRAS. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 644. Dynamic Memory Active to Precharge Command Period register (DYNAMICRAS - address 0x4008 1034) bit description

Bit	Symbol	Description	Reset value
3:0	TRAS	Active to precharge command period. 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.9 Dynamic Memory Self-refresh Exit Time register

The EMCDynamicTSREX register enables you to program the self-refresh exit time, tSREX. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tSREX, for devices without this parameter you use the same value as tXSR. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 645. Dynamic Memory Self Refresh Exit Time register (DYNAMICSREX - address 0x4008 1038) bit description

Bit	Symbol	Description	Reset value
3:0	TSREX	Self-refresh exit time. 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.10 Dynamic Memory Last Data Out to Active Time register

The EMCDynamicTAPR register enables you to program the last-data-out to active command time, tAPR. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tAPR. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 646. Dynamic Memory Last Data Out to Active Time register (DYNAMICAPR - address 0x4008 103C) bit description

Bit	Symbol	Description	Reset value
3:0	TAPR	Last-data-out to active command time. 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.11 Dynamic Memory Data-in to Active Command Time register

The EMCDynamicTDAL register enables you to program the data-in to active command time, tDAL. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tDAL, or tAPW. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 647. Dynamic Memory Data In to Active Command Time register (DYNAMICDAL - address 0x4008 1040) bit description

Bit	Symbol	Description	Reset value
3:0	TDAL	Data-in to active command. 0x0 - 0xE = n clock cycles. The delay is in EMCCLK cycles. 0xF = 15 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.12 Dynamic Memory Write Recovery Time register

The EMCDynamicTWR register enables you to program the write recovery time, tWR. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tWR, tDPL, tRWL, or tRDL. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 648. Dynamic Memory Write Recovery Time register (DYNAMICWR - address 0x4008 1044) bit description

Bit	Symbol	Description	Reset value
3:0	TWR	Write recovery time. 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.13 Dynamic Memory Active to Active Command Period register

The EMCDynamicTRC register enables you to program the active to active command period, tRC. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRC. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 649. Dynamic Memory Active to Active Command Period register (DYNAMICRC - address 0x4008 1048) bit description

Bit	Symbol	Description	Reset value
4:0	TRC	Active to active command period. 0x0 - 0x1E = n + 1 clock cycles. The delay is in EMCCLK cycles. 0x1F = 32 clock cycles (POR reset value).	0x1F
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.14 Dynamic Memory Auto-refresh Period register

The EMCDynamicTRFC register enables you to program the auto-refresh period, and auto-refresh to active command period, tRFC. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRFC, or sometimes as tRC. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 650. Dynamic Memory Auto Refresh Period register (DYNAMICRFC - address 0x4008 104C) bit description

Bit	Symbol	Description	Reset value
4:0	TRFC	Auto-refresh period and auto-refresh to active command period. 0x0 - 0x1E = n + 1 clock cycles. The delay is in EMCCLK cycles. 0x1F = 32 clock cycles (POR reset value).	0x1F
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.15 Dynamic Memory Exit Self-refresh register

The EMCDynamicTXSR register enables you to program the exit self-refresh to active command time, tXSR. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tXSR. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 651. Dynamic Memory Exit Self Refresh register (DYNAMICXSR - address 0x4008 1050) bit description

Bit	Symbol	Description	Reset value
4:0	TXSR	Exit self-refresh to active command time. 0x0 - 0x1E = n + 1 clock cycles. The delay is in EMCCLK cycles. 0x1F = 32 clock cycles (POR reset value).	0x1F
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.16 Dynamic Memory Active Bank A to Active Bank B Time register

The EMCDynamicTRRD register enables you to program the active bank A to active bank B latency, tRRD. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRRD. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 652. Dynamic Memory Active Bank A to Active Bank B Time register (DYNAMICRRD - address 0x4008 1054) bit description

Bit	Symbol	Description	Reset value
3:0	TRRD	Active bank A to active bank B latency 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.17 Dynamic Memory Load Mode register to Active Command Time

The EMCDynamicTMRD register enables you to program the load mode register to active command time, tMRD. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tMRD, or tRSA. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 653. Dynamic Memory Load Mode register to Active Command Time (DYNAMICMRD - address 0x4008 1058) bit description

Bit	Symbol	Description	Reset value
3:0	TMRD	Load mode register to active command time. 0x0 - 0xE = n + 1 clock cycles. The delay is in EMCCLK cycles. 0xF = 16 clock cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.18 Static Memory Extended Wait register

ExtendedWait (EW) bit in the EMCStaticConfig register is set. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. However, if necessary, these control bits can be altered during normal operation. This register is accessed with one wait state.

Table 654. Static Memory Extended Wait register (STATICEXTENDEDWAIT - address 0x4008 1080) bit description

Bit	Symbol	Description	Reset value
9:0	EXTENDEDWAIT	Extended wait time out. 16 clock cycles (POR reset value). The delay is in EMCCLK cycles. 0x0 = 16 clock cycles. 0x1 - 0x3FF = (n+1) x16 clock cycles = (waitSecond * EMCCLK freq) / 16 - 1	0x0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

For example, for a static memory read/write transfer time of 8 μs, and a EMCCLK frequency of 50 MHz, the following value must be programmed into this register:
$(8 \times 10^{-6} \times 50 \times 10^{6}) / 16 - 1 = 24$

33.13.19 Dynamic Memory Configuration registers

The EMCDynamicConfig0-3 registers enable you to program the configuration information for the relevant dynamic memory chip select. These registers are normally only modified during system initialization. These registers are accessed with one wait state.

Table 655 shows the bit assignments for the EMCDynamicConfig0-3 registers.

Table 655. Dynamic Memory Configuration registers (DYNAMICCONFIG[0:3], address 0x4008 1100 (DYNAMICCONFIG0), 0x4008 1120 (DYNAMICCONFIG1), 0x4008 1140 (DYNAMICCONFIG2), 0x4008 1160 (DYNAMICCONFIG3)) bit description

Bit	Symbol	Value	Description	Reset Value
2:0	-		Reserved. Read value is undefined, only zero should be written.	NA
4:3	MD		Memory device.	0
		0x0	SDRAM (POR reset value).	
		0x1	Low-power SDRAM.	
		0x2	Reserved.	
		0x3	Reserved.	
6:5	-		Reserved. Read value is undefined, only zero should be written.	NA
12:7	AM0		See Table 656. 000000 = reset value.[1]	0
13	-		Reserved. Read value is undefined, only zero should be written.	NA
14	AM1		See Table 656. 0 = reset value.	0
18:15	-		Reserved. Read value is undefined, only zero should be written.	NA
19	B		Buffer enable.	0
		0	Buffer disabled for accesses to this chip select (POR reset value).	
		1	Buffer enabled for accesses to this chip select.[2]	
20	P		Write protect.	0
		0	Writes not protected (POR reset value).	
		1	Writes protected.	
31:21	-		Reserved. Read value is undefined, only zero should be written.	NA

[1] The SDRAM column and row width and number of banks are computed automatically from the address mapping.

[2] The buffers must be disabled during SDRAM initialization. The buffers must be enabled during normal operation.

Address mappings that are not shown in Table 656 are reserved.

Table 656. Address mapping

14	12	11:9	8:7	Description	Banks	Row length	Column length
16 bit high-performance bus width (Row, Bank, Column mode)							
0	0	000	00	16 Mbits (2M x 8)	2	11	9
0	0	000	01	16 Mbits (1M x 16)	2	11	8
0	0	001	00	64 Mbits (8M x 8)	4	12	9
0	0	001	01	64 Mbits (4M x 16)	4	12	8
0	0	010	00	128 Mbits (16M x 8)	4	12	10
0	0	010	01	128 Mbits (8M x 16)	4	12	9
0	0	011	00	256 Mbits (32M x 8)	4	13	10
0	0	011	01	256 Mbits (16M x 16)	4	13	9
0	0	100	00	512 Mbits (64M x 8)	4	13	11
0	0	100	01	512 Mbits (32M x 16)	4	13	10
16 bit bus width (Bank, Row, Column mode)							
0	1	000	00	16 Mbits (2M x 8)	2	11	9
0	1	000	01	16 Mbits (1M x 16)	2	11	8
0	1	001	00	64 Mbits (8M x 8)	4	12	9
0	1	001	01	64 Mbits (4M x 16)	4	12	8
0	1	010	00	128 Mbits (16M x 8)	4	12	10
0	1	010	01	128 Mbits (8M x 16)	4	12	9
0	1	011	00	256 Mbits (32M x 8)	4	13	10
0	1	011	01	256 Mbits (16M x 16)	4	13	9
0	1	100	00	512 Mbits (64M x 8)	4	13	11
0	1	100	01	512 Mbits (32M x 16)	4	13	10
32 bit high-performance bus width (Row, Bank, Column mode)							
1	0	000	00	16 Mbits (2M x 8)	2	11	9
1	0	000	01	16 Mbits (1M x 16)	2	11	8
1	0	001	00	64 Mbits (8M x 8)	4	12	9
1	0	001	01	64 Mbits (4M x 16)	4	12	8
1	0	001	10	64 Mbits (2M x 32)	4	11	8
1	0	010	00	128 Mbits (16M x 8)	4	12	10
1	0	010	01	128 Mbits (8M x 16)	4	12	9
1	0	010	10	128 Mbits (4M x 32)	4	12	8
1	0	011	00	256 Mbits (32M x 8)	4	13	10
1	0	011	01	256 Mbits (16M x 16)	4	13	9
1	0	011	10	256 Mbits (8M x 32)	4	13	8
1	0	100	00	512 Mbits (64M x 8)	4	13	11
1	0	100	01	512 Mbits (32M x 16)	4	13	10

Table 656. Address mapping *...continued*

14	12	11:9	8:7	Description	Banks	Row length	Column length
32 bit bus width (Bank, Row, Column mode)							
1	1	000	00	16 Mbits (2M x 8)	2	11	9
1	1	000	01	16 Mbits (1M x 16)	2	11	8
1	1	001	00	64 Mbits (8M x 8)	4	12	9
1	1	001	01	64 Mbits (4M x 16)	4	12	8
1	1	001	10	64 Mbits (2M x 32)	4	11	8
1	1	010	00	128 Mbits (16M x 8)	4	12	10
1	1	010	01	128 Mbits (8M x 16)	4	12	9
1	1	010	10	128 Mbits (4M x 32)	4	12	8
1	1	011	00	256 Mbits (32M x 8)	4	13	10
1	1	011	01	256 Mbits (16M x 16)	4	13	9
1	1	011	10	256 Mbits (8M x 32)	4	13	8
1	1	100	00	512 Mbits (64M x 8)	4	13	11
1	1	100	01	512 Mbits (32M x 16)	4	13	10

A chip select can be connected to a single memory device, in this case the chip select data bus width is the same as the device width. Alternatively the chip select can be connected to a number of external devices. In this case the chip select data bus width is the sum of the memory device data bus widths.

For example, for a chip select connected to:

- a 32 bit wide memory device, choose a 32 bit wide address mapping.
- a 16 bit wide memory device, choose a 16 bit wide address mapping.
- four x 8 bit wide memory devices, choose a 32 bit wide address mapping.
- two x 8 bit wide memory devices, choose a 16 bit wide address mapping.

The SDRAM bank select pins BA1 and BA0 are connected to address lines A14 and A13, respectively.

33.13.20 Dynamic Memory RAS and CAS Delay registers

The EMCDynamicRasCas0-3 registers enable you to program the RAS and CAS latencies for the relevant dynamic memory. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

Note: The values programmed into these registers must be consistent with the values used to initialize the SDRAM memory device.

Table 657. Dynamic Memory RASCAS Delay registers (DYNAMICRASCAS[0:3], address 0x4008 1104 (DYNAMICRASCAS0), 0x4008 1124 (DYNAMICRASCAS1), 0x4008 1144 (DYNAMICRASCAS2), 0x4008 1164 (DYNAMICRASCAS3)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	RAS		RAS latency (active to read/write delay).	11
		0x0	Reserved.	
		0x1	One EMCCLK cycle.	
		0x2	Two EMCCLK cycles.	
		0x3	Three EMCCLK cycles (POR reset value).	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:8	CAS		CAS latency.	11
		0x0	Reserved.	
		0x1	One EMCCLK cycle.	
		0x2	Two EMCCLK cycles.	
		0x3	Three EMCCLK cycles (POR reset value).	
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.21 Static Memory Configuration registers

The EMCStaticConfig0-3 registers configure the static memory configuration. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

Table 658 shows the bit assignments for the EMCStaticConfig0-3 registers. Note that synchronous burst mode memory devices are not supported.

Table 658. Static Memory Configuration registers (STATICCONFIG[0:3], address 0x4008 1200 (STATICCONFIG0), 0x4008 1220 (STATICCONFIG1), 0x4008 1240 (STATICCONFIG2), 0x4008 1260 (STATICCONFIG3)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	MW		Memory width.	0
		0x0	8 bit (POR reset value).	
		0x1	16 bit.	
		0x2	32 bit.	
		0x3	Reserved.	
2	-		Reserved. Read value is undefined, only zero should be written.	NA
3	PM		Page mode. In page mode the EMC can burst up to four external accesses. Therefore devices with asynchronous page mode burst four or higher devices are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.	0
		0	Disabled (POR reset value).	
		1	Asynchronous page mode enabled (page length four).	
5:4	-		Reserved. Read value is undefined, only zero should be written.	NA
6	PC		Chip select polarity. The value of the chip select polarity on power-on reset is 0.	0
		0	Active LOW chip select.	
		1	Active HIGH chip select.	
7	PB		Byte lane state. The byte lane state bit, PB, enables different types of memory to be connected. For byte-wide static memories the BLS3:0 signal from the EMC is usually connected to $\overline{WE}$ (write enable). In this case for reads all the BLS3:0 bits must be HIGH. This means that the byte lane state (PB) bit must be LOW. 16 bit wide static memory devices usually have the BLS3:0 signals connected to the UBn and LBn (upper byte and lower byte) signals in the static memory. In this case a write to a particular byte must assert the appropriate UBn or LBn signal LOW. For reads, all the $\overline{UB}$ and $\overline{LB}$ signals must be asserted LOW so that the bus is driven. In this case the byte lane state (PB) bit must be HIGH.	0
		0	For reads all the bits in BLS3:0 are HIGH. For writes the respective active bits in BLS3:0 are LOW (POR reset value).	
		1	For reads the respective active bits in BLS3:0 are LOW. For writes the respective active bits in BLS3:0 are LOW.	

Table 658. Static Memory Configuration registers (STATICCONFIG[0:3], address 0x4008 1200 (STATICCONFIG0), 0x4008 1220 (STATICCONFIG1), 0x4008 1240 (STATICCONFIG2), 0x4008 1260 (STATICCONFIG3)) bit description

Bit	Symbol	Value	Description	Reset Value
8	EW		Extended wait (EW) uses the EMCStaticExtendedWait register to time both the read and write transfers rather than the EMCStaticWaitRd and EMCStaticWaitWr registers. This enables much longer transactions.[1]	0
		0	Extended wait disabled (POR reset value).	
		1	Extended wait enabled.	
18:9	-		Reserved. Read value is undefined, only zero should be written.	NA
19	B		Buffer enable [2]	0
		0	Buffer disabled (POR reset value).	
		1	Buffer enabled.	
20	P		Write protect	0
		0	Writes not protected (POR reset value).	
		1	Write protected.	
31:21	-		Reserved. Read value is undefined, only zero should be written.	NA

[1] Extended wait and page mode cannot be selected simultaneously.

[2] EMC may perform burst read access even when the buffer enable bit is cleared.

33.13.22 Static Memory Write Enable Delay registers

The EMCStaticWaitWen0-3 registers enable you to program the delay from the chip select to the write enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

Table 659. Static Memory Write Enable Delay registers (STATICWAITWEN[0:3], address 0x4008 1204 (STATICWAITWEN0), 0x4008 1224 (STATICWAITWEN1),0x4008 1244 (STATICWAITWEN2), 0x4008 1264 (STATICWAITWEN3)) bit description

Bit	Symbol	Description	Reset value
3:0	WAITWEN	Wait write enable. Delay from chip select assertion to write enable. 0x0 = One EMCCLK cycle delay between assertion of chip select and write enable (POR reset value). 0x1 - 0xF = (n + 1) EMCCLK cycle delay. The delay is (WAITWEN +1) x tEMCCLK.	0x0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.23 Static Memory Output Enable Delay registers

The EMCStaticWaitOen0-3 registers enable you to program the delay from the chip select or address change, whichever is later, to the output enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

Table 660. Static Memory Output Enable delay registers (STATICWAITOEN[0:3], address 0x4008 1208 (STATICWAITOEN0), 0x0x4008 1228 (STATICWAITOEN1), 0x0x4008 1248 (STATICWAITOEN2), 0x0x4008 1268 (STATICWAITOEN3)) bit description

Bit	Symbol	Description	Reset value
3:0	WAITOEN	Wait output enable. Delay from chip select assertion to output enable. 0x0 = No delay (POR reset value). 0x1 - 0xF = n cycle delay. The delay is WAITOEN x tEMCCLK.	0x0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.24 Static Memory Read Delay registers

The EMCStaticWaitRd0-3 registers enable you to program the delay from the chip select to the read access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. It is not used if the extended wait bit is enabled in the EMCStaticConfig0-3 registers. These registers are accessed with one wait state.

Table 661. Static Memory Read Delay registers (STATICWAITRD[0:3], address 0x4008 120C (STATICWAITRD0), 0x4008 122C (STATICWAITRD1), 0x4008 124C (STATICWAITRD2), 0x4008 126C (STATICWAITRD3)) bit description

Bit	Symbol	Description	Reset value
4:0	WAITRD	Non-page mode read wait states or asynchronous page mode read first access wait state. Non-page mode read or asynchronous page mode read, first read only: 0x0 - 0x1E = (n + 1) EMCCLK cycles for read accesses. For non-sequential reads, the wait state time is (WAITRD + 1) x tEMCCLK. 0x1F = 32 EMCCLK cycles for read accesses (POR reset value).	0x1F [1]
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

[1] The reset value depends on the boot mode.

33.13.25 Static Memory Page Mode Read Delay registers

The EMCStaticWaitPage0-3 registers enable you to program the delay for asynchronous page mode sequential accesses. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This register is accessed with one wait state.

Table 662. Static Memory Page Mode Read Delay registers (STATICWAITPAGE[0:3], address 0x4008 1210 (STATICWAITPAGE0), 4008 1230 (STATICWAITPAGE1), 0x4008 1250 (STATICWAITPAGE2), 0x4008 1270 (STATICWAITPAGE3)) bit description

Bit	Symbol	Description	Reset value
4:0	WAITPAGE	Asynchronous page mode read after the first read wait states. Number of wait states for asynchronous page mode read accesses after the first read: 0x0 - 0x1E = (n+ 1) EMCCLK cycle read access time. For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE + 1) x tEMCCLK. 0x1F = 32 EMCCLK cycle read access time (POR reset value).	0x1F
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.26 Static Memory Write Delay registers

The EMCStaticWaitWr0-3 registers enable you to program the delay from the chip select to the write access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode.These registers are not used if the extended wait (EW) bit is enabled in the EMCStaticConfig register. These registers are accessed with one wait state.

Table 663. Static Memory Write Delay registers (STATICWAITWR[0:3], address 0x4008 1214 (STATICWAITWR0), 0x4008 1234 (STATICWAITWR1), 0x4008 1254 (STATICWAITWR2), 0x4008 1274 (STATICWAITWR3)) bit description

Bit	Symbol	Description	Reset value
4:0	WAITWR	Write wait states. SRAM wait state time for write accesses after the first read: 0x0 - 0x1E = (n + 2) EMCCLK cycle write access time. The wait state time for write accesses after the first read is WAITWR (n + 2) x tEMCCLK. 0x1F = 33 EMCCLK cycle write access time (POR reset value).	0x1F
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

33.13.27 Static Memory Turn Round Delay registers

The EMCStaticWaitTurn0-3 registers enable you to program the number of bus turnaround cycles. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

Table 664. Static Memory Turn-around Delay registers (STATICWAITTURN[0:3], address 0x4008 1218 (STATICWAITTURN0),0x4008 1238 (STATICWAITTURN1), 0x4008 1258 (STATICWAITTURN2), 0x4008 1278 (STATICWAITTURN3)) bit description

Bit	Symbol	Description	Reset value
3:0	WAITTURN	Bus turn-around cycles. 0x0 - 0xE = (n + 1) EMCCLK turn-around cycles. Bus turn-around time is (WAITTURN + 1) x tEMCCLK. 0xF = 16 EMCCLK turn-around cycles (POR reset value).	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

To prevent bus contention on the external memory data bus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses. The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses.

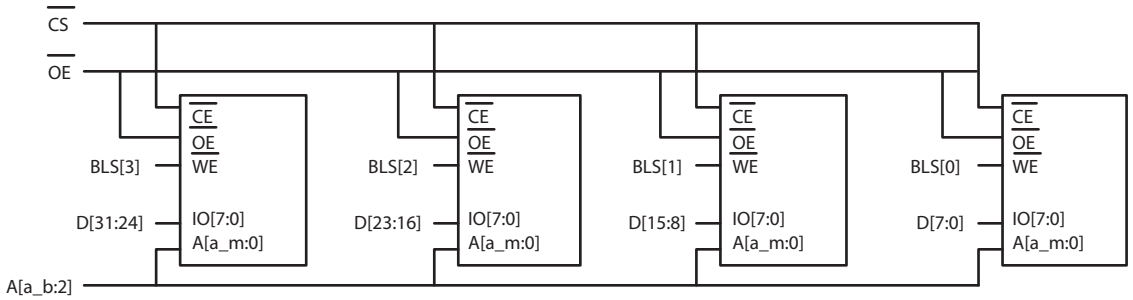
33.14 External static memory interface

External memory interfacing depends on the bank width (32, 16 or 8 bit selected via MW bits in corresponding EMCStaticConfig register).

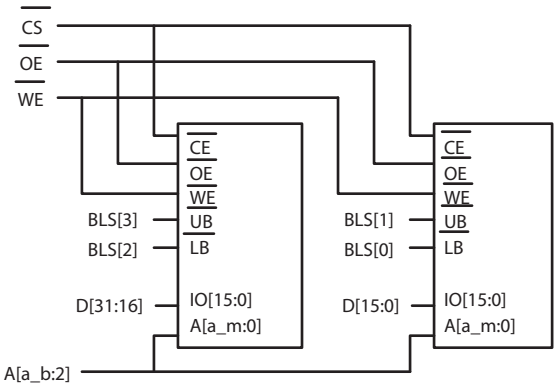
If a memory bank is configured to be 32 bits wide, address lines A0 and A1 can be used as non-address lines. If a memory bank is configured to 16 bits wide, A0 is not required. However, 8 bit wide memory banks do require all address lines down to A0. Configuring the A1 and/or A0 lines to provide address or non-address function is accomplished using the IOCON registers (see Section 10.5).

Symbol "a_b" in the following figures refers to the highest order address line in the data bus. Symbol "a_m" refers to the highest order address line of the memory chip used in the external memory interface.

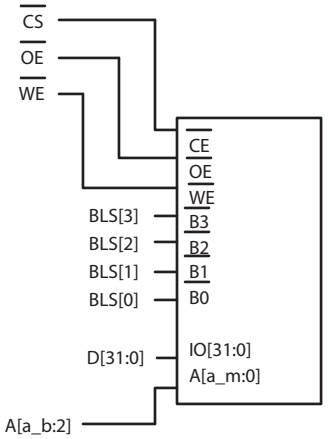
33.14.1 32-bit wide memory bank connection



a. 32 bit wide memory bank interfaced to four 8 bit memory chips



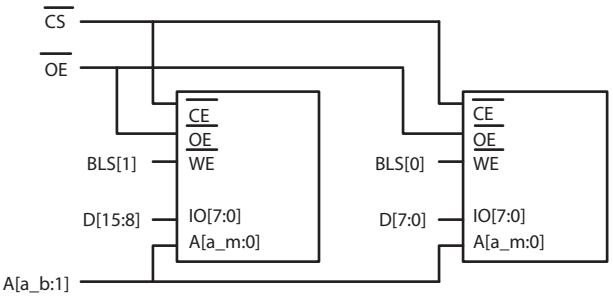
b. 32 bit wide memory bank interfaced to two 16 bit memory chips



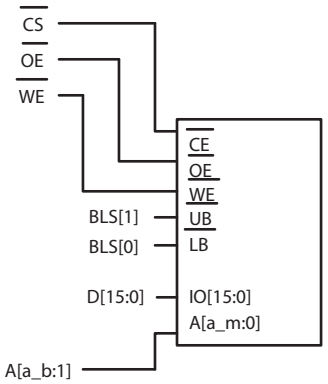
c. 32 bit wide memory bank interfaced to one 32 bit memory chip

Fig 115. 32 bit bank external memory interfaces (bits MW = 10)

33.14.2 16-bit wide memory bank connection



a. 16 bit wide memory bank interfaced to two 8 bit memory chips



b. 16 bit wide memory bank interfaced to a 16 bit memory chip

Fig 116. 16 bit bank external memory interfaces (bits MW = 01)

33.14.3 8-bit wide memory bank connection

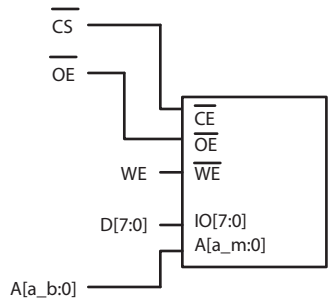


Fig 117. 8 bit bank external memory interface (bits MW = 00)

33.14.4 Memory configuration example

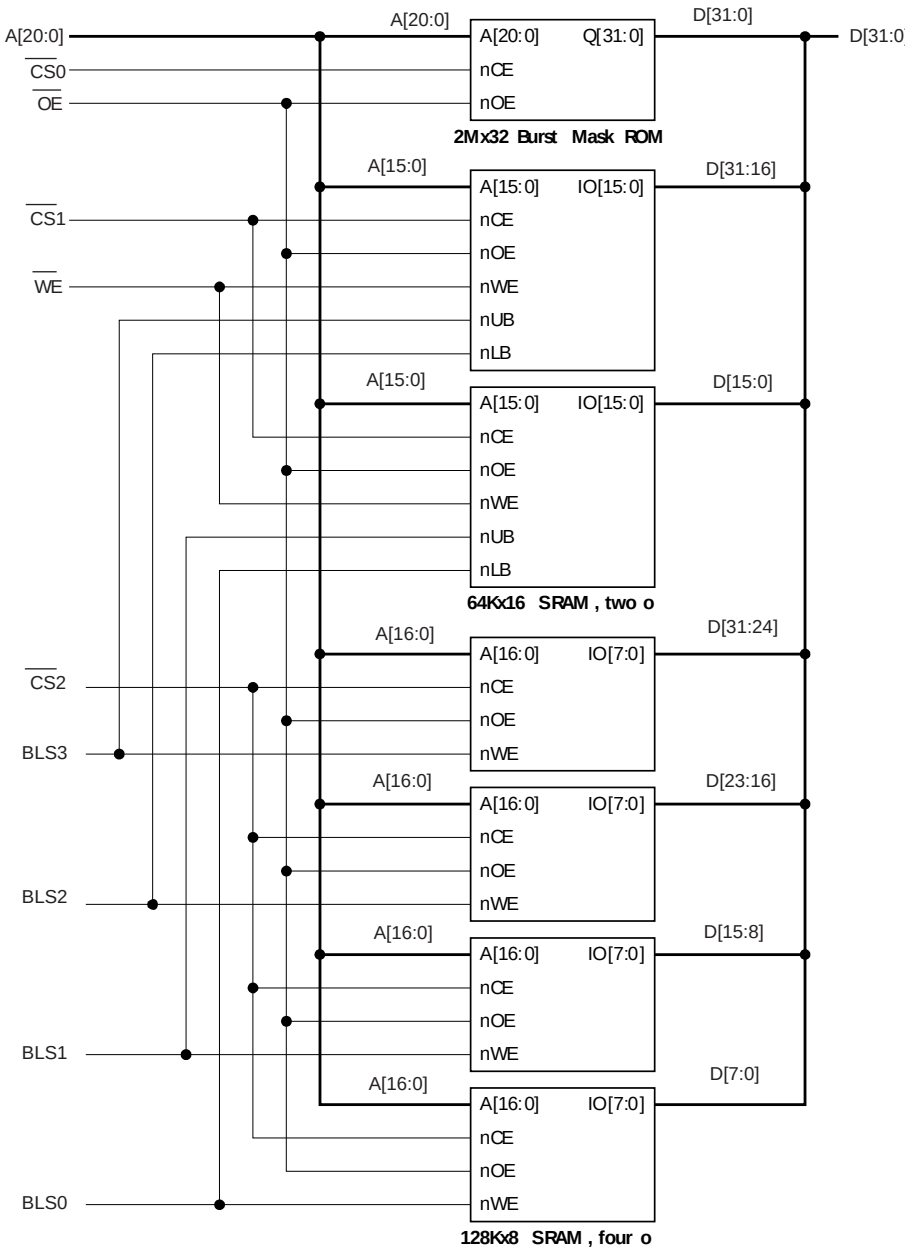


Fig 118. Typical memory configuration diagram

34.1 How to read this chapter

The LCD controller is available on some LPC546xx devices, see Section 1.9 for details.

34.2 Introduction

The LCD controller provides all of the necessary control signals to interface directly to a variety of color and monochrome LCD panels.

34.3 Features

- AHB bus master interface to access frame buffer.
- Setup and control via a separate AHB slave interface.
- Dual 16-deep programmable 64-bit wide FIFOs for buffering incoming display data.
- Supports single and dual-panel monochrome Super Twisted Nematic (STN) displays with 4 or 8-bit interfaces.
- Supports single and dual-panel color STN displays.
- Supports Thin Film Transistor (TFT) color displays.
- Programmable display resolution including, but not limited to: 320x200, 320x240, 640x200, 640x240, 640x480, 800x600, and 1024x768.
- Hardware cursor support for single-panel displays.
- 15 gray-level monochrome, 3375 color STN, and 32K color palettized TFT support.
- 1, 2, or 4 bits-per-pixel (bpp) palettized displays for monochrome STN.
- 1, 2, 4, or 8 bpp palettized color displays for color STN and TFT.
- 16 bpp true-color non-palettized, for color STN and TFT.
- 24 bpp true-color non-palettized, for color TFT.
- Programmable timing for different display panels.
- 256 entry, 16-bit palette RAM, arranged as a 128x32-bit RAM.
- Frame, line, and pixel clock signals.
- AC bias signal for STN, data enable signal for TFT panels.
- Supports little and big-endian, and Windows CE data formats.
- LCD panel clock is generated from the LCD peripheral clock, which, in turn, can be selected in the CGU from a variety of sources, including an LCD clock input pin.

34.3.1 Programmable parameters

The following key display and controller parameters can be programmed:

- Horizontal front and back porch
- Horizontal synchronization pulse width
- Number of pixels per line
- Vertical front and back porch
- Vertical synchronization pulse width
- Number of lines per panel
- Number of pixel clocks per line
- Hardware cursor control.
- Signal polarity, active HIGH or LOW
- AC panel bias
- Panel clock frequency
- Bits-per-pixel
- Display type: STN monochrome, STN color, or TFT
- STN 4 or 8-bit interface mode
- STN dual or single panel mode
- Little-endian, big-endian, or Windows CE mode
- Interrupt generation event

34.3.2 Hardware cursor support

The hardware cursor feature reduces software overhead associated with maintaining a cursor image in the LCD frame buffer.

Without this feature, software needed to:

- Save an image of the area under the next cursor position.
- Update the area with the cursor image.
- Repair the last cursor position with a previously saved image.

In addition, the LCD driver had to check whether the graphics operation had overwritten the cursor, and correct it. With a cursor size of 64x64 and 24-bit color, each cursor move involved reading and writing approximately 75 KB of data.

The hardware cursor removes the requirement for this management by providing a completely separate image buffer for the cursor, and superimposing the cursor image on the LCD output stream at the current cursor (X,Y) coordinate.

To move the hardware cursor, the software driver supplies a new cursor coordinate. The frame buffer requires no modification. This significantly reduces software overhead.

The cursor image is held in the LCD controller in an internal 256x32-bit buffer memory.

34.3.3 Types of LCD panels supported

The LCD controller supports the following types of LCD panel:

- Active matrix TFT panels with up to 24-bit bus interface.
- Single-panel monochrome STN panels (4-bit and 8-bit bus interface).
- Dual-panel monochrome STN panels (4-bit and 8-bit bus interface per panel).
- Single-panel color STN panels, 8-bit bus interface.
- Dual-panel color STN panels, 8-bit bus interface per panel.

34.3.4 TFT panels

TFT panels support one or more of the following color modes:

- 1 bpp, palettized, 2 colors selected from available colors.
- 2 bpp, palettized, 4 colors selected from available colors.
- 4 bpp, palettized, 16 colors selected from available colors.
- 8 bpp, palettized, 256 colors selected from available colors.
- 12 bpp, direct 4:4:4 RGB.
- 16 bpp, direct 5:5:5 RGB, with 1 bpp not normally used. This pixel is still output, and can be used as a brightness bit to connect to the Least Significant Bit (LSB) of RGB components of a 6:6:6 TFT panel.
- 16 bpp, direct 5:6:5 RGB.
- 24 bpp, direct 8:8:8 RGB, providing over 16 million colors.

Each 16-bit palette entry is composed of 5 bpp (RGB), plus a common intensity bit. This provides better memory utilization and performance compared with a full 6 bpp structure. The total number of colors supported can be doubled from 32K to 64K if the intensity bit is used and applied to all three color components simultaneously.

Alternatively, the 16 signals can be used to drive a 5:6:5 panel with the extra bit only applied to the green channel.

34.3.5 Color STN panels

Color STN panels support one or more of the following color modes:

- 1 bpp, palettized, 2 colors selected from 3375.
- 2 bpp, palettized, 4 colors selected from 3375.
- 4 bpp, palettized, 16 colors selected from 3375.
- 8 bpp, palettized, 256 colors selected from 3375.
- 16 bpp, direct 4:4:4 RGB, with 4 bpp not being used.

34.3.6 Monochrome STN panels

Monochrome STN panels support one or more of the following modes:

- 1 bpp, palettized, 2 gray scales selected from 15.
- 2 bpp, palettized, 4 gray scales selected from 15.
- 4 bpp, palettized, 16 gray scales selected from 15.

More than 4 bpp for monochrome panels can be programmed, but using these modes has no benefit because the maximum number of gray scales supported on the display is 15.

34.4 Basic configuration

The LCD controller is configured using the following registers:

1. Clock: In the AHBCLKCTRL2 register (Section 7.5.21), set bit LCD.

   Remark: The LCD is disabled on reset (LCD = 0).

   Select LCD clock using LCDCLKSEL register (Section 7.5.42). Use LCDCLKDIV to divide the above selected clock.
2. See Table 506 and Table 79. Also see Section 28.6.12 for power-up procedure.
3. Pins: Select LCD pins and pin modes through the relevant IOCON registers (Section 10.4.2).
4. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
5. The LCD clock divider is configured in the system configuration block (Section 7.5.60) and the CLKSEL bit in the LCD_POL register (Section 34.7.3).

34.5 Pin description

The largest configuration for the LCD controller uses 31 pins. There are many variants using as few as 10 pins for a monochrome STN panel. Pins are allocated in groups based on the selected configuration. All LCD functions are shared with other chip functions. In Table 665, only the LCD related portion of the pin name is shown.

Remark: To connect the LCD controller to necessary pins, see Section 10.4.2.

Table 665. LCD controller pins

Pin name	Type	Function
LCD_PWR	output	LCD panel power enable.
LCD_DCLK	output	LCD panel clock.
LCD_AC	output	STN AC bias drive or TFT data enable output.
LCD_FP	output	Frame pulse (STN). Vertical synchronization pulse (TFT)
LCD_LE	output	Line end signal
LCD_LP	output	Line synchronization pulse (STN). Horizontal synchronization pulse (TFT)
LCD_VD[23:0]	output	LCD panel data. Bits used depend on the panel configuration.
LCD_CLKIN	input	Optional clock input. Each level on this pin must be at least 1 PCLK in duration in order to be sampled. The maximum frequency must therefore be less than PCLK/2.

34.5.1 Signal usage

The signals that are used for various display types are identified in the following sections.

34.5.1.1 Signals used for single panel STN displays

The signals used for single panel STN displays are shown in Table 666. UD refers to upper panel data.

Table 666. Pins used for single panel STN displays

Pin name	4-bit Monochrome (10 pins)	8-bit Monochrome (14 pins)	Color (14 pins)
LCD_PWR	Y	Y	Y
LCD_DCLK	Y	Y	Y
LCD_AC	Y	Y	Y
LCD_FP	Y	Y	Y
LCD_LE	Y	Y	Y
LCD_LP	Y	Y	Y
LCD_VD[3:0]	UD[3:0]	UD[3:0]	UD[3:0]
LCD_VD[7:4]	-	UD[7:4]	UD[7:4]
LCD_VD[23:8]	-	-	-

34.5.1.2 Signals used for dual panel STN displays

The signals used for dual panel STN displays are shown in Table 667. UD refers to upper panel data, and LD refers to lower panel data.

Table 667. Pins used for dual panel STN displays

Pin name	4-bit Monochrome (14 pins)	8-bit Monochrome (22 pins)	Color (22 pins)
LCD_PWR	Y	Y	Y
LCD_DCLK	Y	Y	Y
LCD_AC	Y	Y	Y
LCD_FP	Y	Y	Y
LCD_LE	Y	Y	Y
LCD_LP	Y	Y	Y
LCD_VD[3:0]	UD[3:0]	UD[3:0]	UD[3:0]
LCD_VD[7:4]	-	UD[7:4]	UD[7:4]
LCD_VD[11:8]	LD[3:0]	LD[3:0]	LD[3:0]
LCD_VD[15:12]	-	LD[7:4]	LD[7:4]
LCD_VD[23:16]	-	-	-

34.5.1.3 Signals used for TFT displays

The signals used for TFT displays are shown in Table 668.

Table 668. Pins used for TFT displays

Pin name	12-bit, 4:4:4 mode (18 pins)	16-bit, 5:6:5 mode (22 pins)	16-bit, 1:5:5:5 mode (24 pins)	24-bit (30 pins)
LCD_PWR	Y	Y	Y	Y
LCD_DCLK	Y	Y	Y	Y
LCD_AC	Y	Y	Y	Y
LCD_FP	Y	Y	Y	Y
LCD_LE	Y	Y	Y	Y
LCD_LP	Y	Y	Y	Y
LCD_VD[1:0]	-	-	-	RED[1:0]
LCD_VD[2]	-	-	Intensity	RED[2]
LCD_VD[3]	-	RED[0]	RED[0]	RED[3]
LCD_VD[7:4]	RED[3:0]	RED[4:1]	RED[4:1]	RED[7:4]
LCD_VD[9:8]	-	-	-	GREEN[1:0]
LCD_VD[10]	-	GREEN[0]	Intensity	GREEN[2]
LCD_VD[11]	-	GREEN[1]	GREEN[0]	GREEN[3]
LCD_VD[15:12]	GREEN[3:0]	GREEN[5:2]	GREEN[4:1]	GREEN[7:4]
LCD_VD[17:16]	-	-	-	BLUE[1:0]
LCD_VD[18]	-	-	Intensity	BLUE[2]
LCD_VD[19]	-	BLUE[0]	BLUE[0]	BLUE[3]
LCD_VD[23:20]	BLUE[3:0]	BLUE[4:1]	BLUE[4:1]	BLUE[7:4]

34.6 LCD controller functional description

The LCD controller performs translation of pixel-coded data into the required formats and timings to drive a variety of single or dual panel monochrome and color LCDs.

Packets of pixel coded data are fed using the AHB interface, to two independent, programmable, 32-bit wide, DMA FIFOs that act as input data flow buffers.

The buffered pixel coded data is then unpacked using a pixel serializer.

Depending on the LCD type and mode, the unpacked data can represent:

- An actual true display gray or color value.
- An address to a 256x16 bit wide palette RAM gray or color value.

In the case of STN displays, either a value obtained from the addressed palette location, or the true value is passed to the gray scaling generators. The hardware-coded gray scale algorithm logic sequences the activity of the addressed pixels over a programmed number of frames to provide the effective display appearance.

For TFT displays, either an addressed palette value or true color value is passed directly to the output display drivers, bypassing the gray scaling algorithmic logic.

In addition to data formatting, the LCD controller provides a set of programmable display control signals, including:

- LCD panel power enable.
- Pixel clock.
- Horizontal and vertical synchronization pulses.
- Display bias.

The LCD controller generates individual interrupts for:

- Upper or lower panel DMA FIFO underflow.
- Base address update signification.
- Vertical compare.
- Bus error.

There is also a single combined interrupt that is asserted when any of the individual interrupts become active.

Figure 119 shows a simplified block diagram of the LCD controller.

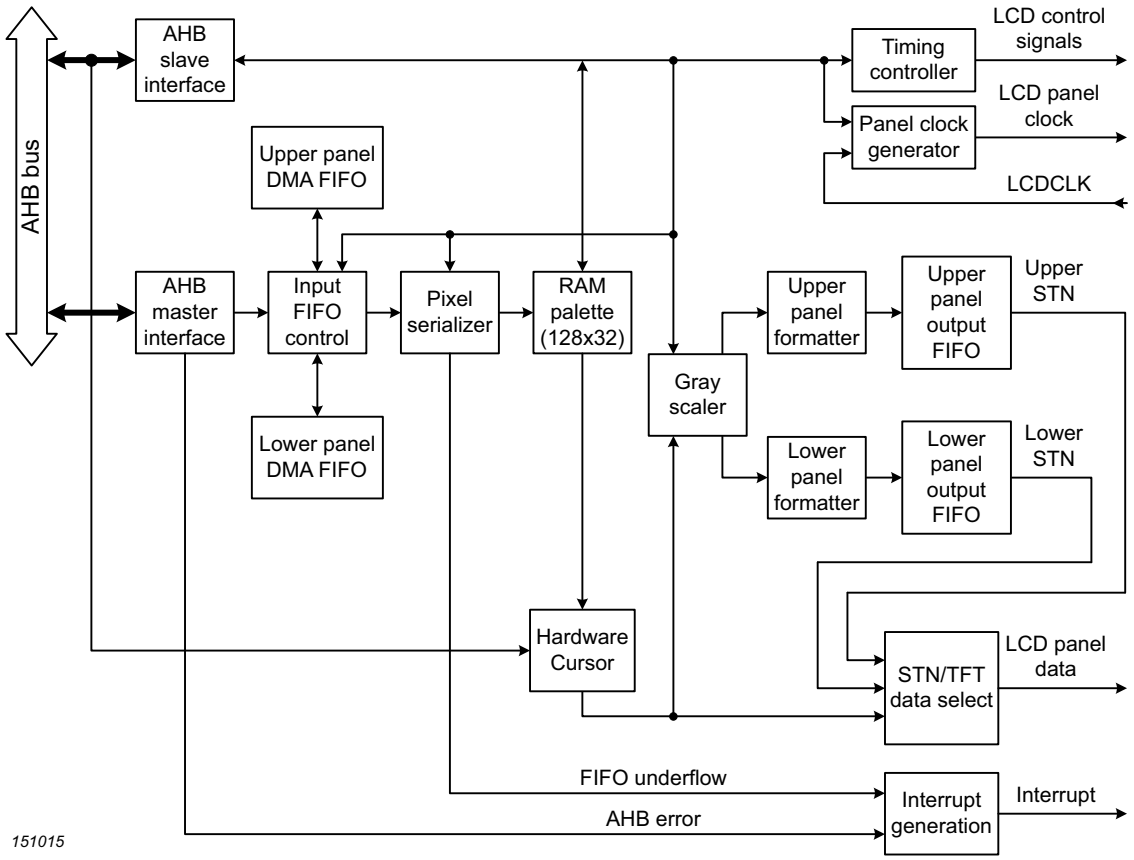


Fig 119. LCD controller block diagram

34.6.1 AHB interfaces

The LCD controller includes two separate AHB interfaces. The first, an AHB slave interface, is used primarily by the CPU to access control and data registers within the LCD controller. The second, an AHB master interface, is used by the LCD controller for DMA access to display data stored in memory elsewhere in the system. The LCD DMA controller can only access the Peripheral SRAMs and the external memory.

34.6.1.1 AMBA AHB slave interface

The AHB slave interface connects the LCD controller to the AHB bus and provides CPU accesses to the registers and palette RAM.

34.6.1.2 AMBA AHB master interface

The AHB master interface transfers display data from a selected slave (memory) to the LCD controller DMA FIFOs. It can be configured to obtain data from the Peripheral SRAMs, various types of off-chip static memory, or off-chip SDRAM.

In dual panel mode, the DMA FIFOs are filled up in an alternating fashion via a single DMA request. In single panel mode, the DMA FIFOs are filled up in a sequential fashion from a single DMA request.

The inherent AHB master interface state machine performs the following functions:

- Loads the upper panel base address into the AHB address incrementer on recognition of a new frame.
- Monitors both the upper and lower DMA FIFO levels and asserts a DMA request to request display data from memory, filling them to above the programmed watermark. the DMA request is reasserted when there are at least four locations available in either FIFO (dual panel mode).
- Checks for 1 KB boundaries during fixed-length bursts, appropriately adjusting the address in such occurrences.
- Generates the address sequences for fixed-length and undefined bursts.
- Controls the handshaking between the memory and DMA FIFOs. It inserts busy cycles if the FIFOs have not completed their synchronization and updating sequence.
- Fills up the DMA FIFOs, in dual panel mode, in an alternating fashion from a single DMA request.
- Asserts the a bus error interrupt if an error occurs during an active burst.
- Responds to retry commands by restarting the failed access. This introduces some busy cycles while it re-synchronizes.

34.6.2 Dual DMA FIFOs and associated control logic

The pixel data accessed from memory is buffered by two DMA FIFOs that can be independently controlled to cover single and dual-panel LCD types. Each FIFO is 16 words deep by 64 bits wide and can be cascaded to form an effective 32-Dword deep FIFO in single panel mode.

Synchronization logic transfers the pixel data from the AHB clock domain to the LCD controller clock domain. The water level marks in each FIFO are set such that each FIFO requests data when at least four locations become available.

An interrupt signal is asserted if an attempt is made to read either of the two DMA FIFOs when they are empty (an underflow condition has occurred).

34.6.3 Pixel serializer

This block reads the 32-bit wide LCD data from the output port of the DMA FIFO and extracts 24, 16, 8, 4, 2, or 1 bpp data, depending on the current mode of operation. The LCD controller supports big-endian, little-endian, and Windows CE data formats.

Depending on the mode of operation, the extracted data can be used to point to a color or gray scale value in the palette RAM or can actually be a true color value that can be directly applied to an LCD panel input.

Table 669 through Table 671 show the structure of the data in each DMA FIFO word corresponding to the endianness and bpp combinations. For each of the three supported data formats, the required data for each panel display pixel must be extracted from the data word.

Table 669. FIFO bits for Little-endian Byte, Little-endian Pixel order

FIFO bit	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
0	p0	p0	p0	p0	p0	p0
1	p1					
2	p2	p1				
3	p3					
4	p4	p2	p1			
5	p5					
6	p6	p3				
7	p7					
8	p8	p4	p2	p1		
9	p9					
10	p10	p5				
11	p11					
12	p12	p6	p3			
13	p13					
14	p14	p7				
15	p15					
16	p16	p8	p4	p2	p1	
17	p17					
18	p18	p9				
19	p19					
20	p20	p10	p5			
21	p21					
22	p22	p11				
23	p23					
24	p24	p12	p6	p3		
25	p25					
26	p26	p13				
27	p27					
28	p28	p14	p7			
29	p29					
30	p30	p15				
31	p31					

Table 670. FIFO bits for Big-endian Byte, Big-endian Pixel order

FIFO bit	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
0	p31	p15	p7	p3	p1	p0
1	p30					
2	p29	p14				
3	p28					
4	p27	p13	p6			
5	p26					
6	p25	p12				
7	p24					
8	p23	p11	p5	p2		
9	p22					
10	p21	p10				
11	p20					
12	p19	p9	p4			
13	p18					
14	p17	p8				
15	p16					
16	p15	p7	p3	p1	p0	
17	p14					
18	p13	p6				
19	p12					
20	p11	p5	p2			
21	p10					
22	p9	p4				
23	p8					
24	p7	p3	p1	p0		
25	p6					
26	p5	p2				
27	p4					
28	p3	p1	p0			
29	p2					
30	p1	p0				
31	p0					

Table 671. FIFO bits for Little-endian Byte, Big-endian Pixel order

FIFO bit	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
0	p7	p3	p1	p0	p0	p0
1	p6					
2	p5	p2				
3	p4					
4	p3	p1	p0			
5	p2					
6	p1	p0				
7	p0					
8	p15	p7	p3	p1		
9	p14					
10	p13	p6				
11	p12					
12	p11	p5	p2			
13	p10					
14	p9	p4				
15	p8					
16	p23	p11	p5	p2	p1	
17	p22					
18	p21	p10				
19	p20					
20	p19	p9	p4			
21	p18					
22	p17	p8				
23	p16					
24	p31	p15	p7	p3		
25	p30					
26	p29	p14				
27	p28					
28	p27	p13	p6			
29	p26					
30	p25	p12				
31	p24					

Table 672 shows the structure of the data in each DMA FIFO word in RGB mode.

Table 672. RGB mode data formats

FIFO data	24-bit RGB	16-bit (1:5:5:5 RGB)	16-bit (5:6:5 RGB)	16-bit (4:4:4 RGB)
0	p0, Red 0	p0, Red 0	p0, Red 0	p0, Red 0
1	p0, Red 1	p0, Red 1	p0, Red 1	p0, Red 1
2	p0, Red 2	p0, Red 2	p0, Red 2	p0, Red 2
3	p0, Red 3	p0, Red 3	p0, Red 3	p0, Red 3
4	p0, Red 4	p0, Red 4	p0, Red 4	p0, Green 0

Table 672. RGB mode data formats

FIFO data	24-bit RGB	16-bit (1:5:5:5 RGB)	16-bit (5:6:5 RGB)	16-bit (4:4:4 RGB)
5	p0, Red 5	p0, Green 0	p0, Green 0	p0, Green 1
6	p0, Red 6	p0, Green 1	p0, Green 1	p0, Green 2
7	p0, Red 7	p0, Green 2	p0, Green 2	p0, Green 3
8	p0, Green 0	p0, Green 3	p0, Green 3	p0, Blue 0
9	p0, Green 1	p0, Green 4	p0, Green 4	p0, Blue 1
10	p0, Green 2	p0, Blue 0	p0, Green 5	p0, Blue 2
11	p0, Green 3	p0, Blue 1	p0, Blue 0	p0, Blue 3
12	p0, Green 4	p0, Blue 2	p0, Blue 1	-
13	p0, Green 5	p0, Blue 3	p0, Blue 2	-
14	p0, Green 6	p0, Blue 4	p0, Blue 3	-
15	p0, Green 7	p0 intensity bit	p0, Blue 4	-
16	p0, Blue 0	p1, Red 0	p1, Red 0	p1, Red 0
17	p0, Blue 1	p1, Red 1	p1, Red 1	p1, Red 1
18	p0, Blue 2	p1, Red 2	p1, Red 2	p1, Red 2
19	p0, Blue 3	p1, Red 3	p1, Red 3	p1, Red 3
20	p0, Blue 4	p1, Red 4	p1, Red 4	p1, Green 0
21	p0, Blue 5	p1, Green 0	p1, Green 0	p1, Green 1
22	p0, Blue 6	p1, Green 1	p1, Green 1	p1, Green 2
23	p0, Blue 7	p1, Green 2	p1, Green 2	p1, Green 3
24	-	p1, Green 3	p1, Green 3	p1, Blue 0
25	-	p1, Green 4	p1, Green 4	p1, Blue 1
26	-	p1, Blue 0	p1, Green 5	p1, Blue 2
27	-	p1, Blue 1	p1, Blue 0	p1, Blue 3
28	-	p1, Blue 2	p1, Blue 1	-
29	-	p1, Blue 3	p1, Blue 2	-
30	-	p1, Blue 4	p1, Blue 3	-
31	-	p1 intensity bit	p1, Blue 4	-

34.6.4 RAM palette

The RAM-based palette is a 256 x 16 bit dual-port RAM physically structured as 128 x 32 bits. Two entries can be written into the palette from a single word write access. The Least Significant Bit (LSB) of the serialized pixel data selects between upper and lower halves of the palette RAM. The half that is selected depends on the byte ordering mode. In little-endian mode, setting the LSB selects the upper half, but in big-endian mode, the lower half of the palette is selected.

Pixel data values can be written and verified through the AHB slave interface. For information on the supported colors, refer to the section on the related panel type earlier in this chapter.

The palette RAM is a dual port RAM with independent controls and addresses for each port. Port1 is used as a read/write port and is connected to the AHB slave interface. The palette entries can be written and verified through this port. Port2 is used as a read-only port and is connected to the unpacker and gray scaler. For color modes of less than 16 bpp, the palette enables each pixel value to be mapped to a 16-bit color:

- For TFT displays, the 16-bit value is passed directly to the pixel serializer.
- For STN displays, the 16-bit value is first converted by the gray scaler.

Table 673 shows the bit representation of the palette data. The palette 16-bit output uses the TFT 1:5:5:5 data format. In 16 and 24 bpp TFT mode, the palette is bypassed and the output of the pixel serializer is used as the TFT panel data.

Table 673. Palette data storage for TFT modes

Bit(s)	Name (RGB format)	Description (RGB format)	Name (BGR format)	Description (BGR format)
4:0	R[4:0]	Red palette data	B[4:0]	Blue palette data
9:5	G[4:0]	Green palette data	G[4:0]	Green palette data
14:10	B[4:0]	Blue palette data	R[4:0]	Red palette data
15	I	Intensity / unused	I	Intensity / unused
20:16	R[4:0]	Red palette data	B[4:0]	Blue palette data
25:21	G[4:0]	Green palette data	G[4:0]	Green palette data
30:26	B[4:0]	Blue palette data	R[4:0]	Red palette data
31	I	Intensity / unused	I	Intensity / unused

The red and blue pixel data can be swapped to support BGR data format using a control register bit 8 (BGR). See the LCD_CTRL register description for more information. Table 674 shows the bit representation of the palette data for the STN color modes.

Table 674. Palette data storage for STN color modes

Bit(s)	Name (RGB format)	Description (RGB format)	Name (BGR format)	Description (BGR format)
0	R[0]	Unused	B[0]	Unused
4:1	R[4:1]	Red palette data	B[4:1]	Blue palette data
5	G[0]	Unused	G[0]	Unused
9:6	G[4:1]	Green palette data	G[4:1]	Green palette data
10	B[0]	Unused	R[0]	Unused
14:11	B[4:1]	Blue palette data	R[4:1]	Red palette data
15	I	Unused	I	Unused
16	-	Unused	-	Unused
20:17	R[3:0]	Red palette data	B[3:0]	Blue palette data
21	-	Unused	-	Unused
25:22	G[3:0]	Green palette data	G[3:0]	Green palette data
26	-	Unused	-	Unused
30:27	B[3:0]	Blue palette data	R[3:0]	Red palette data
31	-	Unused	-	Unused

For monochrome STN mode, only the red palette field bits [4:1] are used. However, in STN color mode the green and blue [4:1] are also used. Only 4 bits per color are used, because the gray scaler only supports 16 different shades per color.

Table 675 shows the bit representation of the palette data for the STN monochrome mode.

Table 675. Palette data storage for STN monochrome mode

Bit(s)	Name	Description
0	-	Unused
4:1	Y[3:0]	Intensity data
16:5	-	Unused
20:17	Y[3:0]	Intensity data
31:21	-	Unused

34.6.5 Hardware cursor

The hardware cursor is an integral part of the LCD controller. It uses the LCD timing module to provide an indication of the current scan position coordinate, and intercepts the pixel stream between the palette logic and the gray scale/output multiplexer.

All cursor programming registers are accessed through the LCD slave interface. This also provides a read/write port to the cursor image RAM.

34.6.5.1 Cursor operation

The hardware cursor is contained in a dual port RAM. It is programmed by software through the AHB slave interface. The AHB slave interface also provides access to the hardware cursor control registers. These registers enable you to modify the cursor position and perform various other functions.

When enabled, the hardware cursor uses the horizontal and vertical synchronization signals, along with a pixel clock enable and various display parameters to calculate the current scan coordinate.

When the display point is inside the bounds of the cursor image, the cursor replaces frame buffer pixels with cursor pixels.

When the last cursor pixel is displayed, an interrupt is generated that software can use as an indication that it is safe to modify the cursor image. This enables software controlled animations to be performed without flickering for frame synchronized cursors.

34.6.5.2 Cursor sizes

Table 676 shows the two cursor sizes that are supported.

Table 676. Supported cursor sizes

X Pixels	Y Pixels	Bits per pixel	Words per line	Words in cursor image
32	32	2	2	64
64	64	2	4	256

34.6.5.3 Cursor movement

The following descriptions assume that both the screen and cursor origins are at the top left of the visible screen (the first visible pixel scanned each frame). Figure 120 shows how each pixel coordinate is assumed to be the top left corner of the pixel.

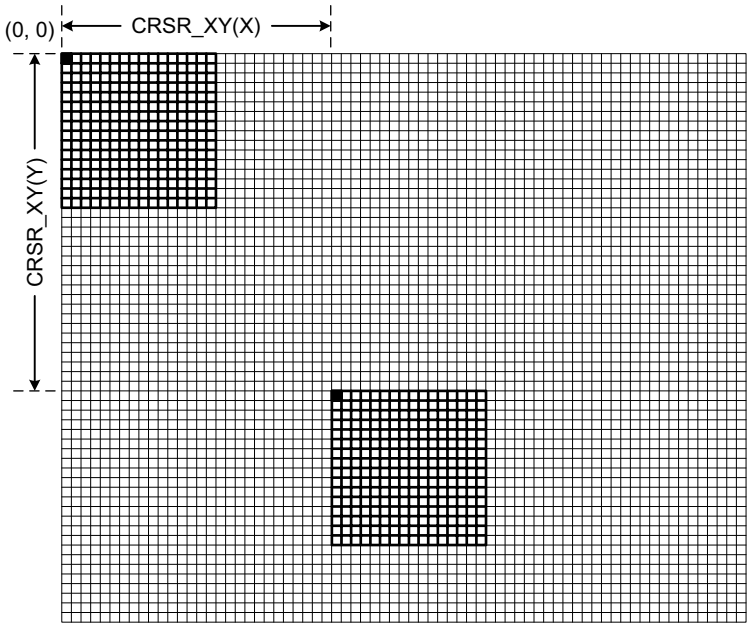


Fig 120. Cursor movement

34.6.5.4 Cursor XY positioning

The CRSR_XY register controls the cursor position on the cursor overlay (see Cursor XY Position register). This provides separate fields for X and Y ordinates.

The CRSR_CFG register (see Cursor Configuration register) provides a FrameSync bit controlling the visible behavior of the cursor.

With FrameSync inactive, the cursor responds immediately to any change in the programmed CRSR_XY value. Some transient smearing effects may be visible if the cursor is moved across the LCD scan line.

With FrameSync active, the cursor only updates its position after a vertical synchronization has occurred. This provides clean cursor movement, but the cursor position only updates once a frame.

34.6.5.5 Cursor clipping

The CRSR_XY register (see Cursor XY Position register) is programmed with positive binary values that enable the cursor image to be located anywhere on the visible screen image. The cursor image is clipped automatically at the screen limits when it extends beyond the screen image to the right or bottom (see X1,Y1 in Figure 121). The checked pattern shows the visible portion of the cursor.

Because the CRSR_XY register values are positive integers, to emulate cursor clipping on the left and top of screen, a Clip Position register, CRSR_CLIP, is provided. This controls which point of the cursor image is positioned at the CRSR_CLIP coordinate. For clipping functions on the Y axis, CRSR_XY(X) is zero, and Clip(X) is programmed to provide the offset into the cursor image (X2 and X3). The equivalent function is provided to clip on the X axis at the top of the display (Y2).

For cursors that are not clipped at the X=0 or Y=0 lines, program the Clip Position register X and Y fields with zero to display the cursor correctly. See Clip(X4,Y4) for the effect of incorrect programming.

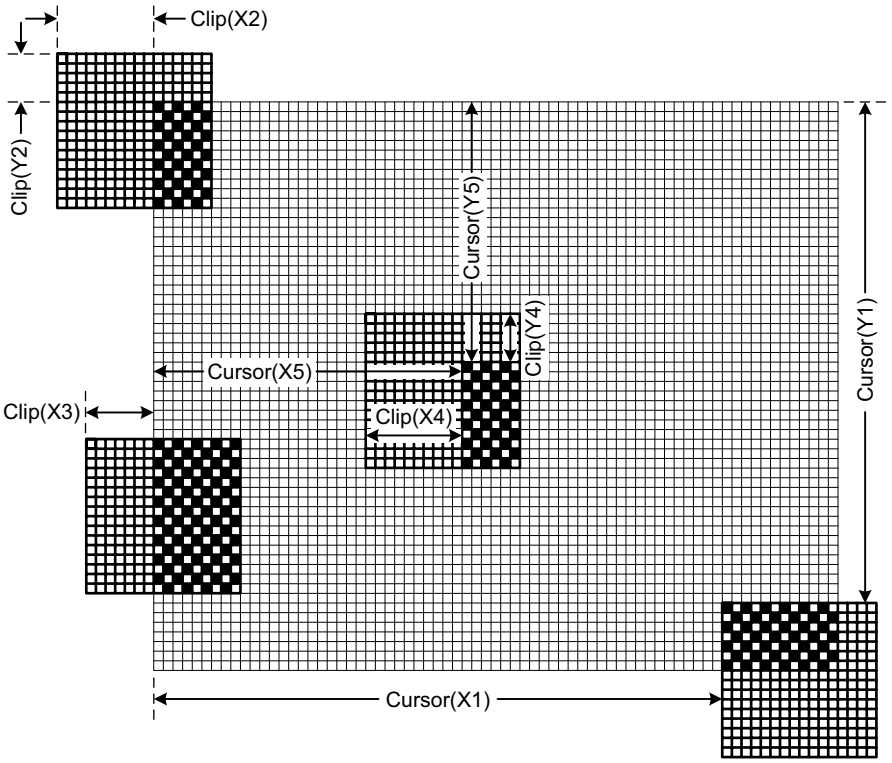


Fig 121. Cursor clipping

34.6.5.6 Cursor image format

The LCD frame buffer supports three packing formats, but the hardware cursor image requirement has been simplified to support only LBBP. This is little-endian byte, big-endian pixel for Windows CE mode.

The Image RAM start address is offset by 0x800 from the LCD base address, as shown in the register description in this chapter.

The displayed cursor coordinate system is expressed in terms of (X,Y). 64 x 64 is an extension of the 32 x 32 format shown in Figure 122.

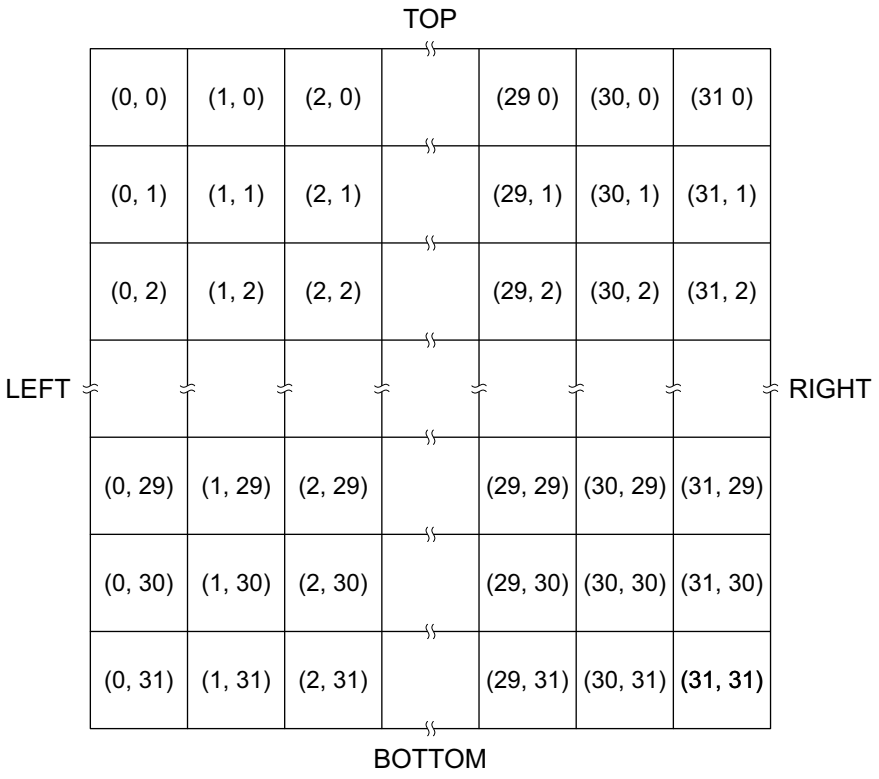


Fig 122. Cursor image format

32 by 32 pixel format

Four cursors are held in memory, each with the same pixel format. Table 677 lists the base addresses for the four cursors.

Table 677. Addresses for 32 x 32 cursors

Offset	Description
0x800	Cursor 0 start address.
0x900	Cursor 1 start address.
0xA00	Cursor 2 start address.
0xB00	Cursor 3 start address.

Table 678 shows the buffer to pixel mapping for Cursor 0.

Table 678. Buffer to pixel mapping for 32 x 32 pixel cursor format

	Offset into cursor memory					
Data bits	**0**	**4**	**(8 * y)**	**(8 * y) +4**	**F8**	**FC**
1:0	(3, 0)	(19, 0)	(3, y)	(19, y)	(3, 31)	(19, 31)
3:2	(2, 0)	(18, 0)	(2, y)	(18, y)	(2, 31)	(18, 31)
5:4	(1, 0)	(17, 0)	(1, y)	(17, y)	(1, 31)	(17, 31)
7:6	(0, 0)	(16, 0)	(0, y)	(16, y)	(0, 31)	(16, 31)

Table 678. Buffer to pixel mapping for 32 x 32 pixel cursor format

	Offset into cursor memory					
Data bits	**0**	**4**	**(8 * y)**	**(8 * y) +4**	**F8**	**FC**
9:8	(7, 0)	(23, 0)	(7, y)	(23, y)	(7, 31)	(23, 31)
11:10	(6, 0)	(22, 0)	(6, y)	(22, y)	(6, 31)	(22, 31)
13:12	(5, 0)	(21, 0)	(5, y)	(21, y)	(5, 31)	(21, 31)
15:14	(4, 0)	(20, 0)	(4, y)	(20, y)	(4, 31)	(20, 31)
17:16	(11, 0)	(27, 0)	(11, y)	(27, y)	(11, 31)	(27, 31)
19:18	(10, 0)	(26, 0)	(10, y)	(26, y)	(10, 31)	(26, 31)
21:20	(9, 0)	(25, 0)	(9, y)	(25, y)	(9, 31)	(25, 31)
23:22	(8, 0)	(24, 0)	(8, y)	(24, y)	(8, 31)	(24, 31)
25:24	(15, 0)	(31, 0)	(15, y)	(31, y)	(15, 31)	(31, 31)
27:26	(14, 0)	(30, 0)	(14, y)	(30, y)	(14, 31)	(30, 31)
29:28	(13, 0)	(29, 0)	(13, y)	(29, y)	(13, 31)	(29, 31)
31:30	(12, 0)	(28, 0)	(12, y)	(28, y)	(12, 31)	(28,31)

64 by 64 pixel format

Only one cursor fits in the memory space in 64 x 64 mode, as detailed in Table 679.

Table 679. Buffer to pixel mapping for 64 x 64 pixel cursor format

	Offset into cursor memory								
Data bits	**0**	**4**	**8**	**12**	**(16 * y)**	**(16 * y) +4**	**(16 * y) + 8**	**(16 * y) + 12**	**FC**
1:0	(3, 0)	(19, 0)	(35, 0)	(51, 0)	(3, y)	(19, y)	(35, y)	(51, y)	(51, 63)
3:2	(2, 0)	(18, 0)	(34, 0)	(50, 0)	(2, y)	(18, y)	(34, y)	(50, y)	(50, 63)
5:4	(1, 0)	(17, 0)	(33, 0)	(49, 0)	(1, y)	(17, y)	(33, y)	(49, y)	(49, 63)
7:6	(0, 0)	(16, 0)	(32, 0)	(48, 0)	(0, y)	(16, y)	(32, y)	(48, y)	(48, 63)
9:8	(7, 0)	(23, 0)	(39, 0)	(55, 0)	(7, y)	(23, y)	(39, y)	(55, y)	(55, 63)
11:10	(6, 0)	(22, 0)	(38, 0)	(54, 0)	(6, y)	(22, y)	(38, y)	(54, y)	(54, 63)
13:12	(5, 0)	(21, 0)	(37, 0)	(53, 0)	(5, y)	(21, y)	(37, y)	(53, y)	(53, 63)
15:14	(4, 0)	(20, 0)	(36, 0)	(52, 0)	(4, y)	(20, y)	(36, y)	(52, y)	(52, 63)
17:16	(11, 0)	(27, 0)	(43, 0)	(59, 0)	(11, y)	(27, y)	(43, y)	(59, y)	(59, 63)
19:18	(10, 0)	(26, 0)	(42, 0)	(58, 0)	(10, y)	(26, y)	(42, y)	(58, y)	(58, 63)
21:20	(9, 0)	(25, 0)	(41, 0)	(57, 0)	(9, y)	(25, y)	(41, y)	(57, y)	(57, 63)
23:22	(8, 0)	(24, 0)	(40, 0)	(56, 0)	(8, y)	(24, y)	(40, y)	(56, y)	(56, 63)
25:24	(15, 0)	(31, 0)	(47, 0)	(63, 0)	(15, y)	(31, y)	(47, y)	(63, y)	(63, 63)
27:26	(14, 0)	(30, 0)	(46, 0)	(62, 0)	(14, y)	(30, y)	(46, y)	(62, y)	(62, 63)
29:28	(13, 0)	(29, 0)	(45, 0)	(61, 0)	(13, y)	(29, y)	(45, y)	(61, y)	(61, 63)
31:30	(12, 0)	(28, 0)	(44, 0)	(60, 0)	(12, y)	(28, y)	(44, y)	(60, y)	(60, 63)

Cursor pixel encoding

Each pixel of the cursor requires two bits of information. These are interpreted as Color0, Color1, Transparent, and Transparent inverted.

In the coding scheme, bit 1 selects between color and transparent (AND mask) and bit 0 selects variant (XOR mask).

Table 680 shows the pixel encoding bit assignments.

Table 680. Pixel encoding

Value	Description
00	Color0. The cursor color is displayed according to the Red-Green-Blue (RGB) value programmed into the CRSR_PAL0 register.
01	Color1. The cursor color is displayed according to the RGB value programmed into the CRSR_PAL1 register.
10	Transparent. The cursor pixel is transparent, so is displayed unchanged. This enables the visible cursor to assume shapes that are not square.
11	Transparent inverted. The cursor pixel assumes the complementary color of the frame pixel that is displayed. This can be used to ensure that the cursor is visible regardless of the color of the frame buffer image.

34.6.6 Gray scaler

A patented gray scale algorithm drives monochrome and color STN panels. This provides 15 gray scales for monochrome displays. For STN color displays, the three color components (RGB) are gray scaled simultaneously. This results in 3375 (15x15x15) colors being available. The gray scaler transforms each 4-bit gray value into a sequence of activity-per-pixel over several frames, relying to some degree on the display characteristics, to give the representation of gray scales and color.

34.6.7 Upper and lower panel formatters

Formatters are used in STN mode to convert the gray scaler output to a parallel format as required by the display. For monochrome displays, this is either 4 or 8 bits wide, and for color displays, it is 8 bits wide. Table 681 shows a color display driven with 2 2/3 pixels worth of data in a repeating sequence.

Table 681. Color display driven with 2 2/3 pixel data

Byte	CLD[7]	CLD[6]	CLD[5]	CLD[4]	CLD[3]	CLD[2]	CLD[1]	CLD[0]
0	P2[Green]	P2[Red]	P1[Blue]	P1[Green]	P1[Red]	P0[Blue]	P0[Green]	P0[Red]
1	P5[Red]	P4q[Blue]	P4[Green]	P4[Red]	P3[Blue]	P3[Green]	P3[Red]	P2[Blue]
2	P7[Blue]	P7[Green]	P7[Red]	P6[Blue]	P6[Green]	P6[Red]	P5[Blue]	P5[Green]

Each formatter consists of three 3-bit (RGB) shift left registers. RGB pixel data bit values from the gray scaler are concurrently shifted into the respective registers. When enough data is available, a byte is constructed by multiplexing the registered data to the correct bit position to satisfy the RGB data pattern of LCD panel. The byte is transferred to the 3-byte FIFO, which has enough space to store eight color pixels.

34.6.8 Panel clock generator

The output of the panel clock generator block is the panel clock, pin LCD_DCLK. The panel clock is a divided version of LCDCLK - the peripheral clock for the LCD block. The source of LCDCLK is selected in the Syscon (CGU) from among a variety of on-chip clock sources or from an external LCD_CLKIN pin.

The panel clock generator can be programmed to output the LCD panel clock in the range of LCDCLK/2 to LCDCLK/1025 to match the bpp data rate of the LCD panel being used.

The CLKSEL bit in the LCD_POL register determines whether the base clock used is CCLK or the LCD_CLKIN pin.

34.6.9 Timing controller

The primary function of the timing controller block is to generate the horizontal and vertical timing panel signals. It also provides the panel bias and enable signals. These timings are all register-programmable.

34.6.10 STN and TFT data select

Support is provided for passive Super Twisted Nematic (STN) and active Thin Film Transistor (TFT) LCD display types:

34.6.10.1 STN displays

STN display panels require algorithmic pixel pattern generation to provide pseudo gray scaling on monochrome displays, or color creation on color displays.

34.6.10.2 TFT displays

TFT display panels require the digital color value of each pixel to be applied to the display data inputs.

34.6.11 Interrupt generation

Four interrupts are generated by the LCD controller, and a single combined interrupt. The four interrupts are:

- Master bus error interrupt.
- Vertical compare interrupt.
- Next base address update interrupt.
- FIFO underflow interrupt.

Each of the four individual maskable interrupts is enabled or disabled by changing the mask bits in the LCD_INT_MSK register. These interrupts are also combined into a single overall interrupt, which is asserted if any of the individual interrupts are both asserted and unmasked. Provision of individual outputs in addition to a combined interrupt output enables use of either a global interrupt service routine, or modular device drivers to handle interrupts.

The status of the individual interrupt sources can be read from the LCD_INTRAW register.

34.6.11.1 Master bus error interrupt

The master bus error interrupt is asserted when an ERROR response is received by the master interface during a transaction with a slave. When such an error is encountered, the master interface enters an error state and remains in this state until clearance of the error has been signaled to it. When the respective interrupt service routine is complete, the master bus error interrupt may be cleared by writing a 1 to the BERIC bit in the LCD_INTCLR register. This action releases the master interface from its ERROR state to the start of FRAME state, and enables fresh frame of data display to be initiated.

34.6.11.2 Vertical compare interrupt

The vertical compare interrupt asserts when one of four vertical display regions, selected using the LCD_CTRL register, is reached. The interrupt can be made to occur at the start of:

- Vertical synchronization.
- Back porch.
- Active video.
- Front porch.

The interrupt may be cleared by writing a 1 to the VcompIC bit in the LCD_INTCLR register.

34.6.11.2.1 Next base address update interrupt

The LCD next base address update interrupt asserts when either the LCDUPBASE or LCDLPBASE values have been transferred to the LCDUPCURR or LCDLPCURR incrementers respectively. This signals to the system that it is safe to update the LCDUPBASE or the LCDLPBASE registers with new frame base addresses if required.

The interrupt can be cleared by writing a 1 to the LNBUIC bit in the LCD_INTCLR register

34.6.11.2.2 FIFO underflow interrupt

The FIFO underflow interrupt asserts when internal data is requested from an empty DMA FIFO. Internally, upper and lower panel DMA FIFO underflow interrupt signals are generated.

The interrupt can be cleared by writing a 1 to the FUFIC bit in the LCD_INTCLR register.

34.6.12 LCD power-up and power-down sequence

The LCD controller requires the following power-up sequence to be performed:

1. When power is applied, the following signals are held LOW:

- LCD_LP
- LCD_DCLK
- LCD_FP
- LCD_AC
- LCD_VD[23:0]
- LCD_LE

2. When LCD power is stabilized, a 1 is written to the LcdEn bit in the LCD_CTRL register. This enables the following signals into their active states:

- LCD_LP
- LCD_DCLK
- LCD_FP
- LCD_AC
- LCD_LE

The LCD_VD[23:0] signals remain in an inactive state.

3. When the signals in step 2 have stabilized, the contrast voltage (not controlled or supplied by the LCD controller) is applied to the LCD panel.

4. If required, a software or hardware timer can be used to provide the minimum display specific delay time between application of the control signals and power to the panel display. On completion of the time interval, power is applied to the panel by writing a 1 to the LcdPwr bit within the LCD_CTRL register that, in turn, sets the LCD_PWR signal high and enables the LCD_VD[23:0] signals into their active states. The LCD_PWR signal is intended to be used to gate the power to the LCD panel.

The power-down sequence is the reverse of the above four steps and must be strictly followed, this time, writing the respective register bits with 0.

Figure 123 shows the power-up and power-down sequences.

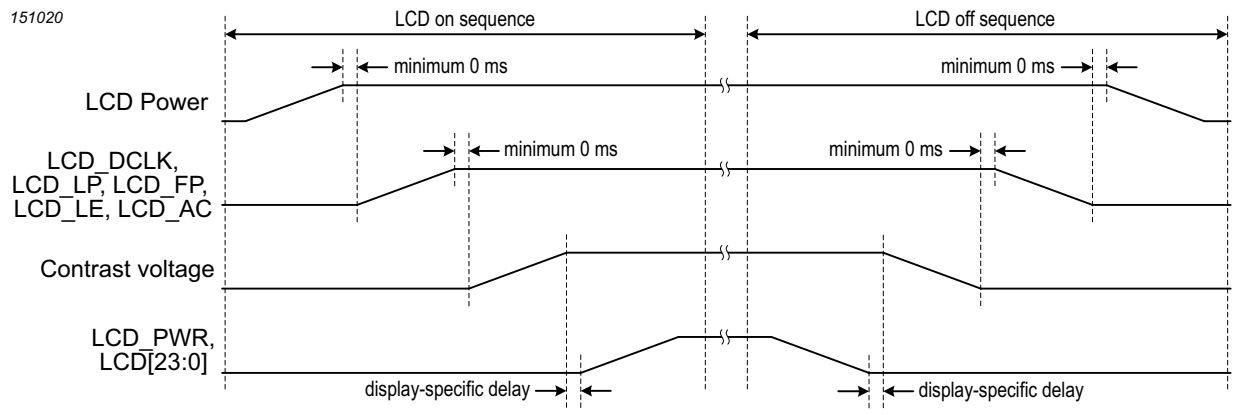


Fig 123. Power-up and power-down sequences

34.7 Register description

Table 682. Register overview: LCD controller (base address 0x4008 3000)

Name	Access	Address offset	Description	Reset value [1]	Section
TIMH	R/W	0x000	Horizontal Timing Control	0	34.7.1
TIMV	R/W	0x004	Vertical Timing Control	0	34.7.2
POL	R/W	0x008	Clock and Signal Polarity Control	0	34.7.3
LE	R/W	0x00C	Line End Control	0	34.7.4
UPBASE	R/W	0x010	Upper Panel Frame Base Address	0	34.7.5
LPBASE	R/W	0x014	Lower Panel Frame Base Address	0	34.7.6
CTRL	R/W	0x018	LCD Control	0	34.7.7
INTMSK	R/W	0x01C	Interrupt Mask	0	34.7.8
INTRAW	RO	0x020	Raw Interrupt Status	0	34.7.9
INTSTAT	RO	0x024	Masked Interrupt Status	0	34.7.10
INTCLR	WO	0x028	Interrupt Clear	-	34.7.11
UPCURR	RO	0x02C	Upper Panel Current Address Value	0	34.7.12
LPCURR	RO	0x030	Lower Panel Current Address Value	0	34.7.13
PAL0 to PAL127	R/W	0x200 to 0x3FC	256x16-bit Color Palette registers	0	34.7.14
CRSR_IMG0 to CRSR_IMG255	R/W	0x800 to 0xBFC	Cursor Image registers	0	34.7.15
CRSR_CTRL	R/W	0xC00	Cursor Control	0	34.7.16
CRSR_CFG	R/W	0xC04	Cursor Configuration	0	34.7.17
CRSR_PAL0	R/W	0xC08	Cursor Palette register 0	0	34.7.18
CRSR_PAL1	R/W	0xC0C	Cursor Palette register 1	0	34.7.19
CRSR_XY	R/W	0xC10	Cursor XY Position	0	34.7.20
CRSR_CLIP	R/W	0xC14	Cursor Clip Position	0	34.7.21
CRSR_INTMSK	R/W	0xC20	Cursor Interrupt Mask	0	34.7.22
CRSR_INTCLR	WO	0xC24	Cursor Interrupt Clear	-	34.7.23
CRSR_INTRAW	RO	0xC28	Cursor Raw Interrupt Status	0	34.7.24
CRSR_INTSTAT	RO	0xC2C	Cursor Masked Interrupt Status	0	34.7.22

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

34.7.1 Horizontal Timing register

The LCD_TIMH register controls the Horizontal Synchronization pulse Width (HSW), the Horizontal Front Porch (HFP) period, the Horizontal Back Porch (HBP) period, and the Pixels-Per-Line (PPL).

Table 683. Horizontal Timing register (TIMH, offset 0x000) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	-
7:2	PPL	Pixels-per-line. The PPL bit field specifies the number of pixels in each line or row of the screen. PPL is a 6-bit value that represents between 16 and 1024 pixels per line. PPL counts the number of panel clocks that occur before the HFP is applied. Program the value required divided by 16, minus 1. Actual pixels-per-line = 16 * (PPL + 1). For example, to obtain 320 pixels per line, program PPL as (320/16) -1 = 19.	0
15:8	HSW	Horizontal synchronization pulse width. The 8-bit HSW field specifies the pulse width of the line clock in passive mode, or the horizontal synchronization pulse in active mode. Program with desired value minus 1.	0
23:16	HFP	Horizontal front porch. The 8-bit HFP field sets the number of panel clock intervals at the end of each line or row of pixels, before the LCD line clock is pulsed. When a complete line of pixels is transmitted to the LCD driver, the value in HFP counts the number of panel clocks to wait before asserting the line clock. HFP can generate a period of 1-256 panel clock cycles. Program with desired value minus 1.	0
31:24	HBP	Horizontal back porch. The 8-bit HBP field is used to specify the number of panel clock periods inserted at the beginning of each line or row of pixels. After the line clock for the previous line has been deasserted, the value in HBP counts the number of panel clocks to wait before starting the next display line. HBP can generate a delay of 1-256 panel clock cycles. Program with desired value minus 1.	0

34.7.1.1 Horizontal timing restrictions

DMA requests new data at the start of a horizontal display line. Some time must be allowed for the DMA transfer and for data to propagate down the FIFO path in the LCD interface. The data path latency forces some restrictions on the usable minimum values for horizontal porch width in STN mode. The minimum values are HSW = 2 and HBP = 2.

Single panel mode:

- HSW = 3 panel clock cycles
- HBP = 5 panel clock cycles
- HFP = 5 panel clock cycles
- Panel Clock Divisor (PCD) = 1 (LCDCLK / 3)

Dual panel mode:

- HSW = 3 panel clock cycles
- HBP = 5 panel clock cycles
- HFP = 5 panel clock cycles
- PCD = 5 (LCDCLK / 7)

If enough time is given at the start of the line, for example, setting HSW = 6, HBP = 10, data does not corrupt for PCD = 4, the minimum value.

34.7.2 Vertical Timing register

The LCD_TIMV register controls the Vertical Synchronization pulse Width (VSW), the Vertical Front Porch (VFP) period, the Vertical Back Porch (VBP) period, and the Lines-Per-Panel (LPP).

Table 684. Vertical Timing register (TIMV, offset 0x004) bit description

Bits	Symbol	Description	Reset value
9:0	LPP	Lines per panel. This is the number of active lines per screen. The LPP field specifies the total number of lines or rows on the LCD panel being controlled. LPP is a 10-bit value allowing between 1 and 1024 lines. Program the register with the number of lines per LCD panel, minus 1. For dual panel displays, program the register with the number of lines on each of the upper and lower panels.	0
15:10	VSW	Vertical synchronization pulse width. This is the number of horizontal synchronization lines. The 6-bit VSW field specifies the pulse width of the vertical synchronization pulse. Program the register with the number of lines required, minus one. The number of horizontal synchronization lines must be small (for example, program to zero) for passive STN LCDs. The higher the value the worse the contrast on STN LCDs.	0
23:16	VFP	Vertical front porch. This is the number of inactive lines at the end of a frame, before the vertical synchronization period. The 8-bit VFP field specifies the number of line clocks to insert at the end of each frame. When a complete frame of pixels is transmitted to the LCD display, the value in VFP is used to count the number of line clock periods to wait. After the count has elapsed, the vertical synchronization signal, LCD_FP, is asserted in active mode, or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates 0–255 line clock cycles. Program to zero on passive displays for improved contrast.	0
31:24	VBP	Vertical back porch. This is the number of inactive lines at the start of a frame, after the vertical synchronization period. The 8-bit VBP field specifies the number of line clocks inserted at the beginning of each frame. The VBP count starts immediately after the vertical synchronization signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit field in passive mode. After this has occurred, the count value in VBP sets the number of line clock periods inserted before the next frame. VBP generates 0 to 255 extra line clock cycles. Program to zero on passive displays for improved contrast.	0

34.7.3 Clock and Signal Polarity register

The LCD_POL register controls various details of clock timing and signal polarity.

Table 685. Clock and Signal Polarity register (POL, offset 0x008) bit description

Bits	Symbol	Description	Reset value
4:0	PCD_LO	Lower five bits of panel clock divisor. The ten-bit PCD field, comprising PCD_HI (bits 31:27 of this register) and PCD_LO, is used to derive the LCD panel clock frequency LCD_DCLK from the input clock, LCD_DCLK = LCDCLK/(PCD+2). For monochrome STN displays with a 4 or 8-bit interface, the panel clock is a factor of four or eight down from the actual individual pixel rate. For color STN displays, 2-2/3 pixels are output per LCD_DCLK cycle, so the panel clock is 0.375 times the pixel rate. For TFT displays, the pixel clock divider can be bypassed by setting the BCD bit in this register. Note: data path latency forces some restrictions on the usable minimum values for the panel clock divider in STN modes: Single panel color mode, PCD = 1 (LCD_DCLK = LCDCLK/3). Dual panel color mode, PCD = 4 (LCD_DCLK = LCDCLK/6). Single panel monochrome 4-bit interface mode, PCD = 2(LCD_DCLK = LCDCLK/4). Dual panel monochrome 4-bit interface mode and single panel monochrome 8-bit interface mode, PCD = 6(LCD_DCLK = LCDCLK/8). Dual panel monochrome 8-bit interface mode, PCD = 14(LCD_DCLK = LCDCLK/16).	0
5	-	Reserved. Read value is undefined, only zero should be written.	-
10:6	ACB	AC bias pin frequency. The AC bias pin frequency is only applicable to STN displays. These require the pixel voltage polarity to periodically reverse to prevent damage caused by DC charge accumulation. Program this field with the required value minus one to apply the number of line clocks between each toggle of the AC bias pin, LCD_AC. This field has no effect if the LCD is operating in TFT mode, when the LCD_AC pin is used as a data enable signal.	0
11	IVS	Invert vertical synchronization. The IVS bit inverts the polarity of the LCD_FP signal. 0 = LCD_FP pin is active HIGH and inactive LOW. 1 = LCD_FP pin is active LOW and inactive HIGH.	0
12	IHS	Invert horizontal synchronization. The IHS bit inverts the polarity of the LCD_LP signal. 0 = LCD_LP pin is active HIGH and inactive LOW. 1 = LCD_LP pin is active LOW and inactive HIGH.	0
13	IPC	Invert panel clock. The IPC bit selects the edge of the panel clock on which pixel data is driven out onto the LCD data lines. 0 = Data is driven on the LCD data lines on the rising edge of LCD_DCLK. 1 = Data is driven on the LCD data lines on the falling edge of LCD_DCLK.	0
14	IOE	Invert output enable. This bit selects the active polarity of the output enable signal in TFT mode. In this mode, the LCD_AC pin is used as an enable that indicates to the LCD panel when valid display data is available. In active display mode, data is driven onto the LCD data lines at the programmed edge of LCD_DCLK when LCD_AC is in its active state. 0 = LCD_AC output pin is active HIGH in TFT mode. 1 = LCD_AC output pin is active LOW in TFT mode.	0
15	-	Reserved. Read value is undefined, only zero should be written.	-

Table 685. Clock and Signal Polarity register (POL, offset 0x008) bit description

Bits	Symbol	Description	Reset value
25:16	CPL	Clocks per line. This field specifies the number of actual LCD_DCLK clocks to the LCD panel on each line. This is the number of PPL divided by either 1 (for TFT), 4 or 8 (for monochrome passive), 2 2/3 (for color passive), minus one. This must be correctly programmed in addition to the PPL bit in the LCD_TIMH register for the LCD display to work correctly.	0
26	BCD	Bypass panel clock divider. Setting this to 1 bypasses the panel clock divider logic. This is mainly used for TFT displays.	0
31:27	PCD_HI	Upper five bits of panel clock divisor. See description for PCD_LO, in bits [4:0] of this register.	0

34.7.4 Line End Control register

The LCD_LE register controls the enabling of line-end signal LCD_LE. When enabled, a positive pulse, four LCDCLK periods wide, is output on LCD_LE after a programmable delay, LED, from the last pixel of each display line. If the line-end signal is disabled it is held permanently LOW.

Table 686. Line End Control register (LE, offset 0x00C) bit description

Bits	Symbol	Description	Reset value
6:0	LED	Line-end delay. Controls Line-end signal delay from the rising-edge of the last panel clock, LCD_DCLK. Program with the number of LCDCLK clock periods minus 1.	0
15:7	-	Reserved. Read value is undefined, only zero should be written.	-
16	LEE	LCD Line end enable. 0 = LCD_LE disabled (held LOW). 1 = LCD_LE signal active.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.5 Upper Panel Frame Base Address register

The LCD_UPBASE register is the color LCD upper panel DMA base address register, and is used to program the base address of the frame buffer for the upper panel. LCDUPBase (and LCDLPBase for dual panels) must be initialized before enabling the LCD controller. The base address must be doubleword aligned.

Optionally, the value may be changed mid-frame to create double buffered video displays. These registers are copied to the corresponding current registers at each LCD vertical synchronization. This event causes the LNBU bit and an optional interrupt to be generated. The interrupt can be used to reprogram the base address when generating double buffered video.

Table 687. Upper Panel Frame Base register (UPBASE, offset 0x010) bit description

Bits	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
31:3	LCDUPBASE	LCD upper panel base address. This is the start address of the upper panel frame data in memory and is doubleword aligned.	0

34.7.6 Lower Panel Frame Base Address register

The LCD_LPBASE register is the color LCD lower panel DMA base address register, and is used to program the base address of the frame buffer for the lower panel. LCDLPBase must be initialized before enabling the LCD controller. The base address must be doubleword aligned.

Optionally, the value may be changed mid-frame to create double buffered video displays. These registers are copied to the corresponding current registers at each LCD vertical synchronization. This event causes the LNBU bit and an optional interrupt to be generated. The interrupt can be used to reprogram the base address when generating double buffered video.

The contents of the LCD_LPBASE register are described in Table 688.

Table 688. Lower Panel Frame Base register (LPBASE, offset 0x014) bit description

Bits	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
31:3	LCDLPBASE	LCD lower panel base address. This is the start address of the lower panel frame data in memory and is doubleword aligned.	0

34.7.7 LCD Control register

The LCD_CTRL register controls the LCD operating mode and the panel pixel parameters.

The contents of the LCD_CTRL register are described in Table 689.

Table 689. LCD Control register (CTRL, offset 0x018) bit description

Bits	Symbol	Description	Reset value
0	LCDEN	LCD enable control bit. 0 = LCD disabled. Signals LCD_LP, LCD_DCLK, LCD_FP, LCD_AC, and LCD_LE are low. 1 = LCD enabled. Signals LCD_LP, LCD_DCLK, LCD_FP, LCD_AC, and LCD_LE are high. See LCD power-up and power-down sequence for details on LCD power sequencing.	0
3:1	LCDBPP	LCD bits per pixel. Selects the number of bits per LCD pixel: 000 = 1 bpp. 001 = 2 bpp. 010 = 4 bpp. 011 = 8 bpp. 100 = 16 bpp. 101 = 24 bpp (TFT panel only). 110 = 16 bpp, 5:6:5 mode. 111 = 12 bpp, 4:4:4 mode.	0
4	LCDBW	STN LCD monochrome/color selection. 0 = STN LCD is color. 1 = STN LCD is monochrome. This bit has no meaning in TFT mode.	0
5	LCDTFT	LCD panel TFT type selection. 0 = LCD is an STN display. Use gray scaler. 1 = LCD is a TFT display. Do not use gray scaler.	0
6	LCDMONO8	Monochrome LCD interface width. Controls whether a monochrome STN LCD uses a 4 or 8-bit parallel interface. It has no meaning in other modes and must be programmed to zero. 0 = monochrome LCD uses a 4-bit interface. 1 = monochrome LCD uses a 8-bit interface.	0
7	LCDDUAL	Single or Dual LCD panel selection. STN LCD interface is: 0 = single-panel. 1 = dual-panel.	0
8	BGR	Color format selection. 0 = RGB: normal output. 1 = BGR: red and blue swapped.	0
9	BEBO	Big-endian Byte Order. Controls byte ordering in memory: 0 = little-endian byte order. 1 = big-endian byte order.	0

Table 689. LCD Control register (CTRL, offset 0x018) bit description

Bits	Symbol	Description	Reset value
10	BEPO	Big-Endian Pixel Ordering. Controls pixel ordering within a byte: 0 = little-endian ordering within a byte. 1 = big-endian pixel ordering within a byte. The BEPO bit selects between little and big-endian pixel packing for 1, 2, and 4 bpp display modes, it has no effect on 8 or 16 bpp pixel formats. See Pixel serializer for more information on the data format.	0
11	LCDPWR	LCD power enable. 0 = power not gated through to LCD panel and LCD_VD[23:0] signals disabled, (held LOW). 1 = power gated through to LCD panel and LCD_VD[23:0] signals enabled, (active). See LCD power-up and power-down sequence for details on LCD power sequencing.	0
13:12	LCDVCOMP	LCD Vertical Compare Interrupt. Generate VComp interrupt at: 00 = start of vertical synchronization. 01 = start of back porch. 10 = start of active video. 11 = start of front porch.	0
15:14	-	Reserved. Read value is undefined, only zero should be written.	-
16	WATERMARK	LCD DMA FIFO watermark level. Controls when DMA requests are generated: 0 = An LCD DMA request is generated when either of the DMA FIFOs have four or more empty locations. 1 = An LCD DMA request is generated when either of the DMA FIFOs have eight or more empty locations.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.8 Interrupt Mask register

The LCD_INTMSK register controls whether various LCD interrupts occur.Setting bits in this register enables the corresponding raw interrupt LCD_INTRAW status bit values to be passed to the LCD_INTSTAT register for processing as interrupts.

The contents of the LCD_INTMSK register are described in Table 690.

Table 690. Interrupt Mask register (INTMSK, offset 0x01C) bit description

Bits	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only zero should be written.	-
1	FUFIM	FIFO underflow interrupt enable. 0: The FIFO underflow interrupt is disabled. 1: Interrupt will be generated when the FIFO underflows.	0
2	LNBUIM	LCD next base address update interrupt enable. 0: The base address update interrupt is disabled. 1: Interrupt will be generated when the LCD base address registers have been updated from the next address registers.	0
3	VCOMPIM	Vertical compare interrupt enable. 0: The vertical compare time interrupt is disabled. 1: Interrupt will be generated when the vertical compare time (as defined by LcdVComp field in the LCD_CTRL register) is reached.	0
4	BERIM	AHB master error interrupt enable. 0: The AHB Master error interrupt is disabled. 1: Interrupt will be generated when an AHB Master error occurs.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.9 Raw Interrupt Status register

The LCD_INTRAW register contains status flags for various LCD controller events. These flags can generate an interrupts if enabled by mask bits in the LCD_INTMSK register.

Table 691. Raw Interrupt Status register (INTRAW, offset 0x020) bit description

Bits	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only zero should be written.	-
1	FUFRIS	FIFO underflow raw interrupt status. Set when either the upper or lower DMA FIFOs have been read accessed when empty causing an underflow condition to occur. Generates an interrupt if the FUFIM bit in the LCD_INTMSK register is set.	
2	LNBURIS	LCD next address base update raw interrupt status. Mode dependent. Set when the current base address registers have been successfully updated by the next address registers. Signifies that a new next address can be loaded if double buffering is in use. Generates an interrupt if the LNBUIM bit in the LCD_INTMSK register is set.	0
3	VCOMPRIS	Vertical compare raw interrupt status. Set when one of the four vertical regions is reached, as selected by the LcdVComp bits in the LCD_CTRL register. Generates an interrupt if the VCompIM bit in the LCD_INTMSK register is set.	0
4	BERRAW	AHB master bus error raw interrupt status. Set when the AHB master interface receives a bus error response from a slave. Generates an interrupt if the BERIM bit in the LCD_INTMSK register is set.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.10 Masked Interrupt Status register

The LCD_INTSTAT register is Read-Only, and contains a bit-by-bit logical AND of the LCD_INTRAW register and the LCD_INTMASK register. A logical OR of all interrupts is provided to the system interrupt controller.

Table 692. Masked Interrupt Status register (INTSTAT, offset 0x024) bit description

Bits	Symbol	Description	Reset value
0	-	Reserved. The value read from a reserved bit is not defined.	-
1	FUFMIS	FIFO underflow masked interrupt status. Set when the both the FUFRIS bit in the LCD_INTRAW register and the FUFIM bit in the LCD_INTMSK register are set.	0
2	LNBUMIS	LCD next address base update masked interrupt status. Set when the both the LNBURIS bit in the LCD_INTRAW register and the LNBUIM bit in the LCD_INTMSK register are set.	0
3	VCOMPMIS	Vertical compare masked interrupt status. Set when the both the VCompRIS bit in the LCD_INTRAW register and the VCompIM bit in the LCD_INTMSK register are set.	0
4	BERMIS	AHB master bus error masked interrupt status. Set when the both the BERRAW bit in the LCD_INTRAW register and the BERIM bit in the LCD_INTMSK register are set.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.11 Interrupt Clear register

The LCD_INTCLR register is Write-Only. Writing a logic 1 to the relevant bit clears the corresponding interrupt.

Table 693. Interrupt Clear register (INTCLR, offset 0x028) bit description

Bits	Symbol	Description
0	-	Reserved. Read value is undefined, only zero should be written.
1	FUFIC	FIFO underflow interrupt clear. Writing a 1 to this bit clears the FIFO underflow interrupt.
2	LNBUIC	LCD next address base update interrupt clear. Writing a 1 to this bit clears the LCD next address base update interrupt.
3	VCOMPIC	Vertical compare interrupt clear. Writing a 1 to this bit clears the vertical compare interrupt.
4	BERIC	AHB master error interrupt clear. Writing a 1 to this bit clears the AHB master error interrupt.
31:5	-	Reserved. Read value is undefined, only zero should be written.

34.7.12 Upper Panel Current Address register

The LCD_UPCURR register is Read-Only, and contains an approximate value of the upper panel data DMA address when read.

Note: This register can change at any time and therefore can only be used as a rough indication of display position.

Table 694. Upper Panel Current Address register (UPCURR, offset 0x02C) bit description

Bits	Symbol	Description	Reset value
31:0	LCDUPCURR	LCD Upper Panel Current Address. Contains the current LCD upper panel data DMA address.	0

34.7.13 Lower Panel Current Address register

The LCD_LPCURR register is Read-Only, and contains an approximate value of the lower panel data DMA address when read.

Note: This register can change at any time and therefore can only be used as a rough indication of display position.

Table 695. Lower Panel Current Address register (LPCURR, offset 0x030) bit description

Bits	Symbol	Description	Reset value
31:0	LCDLPCURR	LCD Lower Panel Current Address. Contains the current LCD lower panel data DMA address.	0

34.7.14 Color Palette registers

The LCD_PAL register contain 256 palette entries organized as 128 locations of two entries per word.

Each word location contains two palette entries. This means that 128 word locations are used for the palette. When configured for little-endian byte ordering, bits [15:0] are the lower numbered palette entry and [31:16] are the higher numbered palette entry. When configured for big-endian byte ordering this is reversed, because bits [31:16] are the low numbered palette entry and [15:0] are the high numbered entry.

Note: Only TFT displays use all of the palette entry bits.

Table 696. Color Palette registers (PAL[0:127], offset 0x200 (PAL0) to 0x3FC (PAL127)) bit description

Bits	Symbol	Description	Reset value
4:0	R04_0	Red palette data. For STN displays, only the four MSBs, bits [4:1], are used. For monochrome displays only the red palette data is used. All of the palette registers have the same bit fields.	0
9:5	G04_0	Green palette data.	0
14:10	B04_0	Blue palette data.	0
15	I0	Intensity / unused bit. Can be used as the LSB of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.	0
20:16	R14_0	Red palette data. For STN displays, only the four MSBs, bits [4:1], are used. For monochrome displays only the red palette data is used. All of the palette registers have the same bit fields.	0
25:21	G14_0	Green palette data.	0
30:26	B14_0	Blue palette data.	0
31	I1	Intensity / unused bit. Can be used as the LSB of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.	0

34.7.15 Cursor Image registers

The CRSR_IMG register area contains 256-word wide values which are used to define the image or images overlaid on the display by the hardware cursor mechanism. The image must always be stored in LBBP mode (little-endian byte, big-endian pixel) mode, as described in Section 34.6.5.6. Two bits are used to encode color and transparency for each pixel in the cursor.

Depending on the state of bit 0 in the CRSR_CFG register (see Cursor Configuration register description), the cursor image RAM contains either four 32x32 cursor images, or a single 64x64 cursor image.

The two colors defined for the cursor are mapped onto values from the CRSR_PAL0 and CRSR_PAL0 registers (see Cursor Palette register descriptions).

Table 697. Cursor Image registers (CRSR_IMG[0:255], offset 0x800 (CRSR_IMG0) to 0xBFC (CRSR_IMG255)) bit description

Bits	Symbol	Description	Reset value
31:0	CRSR_IMG	Cursor Image data. The 256 words of the cursor image registers define the appearance of either one 64x64 cursor, or 4 32x32 cursors.	0

34.7.16 Cursor Control register

The CRSR_CTRL register provides access to frequently used cursor functions, such as the display on/off control for the cursor, and the cursor number.

If a 32x32 cursor is selected, one of four 32x32 cursors can be enabled. The images each occupy one quarter of the image memory, with Cursor0 from location 0, followed by Cursor1 from address 0x100, Cursor2 from 0x200 and Cursor3 from 0x300. If a 64x64 cursor is selected only one cursor fits in the image buffer, and no selection is possible.

Similar frame synchronization rules apply to the cursor number as apply to the cursor coordinates. If CrsrFramesync is 1, the displayed cursor image is only changed during the vertical frame blanking period. If CrsrFrameSync is 0, the cursor image index is changed immediately, even if the cursor is currently being scanned.

Table 698. Cursor Control register (CRSR_CTRL, offset 0xC00) bit description

Bits	Symbol	Description	Reset value
0	CRSRON	Cursor enable. 0 = Cursor is not displayed. 1 = Cursor is displayed.	0
3:1	-	Reserved. Read value is undefined, only zero should be written.	0
5:4	CRSRNUM1_0	Cursor image number. If the selected cursor size is 6x64, this field has no effect. If the selected cursor size is 32x32: 00 = Cursor0. 01 = Cursor1. 10 = Cursor2. 11 = Cursor3.	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	0

34.7.17 Cursor Configuration register

The CRSR_CFG register provides overall configuration information for the hardware cursor.

Table 699. Cursor Configuration register (CRSR_CFG, offset 0xC04) bit description

Bits	Symbol	Description	Reset value
0	CRSRSIZE	Cursor size selection. 0 = 32x32 pixel cursor. Allows for 4 defined cursors. 1 = 64x64 pixel cursor.	0
1	FRAMESYNC	Cursor frame synchronization type. 0 = Cursor coordinates are asynchronous. 1 = Cursor coordinates are synchronized to the frame synchronization pulse.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.18 Cursor Palette register 0

The cursor palette registers provide color palette information for the visible colors of the cursor. Color0 maps through CRSR_PAL0.

The register provides 24-bit RGB values that are displayed according to the abilities of the LCD panel in the same way as the frame buffers palette output is displayed.

In monochrome STN mode, only the upper 4 bits of the Red field are used. In STN color mode, the upper 4 bits of the Red, Blue, and Green fields are used. In 24 bits per pixel mode, all 24 bits of the palette registers are significant.

Table 700. Cursor Palette register 0 (CRSR_PAL0, offset 0xC08) bit description

Bits	Symbol	Description	Reset value
7:0	RED	Red color component	0
15:8	GREEN	Green color component	0
23:16	BLUE	Blue color component.	0
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.19 Cursor Palette register 1

The cursor palette registers provide color palette information for the visible colors of the cursor. Color1 maps through CRSR_PAL1.

The register provides 24-bit RGB values that are displayed according to the abilities of the LCD panel in the same way as the frame buffers palette output is displayed.

In monochrome STN mode, only the upper 4 bits of the Red field are used. In STN color mode, the upper 4 bits of the Red, Blue, and Green fields are used. In 24 bits per pixel mode, all 24 bits of the palette registers are significant.

Table 701. Cursor Palette register 1 (CRSR_PAL1, offset 0xC0C) bit description

Bits	Symbol	Description	Reset value
7:0	RED	Red color component	0
15:8	GREEN	Green color component	0
23:16	BLUE	Blue color component.	0
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.20 Cursor XY Position register

The CRSR_XY register defines the distance of the top-left edge of the cursor from the top-left side of the cursor overlay. refer to the section on Cursor Clipping for more details.

If the FrameSync bit in the CRSR_CFG register is 0, the cursor position changes immediately, even if the cursor is currently being scanned. If Framesync is 1, the cursor position is only changed during the next vertical frame blanking period.

Table 702. Cursor XY Position register (CRSR_XY, offset 0xC10) bit description

Bits	Symbol	Description	Reset value
9:0	CRSRX	X ordinate of the cursor origin measured in pixels. When 0, the left edge of the cursor is at the left of the display.	0
15:10	-	Reserved. Read value is undefined, only zero should be written.	-
25:16	CRSRY	Y ordinate of the cursor origin measured in pixels. When 0, the top edge of the cursor is at the top of the display.	0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.21 Cursor Clip Position register

The CRSR_CLIP register defines the distance from the top-left edge of the cursor image, to the first displayed pixel in the cursor image.

Different synchronization rules apply to the Cursor Clip registers than apply to the cursor coordinates. If the FrameSync bit in the CRSR_CFG register is 0, the cursor clip point is changed immediately, even if the cursor is currently being scanned.

If the Framesync bit in the CRSR_CFG register is 1, the displayed cursor image is only changed during the vertical frame blanking period, providing that the cursor position has been updated since the Clip register was programmed. When programming, the Clip register must be written before the Position register (ClcdCrsrXY) to ensure that in a given frame, the clip and position information is coherent.

The contents of the CRSR_CLIP register are described in Table 703.

Table 703. Cursor Clip Position register (CRSR_CLIP, offset 0xC14) bit description

Bits	Symbol	Description	Reset value
5:0	CRSRCLIPX	Cursor clip position for X direction. Distance from the left edge of the cursor image to the first displayed pixel in the cursor. When 0, the first pixel of the cursor line is displayed.	0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
13:8	CRSRCLIPY	Cursor clip position for Y direction. Distance from the top of the cursor image to the first displayed pixel in the cursor. When 0, the first displayed pixel is from the top line of the cursor image.	0
31:14	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.22 Cursor Interrupt Mask register

The CRSR_INTMSK register is used to enable or disable the cursor from interrupting the processor.

Table 704. Cursor Interrupt Mask register (CRSR_INTMSK, offset 0xC20)

Bits	Symbol	Description	Reset value
0	CRSRIM	Cursor interrupt mask. When clear, the cursor never interrupts the processor. When set, the cursor interrupts the processor immediately after reading of the last word of cursor image.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	-

34.7.23 Cursor Interrupt Clear register

The CRSR_INTCLR register is used by software to clear the cursor interrupt status and the cursor interrupt signal to the processor.

Table 705. Cursor Interrupt Clear register (CRSR_INTCLR, offset 0xC24) bit description

Bits	Symbol	Description
0	CRSRIC	Cursor interrupt clear. Writing a 0 to this bit has no effect. Writing a 1 to this bit causes the cursor interrupt status to be cleared.
31:1	-	Reserved. Read value is undefined, only zero should be written.

34.7.24 Cursor Raw Interrupt Status register

The CRSR_INTRAW register is set to indicate a cursor interrupt. When enabled via the CrsrIM bit in the CRSR_INTMSK register, provides the interrupt to the system interrupt controller.

Table 706. Cursor Raw Interrupt Status register (CRSR_INTRAW, offset 0xC28) bit description

Bits	Symbol	Description	Reset value
0	CRSRRIS	Cursor raw interrupt status. The cursor interrupt status is set immediately after the last data is read from the cursor image for the current frame. This bit is cleared by writing to the CrsrIC bit in the CRSR_INTCLR register.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	-

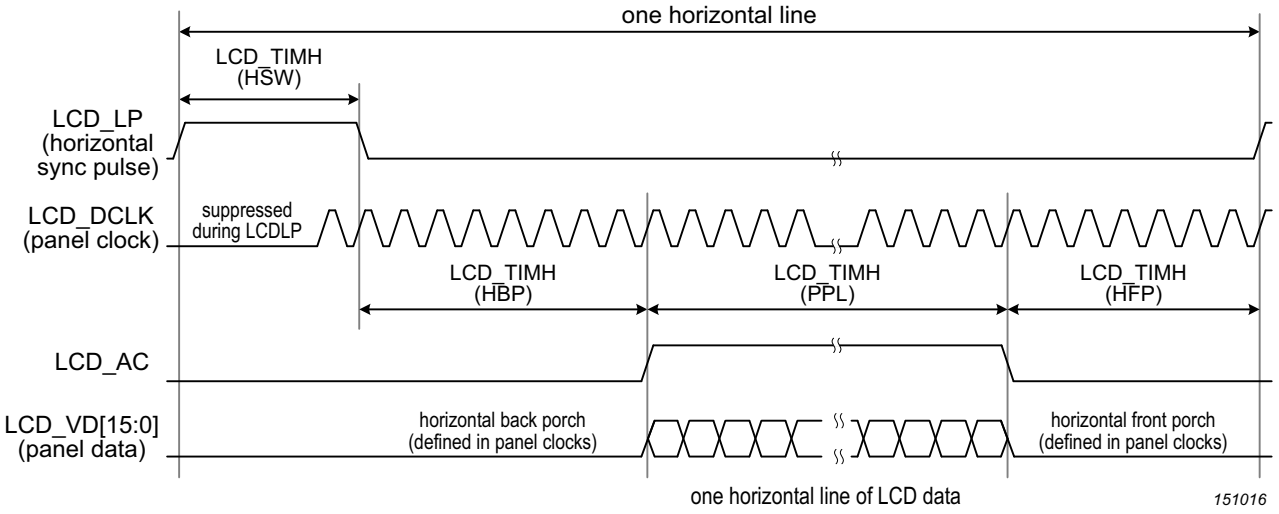
34.7.25 Cursor Masked Interrupt Status register

The CRSR_INTSTAT register is set to indicate a cursor interrupt providing that the interrupt is not masked in the CRSR_INTMSK register.

Table 707. Cursor Masked Interrupt Status register (CRSR_INTSTAT, offset 0xC2C) bit description

Bits	Symbol	Description	Reset value
0	CRSRMIS	Cursor masked interrupt status. The cursor interrupt status is set immediately after the last data read from the cursor image for the current frame, providing that the corresponding bit in the CRSR_INTMSK register is set. The bit remains clear if the CRSR_INTMSK register is clear. This bit is cleared by writing to the CRSR_INTCLR register.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	-

34.8 LCD timing diagrams

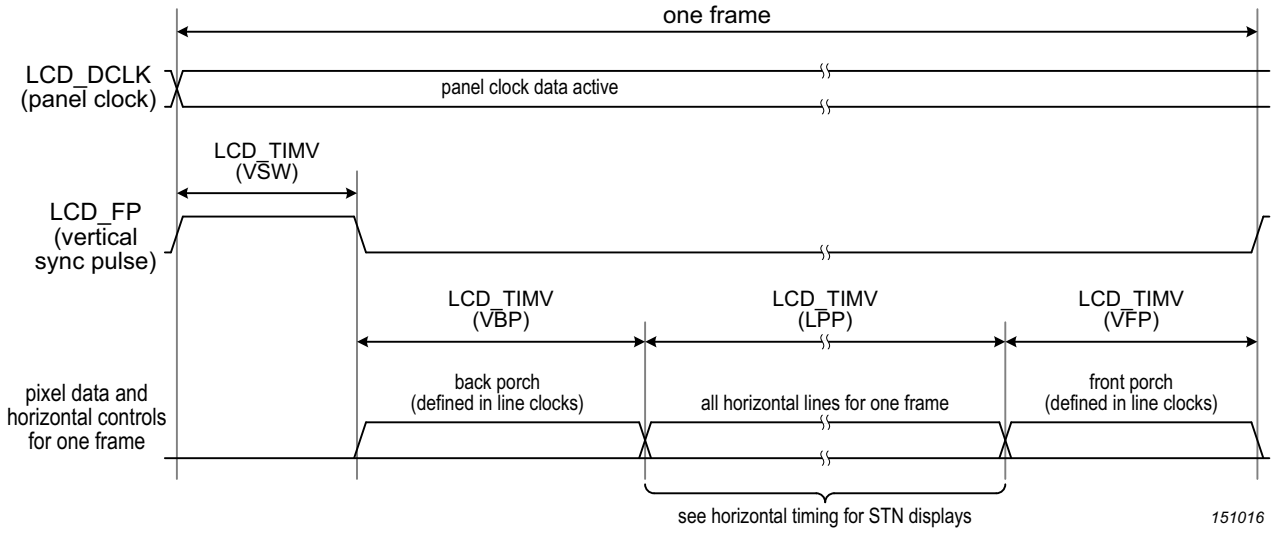


(1) The active data lines will vary with the type of STN panel (4-bit, 8-bit, color, mono) and with single or dual frames.

(2) The duration of the LCD_LP signal is controlled by the HSW field in the LCD_TIMH register.

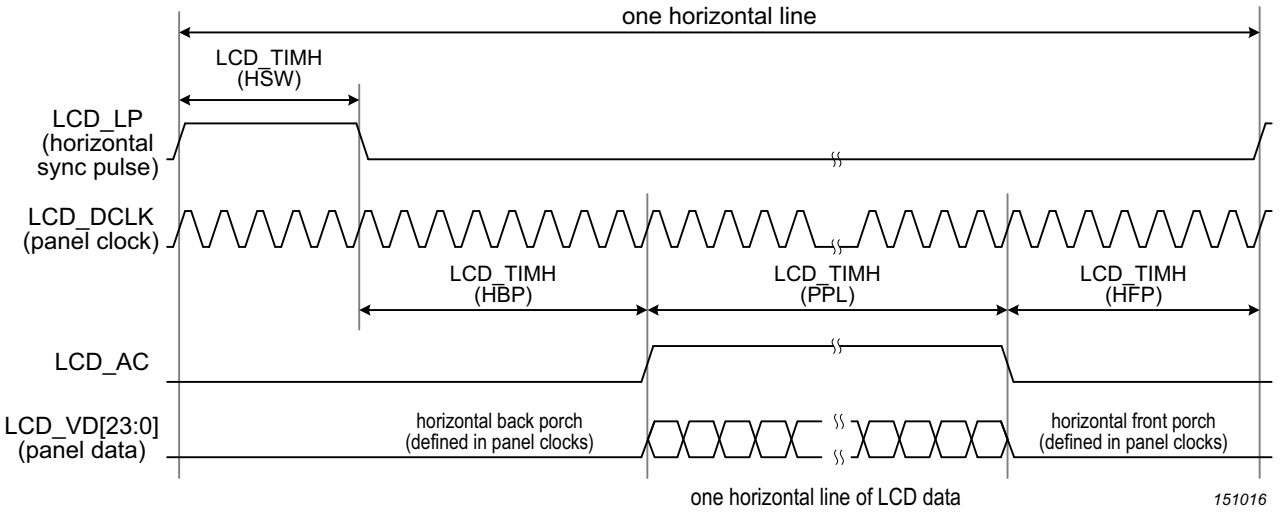
(3) The Polarity of the LCD_LP signal is determined by the IHS bit in the LCD_POL register.

Fig 124. Horizontal timing for STN displays



(1) Signal polarities may vary for some displays.

Fig 125. Vertical timing for STN displays

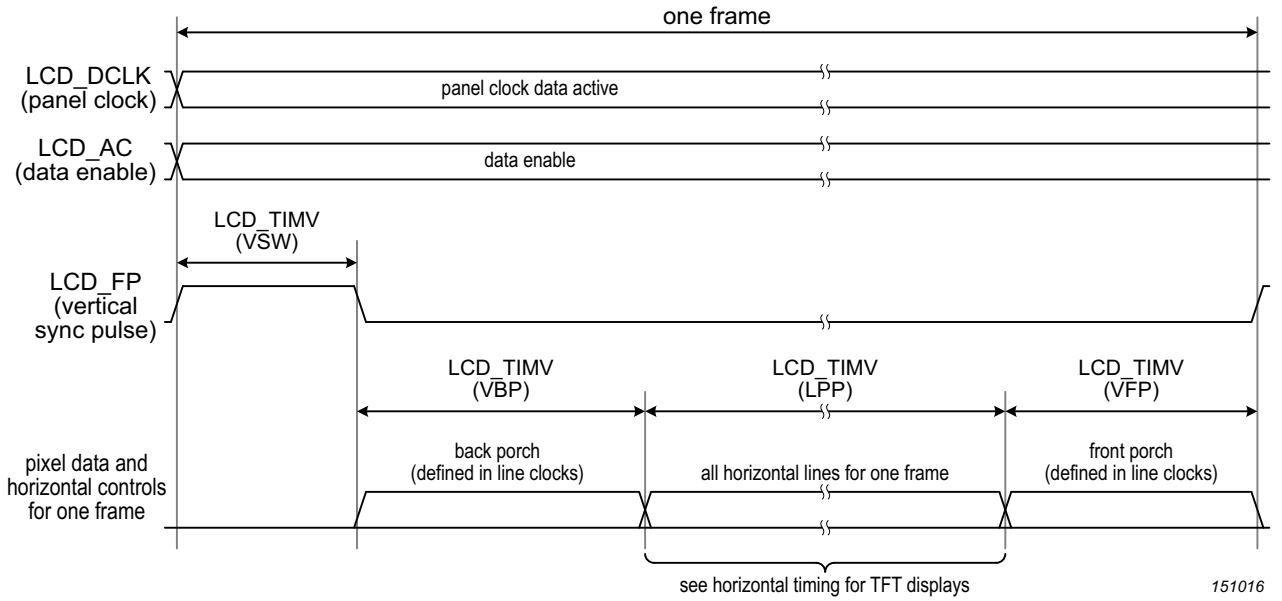


(1) The active data lines will vary with the type of TFT panel.

(2) The duration of the LCD_LP is controlled by the HSW field in the LCD_TIMH register.

(3) The polarity of the LCD_LP signal is determined by the IHS bit in the LCD_POL register.

Fig 126. Horizontal timing for TFT displays



(1) Polarities may vary for some displays.

Fig 127. Vertical timing for TFT displays

34.9 LCD panel signal usage

Table 708. LCD panel connections for STN single panel mode

External pin	4-bit mono STN single panel		8-bit mono STN single panel		Color STN single panel	
	pin used	LCD function	pin used	LCD function	pin used	LCD function
LCD_VD[8] - LCD_VD[23]	-	-	-	-	-	-
LCD_VD[7]	-	-	P4[29]	UD[7]	P4[29]	UD[7]
LCD_VD[6]	-	-	P4[28]	UD[6]	P4[28]	UD[6]
LCD_VD[5]	-	-	P2[13]	UD[5]	P2[13]	UD[5]
LCD_VD[4]	-	-	P2[12]	UD[4]	P2[12]	UD[4]
LCD_VD[3]	P2[9]	UD[3]	P2[9]	UD[3]	P2[9]	UD[3]
LCD_VD[2]	P2[8]	UD[2]	P2[8]	UD[2]	P2[8]	UD[2]
LCD_VD[1]	P2[7]	UD[1]	P2[7]	UD[1]	P2[7]	UD[1]
LCD_VD[0]	P2[6]	UD[0]	P2[6]	UD[0]	P2[6]	UD[0]
LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP
LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC
LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP
LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK
LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE
LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR
LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[0]	LCD_PWR

Table 709. LCD panel connections for STN dual panel mode

External pin	4-bit mono STN dual panel		8-bit mono STN dual panel		Color STN dual panel	
	pin used	LCD function	pin used	LCD function	pin used	LCD function
LCD_VD[16] - LCD_VD[23]	-	-	-	-	-	-
LCD_VD[15]	-	-	P1[29]	LD[7]	P1[29]	LD[7]
LCD_VD[14]	-	-	P1[28]	LD[6]	P1[28]	LD[6]
LCD_VD[13]	-	-	P1[27]	LD[5]	P1[27]	LD[5]
LCD_VD[12]		-	P1[26]	LD[4]	P1[26]	LD[4]
LCD_VD[11]	P4[29]	LD[3]	P1[25]	LD[3]	P1[25]	LD[3]
LCD_VD[10]	P4[28]	LD[2]	P1[24]	LD[2]	P1[24]	LD[2]
LCD_VD[9]	P2[13]	LD[1]	P1[23]	LD[1]	P1[23]	LD[1]
LCD_VD[8]	P2[12]	LD[0]	P1[22]	LD[0]	P1[22]	LD[0]
LCD_VD[7]	-	-	P1[21]	UD[7]	P1[21]	UD[7]
LCD_VD[6]	-	-	P1[20]	UD[6]	P1[20]	UD[6]
LCD_VD[5]	-	-	P2[13]	UD[5]	P2[13]	UD[5]
LCD_VD[4]	-	-	P2[12]	UD[4]	P2[12]	UD[4]
LCD_VD[3]	P2[9]	UD[3]	P2[9]	UD[3]	P2[9]	UD[3]
LCD_VD[2]	P2[8]	UD[2]	P2[8]	UD[2]	P2[8]	UD[2]
LCD_VD[1]	P2[7]	UD[1]	P2[7]	UD[1]	P2[7]	UD[1]
LCD_VD[0]	P2[6]	UD[0]	P2[6]	UD[0]	P2[6]	UD[0]
LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP
LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC
LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP
LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK
LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE
LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD-PWR
LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN

Table 710. LCD panel connections for TFT panels

External pin	TFT 12 bit (4:4:4 mode)		TFT 16 bit (5:6:5 mode)		TFT 16 bit (1:5:5:5 mode)		TFT 24 bit	
	pin used	LCD function	pin used	LCD function	pin used	LCD function	pin used	LCD function
LCD_VD[23]	P1[29]	BLUE3	P1[29]	BLUE4	P1[29]	BLUE4	P1[29]	BLUE7
LCD_VD[22]	P1[28]	BLUE2	P1[28]	BLUE3	P1[28]	BLUE3	P1[28]	BLUE6
LCD_VD[21]	P1[27]	BLUE1	P1[27]	BLUE2	P1[27]	BLUE2	P1[27]	BLUE5
LCD_VD[20]	P1[26]	BLUE0	P1[26]	BLUE1	P1[26]	BLUE1	P1[26]	BLUE4
LCD_VD[19]	-	-	P2[13]	BLUE0	P2[13]	BLUE0	P2[13]	BLUE3
LCD_VD[18]	-	-	-	-	P2[12]	intensity	P2[12]	BLUE2
LCD_VD[17]	-	-	-	-	-	-	P0[9]	BLUE1
LCD_VD[16]	-	-	-	-	-	-	P0[8]	BLUE0
LCD_VD[15]	P1[25]	GREEN3	P1[25]	GREEN5	P1[25]	GREEN4	P1[25]	GREEN7
LCD_VD[14]	P1[24]	GREEN2	P1[24]	GREEN4	P1[24]	GREEN3	P1[24]	GREEN6
LCD_VD[13]	P1[23]	GREEN1	P1[23]	GREEN3	P1[23]	GREEN2	P1[23]	GREEN5
LCD_VD[12]	P1[22]	GREEN0	P1[22]	GREEN2	P1[22]	GREEN1	P1[22]	GREEN4
LCD_VD[11]	-	-	P1[21]	GREEN1	P1[21]	GREEN0	P1[21]	GREEN3
LCD_VD[10]	-	-	P1[20]	GREEN0	P1[20]	intensity	P1[20]	GREEN2
LCD_VD[9]	-	-	-	-	-	-	P0[7]	GREEN1
LCD_VD[8]	-	-	-	-	-	-	P0[6]	GREEN0
LCD_VD[7]	P2[9]	RED3	P2[9]	RED4	P2[9]	RED4	P2[9]	RED7
LCD_VD[6]	P2[8]	RED2	P2[8]	RED3	P2[8]	RED3	P2[8]	RED6
LCD_VD[5]	P2[7]	RED1	P2[7]	RED2	P2[7]	RED2	P2[7]	RED5
LCD_VD[4]	P2[6]	RED0	P2[6]	RED1	P2[6]	RED1	P2[6]	RED4
LCD_VD[3]	-	-	P2[12]	RED0	P4[29]	RED0	P4[29]	RED3
LCD_VD[2]	-	-	-	-	P4[28]	intensity	P4[28]	RED2
LCD_VD[1]	-	-	-	-	-	-	P0[5]	RED1
LCD_VD[0]	-	-	-	-	-	-	P0[4]	RED0
LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP
LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC	P2[4]	LCD_AC
LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP
LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK
LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE
LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR
LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN

35.1 How to read this chapter

The MCAN block is available on some LPC546xx devices. See Table 1 “Ordering information” for details.

35.2 Features

- Conforms to Controller Area Network (CAN) protocol version 2.0 part A, B, and ISO 11898-1.
- CAN FD with up to 64 data bytes supported.
- CAN error logging.
- AUTOSAR support.
- SAE J1939 support.
- Improved acceptance filtering.
- Two configurable Receive FIFOs.
- Separate signaling on reception of High Priority Messages.
- Up to 64 dedicated Receive buffers.
- Up to 32 dedicated Transmit buffers.
- Configurable Transmit FIFO.
- Configurable Transmit Queue.
- Configurable Transmit Event FIFO.
- Message RAM is assigned to on-chip SRAM, accessible by CPU and DMA
- Shared message RAM between the two CAN FD controllers.
- Programmable loop-back test mode.
- Maskable module interrupts.
- Power-down support.
- Debug on CAN support.

35.3 Basic configuration

The two MCAN controllers are configured using the following registers:

- Clock: In the AHBCLKCTRL1 register (Section 7.5.20), set the MCAN bit to enable the function clock to the respective MCAN block being used.

  Remark: The MCAN blocks are disabled on reset (MCAN0 and MCAN1 = 0).

  The MCAN function clock is the clock source used for the MCAN. Use the CANCLKDIV0 and CANCLKDIV1 registers (See Section 7.5.46 and Section 7.5.47) to divide the main clock to reach the required MCAN clock frequency if needed.
- Pins: Select the MCAN pins and pin modes through the relevant IOCON registers (See Section 10.4.2).

- Interrupts are enabled in the NVIC using the appropriate interrupt set enable register.

35.4 General description

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The MCAN controller is designed to provide a full implementation of the CAN protocol according to the CAN Specification Version 2.0 part A, B and to CAN FD Specification V.1.0.

The MCAN controller allows to build powerful local networks with low-cost multiplex wiring by supporting distributed real-time control with a very high level of security. The CAN controller consists of a CAN core, message RAM, control registers, AHB interface, and message handlers for transferring and receiving messages.

The CAN core is the CAN protocol controller and the transfer and receive shift register. The CAN core handles all ISO 11898-1 protocol functions and supports 11-bit and 29-bit identifiers.

The Tx handler transmits messages from the message RAM to the CAN Core as well as providing transmit status information. The Rx handler manages acceptance filtering and transfers received messages from the CAN core to the message RAM as well as providing receive message status information. Acceptance filtering is implemented by a combination of up to 128 filter elements where each element can be configured as a range, bit mask, or dedicated ID filter.

FigureFigure 128 shows the MCAN IP block diagram.

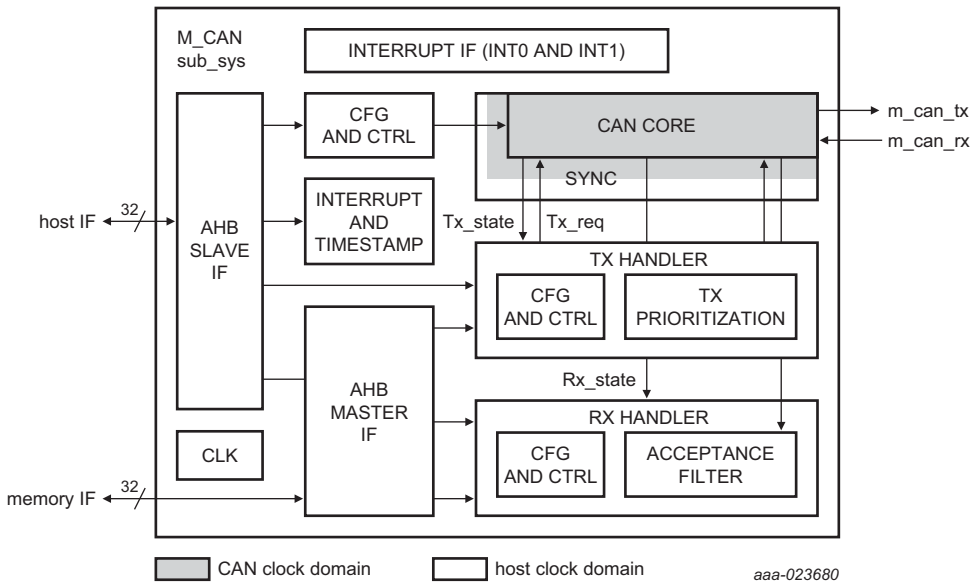


Fig 128. MCAN IP block diagram

35.5 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration, any general purpose SRAM can be used. The base address of the SRAM used for messages is determined by the value in the MRBA register that can be changed by the application.

35.5.1 Message RAM configuration

The message RAM has a width of 32 bits. The MCAN module can be configured to allocate up to 4352 words in the message RAM. It is not necessary to configure each of the sections listed in Figure 129 and there is no restriction with respect to the sequence of the sections.

When operated in CAN FD mode, the required message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx buffers, and Tx buffers via the RXESC and TXESC registers.

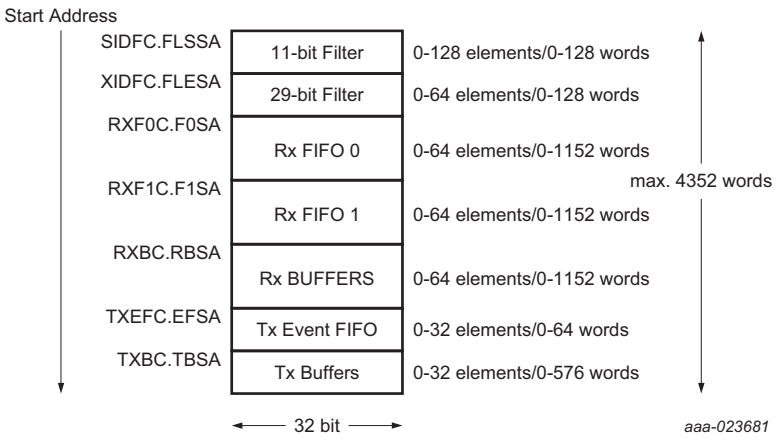


Fig 129. Message RAM configuration

When the MCAN module addresses the message RAM, it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses, that is, bits 15 to 2 are evaluated, the two least significant bits are ignored.

Remark: The MCAN does not check for erroneous configuration of the message RAM. Especially the configuration of the start addresses of different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

35.6 Pin description

Table 711. CAN pin description

Pin	Type	Description
CAN0_TD	O	MCAN0 transmit output
CAN0_RD	I	MCAN0 receive input
CAN1_TD	O	MCAN1 transmit output
CAN1_RD	I	MCAN1 receive input

35.7 Register description

There are two MCAN controllers on the LPC546xx. Each MCAN controller contains its own set of 32-bit wide registers that are related to configuring that particular MCAN controller. See Table 712.

Table 712. Register overview: LPC546xx MCAN controller (MCAN0 base address 0x4009 D000, MCAN1 base address 0x4009 E000)

Name	Access	Offset	Description	Reset value	Section
DBTP	RW	0x00C	Data bit timing and prescaler.	0x0000 0A33	35.8.1
TEST	RW	0x010	Test register.	0	35.8.2
CCCR	RW	0x018	CC control.	0x0000 0001	35.8.3
NBTP	RW	0x01C	Nominal bit timing and prescaler.	0x0600 0A03	35.8.4
TSCC	RW	0x020	Timestamp counter configuration.	0	35.8.5
TSCV	RO	0x024	Timestamp counter value.	0	35.8.6
TOCC	RW	0x028	Timeout counter configuration.	0xFFFF 0000	35.8.7
TOCV	RO	0x02C	Timeout counter value.	0x0000 FFFF	35.8.8
ECR	RO	0x040	Error counter.	0	35.8.9
PSR	RO	0x044	Protocol status.	0x0000 0707	35.8.10
TDCR	RW	0x048	Transmitter delay compensator.	0	35.8.11
IR	RW	0x050	Interrupt.	0	35.8.12
IE	RW	0x054	Interrupt enable.	0	35.8.13
ILS	RW	0x058	Interrupt line select.	0	35.8.14
ILE	RW	0x05C	Interrupt line enable.	0	35.8.15
GFC	RW	0x080	Global filter configuration.	0	35.8.16
SIDFC	RW	0x084	Standard ID filter configuration.	0	35.8.17
XIDFC	RW	0x088	Extended ID filter configuration.	0	35.8.18
XIDAM	RW	0x090	Extended ID and mask.	0x1FFF FFFF	35.8.19
HPMS	RO	0x094	High priority message status.	0	35.8.20
NDAT1	RW	0x098	New data 1.	0	35.8.21
NDAT2	RW	0x09C	New data 2.	0	35.8.22
RXF0C	RW	0x0A0	Rx FIFO 0 configuration.	0	35.8.23
RXF0S	RO	0x0A4	Rx FIFO 0 status.	0	35.8.24
RXF0A	RW	0x0A8	Rx FIFO 0 acknowledge.	0	35.8.25
RXBC	RW	0x0AC	Rx buffer configuration.	0	35.8.26
RXF1C	RW	0x0B0	Rx FIFO 1 configuration.	0	35.8.27
RXF1S	RO	0x0B4	Rx FIFO 1 status.	0	35.8.28
RXF1A	RW	0x0B8	Rx FIFO 1 acknowledge.	0	35.8.29
RXESC	RW	0x0BC	Rx buffer and FIFO element size configuration.	0	35.8.30
TXBC	RW	0x0C0	Tx buffer configuration.	0	35.8.31
TXFQS	RO	0x0C4	Tx FIFO/queue status.	0	35.8.32
TXESC	RW	0x0C8	Tx buffer element size configuration.	0	35.8.33
TXBRP	RO	0x0CC	Tx buffer request pending.	0	35.8.34
TXBAR	RW	0x0D0	Tx buffer add request.	0	35.8.35

Table 712. Register overview: LPC546xx MCAN controller (MCAN0 base address 0x4009 D000, MCAN1 base address 0x4009 E000)

Name	Access	Offset	Description	Reset value	Section
TXBCR	RW	0x0D4	Tx buffer cancellation request.	0	35.8.36
TXBTO	RO	0x0D8	Tx buffer transmission occurred.	0	35.8.37
TXBCF	RO	0x0DC	Tx buffer cancellation finished.	0	35.8.38
TXBTIE	RW	0x0E0	Tx buffer transmission interrupt enable.	0	35.8.39
TXBCIE	RW	0x0E4	Tx buffer cancellation finished interrupt enable.	0	35.8.40
TXEFC	RW	0x0F0	Tx event FIFO configuration.	0	35.8.41
TXEFS	RO	0x0F4	Tx event FIFO status.	0	35.8.42
TXEFA	RW	0x0F8	Tx event FIFO acknowledge.	0	35.8.43
MRBA	RW	0x200	CAN message RAM base address.	0	35.8.44
ETSCC	RW	0x400	External timestamp counter configuration.	0	35.8.45
ETSCV	RW	0x600	External timestamp counter value.	0	35.8.46

35.8 CAN protocol register description

35.8.1 Data bit timing and prescaler register

Write access to the DBTP register is enabled by setting bits CCE and INIT in the CCCR register.

The nominal bit time for CAN is determined by the number of time quanta per bit, and the time each time quantum represents.The time quantum (t_q) may be programmed in the range of 1 to 32 MCAN clock periods:

T_q = (DBRP + 1) MCAN clock periods

A single CAN bit time is made up from 4 segments:

Segment SYNC_SEG is not programmable and is fixed at one time quantum.

Segments PROP_SEG and PHASE_SEG1 are combined in a single bitfield, and is programmable in the range of 1 to 16 time quanta. The sample point is set right after PHASE_SEG1.

Segment PHASE_SEG2 is programmable in the range of 1 to 16 time quanta.

The length of the bit time (in time quanta) is the combined value of these 4 segments:

[SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] t_q = [1 + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] t_q

Therefore, the CAN bit time may be programmed in the range of 3 to 33 time quanta.

The combined value of segments PROP_SEG and PHASE_SEG1 can be programmed using bit field DTSEG1, and segment PHASE_SEG2 can be programmed using bit field DTSEG2.

The CAN hardware interprets these bit fields as the programmed value+1, so the length of the bit time (in time quanta) can be calculated from the programmed values as:

[1 + DTSEG1 + 1 + DTSEG2 + 1] t_q = [DTSEG1 + DTSEG2 + 3] t_q

The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 MCAN clock periods.

T_q = (DBRP + 1) mtq.

DTSEG1 is the sum of prop_seg and phase_Seg1. DTSEG2 is phase_seg2. Therefore, the length of the bit time is:

[DTSEG1 + DTSEG2 + 3] t_q for programmed values, or,

[sync_seg + prop_seg + phase_seg1 + phase_seg2] t_q for functional values.

The information processing time (IPT) is zero, meaning that the data for the next bit is available the first clock edge after the sample point.

Table 713. Data bit timing and prescaler register (DBTP, offset 0x00C) bit description

Bit	Symbol	Value	Description	Reset value
3:0	DSJW	-	Data (re)synchronization jump width. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15. Limits by how many time quanta a bit period may be extended or shortened because of re-synchronization.	0x3
7:4	DTSEG2	-	Data time segment after sample point. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.	0x3
12:8	DTSEG1	-	Data time segment before sample point. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.	0xA
15:13	-	-	Reserved.	-
20:16	DBRP	-	Data bit rate prescaler. The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.Valid values are 0 to 31.	0
22:21	-	-	Reserved.	-
23	TDC		Transmitter delay compensation.	0
		0	Transmitter delay compensation disabled	
		1	Transmitter delay compensation enabled	
31:24	-	-	Reserved	-

Remark: With a MCAN clock of 8 MHz, the reset value of DBTP register will configure the MCAN for a data phase bit rate of 500 kb/s.

Remark: The bit rate configured for the CAN FD data phase via the DBTP register must be higher or equal to the bit rate configured for the arbitration phase via NBTP register.

35.8.2 Test register

Write access to the CAN test register is enabled by setting the TEST bit in the CCCR register. All test register functions are set to their reset values when the TEST bit is reset.

The different test functions can be combined, but when TX bits are programmed with a value that is not 0x0, the message transfer is disturbed.

Table 714. Test register (TEST, offset 0x010) bit description

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved.	-
4	LBCK		Loop back mode.	0
		0	Loop back mode is disabled	
		1	Loop back mode is enabled	
6:5	TX		Control of transmit pin.	0
		0x0	Controller. Level of CAN_TXD is controlled by CAN controller. This is the value at reset.	
		0x1	Sample point. The sample point can be monitored at the CAN_TXD.	
		0x2	Low. CAN_TXD pin is driven LOW/dominant.	
		0x3	High. CAN_TXD is driven HIGH/recessive.	

Table 714. Test register (TEST, offset 0x010) bit description

Bit	Symbol	Value	Description	Reset value
7	RX		Monitors the actual value of the CAN_RXD.	0
		0	Dominant. The CAN bus is dominant (CAN_RXD = 0).	
		1	Recessive. The CAN bus is recessive (CAN_RXD = 1).	
31:8	-	-	Reserved.	-

35.8.3 Control register

The MCAN module has a mechanism to synchronize the two clock domains within itself, which may cause a delay before the value written to INIT bit can be read back. Due to this, it is recommended to read back the value of the INIT bit and confirm that it has been accepted before writing a new value to the INIT bit.

Table 715. Control register (CCCR, offset 0x018) bit description

Bit	Symbol	Value	Description	Reset value
0	INIT		Initialization	1
		0	Normal operation.	
		1	Initialization is started.	
1	CCE		Configuration change enable.	0
		0	No write access. The CPU has no write access to the protected configuration registers.	
		1	Write access. The CPU has write access to the protected configuration registers.	
2	ASM		Restricted operational mode.	0
		0	Normal CAN operation	
		1	Restricted operation mode active	
3	CSA		Clock stop acknowledge.	0
		0	No clock stop acknowledged	
		1	MCAN may be set in power down by stopping the internal MCAN clocks	
4	CSR		Clock stop request	0
		0	No clock stop is requested.	
		1	Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reaches idle.	
5	MON		Bus monitoring mode.	0
		0	Bus monitoring mode is disabled	
		1	Bus monitoring mode is enabled	
6	DAR		Disable automatic retransmission.	0
		0	Automatic retransmission of messages not transmitted successfully enabled	
		1	Automatic retransmission disabled	
7	TEST		Test mode enable.	0
		0	Normal operation	
		1	Test mode is enabled	
8	FDOE		CAN FD operation enable.	0
		0	CAN FD operation is disabled	
		1	CAN FD operation is enabled	

Table 715. Control register (CCCR, offset 0x018) bit description

Bit	Symbol	Value	Description	Reset value
9	BRSE		When CAN FD operation is disabled, this bit is not evaluated	0
		0	Bit rate switching for transmissions is disabled	
		1	Bit rate switching for transmission is enabled.	
11:10	-	-	Reserved.	-
12	PXHD		Protocol exception handling disable. When protocol exception handling is disabled, the MCAN module will transmit an error frame hen it detects a protocol exception condition.	0
		0	Protocol exception handling is enabled	
		1	Protocol exception handling is disabled	
13	EFBI		Edge filtering during bus integration	0
		0	Edge filtering is disabled.	
		1	Two consecutive dominant quanta required to detect an edge for hard synchronization.	
14	TXP		Transmit pause.	0
		0	Transmit pause is disabled	
		1	Transmit pause is enabled	
15	NISO		Non ISO operation. If this bit is set, the MCAN module uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.	0
		0	CAN FD frame format will follow according to ISO11898-1	
		1	CAN FD frame format will follow according to Bosch CAN FD Specification V1.0	
31:16	-	-	Reserved.	-

35.8.4 Nominal bit timing and prescaler register

Write access to the NBTP register is enabled by setting the CCE and INIT bits in the CCCR register.

The CAN bit time may be programmed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 152 MCAN clock periods.

T_q = (DBRP + 1) MCAN clock periods

NTSEG1 is the sum of prop_seg and phase_Seg1. DTSEG2 is phase_seg2. Therefore, the length of the bit time is:

[NTSEG1 + NTSEG2 + 3] t_q for programmed values, or,

[sync_seg + prop_seg + phase_seg1 + phase_seg2] t_q for functional values.

The information processing time (IPT) is zero, meaning that the data for the next bit is available the first clock edge after the sample point.

Table 716. Nominal bit timing and prescaler register (NBTP, offset 0x01C) bit description

Bit	Symbol	Description	Reset value
6:0	NTSEG2	Nominal time segment after sample point. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 127.	0x3
7	-	Reserved.	-
15:8	NTSEG1	Nominal time segment before sample point. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 255.	0xA
24:16	NBRP	Nominal bit rate prescaler. The value by which the oscillator frequency is divided for generating the bit time quanta. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 511.	0
31:25	NSJW	Nominal (re)synchronization jump width. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 127.	0x3

Remark: With a MCAN clock of 8 MHz, the reset value of NBTP register will configure the MCAN for a data phase bit rate of 500 kb/s.

35.8.5 Timestamp counter configuration register

Table 717. Timestamp counter configuration register (TSCC, offset 0x020) bit description

Bit	Symbol	Value	Description	Reset value
1:0	TSS		Timestamp select.	0
		0x0	Timestamp counter value static at 0x0000	
		0x1	Timestamp counter value incremented according to TCP bits	
		0x2	External timestamp counter value used	
		0x3	Timestamp counter value static at 0x0000	
15:2	-	-	Reserved.	
19:16	TCP	-	Timestamp counter prescaler. Configures the timestamp and timeout counters time unit in multiple of CAN bit times. The actual interpretation by that hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.	0
31:20	-	-	Reserved.	-

35.8.6 Timestamp counter value register

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). The timestamp counter will increment depending on the TSS bits in the TSCC register. An overflow will set the TSW interrupt flag in the IR register.

Table 718. Timestamp counter value register (TSCV, offset 0x024) bit description

Bit	Symbol	Description	Reset value
15:0	TSC	Timestamp counter.	0
31:16	-	Reserved.	-

35.8.7 Timeout counter configuration register

Table 719. Timeout counter configuration register (TOCC, offset 0x028) bit description

Bit	Symbol	Value	Description	Reset value
0	ETOC		Enable timeout counter.	0
		0	Timeout counter is disabled	
		1	Timeout counter is enabled	
2:1	TOS		Timeout select. When the timeout counter is operating in continuous mode, a write to the TOCV register presets the counter to the value configured in the TOP bits of the TOCC register and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by the TOP bits. Down-counting is started when the first FIFO element is stored.	0
		0x0	Continuous operation.	
		0x1	Timeout is controlled by Tx event FIFO.	
		0x2	Timeout is controlled by Rx FIFO 0	
		0x3	Timeout is controlled by Rx FIFO 1	
15:3	-	-	Reserved.	-
31:16	TOP	-	Timeout Period. This register holds the start value of the timeout counter to configure the timeout period. This counter counts down.	0xFFFF

35.8.8 Timeout counter value register

Table 720. Timeout counter value register (TOCV, offset 0x02C) bit description

Bit	Symbol	Description	Reset value
15:0	TOC	Timeout counter. The timeout counter is decremented in multiples of CAN bit times depending on the configuration of the TCP bits in the TSCC register. When decremented to zero, the TOO interrupt flag is set in the IR register and the timeout counter is stopped. Start and reset conditions are configured by the TOS bits in the TOC register.	0xFFFF
31:16	-	Reserved.	-

35.8.9 Error counter register

Table 721. Error counter register (ECR, offset 0x040) bit description

Bit	Symbol	Value	Description	Reset value
7:0	TEC	-	Transmit error counter. Current value of the transmit error counter. Current value of the transmit error counter. Valid values are between 0 and 255.	0
14:8	REC	-	Receive error counter. Current value of the receive error counter. Valid values are between 0 and 127.	0
15	RP		Receive error passive	0
		0	Below error level. The receive counter is below the error passive level of 128	
		1	At error level. The receive counter has reached the error passive level of 128	
23:16	CEL	-	CAN error logging. CEL is incremented when TEC or REC is incremented. The counter stops at 0xFF and the next occurrence of a CAN protocol error will set the ELO interrupt flag in the IR register. This counter is reset whenever a read access to these bits is made.	0
31:24	-	-	Reserved.	-

Remark: When the ASM bit in the CCCR register is set, the CAN protocol controller does not increment the TEC and REC bits when a CAN protocol error is detected, but the CEL bits are still incremented.

35.8.10 Protocol status register

Table 722. Protocol status register (PSR, offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
2:0	LEC		Last error code. These bits indicate the type of the last error to occur on the CAN bus. This bit field will be cleared when a message has been transferred without error. The bits in this bit field will be set upon a read access.	0x7
		0x0	No error: No error has occurred since LEC bits has been reset by successful reception or transmission.	
		0x1	Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.	
		0x2	Form error: A fixed format part of a received frame has the wrong format.	
		0x3	AckError: The message transmitted by the M_CAN was not acknowledged by another node.	
		0x4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.	
		0x5	Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).	
		0x6	CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.	
		0x7	NoChange: Any read access to the protocol status register re-initializes the LEC bits to 0x7. When the LEC bits equal the value 0x7, no CAN bus event was detected since the last CPU read access to the protocol status register.	
4:3	ACT		Activity. This register monitors the MCAN communication state.	0
		0x0	Synchronizing – node is synchronizing on CAN communication.	
		0x1	Idle – node is neither receiver nor transmitter.	
		0x2	Receiver – node is operating as receiver	
		0x3	Transmitter – node is operating as transmitter.	
5	EP		Error passive	0
		0	The MCAN is in Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.	
		1	The MCAN is in the Error_Passive state	
6	EW		Warning status.	0
		0	Both error counters are below the Error_Warning limit of 96	
		1	At least one of error counter has reached the Error_Warning limit of 96	
7	BO		Bus off status.	0

Table 722. Protocol status register (PSR, offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
10:8	DLEC	-	Data phase last error code. Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC bits. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. The bits in this bit field will be set upon a read access.	0x7
11	RESI		ESI flag of the last received CAN FD message. This bit is set together with RFDF bits, independent of acceptance filtering. This bit field will be set upon a read access.	0
		0	Last received CAN FD message did not have its ESI flag set	
		1	Last received CAN FD message had its ESI flag set	
12	RBRS		BRS flag of last received CAN FD message. This bit is set together with RFDF bits, independent of acceptance filtering. This bit field will be set upon a read access.	0
		0	Last received CAN FD message did not have its BRS flag set	
		1	Last received CAN FD message had its BRS flag set	
13	RFDF		Received a CAN FD message. This bit is set independent of acceptance filtering. This bit field will be set on a read access.	0
		0	No CAN FD message received since the last CPU reset.	
		1	Message in CAN FD format with FDF flag set has been received.	
14	PXE		Protocol exception event. This bit field will be set upon a read access.	0
		0	No protocol exception event occurred since last read access	
		1	Protocol exception event occurred	
15	-	-	Reserved.	-
22:16	TDCV	-	Transmitter delay compensation value. Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx and m_can_rx and TDCO bits in the TDCR register. The SSP position in the data phase is the number of MCAN clock periods between the start of a transmitted bit and secondary sample point. Valid values are 0 to 127 MCAN clock periods.	
31:23	-	-	Reserved.	-

Remark: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC bits instead of LEC bits. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

Remark: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting the INIT bit in the CCCR register. If the device goes Bus_Off, it will set INIT bit of its own accord, stopping all bus activities. Once the INIT bit has been cleared, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of INIT bit, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the LEC bits to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the Bus_Off recovery sequence. The REC bits in the ECR register is used to count these sequences.

35.8.11 Transmitter delay compensation register

Table 723. Transmitter delay compensation register (TDCR, offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
6:0	TDCF	-	Transmitter delay compensation filter window length. Defines the minimum value for the SSP position, dominant edges on m_can_rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF bits are configured to a value greater than TDCO bits. Valid values are 0 to 127 MCAN clock periods.	0
7	-	-	Reserved.	-
14:8	TDCO	-	Transmitter delay compensation offset. Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 127 MCAN clock periods.	0
31:15	-	-	Reserved.	-

35.8.12 Interrupt register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. The configuration of the IE register controls whether an interrupt is generated. The configuration of the ILS register controls on which interrupt line an interrupt is signaled.

Table 724. Interrupt register (IR, offset 0x050) bit description

Bit	Symbol	Value	Description	Reset value
0	RF0N		Rx FIFO 0 new message.	0
		0	No new message written to Rx FIFO 0	
		1	New message written to Rx FIFO 0	
1	RF0W		Rx FIFO 0 watermark reached.	0
		0	Rx FIFO 0 fill level below watermark	
		1	Rx FIFO 0 fill level reached watermark.	
2	RF0F		Rx FIFO 0 full.	0
		0	Rx FIFO 0 not full	
		1	Rx FIFO 0 full.	
3	RF0L		Rx FIFO 0 message lost.	0
		0	No Rx FIFO 0 message lost	
		1	Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero.	
4	RF1N		Rx FIFO 1 new message.	0
		0	No new message written to Rx FIFO 1	
		1	New message written to Rx FIFO 1	
5	RF1W		Rx FIFO 1 watermark reached.	0
		0	Rx FIFO 1 fill level below watermark	
		1	Rx FIFO 1 fill level reached watermark	
6	RF1F		Rx FIFO 1 full.	0
		0	Rx FIFO 1 not full	
		1	Rx FIFO 1 full.	

Table 724. Interrupt register (IR, offset 0x050) bit description

Bit	Symbol	Value	Description	Reset value
7	RF1L		Rx FIFO 1 message lost.	0
		0	No Rx FIFO 1 message lost	
		1	Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.	
8	HPM		High priority message.	0
		0	No high priority message received	
		1	High priority message received.	
9	TC		Transmission completed.	0
		0	No transmission completed	
		1	Transmission completed.	
10	TCF		Transmission cancellation finished.	0
		0	No transmission cancellation finished	
		1	Transmission cancellation finished	
11	TFE		Tx FIFO empty.	0
		0	Tx FIFO non-empty.	
		1	Tx FIFO empty.	
12	TEFN		Tx event FIFO new entry.	0
		0	Tx event FIFO unchanged	
		1	Tx Handler wrote Tx event FIFO element.	
13	TEFW		Tx event FIFO watermark reached.	0
		0	Tx event FIFO fill level below watermark	
		1	Tx event FIFO fill level reached watermark	
14	TEFF		Tx event FIFO full.	0
		0	Tx event FIFO not full	
		1	Tx event FIFO full	
15	TEFL		Tx event FIFO element lost.	0
		0	No Tx event FIFO element lost	
		1	Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero	
16	TSW		Timestamp wraparound.	0
		0	No timestamp counter wraparound	
		1	Timestamp counter wrapped around	

Table 724. Interrupt register (IR, offset 0x050) bit description

Bit	Symbol	Value	Description	Reset value
17	MRAF		Message RAM access failure. The flag is set when the Rx Handler meets either of the following criteria: The Rx handler has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. The Rx handler was not able to write a message to the Message RAM and the message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the MCAN is switched into Restricted Operation Mode. To leave Restricted Operation Mode, the ASM bit in the CCCR must be cleared.	0
		0	No message RAM access failure occurred	
		1	Message RAM access failure occurred	
18	TOO		Timeout occurred.	0
		0	No timeout	
		1	Timeout reached	
19	DRX		Message stored in dedicated Rx buffer.	0
		0	No Rx buffer updated	
		1	At least one received message stored into an Rx buffer	
20	BEC		Bit error corrected. Message RAM bit error detected and corrected. Controlled by input signal m_can_aeim_berr[0] generated by an optional external parity / ECC logic attached to the message RAM.	0
		0	No bit error detected when reading from message RAM	
		1	Bit error detected and corrected (example, ECC)	
21	BEU		Bit error uncorrected. Message RAM bit error detected, uncorrected. Controlled by input signal m_can_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets INIT bit in the CCCR register to 1. This is done to avoid transmission of corrupted data.	0
		0	No bit error detected when reading from message RAM	
		1	Bit error detected, uncorrected (example, parity logic)	
22	ELO		Error logging overflow.	0
		0	CAN error logging counter did not overflow	
		1	Overflow of CAN error logging counter occurred.	
23	EP		Error passive.	0
		0	Error_Passive status unchanged.	
		1	Error_Passive status changed	
24	EW		Warning status.	0
		0	Error_Warning status unchanged.	
		1	Error_Warning status changed.	

Table 724. Interrupt register (IR, offset 0x050) bit description

Bit	Symbol	Value	Description	Reset value
25	BO		Bus_Off Status.	0
		0	Bus_Off status unchanged	
		1	Bus_Off status changed.	
26	WDI		Watchdog interrupt.	0
		0	No message RAM watchdog event occurred	
		1	Message RAM watchdog event due to missing READY	
27	PEA		Protocol error in arbitration phase.	0
		0	No protocol error in arbitration phase	
		1	Protocol error in arbitration phase detected	
28	PED		Protocol error in data phase.	0
		0	No protocol error in data phase	
		1	Protocol error in data phase detected	
29	ARA		Access to reserved address.	0
		0	No access to reserved address occurred	
		1	Access to reserved address occurred	
31:30	-	-	Reserved.	-

35.8.13 Interrupt enable register

The setting in the interrupt enable register determines which status changes in the interrupt register will be signaled on an interrupt line.

Table 725. Interrupt enable register (IE, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
0	RF0NE		Rx FIFO 0 new message interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
1	RF0WE		Rx FIFO 0 watermark reached interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
2	RF0FE		Rx FIFO 0 full interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
3	RF0LE		Rx FIFO 0 message lost interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled.	
4	RF1NE		Rx FIFO 1 new message interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
5	RF1WE		Rx FIFO 1 watermark reached interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	

Table 725. Interrupt enable register (IE, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
6	RF1FE		Rx FIFO 1 full interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
7	RF1LE		Rx FIFO 1 message lost interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
8	HPME		High priority message interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
9	TCE		Transmission completed interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
10	TCFE		Transmission cancellation finished interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
11	TFEE		Tx FIFO empty interrupt enable.	0
		0	Interrupt disabled.	
		1	Interrupt enabled	
12	TEFNE		Tx event FIFO new entry interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
13	TEFWE		Tx event FIFO watermark reached interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
14	TEFFE		Tx event FIFO full interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
15	TEFLE		Tx event FIFO element lost interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
16	TSWE		Timestamp wraparound interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
17	MRAFE		Message RAM access failure interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
18	TOOE		Timeout occurred interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	

Table 725. Interrupt enable register (IE, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
19	DRXE		Message stored in dedicated Rx buffer interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
20	BECE		Bit error corrected interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
21	BEUE		Bit error uncorrected interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
22	ELOE		Error logging overflow interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
23	EPE		Error passive interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
24	EWE		Warning status interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
25	BOE		Bus_Off Status interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
26	WDIE		Watchdog interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
27	PEAE		Protocol error in arbitration phase interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
28	PEDE		Protocol error in data phase interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
29	ARAE		Access to reserved address interrupt enable.	0
		0	Interrupt disabled	
		1	Interrupt enabled	
31:30	-	-	Reserved.	-

35.8.14 Interrupt line select register

The interrupt line select register assigns an interrupt generated by a specific interrupt flag from the interrupt register to STARTER1 register in SYSCON. See Section 7.5.91. For interrupt generation, the respective interrupt line has to be enabled via the interrupt line enable register.

Table 726. Interrupt line select register (ILS, offset 0x058) bit description

Bit	Symbol	Value	Description	Reset value
0	RF0NL		Rx FIFO 0 new message interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
1	RF0WL		Rx FIFO 0 watermark reached interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
2	RF0FL		Rx FIFO 0 full interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
3	RF0LL		Rx FIFO 0 message lost interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
4	RF1NL		Rx FIFO 1 new message interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
5	RF1WL		Rx FIFO 1 watermark reached interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
6	RF1FL		Rx FIFO 1 full interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
7	RF1LL		Rx FIFO 1 message lost interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
8	HPML		High priority message interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
9	TCL		Transmission completed interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
10	TCFL		Transmission cancellation finished interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
11	TFEL		Tx FIFO empty interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
12	TEFNL		Tx event FIFO new entry interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	

Table 726. Interrupt line select register (ILS, offset 0x058) bit description

Bit	Symbol	Value	Description	Reset value
13	TEFWL		Tx event FIFO watermark reached interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
14	TEFFL		Tx event FIFO full interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
15	TEFLL		Tx event FIFO element lost interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
16	TSWL		Timestamp wraparound interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
17	MRAFL		Message RAM access failure interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
18	TOOL		Timeout occurred interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
19	DRXL		Message stored in dedicated Rx buffer interrupt line	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
20	BECL		Bit error corrected interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
21	BEUL		Bit error uncorrected interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
22	ELOL		Error logging overflow interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
23	EPL		Error passive interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
24	EWL		Warning status interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
25	BOL		Bus_Off Status interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	

Table 726. Interrupt line select register (ILS, offset 0x058) bit description

Bit	Symbol	Value	Description	Reset value
26	WDIL		Watchdog interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
27	PEAL		Protocol error in arbitration phase interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
28	PEDL		Protocol error in data phase interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
29	ARAL		Access to reserved address interrupt line.	0
		0	Interrupt assigned to interrupt line MCANx_INT0	
		1	Interrupt assigned to interrupt line MCANx_INT1	
31:30	-	-	Reserved.	-

35.8.15 Interrupt line enable register

The interrupt line enable register is used to enable or disable the two interrupt lines available.

Table 727. Interrupt line enable register (ILS, offset 0x05C) bit description

Bit	Symbol	Value	Description	Reset value
0	EINT0		Enable interrupt line 0.	0
		0	Interrupt line to MCANx_INT0 is disabled	
		1	Interrupt line to MCANx_INT0 is enabled	
1	EINT1		Enable interrupt line 1.	0
		0	Interrupt line to MCANx_INT1 is disabled	
		1	Interrupt line to MCANx_INT1 is enabled	
31:2	-	-	Reserved.	-

35.8.16 Global filter configuration register

Table 728. Global filter configuration register (GFC, offset 0x080) bit description

Bit	Symbol	Value	Description	Reset value
0	RRFE		Reject remote frames extended.	0
		0	Filter remote frames with 29-bit extended IDs	
		1	Reject all remote frames with 29-bit extended IDs	
1	RRFS		Reject remote frames standard.	0
		0	Filter remote frames with 11-bit standard IDs	
		1	Reject all remote frames with 11-bit standard IDs	

Table 728. Global filter configuration register (GFC, offset 0x080) bit description

Bit	Symbol	Value	Description	Reset value
3:2	ANFE		Accept non-matching frames extended. Defines how receives messages with 29-bit IDs that do not match any element of the filter list are treated.	0
		0x0	Accept in Rx FIFO 0	
		0x1	Accept in Rx FIFO 1	
		0x2	Reject	
		0x3	Reject	
5:4	ANFS		Accept non-matching frames standard. Defines how receives messages with 11-bit IDs that do not match any element of the filter list are treated.	0
		0x0	Accept in Rx FIFO 0	
		0x1	Accept in Rx FIFO 1	
		0x2	Reject	
		0x3	Reject	
31:6	-	-	Reserved.	-

35.8.17 Standard ID filter configuration register

The standard ID filter configuration register controls the filter path for standard messages and settings for the 11-bit standard message ID filter.

Table 729. Standard ID filter configuration register (SIDFC, offset 0x084) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	FLSSA	Filter list standard start address. Start address of the standard message ID filter list	
23:16	LSS	List size standard. 0 = No standard message ID filter 1 – 128 = Number of standard message ID filter elements >128 = Values of greater than 128 are interpreted as 128	
31:24	-	Reserved.	-

35.8.18 Extended ID filter configuration register

The extended ID filter configuration register controls the filter path for extended messages and settings for the 29-bit extended message ID filter.

Table 730. Extended ID filter configuration register (XIDFC, offset 0x088) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	FLESA	Filter list extended start address. Start address of the extended message ID filter list.	
23:16	LSE	List size extended. 0 = No extended message ID filter 1 – 64 = Number of extended message ID filter elements >64 = Values of greater than 64 are interpreted as 64	
31:24	-	Reserved.	-

35.8.19 Extended ID AND mask register

The extended ID AND mask register controls the acceptance filtering of extended frames. The value specified in the EIDM bits is ANDed with the message ID of a received frame. The reset value of the EIDM bits is all 1's, which makes the mask not active.

Table 731. Extended ID AND mask register (XIDAM, offset 0x08C) bit description

Bit	Symbol	Description	Reset value
28:0	EIDM	Extended ID mask.	0x1FFF FFFF
31:29	-	Reserved.	-

35.8.20 High priority message status register

The high priority message status register is updated every time a message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Table 732. High priority message status register (HPMS, offset 0x094) bit description

Bit	Symbol	Value	Description	Reset value
5:0	BIDX	-	Buffer index. Index of the Rx FIFO element to which the message was stored. This is only valid when the MSI bits are configured to store the message in either FIFO 0 or FIFO 1 (MSI = 0x2 or 0x3).	0
7:6	MSI		Message storage indicator.	0
		0x0	No FIFO selected	
		0x1	FIFO message lost	
		0x2	Message stored in FIFO 0	
		0x3	Message stored in FIFO 1	
14:8	FIDX	-	Filter index. Index of matching filter element. The range is 1 minus the LSS bits in the SIDFC register resp. LSE bits in the XIDFC registers.	0
15	FLST		Filter list. Indicates the filter list of the matching filter element.	
		0	Standard filter list	
		1	Extended filter list	
31:16	-	-	Reserved.	-

35.8.21 New data 1 register

The new data 1 register holds the new data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until they are cleared. A flag is cleared by writing a 1 to the corresponding bit position.

Table 733. New data 1 register (NDAT1, offset 0x098) bit description

Bit	Symbol	Description	Reset value
31:0	ND	New data. The most significant bit in this register corresponds to Rx buffer 31 while the least significant bit in this register corresponds to Rx buffer 0.	0

35.8.22 New data 2 register

The new data 2 register holds the new data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until they are cleared. A flag is cleared by writing a 1 to the corresponding bit position.

Table 734. New data 1 register (NDAT1, offset 0x098) bit description

Bit	Symbol	Description	Reset value
31:0	ND	New data. The most significant bit in this register corresponds to Rx buffer 63 while the least significant bit in this register corresponds to Rx buffer 32.	0

35.8.23 Rx FIFO 0 configuration register

Table 735. Rx FIFO 0 configuration register (RXF0C, offset 0x0A0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved.	-
15:2	F0SA	-	Rx FIFO 0 start address. Start address of Rx FIFO 0 in message RAM	0
22:16	F0S	-	Rx FIFO 0 size. 0 = No Rx FIFO 0 1 – 64 = Number of Rx FIFO 0 elements >64 = Values greater than 64 are interpreted as 64 The maximum Rx FIFO 0 elements are the value set in this register minus 1.	
23	-	-	Reserved.	0
30:24	F0WM	-	Rx FIFO 0 watermark. 0 = Watermark interrupt disabled 1 – 64 = Level for RF0W interrupt flag in the IR register >64 = Watermark interrupt disabled	
31	F0OM		FIFO 0 operation mode. The FIFO can be operated in block or overwrite mode.	0
		0	FIFO 0 blocking mode	
		1	FIFO 0 overwrite mode	

35.8.24 Rx FIFO 0 status register

Table 736. Rx FIFO 0 status register (RXF0S, offset 0x0A4) bit description

Bit	Symbol	Value	Description	Reset value
6:0	F0FL	-	Rx FIFO 0 fill level. Number of elements stored in Rx FIFO0, range 0 to 64.	0
7	-	-	Reserved.	-
13:8	F0GI	-	Rx FIFO 0 get index. Rx FIFO 0 read index pointer, range 0 to 63.	0
15:14	-	-	Reserved.	-
21:16	F0PI	-	Rx FIFO 0 put index. Rx FIFO 0 write index pointer, range 0 to 63.	0
23:22	-	-	Reserved.	-
24	F0F		Rx FIFO 0 full.	0
		0	Rx FIFO 0 not full	
		1	Rx FIFO 0 full.	
25	RF0L	-	Rx FIFO 0 message lost. This bit is a copy of the RF0L interrupt flag in the IR register. **Remark:** Overwriting the oldest message when the F0OM bit in the RXF0C register is set to 1 will not set this flag.	0
31:26	-	-	Reserved.	-

35.8.25 Rx FIFO 0 acknowledge register

After a message or sequence of messages have been read from Rx FIFO 0, the buffer index of the last element read from the Rx FIFO 0 must be written to the F0AI bits. This will set the Rx FIFO 0 get index bits to the Rx FIFO 0 acknowledge index bit value + 1 and update the FIFO 0 fill level bits.

Table 737. Rx FIFO 0 acknowledge register (RXF0A, offset 0x0A8) bit description

Bit	Symbol	Description	Reset value
5:0	F0AI	Rx FIFO 0 acknowledge index.	0
31:6	-	Reserved.	-

35.8.26 Rx buffer configuration register

The x buffer configuration register is used to configure the start address of the Rx buffers section in the message RAM.

Table 738. Rx buffer configuration register (RXBC, offset 0x0AC) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	RBSA	Rx buffer start address.	0
31:16	-	Reserved.	-

35.8.27 Rx FIFO 1 configuration register

Table 739. Rx FIFO 1 configuration register (RXF1C, offset 0x0B0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved.	-
15:2	F1SA	-	Rx FIFO 1 start address. Rx FIFO 1 start address. Start address of Rx FIFO 1 in message RAM.	0
22:16	F1S	-	Rx FIFO 1 size. 0 = No Rx FIFO 1 1 – 64 = Number of Rx FIFO 1 elements >64 = Values greater than 64 are interpreted as 64 The maximum Rx FIFO 1 elements are the value set in this register minus 1.	-
23	-	-	Reserved.	-
30:24	F1WM	-	Rx FIFO 1 watermark. 0 = Watermark interrupt disable 1 – 64 = Level for RF1W interrupt flag in the IR register. >64 = Watermark interrupt disabled	0
31	F1OM		FIFO 1 operation mode. The FIFO can be operated in block or overwrite mode.	0
		0	FIFO 1 blocking mode	
		1	FIFO 1 overwrite mode	

35.8.28 Rx FIFO 1 status register

Table 740. Rx FIFO 1 status register (RXF1S, offset 0x0B4) bit description

Bit	Symbol	Value	Description	Reset value
6:0	F1FL	-	Rx FIFO 1 fill level. Number of elements stored in Rx FIFO1, range 0 to 64.	0
7	-	-	Reserved.	-
13:8	F1GI	-	Rx FIFO 1 get index. Rx FIFO 1 read index pointer, range 0 to 63.	0
15:14	-	-	Reserved.	-
21:16	F1PI	-	Rx FIFO 1 put index. Rx FIFO 1 write index pointer, range 0 to 63.	-
23:22	-	-	Reserved.	-
24	F1F		Rx FIFO 1 full.	0
		0	Rx FIFO 1 not full	
		1	Rx FIFO 1 full.	
25	RF1L		Rx FIFO 1 message lost. This bit is a copy of the RF1L interrupt flag in the IR register. **Remark:** overwriting the oldest message when the F1OM bit from the RXF1C register is set to 1 will not set this flag.	
		0	No Rx FIFO 1 message lost.	
		1	Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.	
31:26	-	-	Reserved.	-

35.8.29 Rx FIFO 1 acknowledge register

After a message or sequence of messages have been read from Rx FIFO 1, the buffer index of the last element read from the Rx FIFO 1 has to be written to the F1AI bits. This will set the Rx FIFO 1 get index bits to the Rx FIFO 1 acknowledge index bit value + 1 and update the FIFO 1 fill level bits.

Table 741. Rx FIFO 1 acknowledge register (RXF1A, offset 0x0B8) bit description

Bit	Symbol	Description	Reset value
5:0	F1AI	Rx FIFO 1 acknowledge index.	0
31:6	-	Reserved.	-

35.8.30 Rx buffer and FIFO element size configuration register

The Rx buffer and FIFO element size configuration register configures the number of data bytes belonging to the Rx buffer and Rx FIFO element. Data field sizes that are greater than 8 bytes are intended for CAN FD operation only.

Table 742. Rx buffer and FIFO element size configuration register (RXESC, offset 0x0BC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	F0DS		Rx FIFO 0 data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
3	-	-	Reserved.	-
6:4	F1DS		Rx FIFO 1 data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
7	-	-	Reserved.	-
10:8	RBDS		Rx buffer data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
31:11	-	-	Reserved.	-

Remark: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx buffer or Rx FIFO, only the number of bytes as configured by RXESC register are stored to the Rx buffer resp. Rx FIFO element. The data field of the rest of the frame is ignored.

35.8.31 Tx buffer configuration register

Table 743. Tx buffer configuration register (TXBC, offset 0x0C0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-	-	Reserved.	-
15:2	TBSA	-	Tx buffers start address. Start address of Tx buffers in message RAM.	0
21:16	NDTB	-	Number of dedicated transmit buffers. 0 = No dedicated Tx buffers 1 – 32 = Number of dedicated Tx buffers >32 = Values greater than 32 are interpreted as 32	-
23:22	-	-	Reserved.	-
29:24	TFQS	-	Transmit FIFO/queue size. 0 = No tx FIFO/Queue 1 – 32 = Number of Tx buffers used for Tx FIFO/Queue >32 = Values greater than 32 are interpreted as 32	0
30	TFQM		Tx FIFO/queue mode.	0
		0	Tx FIFO operation	
		1	Tx queue operation	
31	-	-	Reserved.	-

Remark: The sum of TFQS and NDTB bits may not be greater than 32. There is no check for erroneous configurations. The Tx buffers section in the Message RAM starts with the dedicated Tx buffers.

35.8.32 Tx FIFO/queue status register

The Tx FIFO/queue status is related to the pending Tx requests listed in the TXBRP register. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan.

Table 744. Tx FIFO/queue status register (TXFQS, offset 0x0C4) bit description

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved.	-
12:8	TFGI	-	Tx FIFO get index. Tx FIFO read index pointer, range 0 to 31.	-
15:13	-	-	Reserved.	-
20:16	TFQPI	-	Tx FIFO/queue put index. Tx FIFO/queue write index pointer, range 0 to 31.	0
21	TFQF		Tx FIFO/queue full.	0
		0	Tx FIFO/queue not full	
		1	Tx FIFO/queue full	
31:22	-	-	Reserved.	-

Remark: In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get indices indicate the number of the Tx buffer starting with the first dedicated Tx buffers.

For example, a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

35.8.33 Tx buffer element size configuration register

Table 745. Tx buffer element size configuration register (TXESC, offset 0x0C8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	TBDS	-	Tx buffer data field size.	0
		0x0	8 byte data field	
		0x1	12 byte data field	
		0x2	16 byte data field	
		0x3	20 byte data field	
		0x4	24 byte data field	
		0x5	32 byte data field	
		0x6	48 byte data field	
		0x7	64 byte data field	
31:3	-	-	Reserved.	-

Remark: In case the data length code of a Tx buffer element is configured to a value higher than the TBDS bits, the bytes not defined by the Tx buffer are transmitted as “0xCC” as padding.

35.8.34 Tx buffer request pending register

Each Tx buffer has its own transmission request pending bit. The bits are set in the TXBAR register. The bits are reset after a requested transmission has completed or has been cancelled via the TXBCR register.

The TXBRP bits are set only for those Tx buffers configured in the TXBC register. After a TXBRP bit has been set, a Tx scan is started to check the pending Tx request with the highest priority.

A cancellation request resets the corresponding transmission request pending bit in the TXBRP register. In case a transmission has already been started when a cancellation is requested, it is done at the end of the transmission, regardless of the transmission success. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled using the TXBCF register:

- After a successful transmission together with the corresponding TXBTO bit.
- When a transmission has not yet been started at the point of cancellation.
- When the transmission has been aborted due to lost arbitration.
- When an error occurred during frame transmission.

In DAR mode, all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

Table 746. Tx buffer request pending register (TXBRP, offset 0x0CC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TRP		Transmission request pending.	0
		0	No transmission request pending	
		1	Transmission request pending	

Remark: TXBRP bits, which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is cancelled immediately and the corresponding TXBRP bit is reset.

35.8.35 Tx buffer add request register

Each Tx buffer has its own add request bit. The number of Tx buffers is configured in the TXBC register. Writing a 1 will set the corresponding add request bit while writing a 0 has no impact. This allows for setting transmission requests for multiple Tx buffers with one write. The TXBAR bits are only set for those Tx buffers configured in the TXBC register. When no Tx scan is running, the bits are reset immediately, otherwise the bits remain set until the Tx scan process is completed.

Table 747. Tx buffer add request register (TXBAR, offset 0x0D0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	AR		Add request.	0
		0	No transmission request added	
		1	Transmission request added	

Remark: If an add request is applied for a Tx buffer with pending transmission request, the add request is ignored.

35.8.36 Tx buffer cancellation request register

Each Tx buffer has its own add request bit. The number of Tx buffers is configured in the TXBC register. Writing a 1 will set the corresponding add request bit while writing a 0 has no impact. This allows for setting transmission requests for multiple Tx buffers with one write. The TXBCR bits are only set for those Tx buffers configured in the TXBC register. The bits remain set until the corresponding bit in the TXBRP register is reset.

Table 748. Tx buffer cancellation request register (TXBCR, offset 0x0D4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	CR		Cancellation request.	0
		0	No cancellation pending	
		1	Cancellation pending	

35.8.37 Tx buffer transmission occurred register

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding bits in the TXBRP register is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit in the TXBAR register.

Table 749. Tx buffer transmission occurred register (TXBTO, offset 0x0D8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TO		Transmission occurred.	0
		0	No transmission occurred	
		1	Transmission occurred	

35.8.38 Tx buffer cancellation finished register

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding bits in the TXBRP register is cleared after a cancellation was requested in the TXBCR register. In case the corresponding bits in the TXBRP register was not set at the point of cancellation, the cancellation finished bits in this register will be set immediately. The bits are reset when a new transmit cancellation is requested by writing a 1 to the corresponding bit in the TXBAR register.

Table 750. Tx buffer cancellation finished register (TXBCF, offset 0x0DC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TO		Cancellation finished.	0
		0	No transmit buffer cancellation	
		1	Transmit buffer cancellation finished	

35.8.39 Tx buffer transmission interrupt enable register

Table 751. Tx buffer transmission interrupt enable register (TXBTIE, offset 0x0E0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	TIE		Transmission interrupt enable. Each Tx buffer has its own transmission interrupt enable bit.	0
		0	Transmission interrupt disabled	
		1	Transmission interrupt enabled.	

35.8.40 Tx buffer cancellation finished interrupt enable register

Table 752. Tx buffer cancellation finished interrupt enable register (TXBCIE, offset 0x0E4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	CFIE		Cancellation finished interrupt enable. Each Tx buffer has its own transmission interrupt enable bit.	0
		0	Cancellation finished interrupt disabled	
		1	Cancellation finished interrupt enabled	

35.8.41 Tx event FIFO configuration register

Table 753. Tx event FIFO configuration register (TXEFC, offset 0x0F0) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
15:2	EFSA	Event FIFO start address. Start address of the Tx event FIFO in message RAM.	0
21:16	EFS	Event FIFO size. 0 = Tx event FIFO disabled 1 – 32 = Number of Tx event FIFO elements >32 = Values greater than 32 are interpreted as 32 The maximum Tx FIFO elements are the value set in this register minus 1.	0
23:22	-	Reserved.	-
29:24	EFWM	Event FIFO watermark. 0 = Watermark interrupt disabled 1 – 32 = Level for TEFW interrupt flag in the IR register >32 = Watermark interrupt disabled	0
31:30	-	Reserved.	-

35.8.42 Tx event FIFO status register

Table 754. Tx event FIFO status register (TXEFS, offset 0x0F4) bit description

Bit	Symbol	Value	Description	Reset value
5:0	EFFL	-	Event FIFO fill level. Number of elements stored in Tx event FIFO, range 0 to 32.	0
7:6	-	-	Reserved.	-
12:8	EFGI	-	Event FIFO get index. Tx event FIFO read index pointer, range 0 to 31.	0
15:13	-	-	Reserved.	-
21:16	EFPI	-	Event FIFO put index. Tx event FIFO write index pointer, range 0 to 31.	0
23:22	-	-	Reserved.	-
24	EFF		Event FIFO full.	0
		0	Tx event FIFO not full	
		1	Tx event FIFO full	
25	TEFL		Tx event FIFO element lost. This bit is a copy of the TEFL interrupt flag in the IR register. When the TEFL bit is reset, this bit is also reset.	0
		0	No Tx event FIFO element lost.	
		1	Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.	
31:26	-	-	Reserved.	-

35.8.43 Tx event FIFO acknowledge register

After an element or sequence of elements have been read from the Tx event FIFO, the index of the last element read has to be written to the event FIFO acknowledge index bits in the EFAI bits. This will set the Tx event FIFO get index bits to the value stored in the EFAI +1 and update the event FIFO fill level.

Table 755. Tx event FIFO acknowledge register (RXF1A, offset 0x0F8) bit description

Bit	Symbol	Description	Reset value
4:0	EFAI	Event FIFO acknowledge index.	0
31:5	-	Reserved.	-

35.8.44 Message RAM base address register

Table 756. Message RAM base address register (MRBA, offset 0x200) bit description

Bit	Symbol	Description	Reset value
31:0	BA	Base address for the message RAM in the chip memory map.	0

35.8.45 External timestamp counter configuration register

Table 757. External timestamp counter configuration register (ETSCC, offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
10:0	ETCP	-	External timestamp prescaler value. The HCLK is divided down by the prescaler value plus 1 to clock the external timestamp counter.	0
30:11	-	-	Reserved.	-
31	ETCE		External timestamp counter enable.	0
		0	External timestamp counter is disabled	
		1	External timestamp counter is enabled	

35.8.46 External timestamp counter value register

Table 758. External timestamp counter value register (ETSCV, offset 0x600) bit description

Bit	Symbol	Description	Reset value
15:0	ETSC	External timestamp counter. Read to return current counter value. Write to initialize the counter with the specified 16-bit value.	0
31:16	-	Reserved.	-

35.9 Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of an Rx buffer / FIFO element is shown in Figure 130. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the RXESC register.

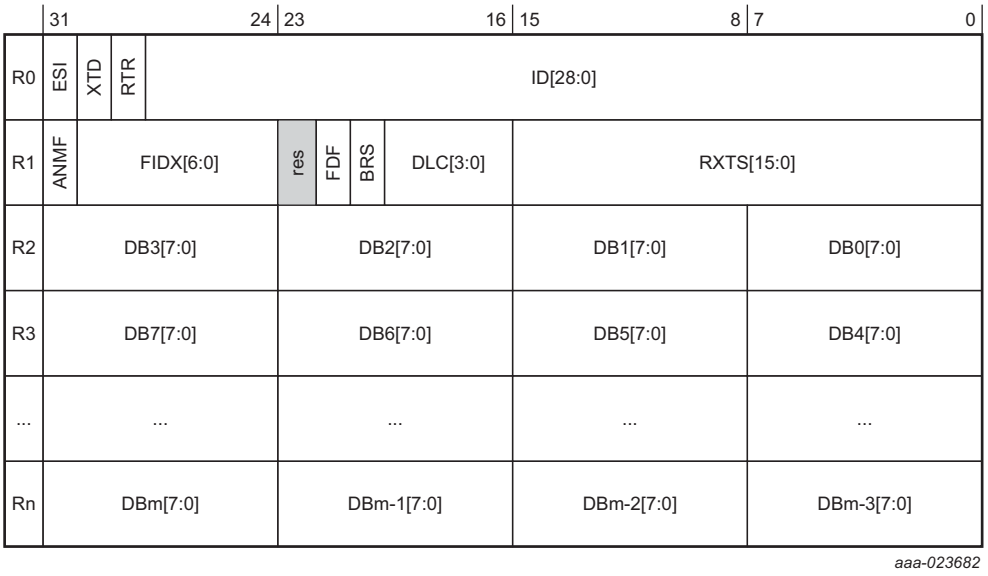


Fig 130. Rx buffer and FIFO element

Table 759. R0 bit description

Bit	Symbol	Value	Description
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.
29	RTR		Remote transmission request. This bit is set depending on whether the received frame is a data frame or a remote frame. **Remark:** There are no remote frames in CAN FD format.
		0	Received frame is a data frame
		1	Received frame is a remote frame
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier
31	ESI		Error state identifier.
		0	Transmitting node is error active
		1	Transmitting node is error passive

Table 760. R1 bit description

Bit	Symbol	Value	Description
15:0	RXTS	-	Rx timestamp. Timestamp counter value captures on start of frame reception. Resolution depending on configuration of timestamp counter prescaler bit in the TSCC register.
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: received frame has 0 - 8 data bytes 9 – 15 = CAN: received frame has 8 data bytes 9 – 15 = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
20	BRS		Bit rate switch.
		0	Frame received without bit rate switching
		1	Frame received with bit rate switching
21	FDF		FD format.
		0	Standard frame format.
		1	CAN FD frame format (new DLC-coding and CRC).
31	ESI		Error state identifier
		0	Transmitting node is error active
		1	Transmitting node is error passive
23:22	-	-	Reserved.
30:24	FIDX	-	Filter index. 0 – 127 = Index of matching Rx acceptance filter element (invalid if the ANMF bit is set) Range is 0 to one minus the LSS bits in the SIDFC register resp. the LSE bit in the XIDFC register
31	ANMF		Accepted non-matching frame. Acceptance of non-matching frames may be enabled via the ANFS and ANFE bits in the GFC register.
		0	Received frame matching filter index FIDX
		1	Received frame did not match any Rx filter element

Table 761. R2 bit description

Bit	Symbol	Description
7:0	DB	Data byte.
15:8	DB	Data byte.
23:16	DB	Data byte.
31:24	DB	Data byte.

Remark: Depending on the configuration of the element size in the RXESC register, between two and sixteen 32-bit words are used for storage of a CAN message's data field.

35.10 Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx Queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx Queue by evaluating the TFQS and NDTB bits in the TXBC register. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the TXESC register.

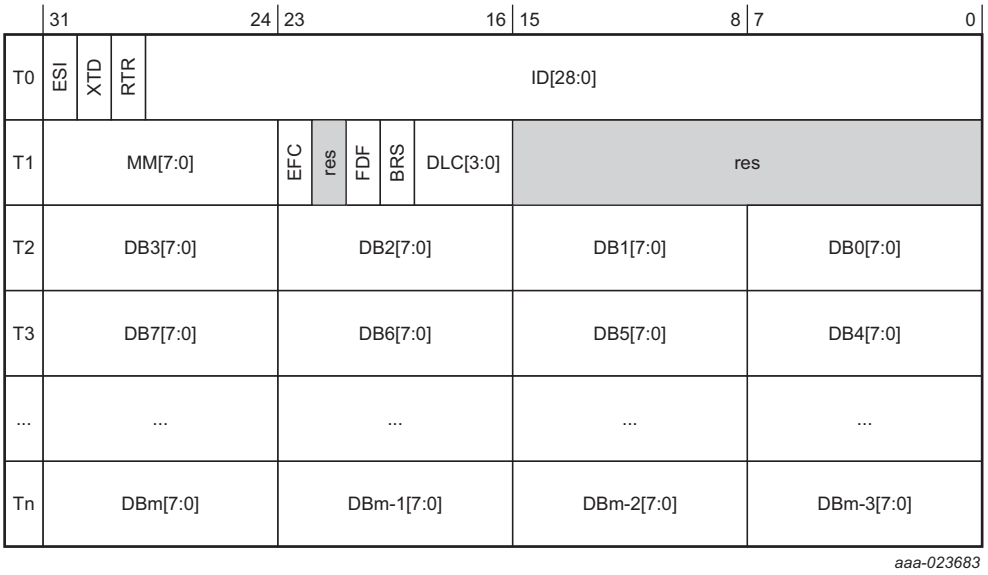


Fig 131. Tx buffer element

Table 762. T0 bit description

Bit	Symbol	Value	Description
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.
29	RTR		Remote transmission request. **Remark:** When this bit is set, the MCAN transmits a remote frame according to ISO11898-1, even if the FDOE bit of the CCCR register enables the transmission in CAN FD format.
		0	Transmit data frame
		1	Transmit remote frame
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier

Table 762. T0 bit description

Bit	Symbol	Value	Description
31	ESI		Error state identifier. **Remark:** The ESI bit of the transmit buffer is OR'd with the passive flag to decide the value of the ESI bit in the transmitted FD frame. Per the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit ESI bit recessive.
		0	ESI bit in CAN FD format depends only on error passive flag
		1	ESI bit in CAN FD format transmitted recessive

Table 763. T1 bit description

Bit	Symbol	Value	Description
15:0	-	-	Reserved.
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: transmit frame has 0 - 8 data bytes 9 – 15 = CAN: transmit frame has 8 data bytes 9 – 15 = CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
20	BRS		Bit rate switch.
		0	CAN FD frames transmitted without bit rate switching
		1	CAN FD frames transmitted with bit rate switching
21	FDF		FD format.
		0	Frame transmitted in classic CAN format
		1	Frame transmitted in CAN FD format
22	-		Reserved.
23	EFC		Event FIFO control.
		0	Do not store Tx events
		1	Store Tx events
31:24	MM	-	Message Marker. Written during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.

Table 764. R2 to Rn bit description

Bit	Symbol	Description
7:0	DB	Data byte.
15:8	DB	Data byte.
23:16	DB	Data byte.
31:24	DB	Data byte.

Remark: Depending on the configuration of the element size in the RXESC register, between two and sixteen 32-bit words are used to store the data field of a CAN message.

35.11 Tx event FIFO element

Each element stores information about transmitted messages that can be used to get information of the order of transmitted messages. Status information about the Tx event FIFO can be obtained from the TXEFS register.

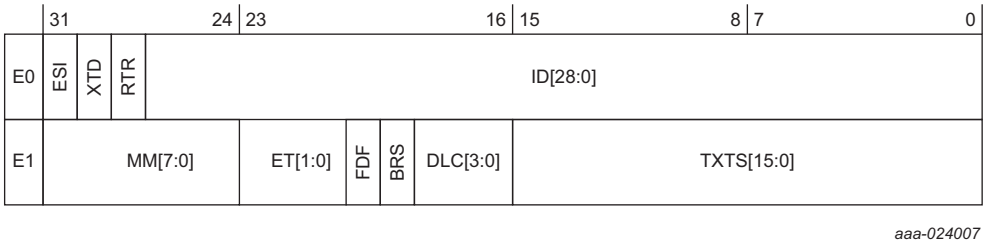


Fig 132. Tx event FIFO element

Table 765. E0 bit description

Bit	Symbol	Value	Description
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.
29	RTR		Remote transmission request.
		0	Data frame transmitted
		1	Remote frame transmitted
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier
31	ESI		Error state identifier.
		0	Transmitted node is error active
		1	Transmitted node is error passive

Table 766. E1 bit description

Bit	Symbol	Value	Description
15:0	TXTS	-	Tx timestamp. Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the TCP bit in the TSCC register.
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: frame with 0 - 8 data bytes transmitted 9 – 15 = CAN: frame with 8 data bytes transmitted 9 – 15 = CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
20	BRS		Bit rate switch.
		0	Frames transmitted without bit rate switching
		1	Frames transmitted with bit rate switching
21	FDF		FD Format.
		0	Standard frame format
		1	CAN FD frame format (new DLC-coding and CRC)

Table 766. E1 bit description

Bit	Symbol	Value	Description
23:22	ET		Event type.
		0x0	Reserved
		0x1	Tx event
		0x2	Transmission in spite of cancellation (always set for transmission in DAR mode)
		0x3	Reserved
31:24	MM	-	Message Marker. Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.

35.12 Standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is equal to the filter list standard start address bit field in the SIDFC register plus the index of the filter element.

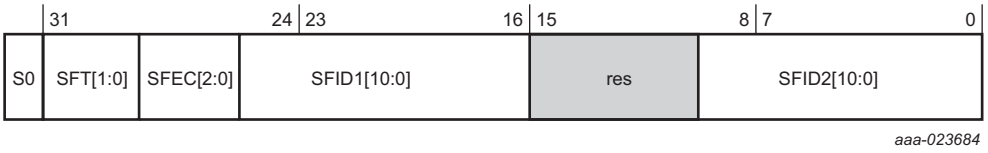


Fig 133. Standard message ID filter element

Table 767. S0 bit description

Bit	Symbol	Value	Description
10:0	SFID2	-	Standard filter ID 2. This bit field has a different meaning depending on the configuration of the SFEC register. 1. SFEC = 0x1 – 0x6, this bit field becomes the second ID of standard ID filter element. 2. SFEC = 0x7, this bit field is used to filter for Rx buffers or for debug messages. Bits 10:9 decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 0x0 = Store message into an Rx buffer 0x1 = Debug message A 0x2 = Debug message B 0x3 = Debug message C Bits 8:6 are used to control the filter event pins at the extension interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN function clock period in case the filter matches. Bits 5:0 define the offset to the Rx buffer start address bit field in the RXBC register for storage of a matching message.
15:11	-	-	Reserved.
26:16	SFID1	-	Standard filter ID 1. First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
29:27	SFEC		Standard filter element configuration. All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If this bit field is set to 0x4, 0x5, or 0x6, a match sets the HPM interrupt flag in the IR register, if enabled, an interrupt is generated. In this case, the HPMS register is updated with the status of the priority match.
		0x0	Disable filter element
		0x1	Store in Rx FIFO 0 if filter matches
		0x2	Store in Rx FIFO 1 if filter matches
		0x3	Reject ID if filter matches
		0x4	Set priority if filter matches
		0x5	Set priority and store in FIFO 0 if filter matches
		0x6	Set priority and store in FIFO 1 if filter matches
		0x7	Store into Rx buffer or as debug message, configuration of the SFT bit field is ignored

Table 767. S0 bit description

Bit	Symbol	Value	Description
31:30	SFT		Standard filter type.
		0x0	Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1)
		0x1	Dual ID filter for SFID1 or SFID2
		0x2	Classic filter: SFID1 = filter, SFID2 = mask
		0x3	Filter element disabled

35.13 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is equal to the filter list extended start address bit field in the XIDFC register plus two times the index of the filter element.

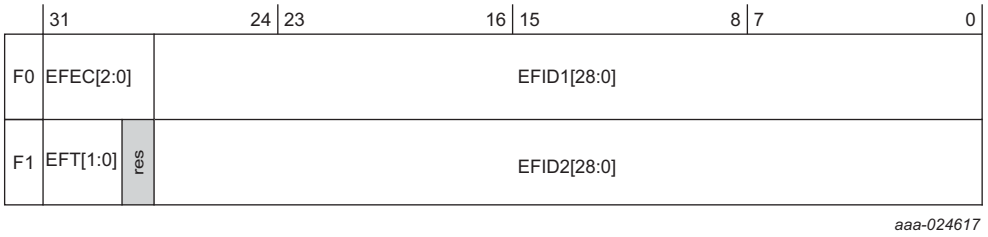


Fig 134. Extended message ID filter element

Table 768. F0 bit description

Bit	Symbol	Value	Description
28:0	EFID1	-	Extended filter ID 1. First ID of extended ID filter element. When filtering for Rx buffers or for debug messages, this field defines the ID of an extended message to be stored. The received identifiers must match exactly when masked with the XIDAM register.
31:29	EFEC		Extended filter element configuration. All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If this bit field is equal to 0x4, 0x5, or 0x6, a match sets the HPM interrupt flag in the IR register and, if enabled, an interrupt is generated. In this case, the HPMS register is updated with the status of the priority match.
		0x0	Disable filter element
		0x1	Store in Rx FIFO 0 if filter matches
		0x2	Store in Rx FIFO 1 if filter matches
		0x3	Reject ID if filter matches
		0x4	Set priority if filter matches
		0x5	Set priority and store in FIFO 0 if filter matches
		0x6	Set priority and store in FIFO 1 if filter matches
		0x7	Store into Rx buffer or as debug message, configuration of the EFT bit field is ignored

Table 769. F1 bit description

Bit	Symbol	Value	Description
28:0	EFID2	-	Extended filter ID 2. This bit field has a different meaning depending on the configuration of the EFEC bit field. 1. EFEC = 0x1 – 0x6, this bit field becomes the second ID of extended ID filter element. 2. EFEC = 0x7, this bit field is used to filter for Rx buffers or debug messages. Bits 10:9 decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 0x0 = Store message into an Rx buffer 0x1 = Debug message A 0x2 = Debug message B 0x3 = Debug message C Bits 8:6 is used to control the filter event pins at the extension interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN function clock period in case the filter matches. Bits 5:0 defines the offset to the Rx buffer start address bit field in the RXBC register for storage of a matching message.
29	-	-	Reserved.
31:30	EFT		Extended filter type.
		0x0	Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1)
		0x1	Dual ID filter for EFID1 or EFID2
		0x2	Classic filter: EFID1 = filter, EFID2 = mask
		0x3	Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), the mask in the XIDAM register is not applied

35.14 Functional description

35.14.1 Operating modes

35.14.1.1 Software initialization

Software initialization is started by setting the INIT bit in the CCCR register, either by software or by a hardware reset, when an uncorrected bit error was detected in the message RAM, or by going Bus_Off. While the INIT bit is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output m_can_tx is recessive. The counters of the error management logic are unchanged. Setting the INIT bit does not change any configuration register. Resetting the INIT bit finishes the software initialization. Afterwards, the bit stream processor synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits, indicating the bus is idle, before it can take part in bus activities and start the message transfer.

Access to the MCAN configuration registers is only enabled when both the INIT and CCE bits are set in the CCCR register.

The CCE bit can only be set or reset while the INIT bit is set. The CCE bit is automatically reset when the INIT bit is reset.

The following registers are reset when the CCE bit is set:

- HPMS – High priority message status.
- RXF0S – Rx FIFO 0 status.
- RXF1S – Rx FIFO 1 status.
- TXFQS – Tx FIFO/queue status.
- TXBTO – Tx buffer transmission occurred.
- TXBRP – Tx buffer request pending.
- TXBCF – Tx buffer cancellation finished.
- TXEFS – Tx event FIFO status.

The timeout counter value in the TOCV register is preset to the value configured by the TOP bits in the TOCC register, while the CCE bit is set in the CCCR register. In addition, the state machines of the Tx and Rx handlers are held in an idle state while the CCE bit is set.

The following registers are only writable while the CCE bit is cleared:

- TXBAR – Tx buffer add request.
- TXBCR – Tx buffer cancellation request.

The TEST and MON bits in the CCCR register can only be set while the INIT and CCE bits are set. Both bits may be reset at any time. The DAR bit can only be set or reset while the INIT and CCE bit are set.

35.14.1.2 Normal operation

When the MCAN is initialized and the INIT bit in the CCCR register is cleared, the MCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into Rx FIFO 0 or Rx FIFO 1.

To transmit messages, dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

35.14.1.3 CAN FD operation

There are two variants in the CAN FD frame transmission. The first is the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the MCAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled in the CCCR register, this causes the operation state to change from Receiver at the next sample point, see the ACT bits in the PSR register. In case Protocol Exception Handling is disabled in the CCCR register, the MCAN will treat a recessive res bit as a form error and will respond with an error frame.

With FDOE bit cleared in the CCCR register, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With FDOE bit set and the BRSE bit cleared in the CCCR register, only bit FDF of a Tx buffer element is evaluated. With FDOE and BRSE bit set in the CCCR register, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Afterwards, all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in the standard CAN. Table 770 shows the DLC codes 9 to 15. In the standard CAN, codes 9 to 15 code a data field of 8 bytes.

Table 770. DLC coding in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the Bit Rate Switch (BRS) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the NBTP register. In the following CAN FD data phase, the data phase bit timing is used as defined by the DBTP register. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. For example, if the CAN clock frequency is 20 MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the Error Status Indicator (ESI) bit is determined by the error state of the transmitter at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, otherwise it is transmitted dominant.

35.14.1.4 Restricted operation mode

Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The REC and TEC error counters in the ECR register are frozen while the CEL bit is set. The MCAN can be put into restricted operation mode by setting the ASM bit in the CCCR register. The bit can only be set when CCE and INIT bits are set in the CCCR register. The bit can be cleared at any time.

Restricted operation mode is automatically entered when the Tx Handler was not able to read data from the message RAM in time. To leave restricted operation mode, the ASM bit in the CCCR register must be cleared.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

If the MCAN is connected to the clock calibration on the CAN unit, the ASM bit in the CCCR register is controlled by input m_can_cok. In case MCAN_COK switches to 0, the ASM bit is set. When the m_can_cok switches back to 1, the ASM bit returns to the previously written value. When there is no clock calibration on the CAN unit's connected input, m_can_cok is hardwired to 1.

Remark: The restricted operation mode must not be combined with the loop back mode, whether internal or external.

35.14.1.5 Bus monitoring mode

The MCAN is set in bus monitoring mode by setting the MON bit in the CCCR registers. In bus monitoring mode (see ISO11898-1, 10.12 Bus monitoring), the MCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the MCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the MCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode, the TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 135 shows the connection of the m_can_tx and m_can_rx signals to the MCAN in bus monitoring mode.

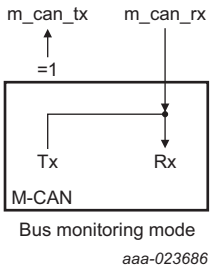


Fig 135. Pin control in bus monitoring mode

35.14.1.6 MCAN power down mode (sleep mode)

The MCAN can be set into power down mode controlled by the CSR bit in the CCCR register.

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the MCAN sets the INIT bit in the CCCR register to 1 to prevent any more CAN transfers. Now the MCAN acknowledges that it is ready for power down by setting the CSA bit to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to INIT bit will have no effect.

To leave the MCAN power down mode, the application has to turn on the module clocks before resetting the CSR bit in the CCCR register. The MCAN acknowledges this by resetting the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

35.14.1.7 Test modes

To enable write access to the test register, the TEST bit in the CCCR register must be set. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin CAN1_TD by programming the TX bit field in the TEST register. Additionally to its default function – the serial data output – it can drive the CAN sample point signal to monitor the bit timing f the MCAN and it can drive constant dominant or recessive values. The actual value at pin CAN1_RD can be read from the RX bit field in the TEST register. Both functions can be used to check the physical layer of the CAN bus.

The synchronization mechanism in the CAN IP may cause a delay of several clock periods between writing to the TX bit field until the new configuration is visible at output pin m_can_tx. This applies also when reading input pin CAN1_RD via the RX bit field.

Remark: Test modes should be used for production tests or self-tests only. The software control for the m_can_tx interferes with all CAN protocol functions. It is not recommended to use test modes for applications.

35.14.1.7.1 External loop back mode

The MCAN can be set in external loop back mode by setting the LBCK bit in the TEST register. In loop back mode, the MCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx buffer or an Rx FIFO. Figure 12 shows the connection of signals m_can_tx and CAN1_RD to the MCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the MCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode the MCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN1_RD input pin is disregarded by the MCAN. The transmitted messages can be monitored at the m_can_tx pin.

35.14.1.7.2 Internal loop back mode

Internal loop back mode is entered by setting the LBCK bit in the TEST register and the MON bit in the CCCR register. This mode can be used for a “Hot Selftest”, meaning, the MCAN can be tested without affecting a running CAN system connected to the pins

m_can_tx and CAN1_RD. In this mode pin CAN1_RD is disconnected from the MCAN and pin m_can_tx is held recessive. Figure 136 shows the connection of m_can_tx and CAN1_RD to the MCAN in case of internal loop back mode.

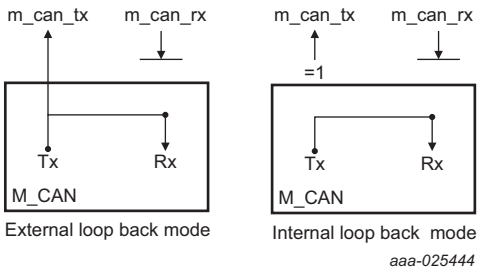


Fig 136. Pin control in loop back modes

35.14.2 Transmitter delay compensation

During the data phase of a CAN FD transmission, only one node transmits while all others are receivers. The length of the bus line has no impact. When transmitting with the CANx_TD pin, the M_CAN receives the transmitted data from its local CAN transceiver via the CANx_RD pin. The received data is delayed by the transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

35.14.2.1 Description

The protocol unit of the MCAN has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting the TDC bit in the DBTP register.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the MCAN's transmit output through the transceiver to the receive input plus the transmitter delay compensation offset as configured by TDCO bit field in the TDCR register. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of MCAN clock periods.

The TDCV bit field in the PSR shows the actual transmitter delay compensation value. This bit field is cleared when INIT bit in the CCCR register is set and is updated at each transmission of an FD frame while TDC bit is set in the DBTP register.

The following boundary conditions have to be considered for the transmitter delay compensation implement in the MCAN:

- The sum of the measured delay from the m_can_tx to the CAN1_RD and the configured transmitter delay compensation offset bit field in the TDCR register has to be less than 6 bit times in the data phase.
- The sum of the measured delay from the m_can_tx to the CAN1_RD and the configured transmitter delay compensation offset bit field in the TDCR register has to be less or equal 127 MCAN clock periods. In case this sum exceeds 127 MCAN clock periods, the maximum value of 127 MCAN clock periods is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter that stops checking of receive bits at the SSPs.

35.14.2.2 Transmitter delay compensation measurement

If the transmitter delay compensation is enabled by setting the TDC bit in the DBTP register, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the MCAN_RX of the transmitter. The resolution of this measurement is one MCAN clock periods.

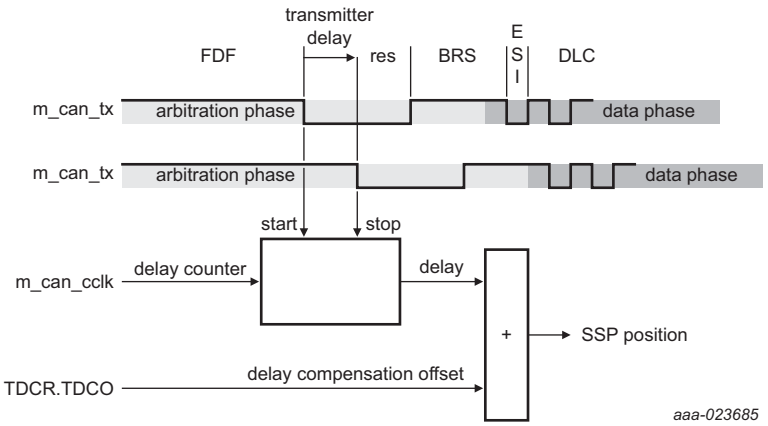


Fig 137. Transmitter delay measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming the TDCF bit field in the TDCR register. This defines a minimum value for the SSP position. Dominant edges on CAN1_RD that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least equal to the TDCF bit field and m_can_rx is low.

35.14.3 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the MCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic

retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled using the DAR bit field in the CCCR register.

35.14.3.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx buffer's Tx request pending bit in the TXBRP register is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is not set.
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is set.
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is not set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is set.

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type = '10' (transmission in spite of cancellation).

35.14.4 Timestamp generation

For timestamp generation, the MCAN supplies a 16-bit wrap-around counter. The TCP bit field in the TSCC register contains a prescaler that can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via the TSC bit field in the TSCV register. A write access to the TSCV register resets the counter to zero.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx buffer / Rx FIFO (RXTS register) or Tx Event FIFO (TXTS register) element.

By programming the TSS bit in the TSCC register, the 16-bit external timestamp counter can be used.

35.14.5 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the MCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP bit field in the TSCC register. The timeout counter is configured via the TOCC register. The actual counter value can be read from TOC bit field in the TSCV. The timeout counter can only be started while the INIT bit in the CCCR register is cleared. It is stopped when the INIT bit is set. For example, when the M_CAN enters a buf_off state.

The operation mode is selected by the TOS bit field in the TOCC register. When operating in continuous mode, the counter starts when the INIT bit is cleared. A write to the TOCV register presets the counter to the value configured by TOP bit field in the TOCC register and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP bit field in the TOCC register. Down-counting is started when the first FIFO element is stored. Writing to TOCV register has no effect.

When the counter reaches zero, the TOO interrupt flag in the IR register is set. In continuous mode, the counter is immediately restarted at the value in the TOP bit field in the TOCC register.

Remark: The clock signal for the timeout counter is derived from the CAN core's sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the bit rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

35.14.6 Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to the Rx buffers or to one of the two Rx FIFOs, and Rx FIFO's put and get indices.

35.14.6.1 Acceptance filtering

The MCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx buffer or to Rx FIFO 0,1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as a range filter, filter for one or two dedicated IDs, or classic bit mask filter.
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled or disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element.

The related configuration registers are:

- Global filter configuration (GFC).
- Standard ID filter configuration (SIDFC).
- Extended ID filter configuration (XIDFC.
- Extended ID AND mask (XIDAM).

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1.
- Store received frame in Rx buffer.
- Store received frame in Rx buffer and generate pulse at filter event pin.
- Reject received frame.
- Set high priority message interrupt flag (HPM bit in the IR register).
- Set high priority message interrupt flag and store the received frame in FOF 0 or FIFO 1.

Acceptance filtering is started after the complete identifier is received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in 32 bit chunks to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx buffer or Rx FIFO:

- Rx buffer
  - New data flag of matching Rx buffer is not set, but the Rx buffer is partially overwritten with received data. For error types, see the LEC and DLEC bit field in the PSR register.
- Rx FIFO
  - Put index of matching Rx FIFO is not updated, but the related Rx FIFO element is partially overwritten with received data. For error types, see the LEC and DLEC bit field in the PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in Section X have to be considered.

Remark: When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independent of the filters used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

35.14.6.1.1 Range filter

The filter matches for all received frames with message IDs in the range defined by the SFID1/SFID2 resp. EFID1/EFID2.

There are two possibilities when range filtering is used together with extended frames:

EFT = “00”: the message ID of received frames is ANDed with the extended ID AND mask register before the range filter is applied.

EFT = “11”: the extended ID AND mask register is not used for range filtering.

35.14.6.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SFID1= SFID2 resp. EFID1= EFID2.

35.14.6.1.3 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering, SFID1/EFID1 is used as message ID filter, while SFID2/EFID2 is used as filter mask.

A 0 bit at the filter mask will mask out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are zero, all message IDs match.

35.14.6.1.4 Standard message ID filtering

Figure 138 shows the flow of standard message ID filtering (11-bit identifier).

Controlled by the GFC and SIDFC registers, the message ID, remote transmission request bit (RTR), and the identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

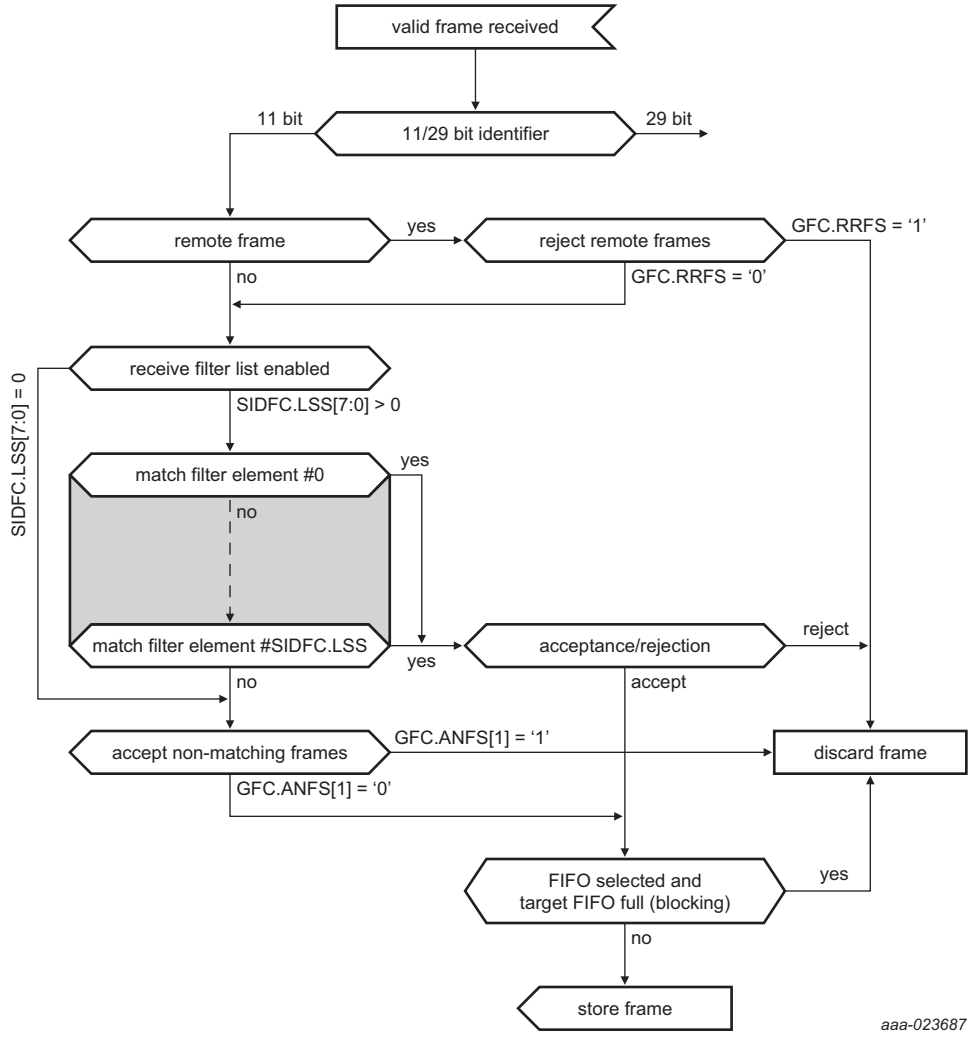


Fig 138. Standard message ID filter path

35.14.6.1.5 Extended message ID filtering

Figure 139 shows the flow of standard message ID filtering (11-bit identifier). The standard message ID filter element is described in Section 35.14.6.1.4 "Standard message ID filtering".

Controlled by the GFC and SIDFC registers, the message ID, remote transmission request bit (RTR), and the identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

The XIDAM register is ANDed with the received identifier before the filter list is execute.

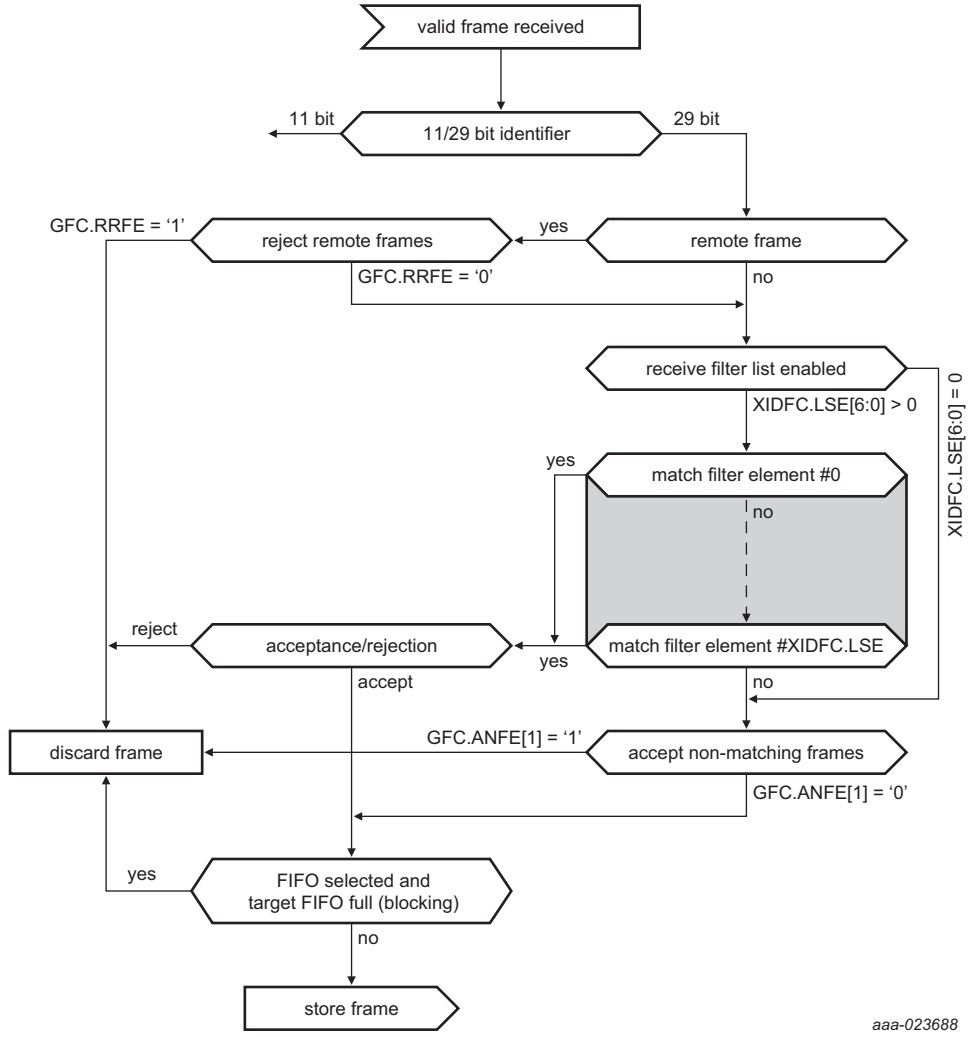


Fig 139. Extended message ID filter path

35.14.6.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via the RXF0C and RXF1C registers.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see Section 35.14.6.1 "Acceptance filtering". The Rx FIFO element is described in Section 35.9 "Rx buffer and FIFO element".

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by Rx FIFO watermark bit fields in the RX FIFO configuration registers, the Rx FIFO watermark interrupt flag in the IR register is set. When the Rx FIFO put index reaches the Rx FIFO get index an Rx FIFO full condition is signaled by Rx FIFO full bit fields in the Rx FIFO status registers. In addition, the Rx FIFO full interrupt flag in the IR register is set.

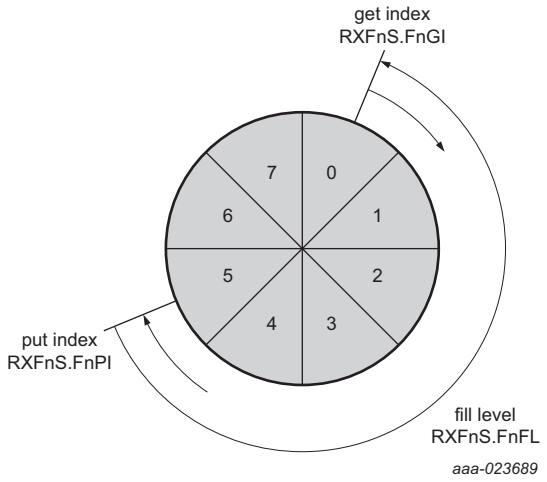


Fig 140. Rx FIFO status

When reading from an Rx FIFO, the Rx FIFO get index bit field in the Rx FIFO status register * FIFO element size has to be added to the corresponding Rx FIFO start address in the Rx FIFO configuration register.

Table 771. Rx buffer / FIFO element size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field (bytes)	FIFO element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

35.14.6.2.1 Rx FIFO blocking mode

The Rx FIFO blocking mode is configured in the RX FIFO configuration registers. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (Rx FIFO put index = the Rx FIFO get index), no more messages are written in the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled when the Rx FIFO full bit fields are set. In addition the Rx FIFO full interrupt flag in the IR register is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled in the Rx FIFO status register. In addition, the Rx FIFO lost interrupt flag in the IR register is set.

35.14.6.2.2 Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured in the Rx FIFO configuration register.

When an Rx FIFO full condition (Rx FIFO put index = the Rx FIFO get index) is signaled by the Rx FIFO full bit fields in Rx FIFO status register, the next message accepted by the FIFO will overwrite the oldest FIFO message. The put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for this is that a received message is written to the message RAM (put index) while the MCU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the MCU accesses the Rx FIFO. Figure 141 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

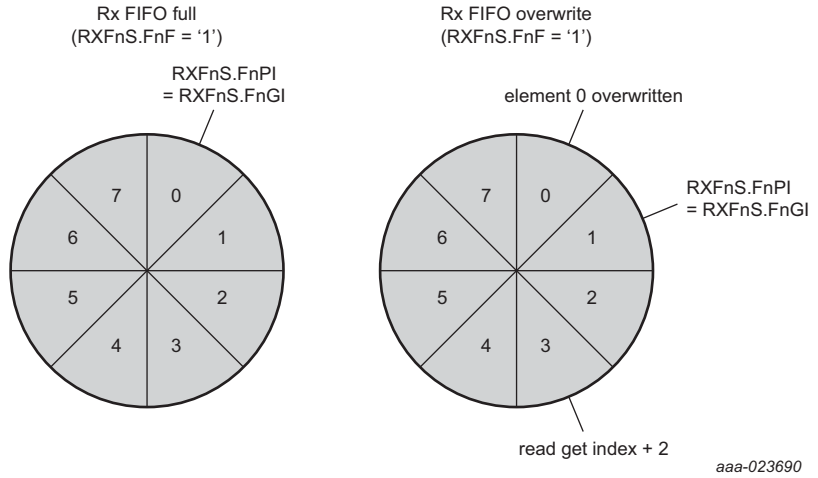


Fig 141. Rx FIFO overflow handling

35.14.6.2.3 Dedicated Rx buffers

The MCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via the RBSA bit in the RXBC register.

For each Rx buffer a Standard or extended message ID filter element with SFEC / EFEC = "111" and SFID2 / EFID2[10:9] = "00" has to be configured.

After a received message is accepted by a filter element, the message is stored into the Rx buffer in the message RAM that is referenced by the filter element. The format is the same as an Rx FIFO element. In addition, the DRX interrupt flag is set in the IR register.

Table 772. Example filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective new data flags are set in the NDAT registers. As long as the new data flag is set, the respective Rx buffer is locked against updates from received matching frames. The new data flags have to be reset by writing a 1 to the respective bit position.

While an Rx buffer's new data flag is set, a message ID filter element referencing this specific Rx buffer will not match, causing the acceptance filtering to continue. Subsequent message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

35.14.6.2.4 Rx buffer handling

- Reset the DRX interrupt flag in the IR register.
- Read data from the NDAT registers.
- Read messages from the message RAM.
- Reset the NDAT flags of processed messages.

35.14.7 Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO. Up to 32 Tx buffers can be set up for message transmission. The CAN mode for transmission (classic CAN or CAN FD) can be configured separately for each Tx buffer element. The Tx buffer element is described in Section 35.10 "Tx buffer element". Figure 19 below describes the possible configurations for frame transmission.

Table 773. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD without bit rate switching

Remark: AUTOSAR requires at least three Tx queue buffers and support for transmit cancellation

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the TXBRP register is updated, or when a transmission has been started.

35.14.7.1 Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

For example, if CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by the TXP bit in the CCCR register. If the bit is set, the MCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, transmit pause is disabled in the CCCR register.

This feature breaks up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

35.14.7.2 Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmissions under complete control of the Host CPU. Each dedicated Tx buffer is configured with a specific message ID. In case multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section id updated, a transmission is requested by the add request bit fields in the TXBAR register. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

Table 774. Tx buffer / FIFO / queue element size

TXESC.TBDS[2:0]	Data Field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8

Table 774. Tx buffer / FIFO / queue element size *…continued*

TXESC.TBDS[2:0]	Data Field (bytes)	Element size (RAM words)
101	32	10
110	48	14
111	64	18

35.14.7.3 Tx FIFO

Tx queue operation is configured by setting the TFQM bit in the TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the Tx FIFO queue put index bit field in the TXFQS register. An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full, the put index is not valid and no more message should be written to the Tx queue until at least one of the requested messages is sent out or a pending transmission request is cancelled.

The application may use the TXBRP register instead of the put index and may place messages to any Tx buffer without pending transmission request.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

35.14.7.4 Tx queue

Tx queue operation is configured by programming the TFQM bit field to a value of one. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case that multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the put index stored in the TFQPI bit field in the TXFQS register. An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (if the TFQF bit field is set in the TXFQS register), the put index is not valid and no more messages should be written to the Tx queue until at least one of the requested messages is sent out or a pending transmission request has been cancelled.

The application may use the TXBRP register instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocated element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx queue buffer is calculated by adding Tx FIFO/queue put index sorted in the TFQPI bit field in the TXFQS register * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

35.14.7.5 Mixed dedicated Tx buffers / Tx FIFO

The Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by the NDTB bit field in the TXBC register. The number of Tx buffers assigned to the Tx FIFO is configured by the TFQS bit field in the TXBC register. In case the TFQS bit field is programmed to zero, only dedicated buffers are used.

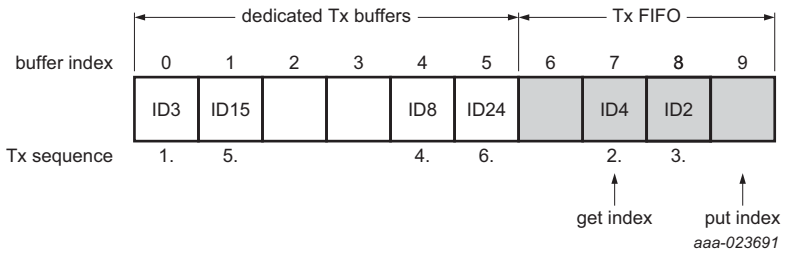


Fig 142. Example of mixed configuration dedicated Tx buffers / Tx FIFO

Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by the TFGI bit field in the TXFS register).
- Buffer with lowest message ID gets highest priority and is transmitted next.

35.14.7.6 Mixed dedicated Tx buffers / Tx queue

The Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by the NDTB bit field in the TXBC register. The number of Tx queue buffers is configured by the TFQS bit field in the TXBC register. In case the TFQS bit field is programmed with a value of zero, only dedicated Tx buffers are used.

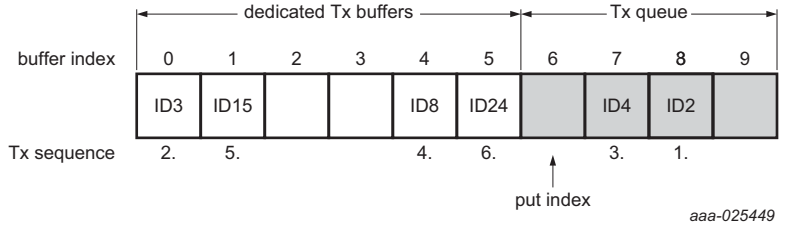


Fig 143. Example of mixed configuration dedicated Tx buffers / Tx queue

Tx prioritization:

- Scan all Tx buffers with activated transmission requests.
- Tx buffer with lowest message ID gets highest priority and is transmitted next.

35.14.7.7 Transmit cancellation

The MCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer, set the corresponding bit in the TXBCR register. Transmit cancellation is not intended for Tx FIFO operation. Successful cancellation is signaled by setting the corresponding bit in the TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding bit in the TXBRP register remains set as long as the transmission is in progress. If the transmission was successful, the corresponding bits in the XBTO and TXBCF register are set. If the transmission was not successful, it is not repeated and only the corresponding bit in the TXBCF register is set.

Remark: In case a pending transmission is cancelled immediately before this transmission could have started, there follows a short time window where no transmission has started even if another message is also pending in this node. This may enable another node to transmit a message, which may have a lower priority than the second message in this node.

35.14.7.8 Tx event handling

To support Tx event handling the MCAN has implemented a Tx event FIFO. After the MCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in Section 35.11.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling, that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF interrupt flag in the IR register, no more elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the IR register.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by the EFWM bit field in the TXEFC register, the TEFW interrupt flag is set in the IR register.

When reading from the Tx event FIFO, two times the Tx event FIFO get index value stored in the EFGI bit field in the TXEFS register has to be added to the Tx event FIFO start address stored in the EFSA bit field in the TXEFC register.

35.14.8 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge. Writing to the FIFO acknowledge index will set the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element is read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO Acknowledge Index.
- When a sequence of elements are read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: Index of the last element read), to update the get index of the FIFO.

Take special care when reading FIFO elements in an arbitrary order (that is, not using get index). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the acknowledge index of the FIFO should not be written because this would set the get index to a wrong position and also alter the fill level of the FIFO. In this case some of the older FIFO elements would be lost.

Remark: The application has to ensure that a valid value is written to the FIFO acknowledge index. The MCAN does not check for erroneous values.

36.1 How to read this chapter

The Ethernet controller is available on all LPC546xx devices.

36.2 Features

- 10/100 Mbit/s.
- Ethernet MAC IEEE 802.3-2008.
- DMA support.
- IEEE 1588-2008 v2 timestamping driven by external PHY clock.
- IEEE 1588 advanced timestamp support (IEEE 1588-2008 v2).
- Quality-of-Service for Audio-Video traffic (802.1AS-2011 and 802.1Qav-2009).
  - Software support for AVB feature is available from NXP Professional Services. See nxp.com for more details.
- Energy efficient Ethernet (IEEE 802.3az-2010) for MII PHY.
- Power management remote wake-up frame and magic packet detection.
- Support both MII and RMII (ver1.2) PHY.
- Supports MDIO (Clause 22 and Clause 45) master interface to manage and configure PHY.
- IP checksum offload support for TCP, UDP, or ICMP payloads encapsulated over IP.
- Supports both full-duplex and half-duplex operation
  - Supports CSMA/CD Protocol for half-duplex operation.
  - Supports IEEE 802.3x pause packets and priority flow control for full-duplex operation.
  - Optional forwarding of received pause control frames to the user application in full-duplex operation.
  - Backpressure support for half-duplex operation.
  - Automatic transmission of zero-quanta pause frame on de-assertion of flow control input in full-duplex operation.

36.3 Basic configuration

The Ethernet controller is configured as follows:

- Clock: In the AHBCLKCTRL2 register (Section 7.5.21), set bit ETH.
- Reset: In the PRESETCTRL2 register (Section 7.5.11), set bit ETH_RST.
- The Ethernet interrupts are connected to interrupt slots # 49, # 50, and # 51 in the NVIC. The Ethernet wake-up interrupt Ethernet wake-up packet indicator is connected to slot # 50 in the NVIC. See Chapter 6 “LPC546xx Nested Vectored Interrupt Controller (NVIC)”.

- Set the Ethernet mode to RMII or MII in the ETHPHYSEL register (see Table 194).
- Set the sideband flow control for each channel. See Section 7.5.75.

36.4 General description

The Ethernet block enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard. The Ethernet interface contains a full featured 10 Mbps or 100 Mbps Ethernet Media Access Controller (MAC) designed to provide optimized performance through the use of DMA hardware acceleration.

Features include control registers, full-duplex or half-duplex operation, flow control, control frames, hardware acceleration for transmit retry, receive packet filtering and wake-up on LAN activity, supporting both wake-up and magic packet frames. Automatic frame transmission and reception with Scatter-Gather DMA off-loads many operations from the CPU.

The Ethernet block is an AHB master connected to the AHB multilayer matrix and has access to internal SRAM and memory connected to the External Memory Controller for Ethernet data, control, and status information. A multi-layer AHB matrix connects the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters.

The Ethernet block interfaces with an off-chip Ethernet PHY using the Media Independent Interface (MII) or Reduced MII (RMII) protocol and with the on-chip Media Independent Interface Management (MIIM) serial bus.

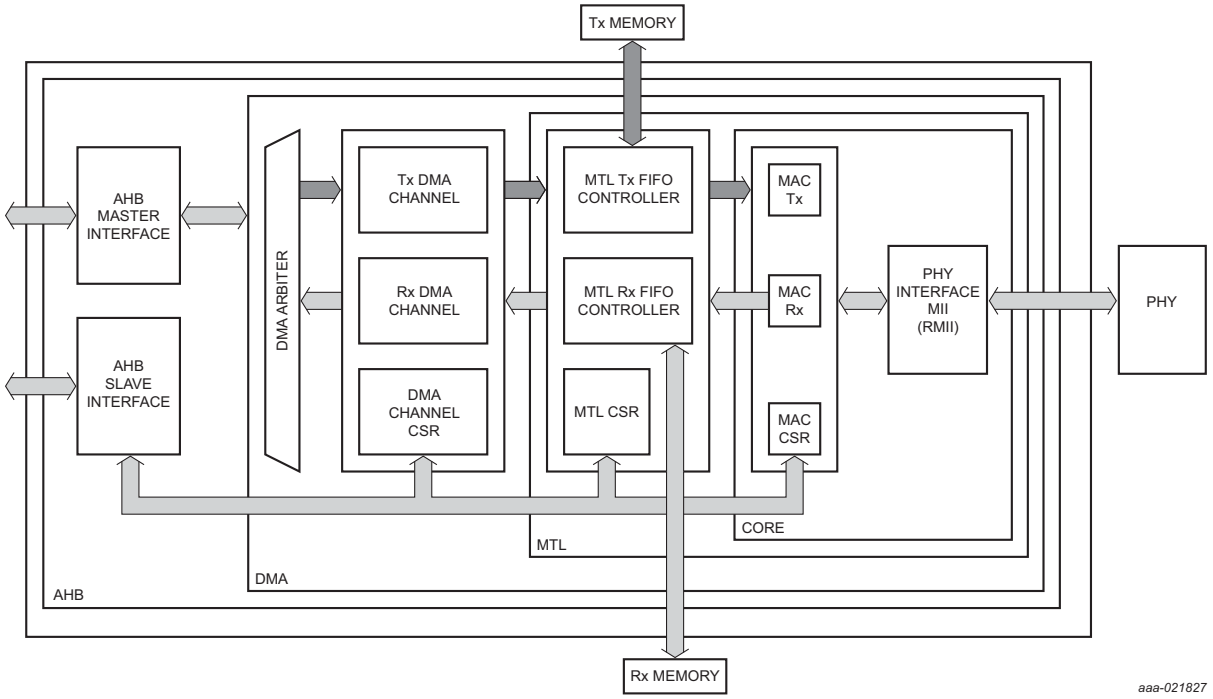


Fig 144. Ethernet block diagram

36.5 Pin description

Table 775. Ethernet pin description

Pin function	Direction	Description
MIIM interface		
ENET_MDIO	I/O	Ethernet MIIM data input and output.
ENET_MDC	O	Ethernet MIIM clock.
RMII interface		
ENET_RXD[1:0]	I	Ethernet receive data.
ENET_TXD[1:0]	O	Ethernet transmit data.
ENET_RX_DV	I	Ethernet receive data valid.
ENET_TX_EN	O	Ethernet transmit data enable.
MII interface		
ENET_RXD[3:0]	I	Ethernet receive data.
ENET_TXD[3:0]	O	Ethernet transmit data.
ENET_COL	I	Ethernet collision detect.
ENET_CRS	I	Ethernet carrier sense.
ENET_TX_ER	O	Ethernet transmit error.
ENET_TX_CLK	I	Ethernet transmit clock.
ENET_RX_CLK	I	Ethernet receive clock.
ENET_RX_ER	I	Ethernet receive error.
ENET_TX_EN	O	Ethernet transmit enable.

36.6 Register description

Table 776. Register overview: Ethernet MAC and DMA (base address 0x4009 2000)

Name	Access	Offset	Description	Reset value	Section
MAC_CONFIG	R/W	0x0000	MAC configuration.	0x8000	36.6.1
MAC_EXT_CONFIG	R/W	0x0004	MAC extended configuration.	0x8000	36.6.2
MAC_FRAME_FILTER	R/W	0x0008	MAC frame filter.	0x0	36.6.3
MAC_WD_TIMEROUT	R/W	0x000C	MAC watchdog timeout.	0x0	36.6.4
MAC_VLAN_TAG	R/W	0x0050	VLAN tag.	0x0	36.6.5
MAC_TX_FLOW_CTRL_Q0	R/W	0x0070	Transmit flow control.	0x0	36.6.6
MAC_TX_FLOW_CTRL_Q1	R/W	0x0074	Transmit flow control.	0x0	36.6.6
MAC_RX_FLOW_CTRL	R/W	0x0090	Receive flow control.	0x0	36.6.7
MAC_TXQ_PRIO_MAP	R/W	0x0098	Transmit Queue priority mapping 0.	0x0	36.6.8
MAC_RXQ_CTRL0	R/W	0x00A0	Receive Queue control 0.	0x0	36.6.9
MAC_RXQ_CTRL1	R/W	0x00A4	Receive Queue control 1.	0x0	36.6.10
MAC_RXQ_CTRL2	R/W	0x00A8	Receive Queue control 2.	0x0	36.6.11
MAC_INTR_STAT	RO	0x00B0	Interrupt status.	0x0	36.6.12
MAC_INTR_EN	R/W	0x00B4	Interrupt enable.	0x0	36.6.13
MAC_RXTX_STAT	RO	0x00B8	Receive transmit status.	0x0	36.6.14
MAC_PMT_CRTL_STAT	R/W	0x00C0	MAC PMT control and status.	0x0	36.6.15
MAC_RWK_PKT_FLT	R/W	0x00C4	Wake-up packet filter.	0x0	36.6.16
MAC_LPI_CTRL_STAT	R/W	0x00D0	LPI control and status.	0x0	36.6.17
MAC_LPI_TIMER_CTRL	R/W	0x00D4	LPI timers control.	0x0	36.6.18
MAC_LPI_ENTR_TIMR	R/W	0x00D8	LPI entry timer.	0x0	36.6.19
MAC_1US_TIC_COUNTR	R/W	0x00DC	MAC 1 μs tick counter.	0x0	36.6.20
MAC_VERSION	RO	0x0110	MAC version.	0x0	36.6.21
MAC_DBG	RO	0x0114	MAC debug	0x0	36.6.22
MAC_HW_FEAT0	RO	0x011C	MAC hardware feature.	0x0201 70E5	36.6.23
MAC_HW_FEAT1	RO	0x0120	MAC hardware feature.	0x0090 0104	36.6.24
MAC_HW_FEAT2	RO	0x0124	MAC hardware feature.	0x0041 1041	36.6.25
MAC_MDIO_ADDR	R/W	0x0200	MIDO address.	0x0	36.6.26
MAC_MDIO_DATA	R/W	0x0204	MDIO data.	0x0	36.6.27
MAC_ADDR_HIGH	R/W	0x0300	MAC address0 high.	0x8000 FFFF	36.6.28
MAC_ADDR_LOW	R/W	0x0304	MAC address0 low.	0xFFFF FFFF	36.6.29
MAC_TIMESTAMP_CTRL	R/W	0x0B00	Timestamp control.	0x2000	36.6.30
MAC_SUB_SCND_INCR	R/W	0x0B04	Sub-second increment.	0x0	36.6.31
MAC_SYS_TIME_SCND	RO	0x0B08	System time seconds.	0x0	36.6.32
MAC_SYS_TIME_NSCND	RO	0x0B0C	System time nanoseconds.	0x0	36.6.33
MAC_SYS_TIME_SCND_UPD	R/W	0x0B10	System time seconds update.	0x0	36.6.34
MAC_SYS_TIME_NSCND_UPD	R/W	0x0B14	System time nanoseconds update.	0x0	36.6.35
MAC_SYS_TIMESTMP_ADDEND	R/W	0x0B18	Timestamp addend.	0x0	36.6.36
MAC_SYS_TIME_HWORD_SCND	R/W	0x0B1C	System time-higher word seconds.	0x0	36.6.37
MAC_SYS_TIMESTMP_STAT	RO	0x0B20	Timestamp status.	0x0	36.6.38

Table 776. Register overview: Ethernet MAC and DMA (base address 0x4009 2000) *…continued*

Name	Access	Offset	Description	Reset value	Section
MAC_Tx_TIMESTAMP_STATUS_NANOSECONDS	RO	0x0B30	Tx timestamp status nanoseconds.	0x0	36.6.39
MAC_Tx_TIMESTAMP_STATUS_SECONDS	RO	0x0B34	Tx timestamp status seconds.	0x0	36.6.40
MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND	R/W	0x0B58	Timestamp ingress correction.	0x0	36.6.41
MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND	R/W	0x0B5C	Timestamp egress correction.	0x0	36.6.42
MTL_OP_MODE	R/W	0x0C00	MTL operation mode.	0x0	36.6.43
MTL_INTR_STAT	RO	0x0C20	MTL interrupt status.	0x0	36.6.44
MTL_RXQ_DMA_MAP	R/W	0x0C30	MTL Rx Queue and DMA channel mapping.	0x0	36.6.45
MTL_TXQ0_OP_MODE	R/W	0x0D00	MTL TxQ0 operation mode.	0x0	36.6.46
MTL_TXQ0_UNDRFLW	RO	0x0D04	MTL TxQ0 underflow.	0x0	36.6.47
MTL_TXQ0_DBG	RO	0x0D08	MTL TxQ0 debug.	0x0	36.6.48
MTL_TXQ0_ETS_STAT	RO	0x0D14	MTL TxQ0 ETS status.	0x0	36.6.50
MTL_TXQ0_QNTM_WGHT	R/W	0x0D18	Queue 0 quantum or weights.	0x0	36.6.51
MTL_TXQ0_INTCTRL_STAT	R/W	0x0D2C	MTL TxQ0 interrupt control status.	0x0	36.6.56
MTL_RXQ0_OP_MODE	R/W	0x0D30	MTL RxQ0 operation mode.	0x0	36.6.57
MTL_RXQ0_MISSPKT_OVRFLW_CNT	R/W	0x0D34	MTL RxQ0 missed packet overflow counter.	0x0	36.6.58
MTL_RXQ0_DBG	R/W	0x0D38	MTL RxQ0 debug.	0x0	36.6.59
MTL_RXQ0_CTRL	R/W	0x0D3C	MTL RxQ0 control.	0x0	36.6.60
MTL_TXQ1_OP_MODE	R/W	0x0D40	MTL TxQ1 operation mode.	0x0	36.6.46
MTL_TXQ1_UNDRFLW	RO	0x0D44	MTL TxQ1 underflow.	0x0	36.6.47
MTL_TXQ1_DBG	RO	0x0D48	MTL TxQ1 debug.	0x0	36.6.48
MTL_TXQ1_ETS_CTRL	R/W	0x0D50	MTL TxQ1 ETS control.	0x0	36.6.49
MTL_TXQ1_ETS_STAT	R/W	0x0D54	MTL TxQ1 ETS status.	0x0	36.6.50
MTL_TXQ1_QNTM_WGHT	R/W	0x0D58	MTL TxQ1 quantum Weight.	0x0	36.6.52
MTL_TXQ1_SNDSLP_CRDT	R/W	0x0D5C	MTL TxQ1 SendSlopCredit.	0x0	36.6.53
MTL_TXQ1_HI_CRDT	R/W	0x0D60	MTL TxQ1 hiCredit.	0x0	36.6.54
MTL_TXQ1_LO_CRDT	R/W	0x0D64	MTL TxQ1 loCredit.	0x0	36.6.55
MTL_TXQ1_INTCTRL_STAT	R/W	0x0D6C	MTL TxQ1 interrupt control status.	0x0	36.6.56
MTL_RXQ1_OP_MODE	R/W	0x0D70	MTL RxQ1 operation mode.	0x0	36.6.57
MTL_RXQ1_MISSPKT_OVRFLW_CNT	R/W	0x0D74	MTL RxQ1 missed packet overflow counter.	0x0	36.6.58
MTL_RXQ1_DBG	R/W	0x0D78	MTL RxQ1 debug.	0x0	36.6.59
MTL_RXQ1_CTRL	R/W	0x0D7C	MTL RxQ1 control.	0x0	36.6.60
DMA_MODE	R/W	0x1000	DMA mode.	0x0	36.6.61
DMA_SYSBUS_MODE	R/W	0x1004	DMA system bus mode.	0x0	36.6.62
DMA_INTR_STAT	R/W	0x1008	DMA interrupt status.	0x0	36.6.63
DMA_DBG_STAT	R/W	0x100C	DMA debug status.	0x0	36.6.64
DMA_CH0_CTRL	R/W	0x1100	DMA channel 0 control.	0x0	36.6.65

Table 776. Register overview: Ethernet MAC and DMA (base address 0x4009 2000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DMA_CH0_TX_CTRL	R/W	0x1104	DMA channel 0 transmit control.	0x0	36.6.66
DMA_CH0_RX_CTRL	R/W	0x1108	DMA channel 0 receive control.	0x0	36.6.67
DMA_CH0_TXDESC_LIST_ADDR	R/W	0x1114	Channel 0 Tx descriptor list address.	0x0	36.6.68
DMA_CH0_RXDESC_LIST_ADDR	R/W	0x111C	Channel 0 Rx descriptor list address.	0x0	36.6.69
DMA_CH0_TXDESC_TAIL_PTR	R/W	0x1120	Channel 0 Tx descriptor tail pointer.	0x0	36.6.70
DMA_CH0_RXDESC_TAIL_PTR	R/W	0x1128	Channel 0 Rx descriptor tail pointer.	0x0	36.6.71
DMA_CH0_TXDESC_RING_LENGTH	R/W	0x112C	Channel 0 Tx descriptor ring length.	0x0	36.6.72
DMA_CH0_RXDESC_RING_LENGTH	R/W	0x1130	Channel 0 Rx descriptor ring length.	0x0	36.6.73
DMA_CH0_INT_EN	R/W	0x1134	Channel 0 interrupt enable.	0x0	36.6.74
DMA_CH0_RX_INT_WDTIMER	R/W	0x1138	Receive interrupt watchdog timer.	0x0	36.6.75
DMA_CH0_SLOT_FUNC_CTRL_STAT	R/W	0x113C	Slot function control and status.	0x0	36.6.76
DMA_CH0_CUR_HST_TXDESC	R/W	0x1144	Channel 0 current host transmit descriptor.	0x0	36.6.77
DMA_CH0_CUR_HST_RXDESC	R/W	0x114C	Channel 0 current host receive descriptor.	0x0	36.6.78
DMA_CH0_CUR_HST_TXBUF	R/W	0x1154	Channel 0 current host transmit buffer address.	0x0	36.6.79
DMA_CH0_CUR_HST_RXBUF	R/W	0x115C	Channel 0 current application receive buffer address.	0x0	36.6.80
DMA_CH0_STAT	R/W	0x1160	Channel 0 DMA status.	0x0	36.6.81
DMA_CH0_MISS_FRAME_CNT	RO	0x116C	Channel 0 missed frame count.	0x0	36.6.82
DMA_CH1_CTRL	R/W	0x1180	DMA channel 1 control.	0x0	36.6.65
DMA_CH1_TX_CTRL	R/W	0x1184	DMA channel 1 transmit control.	0x0	36.6.66
DMA_CH1_RX_CTRL	R/W	0x1188	The DMA channel 1 receive control.	0x0	36.6.67
DMA_CH1_TXDESC_LIST_ADDR	R/W	0x1194	The channel 1 Tx descriptor list address.	0x0	36.6.68
DMA_CH1_RXDESC_LIST_ADDR	R/W	0x119C	The channel 1 Rx descriptor list address.	0x0	36.6.69
DMA_CH1_TXDESC_TAIL_PTR	R/W	0x11A0	The channel 1 Tx descriptor tail pointer.	0x0	36.6.70
DMA_CH1_RXDESC_TAIL_PTR	R/W	0x11A8	The channel 1 Rx descriptor tail pointer.	0x0	36.6.71
DMA_CH1_TXDESC_RING_LENGTH	R/W	0x11AC	Channel 1 Tx descriptor ring length.	0x0	36.6.72
DMA_CH1_RXDESC_RING_LENGTH	R/W	0x11B0	The channel 1 Rx descriptor ring length.	0x0	36.6.73
DMA_CH1_INT_EN	R/W	0x11B4	The channel 1 interrupt enable.	0x0	36.6.74
DMA_CH1_RX_INT_WDTIMER	R/W	0x11B8	The channel 1 receive interrupt watchdog timer.	0x0	36.6.75
DMA_CH1_SLOT_FUNC_CTRL_STAT	R/W	0x11BC	The channel 1 slot function control and status.	0x0	36.6.76
DMA_CH1_CUR_HST_TXDESC	R/W	0x11C4	The channel 1 current host transmit descriptor.	0x0	36.6.77
DMA_CH1_CUR_HST_RXDESC	R/W	0x11CC	The channel 1 current host receive descriptor.	0x0	36.6.78

Table 776. Register overview: Ethernet MAC and DMA (base address 0x4009 2000) *…continued*

Name	Access	Offset	Description	Reset value	Section
DMA_CH1_CUR_HST_TXBUF	R/W	0x11D4	The channel 1 current host transmit buffer address.	0x0	36.6.79
DMA_CH1_CUR_HST_RXBUF	R/W	0x11DC	The channel 1 current host receive buffer address.	0x0	36.6.80
DMA_CH1_STAT	R/W	0x11E0	Channel 1 DMA status.	0x0	36.6.81
DMA_CH1_MISS_FRAME_CNT	RO	0x11EC	Channel 1 missed frame count.	0x0	36.6.82

36.6.1 MAC configuration register

The MAC configuration register establishes receive and transmit operating modes of the MAC.

Table 777. MAC configuration register (MAC_CONFIG, offset 0x0000) bit description

Bit	Symbol	Value	Description	Reset value
0	RE		Receiver enable. When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the MII. When this bit is reset, the MAC receive state machine is disabled after the completion of the reception of the current frame, and will not receive any further frames from the MII.	0
		0	Disabled	
		1	Enabled	
1	TE		Transmitter enable. When this bit is set, the transmit state machine of the MAC is enabled for transmission on the MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and will not transmit any further frames.	0
		0	Disabled	
		1	Enabled	
3:2	PRELEN		Preamble length for transmit packets. These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.	0
		0x00	7 bytes of preamble	
		0x01	5 bytes of preamble	
		0x02	3 bytes of preamble	
		0x03	Reserved	
4	DC		Deferral check. When this bit is set, the deferral check function is enabled in the MAC. The MAC will issue a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status when the transmit state machine is deferred for more than 24,288 bit times in 10/100-Mbps mode. If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active CRS (carrier sense) signal on the MII. Defer time is not cumulative. If the transmitter defers for 10,000 bit times, then transmits, collides, backs off, and then has to defer again after completion of back-off, the deferral timer resets to 0 and restarts. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in half-duplex mode and is reserved (RO) in full-duplex-only configuration.	0
		0	Disable	
		1	Enable	

Table 777. MAC configuration register (MAC_CONFIG, offset 0x0000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
6:5	BL		Back-off limit. The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) the MAC waits before rescheduling a transmission attempt during retries after a collision. These bits are applicable only in half-duplex mode and is reserved (RO) in full-duplex-only configuration.	0
		0x00	k = min (n, 10)	
		0x01	k = min (n, 8)	
		0x02	k = min (n, 4)	
		0x03	k = min (n, 1) where n = retransmission attempt. The random integer r takes the value in the range $0 \leq r \leq 2^k$	
7	-	-	Reserved.	-
8	DR		Disable retry. When this bit is set, the MAC will attempt only one transmission. When a collision occurs on the MII, the MAC will ignore the current frame transmission and report a frame abort with excessive collision error in the transmit frame status. When this bit is reset, the MAC will attempt retries based on the settings of BL. This bit is applicable only to half-duplex mode and is reserved (RO with default value) in full-duplex-only configuration.	0
		0	Enable retry	
		1	Disable retry	
9	DCRS		Disable carrier sense during transmission. When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of loss of carrier or no carrier during transmission. When this bit is reset, the MAC transmitter generates errors because of carrier sense. The MAC can even abort the transmission. This bit is reserved and read-only (RO) in the full-duplex-only configurations.	0
		0	Enable	
		1	Disable	
10	DO		Disable receive own. When this bit is set, the MAC disables the reception of frames when the gmii_txen_o is asserted in half-duplex mode. When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in full-duplex mode.	0
		0	Enable	
		1	Disable	
11	ECRSFD		Enable carrier sense full-duplex mode before transmission. When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When this bit is reset, the MAC transmitter ignores the status of the CRS signal.	0
		0	Disable	
		1	Enable	

Table 777. MAC configuration register (MAC_CONFIG, offset 0x0000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
12	LM		Loopback mode. When this bit is set, the MAC operates in loopback mode at MII. The MII receive clock input is required for the loopback to work properly, as the transmit clock is not looped-back internally.	0
		0	Loopback disable	
		1	Loopback enable	
13	DM		Duplex mode. When this bit is set, the MAC operates in a full-duplex mode where it can transmit and receive simultaneously.	0
		0	Disable	
		1	Enable	
14	FES		Speed. Indicates the speed in Fast Ethernet (MII) mode: This bit is reserved (RO) by default and is enabled only when RMII/SMII is enabled during configuration. This bit generates link speed encoding when TC (bit 24) is set in SMII mode. This signal can optionally be driven as an output signal (mac_speed_o[0]) if the core is configured with an RMII, SMII, SGMII, or RGMII PHY interface selected. This bit is reserved when RMII is enabled during configuration without enabling the "output port for speed selection" option.	0
		0	10 Mbps	
		1	100 Mbps.	
15	PS		Port select.	1
		0	Reserved	
		1	MII (10/100 Mbps operation).	
16	JE		Jumbo frame enable. When this bit is set, MAC allows jumbo frames of 9,018 bytes (9,022 bytes for tagged frames) without reporting a giant frame error in the receive frame status.	0
		0	Disable	
		1	Enable	
17	JD		Jabber disable. When this bit is set, the MAC disables the jabber timer on the transmitter, and can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.	0
		0	Enable	
		1	Disable	
18	BE		Packet burst enable. When this bit is set, the MAC allows packet bursting during transmission in the MII half-duplex mode. This bit is not applicable if the MAC is operating in full-duplex mode.	0
		0	Disable	
		1	Enable	

Table 777. MAC configuration register (MAC_CONFIG, offset 0x0000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
19	WD		Watchdog disable. When this bit is set, the MAC disables the watchdog timer on the receiver, and can receive frames of up to 16,384 bytes. When this bit is reset, the MAC allows no more than 2,048 bytes (10,240 if JE is set high) of the frame being received and cuts off any bytes received after that.	0
		0	Enable	
		1	Disable	
20	ACS		Automatic pad or CRC stripping. When this bit is set, the MAC strips the pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the pad or FCS field. When this bit is reset, the MAC passes all incoming packets to the application, without any modification.	0
		0	Disable	
		1	Enable	
21	CST		CRC stripping for type packets. When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP checksum engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 checksum offload engine is enabled. Note: For information about how the settings of the ACS bit and the CST bit impact the packet length, see packet length based on the and ACS and CST bits.	0
		0	Disable	
		1	Enable	
22	S2KP		IEEE 802.3as support for 2K packets. When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as giant packets. When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the giant packet status, see giant packet status based on S2KP and JE bits. Note: When the JE bit is set, setting this bit has no effect on the giant packet status.	0
		0	Disable	
		1	Enable	

Table 777. MAC configuration register (MAC_CONFIG, offset 0x0000) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
23	GPSLCE		Giant packet size limit control enable. When this bit is set, the MAC considers the value in GPSL field in MAC extended configuration register to declare a received packet as giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When this bit is reset, the MAC considers a received packet as giant packet when its size is greater than 1,518 bytes (1,522 bytes for tagged packet). The watchdog timeout limit, jumbo packet enable and 2K packet enable have higher precedence over this bit, that is, the MAC considers a received packet as giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with jumbo packet enabled and greater than 2,000 bytes with 2K packet enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.	0
		0	Disable	
		1	Enable	
26:24	IPG		Inter-packet gap. These bits control the minimum IPG between packets during transmission. This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG. The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in MAC extended configuration register Table 778 is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given for the EIPG field in the extended configuration register Table 778.	0
		0x00	96 bit times	
		0x01	88 bit times	
		0x02	80 bit times	
		0x03	72 bit times	
		0x04	64 bit times	
		0x05	56 bit times	
		0x06	48 bit times	
		0x07	40 bit times	
27	IPC		Checksum offload. When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.	0
		0	Disable	
		1	Enable	
31:28	-	-	Reserved.	-

36.6.2 MAC extended configuration register

The MAC extended configuration register establishes the operating mode of the MAC.

Table 778. MAC extended configuration register (MAC_EXT_CONFIG, offset 0x0004) bit description

Bit	Symbol	Value	Description	Reset value
13:0	GPSL		Giant packet size limit. If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programmed value.	0
		0	Disabled	
		1	Enabled	
15:14	-	-	Reserved.	-
16	DCRCC		Disable CRC checking for received packets. When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets.	0
		0	Disable	
		1	Enable	
17	SPEN		Slow protocol detection enable. When this bit is set, MAC processes the slow protocol packets (Ether type 0x8809) and provides the Rx status. The MAC discards the slow protocol packets with invalid sub-types. When this bit is reset, the MAC forwards all error-free slow protocol packets to the application. The MAC considers such packets as normal type packets.	0
		0	Disable	
		1	Enable	
18	USP		Unicast slow protocol packet detect. When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC address high Table 804 and MAC address low Table 805 registers. The MAC also detects the slow protocol packets with the slow protocols multicast address (01-80-C2-00-00-02). When this bit is reset, the MAC detects only slow protocol packets with the slow protocol multicast address specified in the IEEE 802.3-2008, Section 5.	0
		0	Disable	
		1	Enable	
31:19	-	-	Reserved.	-

36.6.3 MAC frame filter register

The MAC frame filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as pass bad frames and pass control frames.

Table 779. MAC frame filter register (MAC_FRAME_FILTER, offset 0x0008) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	PR		Promiscuous mode. When this bit is set, the address filter module passes all incoming frames regardless of its destination or source address. The SA/DA filter fails status bits of the receive status word will always be cleared when PR is set.	0	R/W
		0	Disable		
		1	Enable		
2:1	-	-	Reserved.	-	-
3	DAIF		DA inverse filtering. When this bit is set, the address check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed.	0	R/W
		0	Disable		
		1	Enable		
4	PM		Pass all multicast. When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed.	0	R/W
		0	Disable		
		1	Enable		
5	DBF		Disable broadcast frames. When this bit is set, the AFM module filters all incoming broadcast frames. When this bit is reset, the AFM module passes all received broadcast frames.	0	R/W
		0	Disable		
		1	Enable		
7:6	PCF		Pass control frames. These bits control the forwarding of all control frames (including unicast and multicast pause frames). Note that the processing of pause control frames depends only on RFE of the flow control register.	00	R/W
		0x00	MAC filters all control frames from reaching the application.		
		0x01	MAC forwards all control frames except pause control frames to application even if they fail the address filter.		
		0x02	MAC forwards all control frames to application even if they fail the address filter.		
		0x03	MAC forwards control frames that pass the address filter.		
8	SAIF		SA inverse filtering. When this bit is set, the address check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA address filter. When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA address filter.	0	RO
		0	Disable		
		1	Enable		

Table 779. MAC frame filter register (MAC_FRAME_FILTER, offset 0x0008) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
9	SAF		Source address filter enable. When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet. When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx status depending on the SA address comparison. Note: According to the IEEE specification, bit 47 of the SA is reserved. However, in this device, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.	0	RO
30:10	-	-	Reserved.	-	-
31	RA		Receive all. When this bit is set, the MAC receiver module passes to the application all frames received irrespective of whether they pass the address filter. The result of the SA/DA filtering is updated (pass or fail) in the corresponding bits in the receive status word. When this bit is reset, the receiver module passes to the application only those frames that pass the SA/DA address filter.	0	R/W
		0	Disable		
		1	Enable		

36.6.4 MAC watchdog timeout register

The watchdog timeout register controls the watchdog timeout for received packets.

Table 780. MAC watchdog timeout register (MAC_WD_TIMEROUT, offset 0x000C) bit description

Bit	Symbol	Value	Description	Reset value
3:0	WTO		Watchdog timeout. When the PWE bit is set and the WD bit of the MAC configuration register Table 777 is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. **Note:** When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.	0
		0x0	2 KB	
		0x1	3 KB	
		0x3	4 KB	
		0x4	5 KB	
		0x5	6 KB	
		0x6	7 KB	
		0x7	8 KB	
		0x8	9 KB	
		0x9	10 KB	
		0xA	11 KB	
		0xB	12 KB	
		0xC	13 KB	
		0xD	14 KB	
		0xE	16,383 bytes	
		0xF	Reserved	
7:4	-	-	Reserved.	-
8	PWE		Programmable watchdog enable. When this bit is set and the WD bit of the MAC configuration register Table 777 is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in MAC configuration register. SeeTable 777.	0
		0	Disable	
		1	Enable	
31:9	-	-	Reserved.	-

36.6.5 MAC VLAN tag register

The VLAN tag register identifies the IEEE 802.1Q VLAN type packets.

Table 781. MAC VLAN tag register (MAC_VLAN_TAG, offset 0x0050) bit description

Bit	Symbol	Description	Reset value
15:0	VL	VLAN tag identifier for receive packets. This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field: Bits 15:13: User priority. Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI). Bits 11:0: VLAN Identifier (VID) field of VLAN tag. If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with type field value of 0x8100 or 0x88a8 as VLAN packets.	0
16	ETV	Enable 12-bit VLAN tag comparison. When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits 11:0 of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN hash filtering.	0
17	VTIM	VLAN tag inverse match enable. When this bit is set, this bit enables the VLAN tag inverse matching. The packets without matching VLAN tag are marked as matched. When reset, this bit enables the VLAN tag perfect matching. The packets with matched VLAN tag are marked as matched.	0
18	ESVL	Enable S-VLAN. When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.	0
19	ERSVLM	Enable receive S-VLAN match. When this bit is set, the MAC receiver enables filtering or matching for SVLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets. The ERIVLT bit determines the VLAN tag position considered for filtering or matching.	0
20	DOVLTC	Disable VLAN type check. When this bit is set, the MAC does not check whether the VLAN tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN. When this bit is reset, the MAC filters or matches the VLAN tag specified by the ERIVLT bit only when VLAN tag type is similar to the one specified by the ERSVLM bit.	0
22:21	EVLS	Enable VLAN tag stripping on receive. This field indicates the stripping operation on the outer VLAN tag in received packet: 00: Do not strip 01: Strip if VLAN filter passes 10: Strip if VLAN filter fails 11: Always strip	0
23	-	Reserved.	-
24	EVLRXS	Enable VLAN tag in Rx status. When this bit is set, MAC provides the outer VLAN tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN tag in Rx status.	0

Table 781. MAC VLAN tag register (MAC_VLAN_TAG, offset 0x0050) bit description *...continued*

Bit	Symbol	Description	Reset value
25	VTHM	VLAN tag hash table match enable. When this bit is set, the most significant four bits of CRC of VLAN tag are used to index the content of the MAC_VLAN_Hash_Table register. A value of 1 in the VLAN hash table register, corresponding to the index, indicates that the packet matched the VLAN hash table. When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison. When this bit is reset, the VLAN hash match operation is not performed. If the VLAN hash feature is not enabled, this bit is reserved (RO with default value).	0
26	EDVLP	Enable double VLAN processing. When this bit is set, the MAC enables processing of up to two VLAN tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN tag on Tx and Rx (if present).	0
27	ERIVLT	Enable inner VLAN tag. When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.	0
29:28	EIVLS	Enable Inner VLAN tag stripping on receive. This field indicates the stripping operation on inner VLAN tag in received packet: 00: Do not strip 01: Strip if VLAN filter passes 10: Strip if VLAN filter fails 11: Always strip	0
30	-	Reserved.	-
31	EIVLRXS	Enable Inner VLAN tag in Rx status. When this bit is set, the MAC provides the inner VLAN tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN tag in Rx status.	0

36.6.6 MAC transmit flow control registers

There are two flow control registers, one for each queue.

The flow control register controls the generation and reception of the control (pause command) frames by the MAC's flow control module. A write to a register with the busy bit set to 1 triggers the flow control block to generate a pause control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the pause time value from this register is used in the pause time field of the control frame. The busy bit remains set until the control frame is transferred onto the cable. The host must make sure that the busy bit is cleared before writing to the register.

Table 782. MAC transmit flow control register (MAC_TX_FLOW_CTRL_Q0, offset 0x0070 and MAC_TX_FLOW_CTRL_Q1, offset 0x0074) bit description

Bit	Symbol	Value	Description	Reset value
0	FCB		Flow control busy/backpressure activate. This register field can be read by the application (read), can be set to 1 by the application with a register write of 1 (write set), and is cleared to 0 by the core (self clear). The application cannot clear this type of field, and a register write of 0 to this bit has no effect on this field. This bit initiates a pause control frame in full-duplex mode. In full-duplex mode, this bit should be read as 0 before writing to the flow control register. To initiate a pause control frame, the application must set this bit to 1. During a transfer of the control frame, this bit will continue to be set to signify that a frame transmission is in progress. After the completion of pause control frame transmission, the MAC will reset this bit to 0. The flow control register should not be written to until this bit is cleared. In half-duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC Core. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the flow controller input signal for the backpressure function. When the MAC is configured to full- duplex mode, the BPA is automatically disabled.	0
		0	Disable	
		1	Enable	
1	TFE		Transmit flow control enable. In full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC will not transmit any pause frames. In half-duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the backpressure feature is disabled.	0
		0	Disable	
		1	Enable	
3:2	-	-	Reserved.	-
6:4	PLT		Pause low threshold. This field configures the threshold of the pause timer at which the input flow control signal is checked for automatic retransmission of pause frame. The threshold values should be always less than the pause time configured in bits 31:16. For example, if PT = 0x100 (256 slot-times), and PLT = 01, then a second pause frame is automatically transmitted if the flow control signal is asserted at 228 (256 – 28) slot-times after the first pause frame is transmitted.	00
		0x00	Pause time minus 4 slot times (PT - 4 slot times)	
		0x01	Pause time minus 4 slot times (PT - 28 slot times)	
		0x02	Pause time minus 4 slot times (PT - 36 slot times)	
		0x03	Pause time minus 4 slot times (PT - 144 slot times)	
		0x04	Pause time minus 4 slot times (PT - 256 slot times)	
		0x05	Pause time minus 4 slot times (PT - 512 slot times)	
		0x06	Reserved	
		0x07	Reserved	

Table 782. MAC transmit flow control register (MAC_TX_FLOW_CTRL_Q0, offset 0x0070 and MAC_TX_FLOW_CTRL_Q1, offset 0x0074) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	DZPQ		Disable zero-quanta pause. When set, this bit disables the automatic generation of zero-quanta pause control frames on the deassertion of the flow-control signal from the FIFO layer. When this bit is reset, normal operation with automatic zero-quanta pause control frame generation is enabled.	0
15:8	-	-	Reserved.	-
31:16	PT		Pause time. This field holds the value to be used in the pause time field in the transmit control frame. If the pause time bits is configured to be double-synchronized to the MII clock domain, then consecutive writes to this register should be performed only after at least 4 clock cycles in the destination clock domain.	0x0000

36.6.7 MAC receive flow control register

The receive flow control register controls the pausing of MAC transmit based on the received pause packet.

Table 783. MAC Receive flow control register (MAC_RX_FLOW_CTRL, offset 0x0090) bit description

Bit	Symbol	Value	Description	Reset value
0	RFE		Receive flow control enable. When this bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received pause packet and disables its transmitter for a specified (pause) time. When this bit is reset or the MAC is operating in half-duplex mode, the decode function of the pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Tx Queue, with matching priorities, for a duration of received pause time.	0
		0	Disable	
		1	Enable	
1	UP		Unicast pause packet detect. A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect pause packets with unicast address of the station. This unicast address should be as specified in MAC address high register Table 804 and MAC address low register Table 805. When this bit is reset, the MAC only detects pause packets with unique multicast address. Note: The MAC does not process a pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.	0
		0	Disable	
		1	Enable	
31:2	-	-	Reserved.	-

36.6.8 MAC Tx Queue priority mapping0 register

The Tx Queue priority mapping 0 register contains the priority values assigned to Tx Queue 0 and Tx Queue 1.

Table 784. MAC Tx Queue priority mapping register (MAC_TXQ_PRIO_MAP, offset 0x0098) bit description

Bit	Symbol	Description	Reset value
7:0	PSTQ0	Priorities selected in Tx Queue 0. This field holds the priorities assigned to Tx Queue 0 by the software. This field determines if Tx Queue 0 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programmed in this field. If the content of this field is not mutually exclusive to corresponding fields of other Tx Queues, that is, same priority is mapped to multiple Tx Queues, the MAC blocks all queues with matching priority for specified time. This field is reserved and RO when the enable data center bridging option is selected without Tx Queue 0 while configuring the core.	0
15:8	PSTQ1	Priorities selected in Tx Queue 1. This bit is similar to the PSTQ0 bit.	0
31:16	-	Reserved.	-

36.6.9 MAC Rx Queue control 0 register

The Rx Queue control 0 register controls the queue management in the MAC receiver. This register is present only when you select multiple queues in the receive path.

Table 785. MAC Rx Queue control 0 register (MAC_RXQ_CTRL0, offset 0x00A0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	RXQ0EN		Rx Queue 0 enable. This field indicates whether Rx Queue 0 is enabled for AV.	0
		0x00	Disable	
		0x01	Queue 0 enabled for AV	
		Others	Reserved	
3:2	RXQ1EN		Rx Queue 1 enable. This field indicates whether Rx Queue 1 is enabled for AV.	0
		0	Disable	
		0x01	Queue 1 enabled for AV	
		Others	Reserved	
31:4	-	-	Reserved.	-

36.6.10 MAC Rx Queue control 1 register

The Rx Queue control 1 register controls the routing of multicast, broadcast and AV, and untagged packets to the Rx Queues. This register is present only when you select multiple queues in the receive path.

Table 786. MAC Rx Queue control 1 register (MAC_RXQ_CTRL1, offset 0x00A4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	AVCPQ		AV untagged control packets queue. This field specifies the Rx Queue on which the received AV untagged control packets are routed:	0
		0x00	Rx Queue 0	
		0x01	Rx Queue 1. All others reserved.	
3	-	-	Reserved.	-

Table 786. MAC Rx Queue control 1 register (MAC_RXQ_CTRL1, offset 0x00A4) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6:4	AVPTPQ		AV PTP packets queue. This field specifies the Rx Queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. When the AV8021ASMEN bit of MAC timestamp control register Table 806 is set, only untagged PTP over Ethernet packets are routed on an Rx Queue.	0
		0x00	Rx Queue 0.	
		0x01	Rx Queue 1	
		Others	Reserved	
11:7	-	-	Reserved.	-
14:12	UPQ		Untagged packet queue. This field indicates the Rx Queue to which untagged packets are to be routed. Any Rx Queue enabled for generic/AV traffic can be used to route the untagged packets.	0
		0x00	Rx Queue 0	
		0x01	Rx Queue 1	
		Others	Reserved	
15	-	-	Reserved.	-
18:16	MCBCQ		Multicast and broadcast queue. This field specifies the Rx Queue onto which multicast or broadcast packets are routed. Any Rx Queue enabled for generic/AV traffic can be used to route the multicast or broadcast packets.	0
		0x00	Rx Queue 0	
		0x01	Rx Queue 1	
		Others	Reserved	
19	-	-	Reserved.	-
20	MCBCQEN		Multicast and broadcast queue enable. This bit specifies that multicast or broadcast packets routing to the Rx Queue is enabled and the multicast or broadcast packets must be routed to Rx Queue specified in MCBCQ field.	
		0	Disable	
		1	Enable	
31:21	-	-	Reserved.	-

36.6.11 MAC Rx Queue control 2 register

The Rx Queue control 2 register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx Queues 0 to 3. This register is present when multiple Rx Queues are selected while configuring the core.

Table 787. MAC Rx Queue control2 register (MAC_RXQ_CTRL2, offset 0x00A8) bit description

Bit	Symbol	Description	Reset value
7:0	PSRQ0	Priorities selected in the Rx Queue 0. This field decides the priorities assigned to Rx Queue 0. All packets with priorities that match the values set in this field are routed to Rx Queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx Queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx Queues.	0
15:8	PSRQ1	Priorities selected in the Rx Queue 1. This field decides the priorities assigned to Rx Queue 1. All packets with priorities that match the values set in this field are routed to Rx Queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx Queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx Queues. When the DCB feature is selected, this field also determines the priorities to be included in the PFC packet sent to remote station when Rx Queue 1 crosses the flow control threshold settings.	0
23:16	PSRQ2	Priorities selected in the Rx Queue 2. This field decides the priorities assigned to Rx Queue 2. All packets with priorities that match the values set in this field are routed to Rx Queue 2. For example, if PSRQ2[1, 0] are set, packets with USP field equal to 1 or 0 are routed to Rx Queue 2. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx Queues. When the DCB feature is selected, this field also determines the priorities to be included in the PFC packet sent to remote station when Rx Queue 2 crosses the flow control threshold settings. This field is reserved and RO when the number of Rx Queue selected in your core is less than 3.	0
31:24	PSRQ3	Priorities selected in the Rx Queue 3. This field decides the priorities assigned to Rx Queue 3. All packets with priorities that match the values set in this field are routed to Rx Queue 3. For example, if PSRQ3[6, 3] are set, packets with USP field equal to 3 or 6 are routed to Rx Queue 3. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx Queues. When the DCB feature is selected, this field also determines the priorities to be included in the PFC packet sent to remote station when Rx Queue 3 crosses the flow control threshold settings. This field is reserved and RO when the number of Rx Queues selected in your core is less than 4.	0

36.6.12 MAC interrupt status register

The interrupt status register contents identify the events in the MAC-CORE that can generate interrupt.

Table 788. MAC interrupt status register (MAC_INTR_STAT, offset 0x00B0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved.	-
3	PHYIS		PHY interrupt. This bit is set when rising edge is detected on the PHY interrupt input signal. This bit is cleared when this register is read.	
		0	Rising edge not detected	
		1	Rising edge detected	
4	PMTIS		PMT interrupt status. This bit is set whenever a magic packet or Wake-on-LAN frame is received in power down mode (See bits 5 and 6 in Table 791). This bit is cleared when both bits 6 and 5 are cleared because of a read operation to the PMT control and status register.	0
		0	Magic packet or Wake-on-LAN frame not received	
		1	Magic packet or Wake-on-LAN frame received	
5	LPIIS		LPI interrupt status. When the energy efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in the MAC transmitter or receiver. This bit is cleared when the TLPIEN bit of MAC LPI control status register (Table 793) is read.	
		0	No LPI state entry or exit	
		1	LPI state entry or exit	
11:6	-	-	Reserved.	-
12	TSIS		Timestamp interrupt status. This bit is set when any of the following conditions is true: The system time value equals or exceeds the value specified in the target time high and low registers. There is an overflow in the seconds register. This bit is cleared on reading the byte 0 of the timestamp status register (Table 815). When default timestamping is enabled, this bit when set indicates that the system time value equals or exceeds the value specified in the target time registers. In this mode, this bit is cleared after the completion of the read of this interrupt status register[9]. In all other modes, this bit is reserved.	0

Table 788. MAC interrupt status register (MAC_INTR_STAT, offset 0x00B0) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
13	TXSTSIS		Transmit status interrupt. This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the MAC receive transmit status register. See Table 790. Excessive Collision (EXCOL) Late Collision (LCOL) Excessive Deferral (EXDEF) Loss of Carrier (LCARR) No Carrier (NCARR) Jabber Timeout (TJT) This bit is cleared when the corresponding interrupt source bit is read in the MAC receive transmit status register. See Table 790.	0
14	RXSTSIS		Receive status interrupt. This bit indicates the status of received packets. This bit is set when the RWT bit is set in the MAC receive transmit status register. See Table 790. This bit is cleared when the corresponding interrupt source bit is read in the MAC receive transmit status register. See Table 790.	0
31:15	-	-	Reserved.	-

36.6.13 MAC interrupt enable register

The MAC interrupt enable register contains the masks for generating the interrupts.

Table 789. MAC interrupt enable register (MAC_INTR_EN, offset 0x00B4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved.	-
3	PHYIE		PHY interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in MAC interrupt status register Table 788.	
		0	Rising edge not detected	
		1	Rising edge detected	
4	PMTIE		PMT interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in MAC interrupt status register Table 788.	0
		0	Magic packet or Wake-on-LAN frame not received	
		1	Magic packet or Wake-on-LAN frame received	
5	LPIIE		LPI interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in MAC interrupt status register Table 788.	
11:6	-	-	Reserved.	-
12	TSIE		Timestamp interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in MAC interrupt status register Table 788.	0

Table 789. MAC interrupt enable register (MAC_INTR_EN, offset 0x00B4) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
13	TXSTSIE		Transmit status interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the MAC interrupt status register Table 788.	0
14	RXSTSIS		Receive status interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the MAC interrupt status register Table 788.	0
31:15	-	-	Reserved.	-

36.6.14 MAC receive transmit status register

The receive transmit status register contains the receive and transmit error status.

Table 790. MAC receive transmit status register (MAC_RXTX_STAT, offset 0x00B8) bit description

Bit	Symbol	Description	Reset value
0	TJT	PHY interrupt enable. When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in MAC interrupt status register Table 788.	
1	NCARR	No carrier. When the DTXSTS bit is set in the MTL operation mode register Table 820, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission. Access restriction applies. Clears on read. Self-set to 1 on internal event. This bit is reserved in full-duplex mode.	0
2	LCARR	Loss of carrier. When the DTXSTS bit is set in the MTL operation mode register Table 820, this bit indicates that the loss of carrier occurred during packet transmission, that is, the PHY carrier signal was inactive for one or more transmission clock periods during packet transmission. The PHY drives this signal high when the transmit or receive medium is not idle. The PHY drives this signal low when both transmit and receive mediums are idle. This signal is not synchronous to any clock. This bit is valid only for packets transmitted without collision. This bit is reserved in full-duplex mode. Access restriction applies. Clears on read. Self-set to 1 on internal event.	0
3	EXDEF	Excessive deferral. When the DTXSTS bit is set in the MTL operation mode register Table 820 and the DC bit is set in the MAC configuration register Table 820, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when jumbo packet is enabled).This bit is reserved with full-duplex mode. Access restriction applies. Clears on read. Self-set to 1 on internal event.	0
4	LCOL	Late collision. When the DTXSTS bit is set in the MTL operation mode register Table 820, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including preamble in MII mode).This bit is not valid if the underflow error occurs. This bit is reserved in full-duplex mode. Access restriction applies. Clears on read. Self-set to 1 on internal event.	0
5	EXCOL	Excessive collisions. When the DTXSTS bit is set in the MTL operation mode register Table 820, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC configuration register Table 777, this bit is set after the first collision and the packet transmission is aborted.This bit is reserved in full-duplex mode. Access restriction applies. Clears on read. Self-set to 1 on internal event.	0

Table 790. MAC receive transmit status register (MAC_RXTX_STAT, offset 0x00B8) bit description *...continued*

Bit	Symbol	Description	Reset value
7:6	-	Reserved.	-
8	RWT	Receive watchdog timeout. This bit is set when a packet with length greater than 2,048 bytes is received (10,240 bytes when jumbo packet mode is enabled) and the WD bit is reset in the MAC configuration register Table 777. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the MAC configuration register Table 777. Clears on read. Self-set to 1 on internal event.	0
31:9	-	Reserved.	-

36.6.15 MAC PMT control status register

The PMT control and status register controls the PMT functions and provides the PMT interrupt status.

Table 791. MAC PMT control status register (MAC_PMT_CTRL_STAT, offset 0x00C0) bit description

Bit	Symbol	Description	Reset value	Access
0	PWRDWN	Power down. When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet.This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the magic packet enable, global unicast, or remote wake-up packet enable bit is set high.	0	R/W
1	MGKPKTEN	Magic packet enable. When this bit is set, a power management event is generated when the MAC receives a magic packet.	0	R/W
2	RWKPKTEN	Remote wake-up packet enable. When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.	0	R/W
4:3	-	Reserved.	-	-
5	MGKPRCVD	Magic packet received. When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.	0	RO
6	RWKPRCVD	Remote wake-up packet received. When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.	0	RO
8:7	-	Reserved.	-	-
9	GLBLUCAST	Global unicast. When this bit is set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.	0	R/W

Table 791. MAC PMT control status register (MAC_PMT_CTRL_STAT, offset 0x00C0) bit description

Bit	Symbol	Description	Reset value	Access
10	RWKPFE	Remote wake-up packet forwarding enable. When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet. The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high. **Remark:** If magic packet enable and wake-p frame enable are both set along with setting of this bit and magic packet is received prior to wake-up frame, this bit is self-cleared on receiving magic packet, the received magic packet is dropped, and all frames after received magic packet are forwarded to application. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0	R/W
23:11	-	Reserved.	-	-
28:24	RWKPTR	Remote Wake-up FIFO pointer. This field gives the current value (0 to 7) of the remote wake-up packet filter register pointer. When the value of this pointer is equal to 7, the contents of the remote wake-up packet filter register are transferred to the clk_rx_i domain when a write occurs to that register.	0	R/W
30:29	-	Reserved.	-	-
31	RWKFILTRST	Remote wake-up packet filter register pointer reset. When this bit is set, the remote wake-up packet filter register pointer is reset to 3'b000. It is automatically cleared after 1 clock cycle. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0	R/W

36.6.16 MAC remote wake-up packet filter register

This is the address through which the remote wake-up frame filter registers (MAC_RWK_PKT_FLT) are written/read by the application.

To load values in a MAC_RWK_PKT_FLT register, the WKUPFMFILTER_REG0 - WKUPFMFILTER_REG7 is loaded by sequentially loading eight values in address 0x0C4. The current pointer value is reflected in bits 28:24 of the MAC_PMT_Control_Status register.

Remark: Do not use bit-banding for this register.

See Section 36.7.1.2.2 for details.

Table 792. MAC remote wake-up frame filter register (MAC_RWK_PKT_FLT, offset 0x00C4) bit description

Bit	Symbol	Description	Reset value
31:0	ADDR	WKUPFMFILTER address.	0

36.6.17 MAC LPI control status register

The LPI control and status register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read. This register is present only when you select the energy efficient Ethernet feature while configuring the core.

Table 793. MAC LPI control status register (MAC_LPI_CTRL_STAT, offset 0x00D0) bit description

Bit	Symbol	Description	Reset value	Access
0	TLPIEN	Transmit LPI entry. When this bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register.	0	RO
1	TLPIEX	Transmit LPI exit. When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW timer has expired. This bit is cleared by a read into this register.	0	RO
2	RLPIEN	Receive LPI entry. When this bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register. Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.	0	RO
3	RLPIEX	Receive LPI exit. When this bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register. Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.	0	RO
7:4	-	Reserved.	-	-
8	TLPIST	Transmit LPI state. When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.	0	RO
9	RLPIST	Receive LPI state When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.	0	RO
15:10	-	Reserved.	-	-
16	LPIEN	LPI enable. When this bit is set, it instructs the MAC transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.	0	R/W
17	PLS	PHY link status. This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.	0	R/W
18	-	Reserved.	-	-

Table 793. MAC LPI control status register (MAC_LPI_CTRL_STAT, offset 0x00D0) bit description *…continued*

Bit	Symbol	Description	Reset value	Access
19	LPITXA	LPI Tx automate. This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the transmit side. If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of MTL TxQ0 operation mode register Table 820, when the MAC is in the LPI mode, it exits the LPI mode. When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.	0	R/W
20	LPIATE	LPI timer enable. This bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPITXEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the MAC LPI entry timer register Table 795. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPITXEN bit. This enables the re-entry into LPI state when it is IDLE again. When LPITE is 0, the LPI Auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPITXEN bit descriptions.	0	R/W
21	LPITCSE	LPI Tx clock stop enable. When this bit is set, the MAC asserts LPI Tx clock fating control signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert LPI Tx clock fating control signal high after it enters Tx LPI mode. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.	0	R/W
31:22	-	Reserved.	-	-

36.6.18 MAC LPI timers control register

The LPI timers control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.The LPI timers control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

Table 794. MAC LPI timer control register (MAC_LPI_TIMER_CTRL, offset 0x00D4) bit description

Bit	Symbol	Description	Reset value
15:0	TWT	LPI TW timer. This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.	0
25:16	LST	LPI LS timer. This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS timer is 1,000 (1 sec) as defined in the IEEE standard.	0
31:26	-	Reserved.	-

36.6.19 MAC LPI entry timer register

This register controls the Tx LPI entry timer. This counter is enabled only when bit 20 (LPITE) bit of MAC LPI control status register is set to 1.

Table 795. MAC LPI entry timer register (MAC_LPI_ENTR_TIMER, offset 0x00D4) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved.	-
19:3	LPIET	LPI entry timer. This field specifies the time in microseconds the MAC will wait to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1. Bits 2:0 are read-only so that the granularity of this timer is in steps of 8 microseconds.	0
31:20	-	Reserved.	-

36.6.20 MAC 1US tick counter register

This register controls the generation of the Reference time (1 microsecond tic) for all the LPI timers. This timer has to be programmed by the software initially.

Table 796. MAC 1US Tick Counter register (MAC_1US_TIC_COUNTR, offset 0x00DC) bit description

Bit	Symbol	Description	Reset value
11:0	TIC_1US_CNTR	1US TIC counter. The application must program this counter so that the number of clock cycles of CSR clock is 1 μs (subtract 1 from the value before programming). For example if the CSR clock is 100 MHz then this field needs to be programmed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1US events that are used to update some of the EEE related counters.	0x63
-	-	Reserved.	-

36.6.21 MAC version register

The version register identifies the version of the Ethernet block. This register contains two bytes: one that NXP uses to identify the core release number, and the other that you set while configuring the core.

Table 797. MAC Version register (MAC_VERSION, offset 0x0110) bit description

Bit	Symbol	Description	Reset value
7:0	SNPVER	NXP defined version.	0x10
15:8	USERVER	User defined version.	1
31:16	-	Reserved.	-

36.6.22 MAC debug register

The debug register provides the debug status of various MAC blocks.

Table 798. MAC Debug register (MAC_DBG, offset 0x0114) bit description

Bit	Symbol	Value	Description	Reset value
0	REPESTS		MAC MII receive protocol engine status. When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.	0
2:1	RFCFCSTS		MAC receive packet controller FIFO status. When this bit is set, this field indicates the active state of the small FIFO read and write controllers of the MAC receive packet controller module.	0
15:3	-	-	Reserved.	-
16	TPESTS		MAC MII transmit protocol engine status. When this bit is set, it indicates that the MAC or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.	0
18:17	TFCSTS		MAC transmit packet controller status. This field indicates the state of the MAC transmit packet controller module.	
		0x00	Idle state.	
		0x01	Waiting for one of the following: status of the previous packet OR IPG or backoff period to be over.	
		0x02	Generating and transmitting a pause control packet (in full-duplex mode).	
		0x03	Transferring input packet for transmission.	
31:19	-	-	Reserved.	-

36.6.23 MAC HW feature0 register

The MAC HW Feature0 register indicates the presence of the optional features or functions.

The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Table 799. MAC HW Feature0 register (MAC_HW_FEAT0, offset 0x011C) bit description

Bit	Symbol	Value	Description	Reset value
0	MIISEL		10 or 100 Mbps support. This bit is set to 1 as10/100 Mbps the mode of operation.	1
1	-	-	Reserved.	-
2	HDSEL		Half-duplex Support.	0
		0	half-duplex mode is not selected.	
		1	half-duplex mode is selected.	
3	-	-	Reserved.	-

Table 799. MAC HW Feature0 register (MAC_HW_FEAT0, offset 0x011C) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
4	VLHASH		Hash table based filtering option.	
		0	Disabled	
		1	Enabled	
5	SMASEL		SMA (MDIO) interface.	
		0	Disable station management (MDIO interface)	
		1	Enable Station Management (MDIO interface)	
6	RWKSEL		PMT remote wake-up packet detection.	1
		0	Disabled	
		1	Enabled	
7	MGKSEL		PMT magic packet detection.	1
		0	Disabled	
		1	Enabled	
8	MMCSEL		RMON module enable. This bit is set to 1 when the enable MAC management counters (MMC) option is selected.	0
		0	Disabled	
		1	Enabled	
9	ARPOFFSEL		ARP offload enabled. This bit is set to 1 when the enable IPv4 ARP offload option is selected.	0
		0	Disabled	
		1	Enabled	
11:10	-	-	Reserved.	-
12	TSSEL		IEEE 1588-2008 timestamp support.	1
		0	Disabled	
		1	Enabled	
13	EEESEL		Energy Efficient Ethernet support	1
		0	Disabled	
		1	Enabled	
14	TXCOESEL		Transmit checksum offload support	1
		0	Disabled	
		1	Enabled	
15	-	-	Reserved.	-
16	RXCOESEL		Receive checksum offload support.	1
		0	Disabled	
		1	Enabled	
24:17	-	-	Reserved.	-
26:25	TSSTSSEL		Timestamp system time source.	1
		0x01	Internal	
		Others	Reserved	
27	-	-	Reserved.	-

Table 799. MAC HW Feature0 register (MAC_HW_FEAT0, offset 0x011C) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
30:28	ACTPHYSEL		Active PHY selected.	0x00
		0x0	MII	
		0x4	RMII. All other values reserved.	
31	-	-	Reserved.	-

36.6.24 MAC HW feature1 register

The MAC HW feature1 register indicates the presence of the optional features or functions.

The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Table 800. MAC HW Feature1 register (MAC_HW_FEAT1, offset 0x0120) bit description

Bit	Symbol	Value	Description	Reset value
4:0	RXFIFOSIZE		MTL Receive FIFO size. This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7	0x4
		0x04	2,048 bytes. All other values reserved.	
5	-	-	Reserved.	-
10:6	TXFIFOSIZE		MTL Transmit FIFO size. This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:	0x4
		0x04	2,048 bytes. All other values reserved.	
11	OSTEN		One-step timestamping feature.	0
		0	Disable	
		1	Enable	
12	PTOEN		PTP offLoad feature.	0
		0	Disable	
		1	Enable	
13	ADVTHWORD		IEEE 1588 high word register feature	0
		0	Disable	
		1	Enable	
15:14	ADDR64		Address width. This field indicates the configured address width.	0
		0x0	32. All other values reserved.	
16	DCBEN		Data center bridging feature.	0
		0	Disable	
		1	Enable	
17	SPEN		Split header structure feature.	0
		0	Disable	
		1	Enable	
18	TSOEN		TCP segment offload feature.	0
		0	Disable	
		1	Enable	

Table 800. MAC HW Feature1 register (MAC_HW_FEAT1, offset 0x0120) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
19	DBGMEMA		DMA debug register feature.	0
		0	Disable	
		1	Enable	
20	AVSEL		Audio video bridging feature.	1
		0	Disable	
		1	Enable	
23:21	-	-	Reserved.	-
25:24	HASHTBLSZ		Hash table size.	0
		0x00	No hash table. All other values reserved.	
26	-	-	Reserved.	-
30:27	L3_L4_FILTER		Total number of L3 and L4 filters.	0
		0x00	No L3 and L4 filters. All other values reserved.	
31	-	-	Reserved.	-

36.6.25 MAC HW feature2 register

The MAC HW Feature2 register indicates the presence of the optional features or functions.

The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Table 801. MAC HW Feature2 register (MAC_HW_FEAT2, offset 0x0124) bit description

Bit	Symbol	Value	Description	Reset value
3:0	RXQCNT		Number of MTL Rx Queues.	1
		0x1	2 MTL Rx Queues. All other values reserved.	
5:4	-		Reserved.	0
9:6	TXQCNT		Number of MTL Tx Queues.	1
		0x1	2 MTL Tx Queues.	
11:10	-		Reserved.	0
15:12	RXCHCNT		Number of DMA receive channels.	1
		0x001	2 DMA Rx channels. All other values reserved.	
17:16	-	-	Reserved.	-
21:18	TXCHCNT		Number of DMA transmit channels.	1
		0x001	2 DMA Tx channels. All other values reserved.	
23:22	-	-	Reserved	-
26:24	PPSOUTNUM		Number of PPS outputs.	0
		0x000	No PPS output. All other values reserved.	
27	-	-	Reserved.	-
30:28	AUXSNAPNUM		Number of auxiliary snapshot inputs.	0
		0x000	No auxiliary input. All other values reserved.	
31	-	-	Reserved.	-

36.6.26 MAC MDIO address register

The MDIO address register controls the management cycles to external PHY through a management interface.

Table 802. MAC MDIO address register (MAC_MDIO_ADDR, offset 0x0200) bit description

Bit	Symbol	Value	Description	Reset value
0	MB		MII busy. The application sets this bit to instruct the SMA to initiate a read or write access to the MDIO slave. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in MAC MDIO address register Table 802 and MAC MDIO data registers Table 803 as long as this bit is set. For write transfers, the application must first write 16-bit data in the GD field in MAC MDIO data register Table 803 before setting this bit. When a read transfer is completed (MB=0), the data read from the PHY register is valid in the GD field of the MAC MDIO data register Table 803. Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0
		0	Not busy	
		1	Busy	
1	-	-	Reserved.	-
3:2	MOC		MII operation command. This field indicates the operation command to the PHY.	0
		0x0	Reserved	
		0x1	Write	
		0x2	Reserved	
		0x3	Read	
7:4	-	-	Reserved.	-
11:8	CR		CSR clock range.The CSR clock range selection determines the frequency of the MDC clock according to the CSR clock frequency used in design. The CSR clock is same as Core Clock.	0
		0x0	CSR clock = 60-100 MHz; MDC clock = CSR clock/42	
		0x1	CSR clock = 100-150 MHz; MDC clock = CSR clock/62	
		0x2	CSR clock = 20-35 MHz; MDC clock = CSR clock/16	
		0x3	CSR clock = 35-60 MHz; MDC clock = CSR clock/26	
14:12	NTC		Number of training clocks. This field controls the number of trailing clock cycles generated on MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.	
15	-	-	Reserved.	-
20:16	RDA		Register/device address. These bits select the PHY register in selected PHY device.	0
25:21	PA		Physical layer address. This field indicates which PHY devices (out of 32 devices) the MAC is accessing.	

Table 802. MAC MDIO address register (MAC_MDIO_ADDR, offset 0x0200) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
26	BTB		Back to back transactions. When this bit is set and the NTC has value greater than 0, then the MAC will inform the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which will be executed immediately irrespective of the number trailing clocks generated for the previous frame. When this bit is reset, then the read/write command completion (MB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. This bit must not be set when NTC=0.	
27	PSE		Preamble suppression enable. When this bit is set, the SMA will suppress the 32-bit preamble and transmit MDIO frames with only 1 preamble bit. When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.	
31:28	-	-	Reserved.	-

36.6.27 MAC MDIO data register

The MDIO data register stores the write data to be written to the PHY register located at the address specified in MDIO address register. This register also stores the read data from the PHY register located at the address specified by MDIO address register.

Table 803. MAC MDIO data register (MAC_MDIO_DATA, offset 0x0204) bit description

Bit	Symbol	Description	Reset value
15:0	MD	MII data. This field contains the 16-bit data value read from the PHY after a management read operation or the 16-bit data value to be written to the PHY before a management write operation.	0
31:16	-	Reserved.	-

36.6.28 MAC address high register

The MAC address high register holds the upper 16 bits of the 6-byte first MAC address of the station. Note that the first DA byte that is received on the MII interface corresponds to the LS Byte (bits 7:0) of the MAC address low register. For example, if 0x1122 3344 5566 is received (0x11 is the first byte) on the MII as the destination address, then the MAC address register[47:0] is compared with 0x6655 4433 2211.

If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when bits 31:24 (in little-endian mode) or bits 7:0 (in Big-Endian mode) of the MAC address low register are written to.

Remark: Consecutive writes to this address low register should be performed only after at least 4 clock cycles in the destination clock domain for proper synchronization updates.

Table 804. MAC address high register (MAC_ADDR_HIGH, offset 0x0300) bit description

Bit	Symbol	Description	Reset value	Access
15:0	A47_32	MAC address0[47:32]. This field contains the upper 16 bits (47:32) of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the transmit flow control (PAUSE) frames.	0xFFFF	R/W
16	DCS	DMA channel select. This field contains the DMA channel number to which the Rx packet whose DA matches the MAC address content is routed.		
30:17	-	Reserved.	-	-
31	AE	Address enable. This bit is always 1.	1	RO

36.6.29 MAC address low register

The MAC address low register holds the lower 32 bits of the 6-byte first MAC address of the station.

Table 805. MAC address low register (MAC_ADDR_LOW, offset 0x0304) bit description

Bit	Symbol	Description	Reset value
31:0	A31_0	MAC address0[31:0]. This field contains the lower 32 bits of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the transmit flow control (PAUSE) frames.	0xFFFF FFFF

36.6.30 MAC IEEE 1588 timestamp control register

This register controls the operation of the system time generator and processing of PTP packets for timestamping in the receiver.

Table 806. MAC timestamp control register (MAC_TIMESTAMP_CTRL, offset 0x0B00) bit description

Bit	Symbol	Description	Reset value
0	TSENA	Enable timestamp. When this bit is set, the timestamp is added for transmit and receive packets. When disabled, timestamp is not added for transmit and receive packets and the timestamp generator is also suspended. You need to initialize the timestamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 packets only if this bit is set.	0
1	TSCFUPDT	Fine or coarse timestamp update. When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.	0
2	TSINIT	Initialize timestamp. When this bit is set, the system time is initialized (overwritten) with the value specified in the MAC register 80 (system time seconds update register) and MAC register 81 (system time nanoseconds update register). This bit should be zero before it is updated. This bit is reset when the initialization is complete. The timestamp higher word register (if enabled during core configuration) can only be initialized.	0

Table 806. MAC timestamp control register (MAC_TIMESTAMP_CTRL, offset 0x0B00) bit description *...continued*

Bit	Symbol	Description	Reset value
3	TSUPDT	Update timestamp. When this bit is set, the system time is updated (added or subtracted) with the value specified in MAC system time seconds update Table 811 and MAC system time nanoseconds update Table 812. This bit should be zero before updating it. This bit is reset when the update is complete in hardware. The timestamp higher word register (if enabled during core configuration) is not updated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0
4	TSTRIG	Enable timestamp interrupt trigger. When this bit is set, the timestamp interrupt is generated when the system time becomes greater than the value written in the target time register. This bit is reset after the timestamp trigger interrupt is generated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0
5	TADDREG	Update addend register. When this bit is set, the content of the timestamp addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0
7:6	-	Reserved.	-
8	TSENALL	Enable timestamp for all packets. When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.	0
9	TSCTRLSSR	Timestamp digital or binary rollover control. When this bit is set, the timestamp low register rolls over after 0x3B9AC9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (high) seconds. When this bit is reset, the rollover value of sub-second register is 0x7FFFFFFF. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.	0
10	TSVER2ENA	Enable PTP packet processing for version 2 format. When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in PTP processing and control section	0
11	TSIPENA	Enable processing of PTP over Ethernet packets. When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.	0
12	TSIPV6ENA	Enable processing of PTP packets sent over 1Pv6-UDP. When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.	0
13	TSIPV4ENA	Enable processing of PTP packets sent over IPv4-UDP. When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.	1

Table 806. MAC timestamp control register (MAC_TIMESTAMP_CTRL, offset 0x0B00) bit description *...continued*

Bit	Symbol	Description	Reset value
14	TSEVTENA	Enable timestamp snapshot for event messages. When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Table 807 "Timestamp snapshot dependency on register bits".	0
15	TSMSTRENA	Enable snapshot for messages relevant to master. When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.	0
17:16	SNAPTYPSEL	Select PTP packets for taking snapshots. These bits, along with bits 15 and 14, decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Table 807 "Timestamp snapshot dependency on register bits".	0
18	TSENMACADDR	Enable MAC address for PTP packet filtering. When this bit is set, the DA MAC address (that matches any MAC address register) is used to filter the PTP packets when PTP is directly sent over Ethernet. When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet. For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching. For PTP offload, only MAC address register 0 is considered for unicast destination address matching.	0
23:19	-	Reserved.	-
24	TXTTSSTSM	Transmit timestamp status mode. When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSTSMIS bit of the MAC_TxTimestamp_Status_Nanoseconds register. When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSTSHI bit of the MAC_TxTimestamp_Status_Seconds register.	0
27:25	-	Reserved.	-
28	AV8021ASMEN	AV 802.1AS mode enable. When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS mode of operation. When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of this bit.	0
31:29	-	Reserved.	-

Table 807 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in MAC_Timestamp_Control register.

Table 807. Timestamp snapshot dependency on register bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP Messages
00	x	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	x	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	x	x	SYNC,Pdelay_Req,
11	x	x	Pdelay_Resp Delay_Req

36.6.31 Sub-second increment register

In coarse update mode (TSCFUPDT bit in Table 806), the value in this register is added to the system time every clock cycle. In fine update mode, the value in this register is added to the system time whenever the accumulator gets an overflow.

Table 808. Sub-second increment register (MAC_SUB_SCND_INCR, offset 0x0B04) bit description

Bit	Symbol	Description	Reset value
15:0	-	Reserved.	0
23:16	SSINC	Sub-second increment value. The value programmed in this register is accumulated with the contents of the sub-second register. For example, to achieve an accuracy of 20 ns, the value to be programmed is 20 (program 0x14 with a 50 MHz reference clock if 1 ns accuracy is selected).	0
31:24	-	Reserved.	0

36.6.32 System time seconds register

The system time seconds register, along with system time nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time due to clock domain transfer latencies.

Table 809. System time seconds register (MAC_SYS_TIME_SCND, offset 0x0B08) bit description

Bit	Symbol	Description	Reset value
31:0	TSS	Timestamp second. The value in this field indicates the current value in seconds of the system time maintained by the MAC.	0

36.6.33 System time nanoseconds register

The system time nanoseconds register, along with system time seconds register, indicates the current value of the system time maintained by the MAC

Table 810. System time nanoseconds register (MAC_SYS_TIME_NCND, offset 0x0B0C) bit description

Bit	Symbol	Description	Reset value
30:0	TSSS	Timestamp sub seconds. The value in this field has the sub second representation of time, with an accuracy of 0.46 nano-second. (When TSCTRLSSR in the MAC IEEE1588 timestamp control register Table 806 is set, each bit represents 1 ns and the maximum value will be 0x3B9A C9FF, after which it rolls-over to zero).	0
31	-	Reserved.	-

36.6.34 System time seconds update register

The system time seconds update register, along with the system time nanoseconds update register, initialize or update the system time maintained by the MAC. These two registers must be written before setting the TSINIT or TSUPDT bits in the timestamp control register.

Table 811. System time seconds update register (MAC_SYS_TIME_SCND_UPD, offset 0x0B10) bit description

Bit	Symbol	Description	Reset value
31:0	TSS	Timestamp second. The value in this field indicates the time, in seconds, to be initialized or added to the system time.	0

36.6.35 System time nanoseconds update register

This register contains 32 bits of the nano-seconds field to be written to, added to, or subtracted from the system time value.

Table 812. System time nanoseconds update register (MAC_SYS_TIME_NSCND_UPD, offset 0x0B14) bit description

Bit	Symbol	Description	Reset value
30:0	TSSS	Timestamp sub seconds. The value in this field has the sub second representation of time, with an accuracy of 0.46 nano-second. (When TSCTRLSSR is set in the timestamp control register, each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF.)	0
31	ADDSUB	Add or subtract time. When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.	0

36.6.36 Timestamp addend register

This register is used by the software to readjust the clock frequency linearly to match the master clock frequency.

This register value is used only when the system time is configured for fine update mode (TSCFUPDT bit in Table 806). This register content is added to a 32-bit accumulator in every clock cycle and the system time is updated whenever the accumulator overflows.

Table 813. Timestamp addend register (MAC_SYS_TIMESTMP_ADDEN, offset 0x0B18) bit description

Bit	Symbol	Description	Reset value
31:0	TSAR	Timestamp addend. This register indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.	0

36.6.37 System time higher words seconds register

This register contains the most significant 16-bits of the timestamp seconds value.

Table 814. System time higher words seconds register (MAC_SYS_TIME_HWORD_SCND, offset 0x0B1C) bit description

Bit	Symbol	Description	Reset value
15:0	TSHWR	Timestamp higher word. Contains the most significant 16-bits of the timestamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the system time seconds register.	0
31:16	-	Reserved.	-

36.6.38 Timestamp status register

This register contains the PTP status. All bits except bits 27:25 gets cleared after this register is read by the host.

The register field can be read by the application (Read), can be set to 1 by the core on a certain internal event (Self Set), and is automatically cleared to 0 on a register read. A register write of 0 has no effect on this field.

Table 815. Timestamp status register (MAC_SYS_TIMESTMP_STAT, offset 0x0B20) bit description

Bit	Symbol	Description	Reset value
0	TSSOVF	Timestamp seconds overflow. When set, indicates that the seconds value of the timestamp has overflowed beyond 0xFFFF FFFF.	0
31:1	-	Reserved.	-

36.6.39 Tx timestamp status nanoseconds register

This register contains the timestamp captured for transmit packets when Tx status is disabled.

Table 816. Tx timestamp status nanoseconds register (MAC_Tx_TIMESTMP_STAT_NANOSECONDS, offset 0x0B30) bit description

Bit	Symbol	Description	Reset value
30:0	TXTSSTSLO	Transmit timestamp status low. This field contains the 31 bits of the Nanoseconds field of the transmit packet's captured timestamp.	0
31	TXTSSTSMIS	Transmit timestamp status missed. When this bit is set, it indicates one of the following: The timestamp of the current packet is ignored if TXTSSTSM bit of the MAC_Timestamp_Control register is reset. The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the MAC_Timestamp_Control register is set.	0

36.6.40 Tx timestamp status seconds register

This register contains the timestamp captured for transmit packets when Tx status is disabled.

Table 817. Tx timestamp status seconds register (MAC_Tx_TIMESTMP_STAT_SECONDS, offset 0x0B34) bit description

Bit	Symbol	Description	Reset value
31:0	TXTSSTSHI	Transmit timestamp status high. This field contains the lower 32 bits of the seconds field of transmit packet's captured timestamp.	0

36.6.41 MAC timestamp ingress correction nanosecond register

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

Table 818. MAC timestamp ingress corr nanosecond (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND offset 0x0B58) bit description

Bit	Symbol	Description	Reset value
31:0	TSIC	Transmit ingress correction. This field contains the ingress path correction value as defined by the ingress correction expression.	0

36.6.42 MAC timestamp egress correction nanosecond register

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

Table 819. MAC timestamp egress corr nanosecond (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND offset 0x0B5C) bit description

Bit	Symbol	Description	Reset value
31:0	TSEC	Transmit egress correction. This field contains the nanoseconds part of the egress path correction value as defined by the egress correction expression.	0

36.6.43 MTL operation mode register

The operation mode register establishes the transmit and receive operating modes and commands.

Table 820. MTL operation mode register (MTL_OP_MODE, offset 0x0C00) bit description

Bit	Symbol	Description	Reset value	Access
0	-	Reserved.	-	-
1	DTXSTS	Drop transmit status. When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.	0	R/W
2	RAA	Receive arbitration algorithm. This field is used to select the arbitration algorithm for the Rx side. 0: Strict priority (SP) 1: Weighted Strict Priority (WSP) Queue 0 has the lowest priority and the last queue has the highest priority.	0	RO
4:3	-	Reserved.	-	-

Table 820. MTL operation mode register (MTL_OP_MODE, offset 0x0C00) bit description *…continued*

Bit	Symbol	Description	Reset value	Access
6:5	SCHALG	Tx scheduling algorithm. This field indicates the algorithm for Tx scheduling: 0x00: WRR algorithm 0x1: Reserved 0x2: Reserved 0x3: Strict priority algorithm	0	R/W
7	-	Reserved.	-	-
8	CNTPRST	Counters preset. When this bit is set, MTL TxQ0 Underflow register (Table 824) and MTL_TxQ1_Underflow (Table 824) registers are initialized/preset to 0x7F0. Missed packet and overflow packet counters in MTL RxQ0 missed packet overflow counter Table 835, MTL RxQ1 missed packet overflow counter Table 835 registers are initialized/preset to 0x7F0. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0	R/W
9	CNTCLR	Counters reset. When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	0	R/W
31:10	-	Reserved.	0	RO

36.6.44 MTL interrupt status register

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

Table 821. MTL interrupt status register (MTL_INTR_STAT, offset 0x0C20) bit description

Bit	Symbol	Description	Reset value
0	Q0IS	Queue 0 interrupt status. This bit indicates that there is an interrupt from Queue 0. To reset this bit, the application must read Queue 0 interrupt control and status register to get the exact cause of the interrupt and clear its source.	0
1	Q1IS	Queue 1 interrupt status. This bit indicates that there is an interrupt from Queue 1. To reset this bit, the application must read the Queue 1 interrupt control and status register to get the exact cause of the interrupt and clear its source.	0
31:2	-	Reserved.	0

36.6.45 MTL Rx Queue and DMA channel mapping register

Table 822. MTL Rx Queue and DMA channel mapping register (MTL_RXQ_DMA_MAP, offset 0x0C30) bit description

Bit	Symbol	Description	Reset value
0	Q0MDMACH	Queue 0 mapped to DMA channel. This field controls the routing of the packet received in Queue 0 to the DMA channel: 0: DMA channel 0 1: DMA channel 1 This field is valid when the Q0DDMACH field is reset.	0
3:1	-	Reserved.	-
4	Q0DDMACH	Queue 0 enabled for DA-based DMA channel selection. When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA channel programmed in the Q0MDMACH field.	0
7:5	-	Reserved.	-
8	Q1MDMACH	Queue 1 mapped to DMA channel. This field controls the routing of the received packet in Queue 1 to the DMA channel: 0: DMA channel 0 1: DMA channel 1 This field is valid when the Q1DDMACH field is reset.	0
11:9	-	Reserved.	-
12	Q1DDMACH	Queue 1 enabled for DA-based DMA channel selection. When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA channel programmed in the Q1MDMACH field (bits 10:8).	0
31:13	-	Reserved.	-

36.6.46 MTL TxQ operation mode register

The queue transmit operation mode register establishes the Tx Queue operating modes and commands.

Table 823. MTL TxQ operation mode register (MTL_TXQ0_OP_MODE, offset 0x0D00) and MTL_TXQ1_OP_MODE, offset 0x0D40 bit description

Bit	Symbol	Description	Reset value
0	FTQ	Flush Tx Queue. When this bit is set, the Tx Queue controller logic is reset to its default values. Therefore, all the data in the Tx Queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, in MTL TxQ0 operation mode you should not write to the MTL TxQ1 operation mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note:** The flush operation is complete only when the Tx Queue is empty and the application has accepted the pending Tx status of all transmitted packets. To complete this flush operation, the PHY Tx clock should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.	-
1	TSF	Transmit store and forward. When this bit is set, the transmission starts when a full packet resides in the MTL Tx Queue. When this bit is set, the TTC values specified in bits 6:4 of this register are ignored. This bit should be changed only when the transmission is stopped.	0
3:2	TXQEN	Tx Queue enable. This field is used to enable/disable the Tx Queue 0. 0x0: Not enabled 0x2: Enabled Others: Reserved	0
6:4	TTC	Transmit threshold control. These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 0x0: 32 0x1: 64 0x2: 96 0x3: 128 0x4: 192 0x5: 256 0x6: 384 0x7: 512	0
15:7	-	Reserved.	-
18:16	TQS	Tx Queue size. This field indicates the size of the allocated Tx Queues in blocks of 256 bytes.	0
31:19	-	Reserved.	-

36.6.47 MTL TxQ underflow register

The Queue Underflow Counter register contains the counter for packets aborted because of Tx Queue underflow and packets missed because of Rx Queue packet flush.

Table 824. MTL TxQ Underflow register (MTL_TXQ0_UNDRFLW, offset 0x0D04 and MTL_TXQ1_UNDRFLW, offset 0x0D44) bit description

Bit	Symbol	Description	Reset value
10:0	UFFRMCNT	Underflow packet counter. This field indicates the number of packets aborted by the controller because of Tx Queue underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.	0
11	UFCNTOVF	Overflow bit for underflow packet counter. This bit is set every time the Tx Queue underflow packet counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event	0
31:12	-	Reserved.	-

36.6.48 MTL TxQ debug register

The queue transmit debug register gives the debug status of various blocks related to the Tx Queue.

Table 825. MTL TxQ debug register (MTL_TXQ0_DBG, offset 0x0D08 and MTL_TXQ1_DBG, offset 0x0D48) bit description

Bit	Symbol	Description	Reset value
0	TXQPAUSED	Tx Queue in pause. When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the pause condition (in the full-duplex only mode) because of the following: - Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled - Reception of 802.3x pause packet when PFC is disabled	0
2:1	TRCSTS	MTL Tx Queue read controller status. This field indicates the state of the Tx Queue read controller: 00: idle state 01: read state (transferring data to the MAC transmitter) 10: waiting for pending Tx status from the MAC transmitter 11: flushing the Tx Queue because of the packet abort request from the MAC	0
3	TWCSTS	MTL Tx Queue write controller status. When high, this bit indicates that the MTL Tx Queue write controller is active, and it is transferring the data to the Tx Queue.	0
4	TXQSTS	MTL Tx Queue not empty status. When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission.	0
5	TXSTSFSTS	MTL Tx status FIFO full status. When high, this bit indicates that the MTL Tx status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.	0
15:6	-	Reserved.	-
18:16	PTXQ	Number of packets in the Tx Queue. This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL operation mode register Table 820 is set to 1, this field does not reflect the number of packets in the Tx Queue.	0

Table 825. MTL TxQ debug register (MTL_TXQ0_DBG, offset 0x0D08 and MTL_TXQ1_DBG, offset 0x0D48) bit description *...continued*

Bit	Symbol	Description	Reset value
19	-	Reserved.	0
22:20	STSXSTSF	Number of status words in Tx status FIFO of queue. This field indicates the current number of status in the Tx status FIFO of this queue. When the DTXSTS bit of MTL operation mode register Table 820 is set to 1, this field does not reflect the number of status words in Tx status FIFO.	0
31:23	-	Reserved.	-

36.6.49 MTL TxQ1 ETS control register

The Queue 1 ETS control register controls the enhanced transmission selection operation.

Table 826. MTL TxQ1 ETS control register (MTL_TXQ1_ETS_CTRL, offset 0x0D50) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved.	-	-
2	AVALG	AV algorithm. When Queue 1 is programmed for AV, this field configures the scheduling algorithm for this queue: This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected. If AV feature is not selected for this queue, this field is reserved and RO.	-	R/W
3	CC	Credit control. When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in channel 1. The credit accumulates even when there is no packet waiting in channel 1 and another channel is transmitting. When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in channel 1. When there is no packet waiting in channel 1 and other channel is transmitting, no credit is accumulated.	-	R/W
6:4	SLC	Slot count. If the credit-based shaper algorithm is enabled, the software can program the number of slots (of 125 us duration) over which the average transmitted bits per slot, provided in the MTL TxQ1 ETS status register Table 827 , need to be computed for Queue 1. The encoding is as follows: 0x0: 1 Slot 0x1: 2 Slots 0x2: 4 Slots 0x3: 8 Slots 0x4: 16 Slots Others: Reserved If the AV feature is not selected, this field is reserved and RO.	0	RO
31:7	-	Reserved.	-	-

36.6.50 MTL TxQ ETS status register

The Queue ETS status register provides the average traffic transmitted in queue.

Table 827. MTL TxQ ETS status register (MTL_TXQ0_ETS_STAT, offset 0x0D14 and MTL_TXQ1_ETS_STAT, offset 0x0D54) bit description

Bit	Symbol	Description	Reset value
23:0	ABS	Average bits per slot. This field contains the average transmitted bits per slot.	-
31:24	-	Reserved.	-

36.6.51 MTL TxQ0 quantum weight register

The queue 0 quantum or weights register contains the quantum value for Deficit Weighted Round Robin (DWRR), weights for the Weighted Round Robin (WRR), and Weighted Fair Queuing (WFQ) for Queue 0.

Table 828. MTL TxQ0 quantum weight register (MTL_TXQ0_QNTM_WGHT, offset 0x0D18 bit description

Bit	Symbol	Description	Reset value
20:0	ISCQW	Quantum or weights. When the DCB operation is enabled with DWRR algorithm for Queue 0 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x13 12D0 bytes. When DCB operation is enabled with WFQ algorithm for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits 20:14 must be written to zero. When DCB operation is enabled with WRR algorithm for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits 20:7 must be written to zero.	0
31:21	-	Reserved.	-

36.6.52 MTL TxQ1 quantum weight register

The Queue 1 idleSlopeCredit, Quantum, or Weights register provides the average traffic transmitted in Queue 1.

Table 829. MTL TxQ1 Quantum Weight register (MTL_TXQ1_QNTM_WGHT, offset 0x0D58) bit description

Bit	Symbol	Description	Reset value
20:0	ISCQW	idleSlopeCredit, Quantum, or Weights idleSlopeCredit When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1,000 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1,000 Mbps mode and 0x1000 in 100 Mbps mode. Bits 20:14 must be written to zero. Quantum When the DCB feature is enabled with DWRR algorithm for Queue 1 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x13 12D0 bytes. Weights When DCB operation is enabled with WFQ algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits 20:14 must be written to zero. When DCB operation is enabled with WRR algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits 20:7 must be written to zero.	0
31:21	-	Reserved.	-

36.6.53 MTL TxQ1 SendSlopeCredit register

The SendSlopeCredit register contains the SendSlope credit value required for the credit-based shaper algorithm for the queue.

Table 830. MTL TxQ1 SendSlopCredit register (MTL_TXQ1_SNDSLP_CRDT, offset 0x0D5C) bit description

Bit	Symbol	Description	Reset value
13:0	SSC	sendSlopeCredit. When AV operation is enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns and 8 ns for 100 Mbps and 1,000Mbps respectively) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1,024.The maximum value is port transmit rate, that is, 0x2000 in 1,000 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when channel 1 is selected for transmission. This bit is reserved when the AV feature is not selected.	0
31:14	-	Reserved.	-

36.6.54 MTL TxQ1 hiCredit register

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the queue.

Table 831. MTL TxQ1 hiCredit register (MTL_TXQ1_HI_CRDT, offset 0x0D60) bit description

Bit	Symbol	Description	Reset value
28:0	HC	hiCredit. When the AV feature is enabled, this field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is: 131,072 * 1,024 = 134,217,728 or 0x0800 0000.	0
31:29	-	Reserved.	-

36.6.55 MTL TxQ1 loCredit register

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the queue.

Table 832. MTL TxQ1 loCredit register (MTL_TXQ1_LO_CRDT, offset 0x0D64) bit description

Bit	Symbol	Description	Reset value
28:0	LC	loCredit. When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is max frame size transmitted from this queue, that is, 16,384 * 8 * 1,024 = 134,217,728 or 0x0800 0000. Because it is a negative value, the programmed value is 2's complement of the value, that is, 0x1800 0000. This bit is reserved when the AV feature is not selected.	0
31:29	-	Reserved.	-

36.6.56 MTL TxQ interrupt control status register

This register contains the interrupt enable and status bits for the queue interrupts.

Table 833. MTL TxQ interrupt control status register (MTL_TXQ0_INTCTRL_STAT, offset 0x0D2C and MTL_TXQ1_INTCTRL_STAT, offset 0x0D6C) bit description

Bit	Symbol	Description	Reset value	Access
0	TXUNFIS	Tx Queue underflow interrupt status. This bit indicates that the Tx Queue had an underflow while transmitting the packet. Transmission is suspended and an underflow error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.	-	R/W
1	ABPSIS	Average bits per slot interrupt status. When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit.	0	R/W
7-2	-	Reserved.	0	RO
8	TXUIE	Tx Queue underflow interrupt enable. When this bit is set, the Tx Queue underflow interrupt is enabled. When this bit is reset, the Tx Queue underflow interrupt is disabled.	0	R/W
9	ABPSIE	Average bits per slot interrupt enable. When this bit is set, the MAC asserts the interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event.	0	R/W
15-10	-	Reserved.	-	-
16	RXOVFIS	Rx Queue overflow interrupt status. This bit indicates that the Rx Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.	0	R/W
23:17	-	Reserved.	0	RO
24	RXOIE	Rx Queue overflow interrupt enable. When this bit is set, the Rx Queue overflow interrupt is enabled. When this bit is reset, the Rx Queue overflow interrupt is disabled.	0	R/W
31:25	-	Reserved.	-	-

36.6.57 MTL RxQ operation mode register

The queue receive operation mode register establishes the Rx Queue operating modes and command.

Table 834. MTL RxQ operation mode register (MTL_RXQ0_OP_MODE, offset 0x0D30 and MTL_RXQ1_OP_MODE, offset 0x0D70) bit description

Bit	Symbol	Description	Reset value	Access
1:0	RTC	Rx Queue threshold control. These bits control the threshold level of the MTL Rx Queue (in bytes): 00: 64 01: 32 10: 96 11: 128 The packet received is transferred to the application or DMA when the packet size within the MTL Rx Queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.	0	R/W
2	-	Reserved.	-	-
3	FUP	Forward undersized good packets. When this bit is set, the Rx Queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx Queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx threshold, for example, RTC = 01.	0	R/W
4	FEP	Forward error packets. When this bit is reset, the Rx Queue drops packets with error status (CRC error, MII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx Queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx Queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.	0	R/W
5	RSF	Rx Queue store and forward. When this bit is set, the Ethernet block on this chip reads a packet from the Rx Queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx Queue operates in the threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.	0	R/W
6	DIS_TCP_EF	Disable dropping of TCP/IP checksum error packets. When this bit is set, the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset.	0	R/W
19:7	-	Reserved.	0	RO
22:20	RQS	This field indicates the size of the allocated Rx Queues in blocks of 256 bytes. The twentieth bit is the starting bit of this field.	0	R/W
31:23	-	Reserved.	-	-

36.6.58 MTL RxQ missed packet overflow counter register

The queue missed packet and overflow counter register contains the counter for packets missed because of Rx Queue packet flush and packets discarded because of Rx Queue overflow.

Table 835. MTL RxQ missed packet overflow counter register (MTL_RxQ0_MISSPKT_OVRFLW_CNT, offset 0x0D34 and MTL_RxQ1_MISSPKT_OVRFLW_CNT, offset 0x0D74) bit description

Bit	Symbol	Description	Reset value
10:0	OVFPKTCNT	Overflow packet counter. This field indicates the number of packets discarded by the Ethernet block because of Rx Queue overflow. This counter is incremented each time the Ethernet block discards an incoming packet because of overflow. This counter is reset when this register is read	0
11	OVFCNTOVF	Overflow counter overflow bit. When set, this bit indicates that the Rx Queue overflow packet Counter field crossed the maximum limit.	0
31:12	-	Reserved.	-

36.6.59 MTL RxQ debug register

The queue receive debug register gives the debug status of various blocks related to the Rx Queue.

Table 836. MTL RxQ Debug register (MTL_RXQ0_DBG, offset 0x0D38 and MTL_RXQ1_DBG, offset 0x0D78) bit description

Bit	Symbol	Description	Reset value
0	RWCSTS	MTL Rx Queue write controller active status. When high, this bit indicates that the MTL Rx Queue write controller is active, and it is transferring a received packet to the Rx Queue.	0
2:1	RRCSTS	MTL Rx Queue read controller state. This field gives the state of the Rx Queue read controller: 00: Idle state 01: Reading packet data 10: Reading packet status (or timestamp) 11: Flushing the packet data and status	0
3	-	Reserved.	-
5:4	RXQSTS	MTL Rx Queue fill-level status. This field gives the status of the fill-level of the Rx Queue: 0x0: Rx Queue empty 0x1: Rx Queue fill-level below flow-control deactivate threshold 0x2: Rx Queue fill-level above flow-control activate threshold 0x3: Rx Queue full	0
15:6	-	Reserved.	0
29:16	PRXQ	Number of packets in Rx Queue. This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256 KB/16 B = 16K packets, that is, max queue size/min packet size.	0
31:30	-	Reserved.	-

36.6.60 MTL RxQ control register

The queue receive control register controls the receive arbitration and passing of received packets to the application.

Table 837. MTL RxQ control register (MTL_RXQ0_CTRL, offset 0x0D3C and MTL_RXQ1_CTRL, offset 0x0D7C) bit description

Bit	Symbol	Description	Reset value
2:0	RXQ_WEGT	Rx Queue weight. This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_FKT_ARBIT) allocated to the queue in one arbitration cycle.	0
3	RXQ_FRM_ARBIT	Rx Queue packet arbitration. When this bit is set, the The Ethernet block drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the Ethernet block drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: • PBL amount of data, or, • Complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet block drives the complete status (including timestamp status) during first PBL request for the packet (in store and forward mode) or the last PBL request for the packet (in threshold mode).	
31:4	-	Reserved.	-

36.6.61 DMA mode register

The mode register establishes the transmit and receive operating modes and commands. This register should be the last CSR to be written as part of DMA initialization.

Table 838. DMA mode register (DMA_MODE, offset 0x1000) bit description

Bit	Symbol	Description	Reset value
0	SWR	Software reset. When this bit is set, the MAC and the OMA controller reset the logic and all internal registers of the OMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all Ethernet Block clock domains. Before reprogramming any Ethernet Block register, a value of 0 should be read in this bit. **Remark:** The reset operation is complete only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.	0
1	DA	DMA Tx or Rx arbitration scheme. This bit specifies the arbitration scheme between the transmit and receive paths of all channels: The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path. 0: Weighted round-robin with Rx:Tx or Tx:Rx 1: Fixed priority The priority between the paths is according to the priority specified in bits 14:12 and the priority weight is specified in the TXPR bit.	0
4:2	TAA	Transmit arbitration algorithm. This field is used to select the arbitration algorithm for the transmit side when multiple Tx DMAs are selected. 000: Fixed priority In fixed priority, channel 0 has the lowest priority and the channel 1has the highest priority. 001: Weighted strict priority (WSP) 010: Weighted round-robin (WRR) Others: Reserved	0
10:5	-	Reserved.	-
11	TXPR	Transmit priority. When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.	0
14:12	PR	Priority ratio. These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx: Rx depending on whether the TXPR bit is reset or set. 0x0: The priority ratio is 1:1 0x2: The priority ratio is 3: 1 0x3: The priority ratio is 4:1 0x4: The priority ratio is 5: 1 0x5: The priority ratio is 6: 1 0x6: The priority ratio is 7:1 0x7: The priority ratio is 8: 1	0
31:15	-	Reserved.	-

36.6.62 DMA system bus mode register

The system bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

Table 839. DMA system bus mode register (DMA_SYSBUS_MODE, offset 0x1004) bit description

Bit	Symbol	Description	Reset value	Access
0	FB	Fixed burst length. When this bit is set to 1, the AHB master will initiate burst transfers of specified length (INCRx or SINGLE). When this bit is set to 0, the AHB master will initiate transfers of unspecified length (INCR) or SINGLE transfers.	0	R/W
11:1	-	Reserved.	0	-
12	AAL	Address-aligned beats. When this bit is set to 1, the AHB master performs address-aligned burst transfers on read and write channels.	0	R/W
13	-	Reserved.	0	RO
14	MB	Mixed burst. When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).	0	R/W
15	RB	Rebuild INCRx burst. When this bit is set high and the AHB master gets SPLIT, RETRY, or EarlyBurst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLEtransfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.	0	R/W
31:16	-	Reserved.	-	-

36.6.63 DMA interrupt status register

The status register contains all the status bits that the DMA reports to the host. This register is usually read by the Software driver during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits in this register are not cleared when read. Writing 1 to (unreserved) bits in this register (bits 16:0) clears them and writing 0 has no effect. Each field (bits 16:0) can be masked by masking the appropriate bit in the DMA interrupt enable register Table 851.

This fields in this register can be read by the application (Read), can be set to 1 by the Ethernet core on a certain internal event (Self Set), and can be cleared to 0 by the application with a register write of 1 (Write Clear). A register write of 0 has no effect on the field.

Table 840. DMA interrupt status register (DMA_INTR_STAT, offset 0x1008) bit description

Bit	Symbol	Description	Reset value
0	DC0IS	DMA channel 0 interrupt status. This bit indicates an interrupt event in DMA channel 0. To reset this bit to 0, the software must read the corresponding register in DMA channel 0 to get the exact cause of the interrupt and clear its source.	0
1	DC1IS	DMA channel 1 interrupt status. This bit indicates an interrupt event in DMA channel 1. To reset this bit to 0, the software must read the corresponding register in DMA channel 1 to get the exact cause of the interrupt and clear its source.	0
15:2	-	Reserved.	-

Table 840. DMA interrupt status register (DMA_INTR_STAT, offset 0x1008) bit description *…continued*

Bit	Symbol	Description	Reset value
16	MTLIS	MTL interrupt status. This bit indicates an interrupt event in the MTL. To reset this bit to 0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.	0
17	MACIS	MAC interrupt status. This bit indicates an interrupt event in the MAC. To reset this bit to 0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.	0
31:18	-	Reserved.	-

36.6.64 DMA debug status register

The debug status register gives the receive and transmit process status for DMA channel 0 to channel 1 for debugging purposes.

Table 841. DMA debug status register (DMA_DBG_STAT, offset 0x100C) bit description

Bit	Symbol	Description	Reset value
0	AHSTS	AHB master status. When high, this bit indicates that the AHB master FSMs are in the non-idle state.	0
7:1	-	Reserved.	-
11:8	RPS0	DMA channel 0 receive process state. This field indicates the Rx DMA FSM state for channel 0: 0x0: Stopped (reset or stop receive command issued) 0x1: Running (fetching Rx Transfer) 0x2: Reserved 0x3: Running (waiting for Rx packet) 0x4: Suspended (Rx unavailable) 0x5: Running (closing the Rx) 0x6: Timestamp write state 0x7: Running (transferring the received packet data from the Rx buffer to the system memory) This field does not generate an interrupt.	0
15:12	TPS0	DMA channel 0 transmit process state. This field indicates the Tx DMA FSM state for channel 0: 000: Stopped (reset or stop transmit command issued) 0x1: Running (fetching Tx transfer) 0x2: Running (waiting for status) 0x3: Running (reading data from system memory buffer and queuing it to the Tx buffer (Tx FIFO)) 0x4: Timestamp write state 0x5: Reserved for future use 0x6: Suspended (Tx unavailable or Tx buffer underflow) 0x7: Running (closing Tx) This field does not generate an interrupt.	0

Table 841. DMA debug status register (DMA_DBG_STAT, offset 0x100C) bit description *…continued*

Bit	Symbol	Description	Reset value
19:16	RPS1	DMA channel 1 receive process state. This field indicates the Rx DMA FSM state for channel 1. This field is similar to the RPS0 field.	0
23:20	TPS1	DMA channel 1 transmit process state. This field indicates the Tx DMA FSM state for channel 1. This field is similar to the TPS0 field.	0
31:24	-	Reserved.	-

36.6.65 DMA channel control register

The DMA channel control register specifies the Maximum Segment Size (MSS) value for segmentation, length to skip between two s, and also the features such as header splitting and 8xPBL mode.

Table 842. DMA channel control register (DMA_CH0_CTRL, offset 0x1100 and DMA_CH1_CTRL, offset 0x1180) bit description

Bit	Symbol	Description	Reset value
15:0	-	Reserved.	-
16	PBLx8	8xPBL mode. When this bit is set, the PBL value programmed in bits 21:16 in DMA channel transmit control Table 843 is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.	0
17	-	Reserved.	-
20:18	DSL	Skip length. This bit specifies the word, dword, or lword number (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchaineds. The address skipping starts from the end of the current to the start of the next. When the DSL value is equal to 0, the DMA takes the table as contiguous.	
31:21	-	Reserved.	-

36.6.66 DMA channel transmit control register

The DMA channel transmit control register controls the Tx features such as PBL, TCP segmentation, and Tx channel weights.

Table 843. DMA channel transmit control register (DMA_CH0_TX_CTRL, offset 0x1104 and DMA_CH1_TX_CTRL, offset 0x1184) bit description

Bit	Symbol	Description	Reset value
0	ST	Start or stop transmission command. When this bit is set, transmission is placed in the running state. The DMA checks the transmit list at the current position for a packet to be transmitted. The DMA tries to acquire from either of the following positions: • The current position in the list This is the base address of the transmit list set by the DMA channel transmit descriptor list address register Table 845. • The position at which the transmission was previously stopped. If the DMA does not own the current, the transmission enters the suspended state and the TBU bit of the DMA channel status register Table 858 is set.The start transmission command is effective only when the transmission is stopped. If the command is issued before setting the DMA channel transmit descriptor list address register Table 845, the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the stopped state after completing the transmission of the current packet. The next position in the transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, DMA channel transmit descriptor list address register Table 845 needs to be programmed with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the suspended state.	0
3:1	TCW	Transmit channel weight. This field indicates the weight assigned to the corresponding transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.	0
4	OSF	Operate on second frame. When this bit is set, it instructs the DMA to process the second packet of the transmit data even before the status for the first packet is obtained.	0
15:5	-	Reserved.	-
21:16	TxPBL	Transmit programmable burst length. These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block read or write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the PBLx8 mode in DMA channel control register Table 842. 2. Set the PBL.	0
31:22	-	Reserved.	-

36.6.67 DMA channel receive control register

The DMA channel receive control register controls the Rx features such as PBL, buffer size, and extended status.

Table 844. DMA channel receive control register (DMA_CH0_RX_CTRL, offset 0x1108 and DMA_CH1_RX_CTRL, offset 0x1188) bit description

Bit	Symbol	Description	Reset value
0	SR	Start or stop receive. When this bit is set, the DMA tries to acquire the from the receive list and processes the incoming packets. The DMA tries to acquire from either of the following positions: • The current position in the list. This is the address set by the DMA channel transmit descriptor list address register Table 845. • The position at which the Rx process was previously stopped. If the DMA does not own the current, the reception is suspended and the RBU bit of the DMA channel status register Table 845 is set. The start receive command is effective only when the reception is stopped. If the command is issued before setting the DMA channel transmit descriptor list address register Table 845, the DMA behavior is unpredictable. When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next position in the receive list is saved, and it becomes the current position after the Rx process is restarted. The stop receive command is effective only when the Rx process is in the running (waiting for Rx packet) or suspended state.	0
2:1	-	Reserved.	-
14:3	RBSZ	Receive buffer size. This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. Note: The buffer size must be a multiple of 4, This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, it may result into undefined behavior. The LSB bits (1:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).	0
15	-	Reserved.	-

Table 844. DMA channel receive control register (DMA_CH0_RX_CTRL, offset 0x1108 and DMA_CH1_RX_CTRL, offset 0x1188) bit description *…continued*

Bit	Symbol	Description	Reset value
21:16	RxPBL	Receive programmable burst length. These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block read or write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the PBLx8 mode in the DMA channel control register Table 842. 2. Set the PBL.	
30:22	-	Reserved.	-
31	RPF	DMA Rx channel 0 packet flush. When this bit is set to 1, the DMA will automatically flush the packet from the Rx Queues destined to DMA Rx channel 0 when the DMA Rx channel 0 is stopped after a system bus error has occurred. The flushing happens on the read side of the Rx Queue. When this bit is set to 0 the Ethernet Core will not flush the packet in the Rx Queue destined to DMA Rx channel 0 after the DMA is stopped due to a system bus error.	0

36.6.68 DMA channel transmit descriptor list address register

The transmit list address register points to the start of the transmit List. The lists reside in the host's physical memory space and must be Word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low. Writing to this register is permitted only when transmission has stopped. When stopped, this register can be written before the transmission Start command is given.

Table 845. DMA transmit descriptor list address register (DMA_CH0_TXDESC_LIST_ADDR, offset 0x1114 and DMA_CH1_TXDESC_LIST_ADDR, offset 0x1194) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
31:2	STL	Start of transmit list. This field contains the base address of the first in the transmit list. The LSB bit 1 will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are read-only.	0

36.6.69 DMA receive descriptor list address register

The receive list address register points to the start of the receive list. The lists reside in the host's physical memory space and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive start command is given.

Table 846. DMA receive descriptor list address register (DMA_CH0_RXDESC_LIST_ADDR,offset 0x111C and DMA_CH1_RXDESC_LIST_ADDR,offset 0x119C) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
31:2	SRL	Start of receive list. This field contains the base address of the first descriptor in the receive list. The LSB bit 1 will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are read-only.	0

36.6.70 DMA channel transmit tail pointer register

The channel Tx tail pointer register points to an offset from the base and indicates the location of the last valid. When this register is read, it always returns zero.

Table 847. DMA channel transmit tail pointer register (DMA_CH0_TXDESC_TAIL_PTR, offset 0x1120 and DMA_CH1_TXDESC_TAIL_PTR, offset 0x11A0) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
31:2	TDTP	Transmit tail pointer. This field contains the tail pointer for the Tx ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers. The width of this field depends on the configuration:	0

36.6.71 DMA channel receive tail pointer register

The channel Rx tail pointer points to an offset from the base and indicates the location of the last valid.

Table 848. DMA channel receive tail pointer register (DMA_CH0_RXDESC_TAIL_PTR, offset 0x1128 and DMA_CH1_RXDESC_TAIL_PTR, offset 0x11A8) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved.	-
31:2	RDTP	Receive tail pointer. This field contains the tail pointer for the Rx ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.	0

36.6.72 DMA channel transmit ring length register

The Tx ring length register contains the length of the transmit ring.

Table 849. DMA channel transmit ring length register (DMA_CH0_TXDESC_RING_LEN, offset 0x112C and DMA_CH1_TXDESC_RING_LEN, offset 0x11AC) bit description

Bit	Symbol	Description	Reset value
9:0	TDRL	Transmit ring length. This field sets the maximum number of Tx descriptors in the circular ring. The maximum number of descriptors is limited to 1 K s. NXP recommends a minimum ring length of 4.	0
31:10	-	Reserved.	-

36.6.73 DMA channel receive ring length register

The channel Rx ring length register contains the length of the receive circular ring.

Table 850. DMA channel receive ring length register (DMA_CH0_RXDESC_RING_LEN, offset 0x1130 and DMA_CH1_RXDESC_RING_LEN, offset 0x11B0) bit description

Bit	Symbol	Description	Reset value
9:0	RDRL	Receive ring length. This register sets the maximum number of Rx descriptors in the circular ring. The maximum number of descriptors is limited to 1K descriptors.	0
31:10	-	Reserved.	-

36.6.74 DMA channel interrupt enable register

The channel interrupt enable register enables the interrupts reported by the status register.

Table 851. DMA interrupt enable register (DMA_CH0_INT_EN, offset 0x1134 and DMA_CH1_INT_EN, offset 0x11B4) bit description

Bit	Symbol	Description	Reset value
0	TIE	Transmit interrupt enable. When this bit is set with normal interrupt summary enable (bit 16 in this register), transmit interrupt is enabled. When this bit is reset, transmit interrupt is disabled.	0
1	TSE	Transmit stopped enable. When this bit is set with abnormal interrupt summary enable (bit 15 in this register), transmission stopped interrupt is enabled. When this bit is reset, transmission stopped interrupt is disabled.	0
2	TBUE	Transmit buffer unavailable enable. When this bit is set with normal interrupt summary enable (bit 16 in this register), transmit buffer unavailable interrupt is enabled. When this bit is reset, transmit buffer unavailable interrupt is disabled.	0
5:3	-	Reserved.	-
6	RIE	Receive interrupt enable. When this bit is set with abnormal interrupt summary enable (bit 16 in this register), receive interrupt is enabled. When this bit is reset, receive interrupt is disabled.	0
7	RBUE	Receive buffer unavailable enable. When this bit is set with abnormal interrupt summary enable (bit 15 in this register), receive buffer unavailable interrupt is enabled. When this bit is reset, the receive buffer unavailable interrupt is disabled.	0
8	RSE	Received stopped enable. When this bit is set with abnormal interrupt summary enable (bit 15 in this register), receive stopped interrupt is enabled. When this bit is reset, receive stopped interrupt is disabled.	0
9	RWTE	Receive watchdog timeout enable. When this bit is set with abnormal interrupt summary enable (bit 15 in this register), the receive watchdog timeout interrupt is enabled. When this bit is reset, receive watchdog timeout interrupt is disabled.	0
10	ETIE	Early transmit interrupt enable. When this bit is set with an abnormal interrupt summary enable (bit 15 in this register), early transmit interrupt is enabled. When this bit is reset, early transmit interrupt is disabled.	0
11	ERIE	Early receive interrupt enable. When this bit is set with normal interrupt summary enable (bit 16 in this register), early receive interrupt is enabled. When this bit is reset, early receive interrupt is disabled.	0
12	FBEE	Fatal bus error enable. When this bit is set with abnormal interrupt summary enable (bit 15 in this register), the fatal bus error interrupt is enabled. When this bit is reset, fatal bus error enable interrupt is disabled.	0
13	-	Reserved.	-

Table 851. DMA interrupt enable register (DMA_CH0_INT_EN, offset 0x1134 and DMA_CH1_INT_EN, offset 0x11B4) bit description *...continued*

Bit	Symbol	Description	Reset value
14	AIE	Abnormal interrupt summary enable. When this bit is set, an abnormal interrupt summary is enabled. When this bit is reset, an abnormal interrupt is disabled. This bit enables the following bits: DMA channel status register Table 858, bit 1: transmit process stopped DMA channel status register Table 858, bit 7: receiver buffer unavailable DMA channel status register Table 858, bit 8: receive process stopped DMA channel status register Table 858, bit 9: receive watchdog timeout DMA channel status register Table 858, bit 10: early transmit interrupt DMA channel status register Table 858, bit 13: fatal bus error	0
15	NIE	Normal interrupt summary enable. When this bit is set, a normal interrupt is enabled. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits: DMA channel status register Table 858, bit 0: transmit interrupt DMA channel status register Table 858, bit 2: transmit buffer unavailable DMA channel status register Table 858, bit 6: receive interrupt DMA channel status register Table 858, bit 14: early receive interrupt	0
31:16	-	Reserved.	-

36.6.75 DMA receive interrupt watchdog timer register

The receive interrupt watchdog timer register indicates the watchdog timeout for receive interrupt (RI) from the DMA. This register, when written with non-zero value, enables the watchdog timer for RI (bit 6 in the DMA channel status register Table 858).

Table 852. DMA receive interrupt watchdog timer register (DMA_CH_RX_INT_WDTIMER, CH0 offset 0x1138 and CH1, offset 0x11B8) bit description

Bit	Symbol	Description	Reset value
7:0	RIWT	Receive interrupt watchdog timer count. Indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the RxDMA completes the transfer of a frame for which the RI status bit is not set due to the setting in the corresponding descriptor RDES3[30]. When the watch-dog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when RI bit is set high due to automatic setting of RI as per RDES3[30] of any received frame.	0
31:8	-	Reserved.	-

36.6.76 DMA slot function control register

The slot function control and status register contains the control bits for slot function and the status for transmit path.

Table 853. DMA slot function control register (DMA_CH_SLOT_FUNC_CTRL_STAT, CH0 offset 0x113C,CH1 offset 0x11BC) bit description

Bit	Symbol	Description	Reset value	Access
0	ESC	Enable slot comparison. When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is: • equal to the reference slot number or, • ahead of the reference slot number by one slot When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.	0	R/W
1	ASC	Advance slot check. When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is • equal to the reference slot number given in the RSN field or, • ahead of the reference slot number by up to two slots This bit is applicable only when the ESC bit is set.	0	R/W
15:2	-	Reserved.	0	RO
19:16	RSN	Reference slot number. This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.	0	RO
31:20	-	Reserved.	-	-

36.6.77 DMA channel current host transmit register

The current host transmit descriptor register points to the start address of the current transmit read by the DMA.

Table 854. DMA current host transmit register (DMA_CH_CUR_HST_TXDES, CH0 offset 0x1144 and CH1 offset 0x11C4) bit description

Bit	Symbol	Description	Reset value
31:0	HTD	Host transmit descriptor address pointer. Cleared on reset. Pointer updated by DMA during transmit operation.	0

36.6.78 DMA channel current host receive register

The current host receive descriptor register points to the start address of the current receive read by the DMA.

Table 855. DMA current host receive register (DMA_CH_CUR_HST_RXDES, CH0 offset 0x114C and CH1 offset 0x11CC) bit description

Bit	Symbol	Description	Reset value
31:0	HRD	Host receive descriptor address pointer. Cleared on reset. Pointer updated by DMA during receive operation.	0

36.6.79 DMA channel current host transmit buffer address register

The current host transmit buffer address register points to the current transmit buffer address being read by the DMA.

Table 856. DMA current host transmit buffer address register (DMA_CH_CUR_HST_TXBUF, CH0 offset 0x1154 and CH1 offset 0x11D4) bit description

Bit	Symbol	Description	Reset value
31:0	HTB	Host transmit buffer address pointer. Cleared on reset. Pointer updated by DMA during operation.	0

36.6.80 DMA channel current host receive buffer address register

The current host receive buffer address register points to the current receive buffer address being read by the DMA.

Table 857. DMA Current host receive buffer address register (DMA_CH_CUR_HST_RXBUF, CH0 offset 0x115C and CH1 offset 0x11DC) bit description

Bit	Symbol	Description	Reset value
31:0	HRB	Host receive buffer address pointer. Cleared on reset. Pointer updated by DMA during operation.	0

36.6.81 DMA channel status register

The software driver (application) reads the status register during interrupt service routine or polling to determine the status of the DMA.

Table 858. DMA channel status register (DMA_CH_STAT, CH0 offset 0x1160 and CH1 offset 0x11E0) bit description

Bit	Symbol	Description	Reset value
0	TI	Transmit interrupt. This bit indicates that the packet transmission is complete. When transmission is complete, bit 31 of TDES1 is reset in the first, and the specific packet status information is updated in the descriptor.	0
1	TPS	Transmit process stopped. This bit is set when the transmission is stopped.	0
2	TBU	Transmit buffer unavailable. This bit indicates that the application owns the next descriptor in the transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSO field of the DMA channel debug status register Table 841 explains the transmit process state transitions. To resume processing the transmits, the application should do the following: 1. Change the ownership of the by setting bit 31 of TDES0. 2. Issue a transmit poll demand command. For ring mode, the application should advance the transmit tail pointer register of a channel.	0
5:3	-	Reserved.	-
6	RI	Receive interrupt. This bit indicates that the packet reception is complete. When packet reception is complete, bit 31 of RDES1 is reset in the last, and the specific packet status information is updated in the descriptor. The reception remains in the running state.	0

Table 858. DMA channel status register (DMA_CH_STAT, CH0 offset 0x1160 and CH1 offset 0x11E0) bit description

Bit	Symbol	Description	Reset value
7	RBU	Receive buffer unavailable. This bit indicates that the application owns the next in the receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx data, the application should change the ownership and issue a receive poll demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the receive tail pointer register of a channel. This bit is set only when the DMA owns the previous Rx.	0
8	RPS	Receive process stopped. This bit is asserted when the Rx process enters the stopped state.	0
9	RWT	Receive watchdog time out. This bit is asserted when a packet with length greater than 2,048 bytes(10,240 bytes when jumbo packet mode is enabled) is received.	0
10	ETI	Early transmit interrupt. This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO.	0
11	ERI	Early receive interrupt. This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit.	0
12	FBE	Fatal bus error. This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.	0
13	-	Reserved.	-
14	AIS	Abnormal interrupt summary. Abnormal interrupt summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the DMA channel interrupt enable register Table 851: Bit 1: transmit process stopped Bit 7: receive buffer unavailable Bit 8: receive process stopped Bit 10: early transmit interrupt Bit 12: fatal bus error Only unmasked bits affect the abnormal interrupt summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.	0

Table 858. DMA channel status register (DMA_CH_STAT, CH0 offset 0x1160 and CH1 offset 0x11E0) bit description

Bit	Symbol	Description	Reset value
15	NIS	Normal interrupt summary. Normal interrupt summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the DMA channel interrupt enable register Table 851: Bit 0: transmit interrupt Bit 2: transmit buffer unavailable Bit 6: receive interrupt Bit 11: early receive interrupt Only unmasked bits (interrupts for which interrupt enable is set in DMA channel interrupt enable register Table 851) affect the normal interrupt summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.	0
18:16	EB	DMA error bits. This field indicates the type of error that caused a bus error. For example, error response on the AHB 0 interface. Bit 18: • 1: error during data transfer by the DMA • 0: no error during data transfer by the DMA Bit 17 • 1: error during descriptor access • 0: error during data buffer access Bit 16: • 1: error during read transfer • 0: error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt. This field indicates the type of error that caused a bus error. For example, error response on the AHB 0 interface.	0
31:19	-	Reserved.	-

36.6.82 DMA channel miss frame count register

This register has the number of packet counter that got dropped by the DMA either because of bus error or programing RPF field in DMA_CH0_Rx_Control and DMA_CH1_Rx_Control registers.

Table 859. DMA channel miss frame count (DMA_CH0_MISS_FRAME_CNT, offset 0x116C and DMA_CH1_MISS_FRAME_CNT, offset 0x11EC) bit description

Bit	Symbol	Description	Reset value
10:0	MFC	Dropped packet counters. This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in DMA_CH0_Rx_Control register. The counter gets cleared when this register is read. Clears on read. Self-set to 1 on internal event.	0

Table 859. DMA channel miss frame count (DMA_CH0_MISS_FRAME_CNT, offset 0x116C and DMA_CH1_MISS_FRAME_CNT, offset 0x11EC) bit description *…continued*

Bit	Symbol	Description	Reset value
14:11	-	Reserved.	-
15	MFCO	Overflow status of the MFC counter. When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read. Clears on read. Self-set to 1 on internal event.	
31:16	-	Reserved.	-

36.7 Functional description

36.7.1 Power management block and low power modes

This section describes the power management (PMT) mechanisms supported by the MAC. PMT supports the reception of network (remote) wake-up frames and magic packet frames. PMT does not perform the clock gate function, but generates interrupts for wake-up frames and magic packets received by the MAC. The PMT block sits on the receiver path of the MAC and is enabled with remote wake-up frame enable and magic packet enable. These enables are in the PMT control and status register and are programmed by the application.

When the power-down mode is enabled in the PMT, then all received frames are dropped by the Ethernet core and they are not forwarded to the application. The Ethernet comes out of the power-down mode only when either a magic packet or a remote wake-up frame is received and the corresponding detection is enabled.

36.7.1.1 Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The Ethernet block supports the IEEE 802.3az-2010 for EEE. The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionality to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY. The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

36.7.1.1.1 Transmit path functions

In the transmit path, the software must set the LPIEN bit of the MAC LPI control status register Table 793 to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER(LST) field of MAC LPI timers control register Table 794. The PHY link status bit of the LPI control and status register Table 793 indicates the link status of the PHY.

Remark: According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the PHY must not stop the transmit clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts transmit enable (TX_EN).
2. Asserts transmit error (TX_ER)
3. Sets TXD[3:0] to 0x1 **Note:** The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

4. Updates the status (TLPIEN bit of MAC LPI control status register Table 793) and generates an interrupt. To bring the PHY out of the LPI state, that is, when the software resets the LPIEN bit, the MAC performs the following tasks:
   a. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern
   b. Starts the LPI TW TIMER
   c. Updates the LPI exit status (TLPIEX bit of the MAC LPI control status register Table 793) and generates an interrupt

   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the MAC LPI Timers control register Table 794.

36.7.1.1.2 Automated entry/exit of LPI mode in TX path

The MAC transmitter can be programmed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period of time or has a packet to transfer. These modes are enabled and controlled by MAC LPI control status register Table 793.

When LPITXA (bit 19) and LPITXEN (bit 16) of MAC LPI control status register Table 793 are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter will exit the LPI IDLE state and clear the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer. In addition to the above, when bit 20 (LPIATE) is also set, the MAC transmitter will enter LPI IDLE state only if the transmit path remains in idle state (no activity) for the time period indicated by the value in MAC LPI Entry Timer. In this mode also, the MAC transmitter will exit the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPITXEN bit is not cleared but remains active so that reentry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both LPIATE and LPITXA bits are cleared, you can directly control the entry and exit of LPI IDLE state by programming the LPITXEN bit.

36.7.1.1.3 Receive path functions

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[7:0] to 0x01.
3. The PHY de-asserts RX_DV. Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.
4. The MAC updates the RLPIEN bit of the MAC LPI control status register and immediately generates an interrupt.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the MAC LPI control status register and generates an interrupt immediately. The sideband signal (synchronous to Rx clock) is also asserted.

36.7.1.1.4 LPI Timers

The transmitter maintains the following two timers that are loaded with the respective values from the MAC_LPI_Timers_Control register:

- LPI LS Timer
  - The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. You can enable the monitoring of the LUD bit of the MAC_PHYIF_Control_Status register by setting the PLSEN bit of MAC_LPI_Control_Status register. The link status is indicated to the MAC by the LNKSTS bit of the MAC_PHYIF_Control_Status register or the value programmed by the software in the PLS bit of the MAC_LPI_Control_Status register. If the link status is not available in the MAC_PHYIF_Control_Status, the software should get the PHY link status by reading the PHY register and accordingly update the PLS bit. This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in bits 25:16 of the MAC_LPI_Timers_Control register. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1,023 ms.
- LPI TW Timer
  - The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in bits 15:0 of MAC register 53 (LPI timers control register). The terminal count of the timer is the value of resolved transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the

    MAC resumes its normal operation after the TW timer reaches the terminal count. The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bits wide. Therefore, the software can program up to 65,535 μs.

36.7.1.1.5 LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The low power interrupt is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the MAC LPI control status register Table 793.

36.7.1.2 Power Management and Wake-up

The power management (PMT) block supports the reception of magic packets as wake up packets. The power management (PMT) block also supports the reception of network (remote) wake-up packets as wake up packets.

36.7.1.2.1 Magic packet detection

The magic packet frame is based on a method that uses Advanced Micro Device's magic packet technology to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

Only magic packets that are addressed to the device or a broadcast address will be checked to determine whether they meet the wake-up requirements. Magic packets that pass the address filtering (unicast or broadcast) will be checked to determine whether they meet the remote wake-on-LAN data format of 6 bytes of all ones followed by a MAC address appearing 16 times.

The application enables magic packet wake-up by writing a 1 to bit 1 of the PMT control and status register. The PMT block constantly monitors each frame addressed to the node for a specific magic packet pattern. Each frame received is checked for a 0xFFFF FFFF FFFF pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the 0xFFFF FFFF FFFF pattern is scanned for again in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (0xFFFF FFFF FFFF). The device will also accept a multicast frame, as long as the 16 duplications of the MAC address are detected.

If the MAC address of a node is 0x0011 2233 4455, then the MAC scans for the data sequence:

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

...CRC

Magic packet detection is updated in the PMT control and status register for magic packet received. A PMT interrupt to the application triggers a read to the PMT CSR to determine whether a magic packet frame has been received. When you enable the power-down mode in the PMT block, the MAC drops all received packets and does not forward any packet to the MTL Rx FIFO or the application. The MAC comes out of the power-down mode only when a magic packet is received and the corresponding detection is enabled.

36.7.1.2.2 Remote wake-up frame registers

The register WKUPFMFILTER_REG, address offset (0x0C4), loads the wake-up frame filter register. To load values in a wake-up frame filter register, the entire register (WKUPFMFILTER_REG) must be written. The WKUPFMFILTER_REG register is loaded by sequentially loading the eight register values in address offset (0x0C4) for WKUPFMFILTER_REG0, WKUPFMFILTER_REG1,... WKUPFMFILTER_REG7, respectively. WKUPFMFILTER_REG is read in the same way.

Remark: The internal counter to access the appropriate WKUPFMFILTER_REG is incremented when lane 3 (or lane 0 in big-endian) is accessed by the CPU. This should be kept in mind if you are accessing these registers in byte or half-word mode.

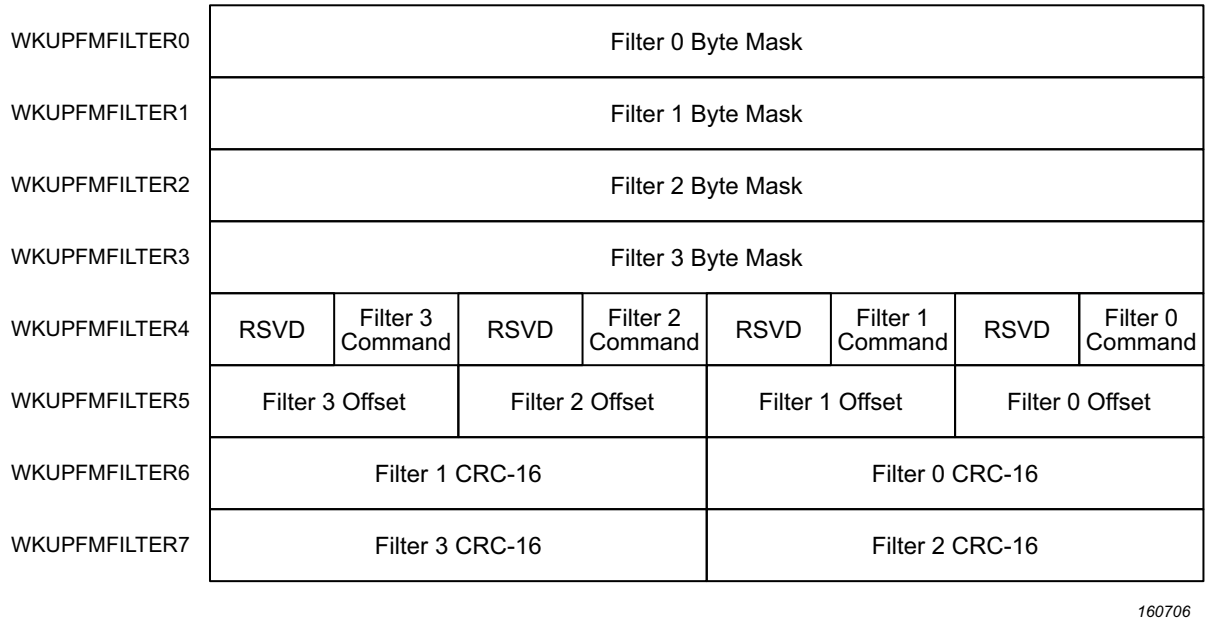


Fig 145. Wake-up frame filter register

Filter i byte mask

This register defines which bytes of the frame are examined by filter i (0, 1, 2, and 3) in order to determine whether or not the frame is a wake-up frame. The MSB (thirty-first bit) must be zero. Bit j [30:0] is the Byte Mask. If bit j (byte number) of the byte mask is set, then Filter i Offset + j of the incoming frame is processed by the CRC block; otherwise Filter i Offset + j is ignored.

Filter i command

This 4-bit command controls the filter i operation. Bit 3 specifies the address type, defining the pattern's destination address type. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and bit 1 are reserved. Bit 0 is the enable for filter i; if bit 0 is not set, filter i is disabled.

Filter i offset

This register defines the offset (within the frame) from which filter i examines the frames. This 8-bit pattern offset is the offset for the filter i first byte to be examined. The minimum allowed is 12, which refers to the 13th byte of the frame. The offset value 0 refers to the first byte of the frame.

Filter i CRC-16

This register contains the CRC_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

36.7.1.2.3 Remote wake-up detection

The power management (PMT) block supports the reception of network (remote) wake-up packets as wake up packets.

When the MAC is in sleep mode and the remote wake-up bit is enabled in PMT control and status register (0x00C0), normal operation is resumed after receiving a remote wake-up frame. The application writes all eight wake-up filter registers by performing a sequential write to address (0x00C4). The application enables remote wake-up by writing a 1 to bit 2 of the PMT control and status register.

PMT supports 16 programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of filter command, and if filter CRC-16 matches the incoming examined pattern, then the wake-up frame is received.

Filter_offset (minimum value 12, which refers to the 13th byte of the frame) determines the offset from which the frame is to be examined. Filter byte mask determines which bytes of the frame must be examined. The 31st bit of byte mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The wake-up frame is checked only for length error, FCS error, dribble bit error, MII error, collision, and to ensure that it is not a runt frame. Even if the wake-up frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. Wake-up frame detection is updated in the PMT control and status register for every remote wake-up frame received. A PMT interrupt to the application triggers a read to the PMT control and status register to determine reception of a wake-up frame.

36.7.1.2.4 PMT interrupts

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. When software resets the PWRDWN bit in remote wake-Up packet detection register, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

When you enable the power-down mode in the PMT block, the MAC drops all received packets and does not forward any packet to the MTL Rx FIFO or the application. The MAC comes out of the power-down mode only when a remote wake-up packet is received and the corresponding detection is enabled. The PMT block is available in the receive path of MAC.

36.7.1.3 System considerations during power-down

MAC neither gates nor stops clocks when power-down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with ENET_RX_CLK during power-down mode because it is involved in magic packet/wake-on-LAN frame detection. However, the transmit path and the application path clocks can be gated off during deep-sleep mode.

The PMT interrupt is asserted when a valid wake-up frame is received. This signal is generated in the receive clock domain.

The recommended power-down and wake-up sequence is as follows.

1. Disable the transmit DMA and wait for any previous frame transmissions to complete. These transmissions can be detected when transmit interrupt (see DMA channel status register bit TI Table 858) is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the MAC configuration register Table 777.

3. Wait until the receive DMA empties all the frames from the Rx FIFO.This can be done by reading the appropriate bits of Debug registers in the DMA and MTL CSR space.
4. Enable power-down mode by appropriately configuring the PMT registers.
5. Enable the MAC receiver and enter power-down mode.
6. Gate the application and transmit clock inputs to the core (and other relevant clocks in the system) to reduce power and enter sleep mode.
7. On receiving a valid wake-up frame, the MAC asserts the PMT interrupt signal and exits power-down mode.
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core.
9. Read the PMT status register to clear the interrupt, then enable the other modules in the system and resume normal operation.

36.7.2 Flow control

This section describes the flow control for transmit and receive paths.

36.7.2.1 Transmit flow control

The transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.To independently enable transmit flow control for each Rx Queue Set the TFE bit in corresponding MAC queue Tx flow control register. The transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

36.7.2.1.1 Flow control in full-duplex mode

In full-duplex mode, one of the following packet types are used for flow control:

- IEEE 802.3x pause packets.
- Priority flow control (PFC) packets.

The PFC packets are used only when the enable data center bridging option is selected. The PFCE bit of MAC_Rx_Flow_Ctrl register determines whether PFC packets or IEEE 802.3x pause control packets are used for flow control. If the PFCE bit is set, the MAC sends the PFC packets.

Table 860 describes the fields of a pause packet.

Table 860. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC	Control opcode Contains 0001 for IEEE 802.3x pause control packets; 0101 for PFC packets
PT	Contains pause time specified in the PT field of the MAC_Tx_FLOW_CTRL Q# register

When the FCB bit is set, the MAC generates and transmits a single pause packet. If the FCB bit is set again after the pause packet transmission is complete, the MAC sends another pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted pause packet, the application should program the pause time register with appropriate value and then again set the FCB bit. Similarly, when the mti_flowctrl_i signal is asserted, the MAC generates and transmits a single pause packet. If the mti_flowctrl_i signal remains asserted at a configurable number of slot times before the pause time runs out, the MAC transmits a second pause packet. This process is repeated as long as the mti_flowctrl_i signal remains active. If the mti_flowctrl_i signal goes inactive prior to the sampling time, the MAC transmits a pause packet with zero pause time (if the DZPQ bit in MAC_Q#_Tx_Flow_Ctrl register is set to 0) to indicate to the remote end that the receive buffer is ready to receive new data packets.

For PFC packets, you can specify the priority, pause time, and other controls for a queue in MAC_Q#_Tx_Flow_Ctrl register. The MAC supports independent flow control for each Rx Queue. There is one trigger input for each Rx Queue. Therefore, when multiple triggers come simultaneously, the MAC sends one PFC packet for each trigger. Based on the hardware trigger or software trigger through respective queues, the MAC sends a PFC packet with programmed pause time and priority. If multiple priorities are programmed in same Rx Queue, multiple priorities are set for the PFC packet with same pause time values. A separate pause timer is available for each Rx Queue.

36.7.2.1.2 Flow control in half-duplex mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 15-2 describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EFC bit of MTL_RxQ0_Operation_Mode register
- TFE bit of MAC_TX_FLOW_CTRL_Q0 register
- DM bit of the MAC configuration register flow control is similar for all queues.

Flow control is similar for all queues.

Table 861. Tx MAC flow control

EFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1.

Table 861. Tx MAC flow control

EFC	TFE	DM	Description
1	1	0	The MAC transmitter performs backpressure when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx performs backpressure when Rx Queue level crosses the threshold set by bits 10:8 of MTL_RxQ0_Operation_Mode register.
0	1	1	The MAC transmitter sends the pause packet when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1.
1	1	1	The MAC transmitter sends the pause packet when bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx sends a pause packet when Rx Queue level crosses the threshold set by bits 10:8 of MTL_RxQ0_Operation_Mode register.

36.7.2.2 Receive flow control

In the receive path, the flow control is functional only in the full-duplex mode. If any pause packet is received in the half-duplex mode, the packet is considered as a normal control packet.To enable the pause flow control Set the RFE bit in the MAC Rx flow control register.

36.7.3 Transmit and Receive FIFOs

The Transmit FIFO (Tx FIFO) buffers the data transferred from the host controller to the Ethernet block. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they can be transferred to the host controller. These are asynchronous FIFOs because they also transfer the data between the host controller clock and the MAC line clocks. Tx memory and Rx memory are required to be two-ported RAM of 35-bit width for 32 bit data bus widths, respectively. The extra bits are used for storing the byte-enable information.

When multiple queues are selected, all Tx Queue share the Tx FIFO memory and all Rx Queue share the Rx FIFO memory. The application can program the size of the FIFO memory allocated to each Tx or Rx Queue.

36.7.4 Multiple channels and queues support

The Ethernet block supports up to 2 queues and channels on Tx and Rx paths. The number of Tx channels are equal to the number of Tx Queues.

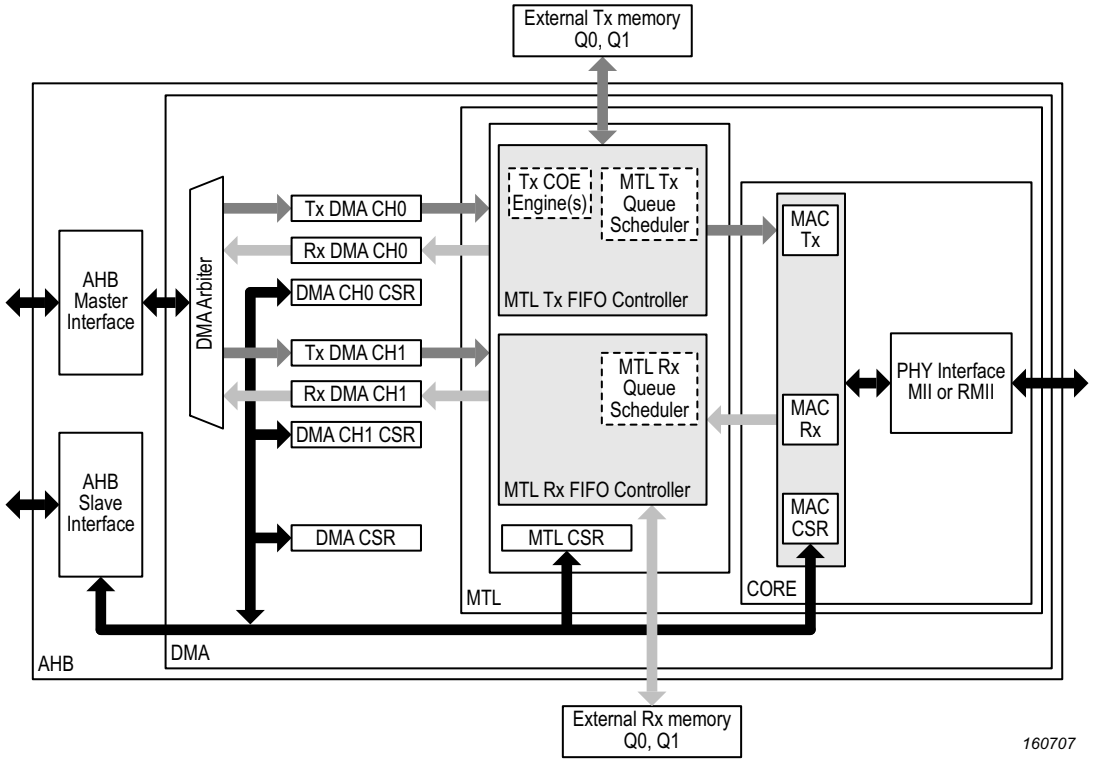


Fig 146. Multiple channels and queues

36.7.4.1 Support in the transmit path

Ethernet block supports up to two Tx Queues. The fixed priority scheme is the default priority scheme for the DMA channels. In fixed priority scheme, the channel with highest priority always wins the arbitration when it requests the bus. Channel 0 has lower priority than channel 1.In weighted strict priority (WSP), the weight corresponds to the number of burst transfers given to a channel in one arbitration cycle. The unused burst transfers of one or more channels are reallocated based on the priority. The channel with highest priority gets the unused burst transfer time before it is allocated to a channel with next highest priority.

In weighted round robin (WRR), all channels are serviced in round-robin order according to the weights settings.The TCW field of corresponding DMA channel transmit control register provides the weight for each transmit channel. The configured weights correspond to the number of burst transfers given to a channel in one arbitration cycle. The unused or excess burst transfers are distributed equally to all channels.

36.7.4.2 Support in the receive path

In the Rx direction, the MTL Rx controller selects the Rx DMA for which it is transferring or reading the data from the Rx FIFO memory. This scheduling is based on the programming done in the respective MTL RxQ control register.

Each Rx DMA indicates when it is ready to transfer data and the size of the burst-length (number of beats) that it has to transfer. The scheduler checks whether sufficient data (of requested burst length) is available to be transferred to these DMAs and then selects the Rx DMA that gets serviced using the programmed priorities.

36.7.4.3 Priority scheme for Tx DMA and Rx DMA

In the DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channels for accessing descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the DMA mode register Table 838 specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a channel.

If the Tx DMA and Rx DMA of a channel are enabled, user can specify which DMA gets the bus when the channel gets the control of the bus. User can set the priority between the corresponding Tx DMA and Rx DMA by using the TXPR field of the DMA mode register. For round-robin arbitration, user can use the PR field of the DMA mode register. Table 862 to specify the weighted priority between the Tx DMA and Rx DMA.

Table 862. Priority Scheme for Tx DMA and Rx DMA

Bit 14	Bit 13	Bit 12	Bit11	Bit 1	Priority Scheme
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	0	Rx has priority over Tx in ratio 2:1
0	1	0	0	0	Tx has priority over Rx in ratio 3:1
0	1	1	0	0	Tx has priority over Rx in ratio 4:1
1	0	0	0	0	Tx has priority over Rx in ratio 5:1
1	0	1	0	0	Tx has priority over Rx in ratio 6:1
1	1	0	0	0	Tx has priority over Rx in ratio 7:1
1	1	1	0	0	Tx has priority over Rx in ratio 8:1
x	x	x	1	1	Tx always has priority over Rx
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
0	0	1	1	0	Tx has priority over Rx in ratio 2:1
0	1	0	1	0	Tx has priority over Rx in ratio 3:1
0	1	1	1	0	Tx has priority over Rx in ratio 4:1
1	0	0	1	0	Tx has priority over Rx in ratio 5:1
1	0	1	1	0	Tx has priority over Rx in ratio 6:1
1	1	0	1	0	Tx has priority over Rx in ratio 7:1
1	1	1	1	0	Tx has priority over Rx in ratio 8:1

Remark: Note: Bits 14, 13, 12, 11, and 1 are not valid if one of the paths (transmit or receive) is not present in a channel. The transmit and receive paths are always present in channel 0.

36.7.4.4 Multiple queues and channels support in Ethernet MTL

The Ethernet block supports up to two Tx and Rx Queues. You can control a Tx Queue through corresponding MTL TxQ operation mode register. Also you can control an Rx Queue through corresponding MTL RxQ operation mode register.

36.7.4.5 Rx Queue to DMA mapping

The packets in the MTL Rx Queues can be routed to any one of the two DMA channels by programming the MTL RxQ DMA map registers for queues 0, 1. The following types of Rx Queue to DMA mapping is possible through programming:

1. Static mapping
2. Dynamic (per packet) mapping

36.7.4.5.1 Static mapping

The In this mode, all the packets of a Rx Queue is connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH (bits 3:0) and Q0DDMACH (bit 7 = 0) of the MTL RxQ DMA Map0 register. Similarly, packets from other Rx Queues can be routed to any DMA channel by programming register fields corresponding to each queue.

36.7.4.5.2 Dynamic (per packet) mapping

In this mode, the destination DMA channel of a packet being read from a Rx Queue is not constant but decided independently for each packet. For example, if you set the Q1DDMACH bit of the MTL RxQ DMA map register, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC Core receiver for each packet, based on Ethernet DA-based DMA Selection. The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding DCS field (when enabled) determines the destination DMA channel number. If above operations is unable to make a successful match/decision, then the packet is routed to DMA channel 0 by default.

36.7.4.6 Rx side routing from MAC to queues

The MAC routes the Rx packets to the Rx Queues based on following packet types:

1. AV control packets
2. Untagged IEEE 1588 PTP over Ethernet packets

The AV control Rx packets can be routed based on the Rx Queue number specified in the AVCPQ field in MAC RxQ Ctrl1 register and corresponding Rx Queue is enabled for AV through corresponding RXQEN field in MAC RxQ Ctrl0 register.
The untagged IEEE 1588 PTP over Ethernet Rx packets can be routed based on the Rx Queue number specified in the AVPTPQ field in MAC RxQ Ctrl1 register and the corresponding Rx Queue is enabled for AV through corresponding RXQEN field in MAC RxQ Ctrl0 register. This type of Rx packet routing is available when AV feature and multiple Rx Queues are selected in the configuration.
If the Rx packet cannot be classified in any of the defined packet type for routing, it will be routed through Rx Queue 0.

36.7.4.7 Rx side arbitration between DMA and MTL

After completing the current packet processing, the DMA channel controller fetches the next descriptor. After the descriptor fetching is complete, the DMA channel controller evaluates the amount of data to be to be transferred to the Rx buffer based on the programmed PBL and Rx buffer length. Accordingly, it requests the MTL to transfer the data.After servicing the current request, the MTL Rx Queue arbitration scheme selects Rx Queue based on the arbitration scheme (RAA field of the MTL operation mode register)

and the weights programmed in the corresponding queue receive control register. The arbitration is done among queues for which DMA is ready to service. After the Rx Queue is selected, PBL (Programmable Burst Length) amount of data is read out from that queue and is routed to the Rx DMA channel based on the Rx channel selection criteria.

The arbitration takes place when every PBL data transfer is completed and descriptors are ready for the processing from at least one DMA channel.

36.7.4.8 Tx side arbitration between DMA and MTL

Because the number of Tx DMA channels is always equal to the number of Tx Queues in the MTL, they are always mapped directly. For instance, each Tx DMA pushes data into its respective Tx Queue assigned to it. The main reason for this strategy is to optimize and increase the efficiency of the DMA data transfer, as well as to simplify the TX checksum engines in the MTL.

The data inside each Tx Queue is stored in packets. Hence, if two DMAs are allowed to transfer data into the same queue, when a Tx DM A starts a packet transfer, the other DMA cannot transfer data unless the previous packet is completely pushed-in. This means that the second DMA remains idle until the first packet is transferred. Hence, each DMA is always connected directly to its corresponding Tx Queue.

36.7.4.9 Audio Video Bridging

The Ethernet block supports the AV data transfer in 100 Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

1. IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
2. IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network.
3. IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

36.7.4.9.1 Transmit path functions

When the AV feature is selected, the transmit paths of Queue 0 and the highest enabled queue are enabled by default. The transmit path of Queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programmed threshold limit.

The transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

1. The queue contains one or more packets.
2. The credit for the queue is positive as per the algorithm.

The credit-based shaper algorithm can be disabled for all queues or lower-priority queues. For example, you can disable the credit-based shaper algorithm for Queue 1 and Queue 2 or only for Queue 1. You should not disable the credit-based shaper algorithm for Queue 2 and enable it for Queue 1. When you disable the credit-based shaper algorithm for a queue, the channel uses the default strict priority algorithm. Each transmit DMA has a separate descriptor chain for fetching the transmit data. The transmit channel that gets the access to the system bus depends on the DMA arbiter.

The transmit path has a shared FIFO (MTL layer) for each queue, as shown in Figure 146. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

1. If the packet is available in the channel and has a positive or zero credit.
2. If the higher priority queue has no packet waiting in the FIFO.

If the credit-based shaper algorithm is disabled for all queues, the packet to be transmitted from a queue is selected based on the priority scheme.

36.7.4.9.2 Receive path functions

When the AV feature is selected, the receive path of Queue 0 is enabled by default. All traffic is received on this channel. You can enable the receive paths of additional queues. By enabling the receive paths of multiple channels, you can demultiplex the received data to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN Priority tag value. This status is updated in the extended status field of the receive descriptor as explained in “receive descriptor”. All received packets with the Ether Type field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

1. AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits 15:8 in the transmit flow control register of the corresponding queue.
2. AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on Queue 0 by default. To receive these packets on any other queue, you can program bits 2:0 of the MAC Rx Queue control 1 register Table 785. Similar to the AV data packets, the tagged AV control packets are received based on the programmed priority value.

In addition to the AV packets, can be received the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on Queue 0. To receive these packets on any other queue, you need to program bits 6:4 of the MAC Rx Queue

control1 register Table 785.

36.7.4.9.3 Credit based shaper algorithm

The MTL queue scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in Queue 0. You can program the additional queues to use the credit-based shaper algorithm.The following sections provide information about how you can implement the credit-based shaper algorithm:

1. Credit value
2. idleSlopeCredit and sendSlopeCredit Values
3. Bandwidth status

Credit value: The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps. The credit to be added or subtracted per cycle can be fractional based on the required idleSlope and sendSlope values. See Table 863.

Table 863. Credit value Per transmit cycle example

Mode	Values	Description
100 Mbps	portTransmitRate = 100 Mbps	
	idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class)	credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted.
	sendSlope = 30 Mbps	credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA AHB application bus.

The credit for a queue builds up only when the packet is available but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet block supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. You can enable this mode with bit 1 of the CBS control registers of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit Values:

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1,024, such as, 2.8 x 1,024 = 2,867 and 1.2 x 1,024 = 1,229. In addition, the software must program the hiCredit and loCredit values, scaled by 1,024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and

loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12,000 x 1,024 bits and two's complement of 3,036 x1,024, respectively.

Bandwidth status:

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register gives the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in bits 6:4 of the CBS control register of the respective queue. For example, if you have programmed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, Queue 1 the credit-based shaper algorithm (with reserved bandwidth of 50%) and Queue 0 uses the strict-priority algorithm. If Queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by Queue 0.

36.7.4.9.4 Slot number function with Audio Video Bridging

In AV mode the slot number function to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.slot number can be program to a particular slot at which the DMA should fetch the data from system memory in the transmit descriptor Word 3 (TDES3). This 4-bit field allows the application to schedule data up to 16 slots of 125 µs each. This field is applicable only for the AV channels.When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally

generated reference slot interval. The slot interval is a counter that is updated every 125 µs of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state till there is a match.
To enable the DMA to fetch the data only if it matches the current slot or the next two slots, you can program bit 1 of the slot function control and status register of the corresponding DMA channel. You can enable the check for slot number by setting bit 0 of the slot function control and status register of corresponding DMA channel. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, bits 19:16 indicate the value of the reference slot number in DMA.

Remark: Remark: If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

Check the following sections for AV programming:

Programming guidelines for initializing the DMA Section 36.7.11.9.1.

Programming guidelines for enabling slot number checking Section 36.7.11.9.2.

Programming guidelines for enabling average bits per slot reporting Section 36.7.11.9.3.

36.7.4.10 Queue modes

You can enable a Tx Queue for generic, AV, or all types of traffic. When you select multiple Tx Queues, the queues are enabled for generic queuing using WRR or WSP algorithms. The queuing is based on the CBS or SP algorithms.You can enable an Rx Queue for generic, AV, or all types of traffic. A particular queue enabled for generic or AV based routing is determined by the RXQEN field of corresponding queue. When you select the enable audio video bridging option, AV is enabled for all selected Rx Queues.

The AV control packets (tagged or untagged) are routed based on the AVCPQ field of the MAC Rx Queue control 1 register Table 785. The PTP over Ethernet packets are routed based on the AVPTPQ field of MAC Rx Queue control 1 register Table 785. You can program the priority of a Tx Queue in the MAC TxQ Priority Map register. See Table 784. You can program the priority of an Rx Queue in bits 15:8 of corresponding Rx flow control register. See Table 787. The priority should be assigned in the following order:

1. AV queue (high priority)
2. Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programmed queue on Tx side. The MAC uses the programmed priorities for blocking the Tx Queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

36.7.5 TCP/IP offloading

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has a checksum offload engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path. By using the Checksum Offload features, the software can offload the task of checksum insertion and checking to the hardware. This feature helps in improving the throughput.

In the transmit path MAC calculates the checksum and inserts it in the Tx packet. In the receive path, MAC verifies the received checksum with the internally calculated checksum and provides the status.

36.7.5.1 Transmit checksum offload engine

The MAC has a checksum offload engine (COE) to support checksum calculation and insertion in the transmit path, using which, the software can offload the task of checksum insertion to the hardware. In the transmit path MAC calculates the checksum and inserts it in the Tx packet. This feature helps in improving the throughput.The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits 17:16).

The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field.It must be made sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid dead-lock. When reading starts, the COE fails and consequently all succeeding packets may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the packets that are less than the following number of bytes in size:

TXFIFO_SIZE – ((PBL + 6)*(DATAWIDTH/8))

The PBL is the programmed burst length in DMA Bus mode register.

36.7.5.1.1 IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit header checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the type field of Ethernet packet has the value 0x0800 and the version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

36.7.5.1.2 TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP checksum engine processes the IPv4 or IPv6 header and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the checksum field of the input packet. This engine includes the checksum field in the checksum calculation, and then replaces the checksum field with the final calculated checksum.
- The engine ignores the checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

The result of this operation is indicated by the payload checksum error status bit in the transmit status vector (bit 12 in Table 881). This engine sets the payload checksum error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store and forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the payload length field in the IP header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 864. Transmit checksum offload engine functions for different frame types

Frame type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 frame	No	No
IPv4 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K frames is enabled in MAC) but less than or equal to the frame size constraint specified in 36.7.5.1	Yes	Yes
IPv6 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K frames is enabled in MAC) but less than or equal to the frame size constraint specified in 36.7.5.1	N/A	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 frame has IP options (IP header is longer than 20 bytes)	Yes	Yes
Frame is an IPv4 fragment	Yes	No
IPv6 frame with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 main header) • Hop-by-hop options (in IPv6 extension header) • Destinations options • Routing (with segment left 0) • Routing (with segment left > 0) • TCP, UDP, or ICMP • Authentication • Any other next header field in main or extension headers	N/A	 Yes No Yes No No Yes Yes No
IPv4 frame has TCP header with Options fields.	Yes	Yes
IPv4 Tunnels: • IPv4 frame in an IPv4 tunnel • IPv6 frame in an IPv4 tunnel	 • Yes (IPv4 tunnel header) • Yes (IPv4 tunnel header)	 • No • No
IPv4 frame has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 frame has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	N/A	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	N/A	No

36.7.5.2 Receive checksum offload engine

The MAC has a checksum offload engine (COE) to support packet error detection in the receive path. In the receive path, MAC verifies the received checksum with the internally calculated checksum and provides the status. Here, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP header error bit is set for any mismatch between the indicated payload type (Ethernet type field) and the IP header

version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header). This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a payload checksum error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 865. Receive checksum offload engine functions for different frame types

Frame type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6	No	No
IPv4 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 frame is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K frames is enabled in the MAC)	N/A	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 frame has IP options (IP header is longer than 20 bytes)	Yes	Yes
Frame is an IPv4 fragment	Yes	No
IPv6 frame with the following next header fields in main or extension headers: Hop-by-hop options (in IPv6 main header) Hop-by-hop options (in IPv6 extension header) Destinations options Routing (with segment left 0) Routing (with segment left > 0) TCP, UDP, or ICMP Any other next header field in main or extension headers	N/A	 Yes No Yes Yes No Yes No
IPv4 frame has TCP header with Options fields.	Yes	Yes
IPv4 Tunnels: • IPv4 frame in an IPv4 tunnel • IPv6 frame in an IPv4 tunnel	 • Yes (IPv4 tunnel header) • Yes (IPv4 tunnel header)	 • No • No
IPv6 Tunnels: • IPv4 frame in an IPv6 tunnel • IPv6 frame in an IPv6 tunnel	N/A	 • No • No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	N/A	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	N/A	No

36.7.6 Loopback mode

The MAC supports loopback of transmitted packets to its receiver. To enable this feature, program the LM bit of the MAC configuration register.Lopback mode can be enabled for all PHY interfaces. The data is always looped back on the MII interface irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding interface block.

36.7.6.1 Guidelines for using loopback mode

The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip (for example, in FPGA setup), the Tx and Rx clocks should be externally generated and should provide these clocks to the MAC.
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO.

The transmit and receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the PHY transmit data to the receive path.

The depth of FIFO is nine. The FIFO is free-running to write on the write clock and read on every read clock.

At the start of each packet read out of the FIFO, the write and read pointers get re-initialized to have an offset of 4. This avoids overflow or underflow during a packet transfer. This also ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO. At the end of every received packet, the receive protocol engine module generates received packet status and sends it to the receive packet controller module. The control, missed packet, and filter fail status are added to the receive status in the receive packet controller module.The MAC does not process ARP or PMT packets that are looped back.

36.7.7 PHY interfaces

36.7.7.1 Station management agent

The host controller can access the PHY registers through the Station Management Agent (SMA) module. SMA is a two-wire station management interface (MIM).

SMA module supports accessing up to 32 PHYs. The host controller can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module.

36.7.8 IEEE 1588 timestamps

The IEEE 1588 standard defines a protocol, Precision Time Protocol (PTP), that enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The Ethernet block supports both the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2) standards. The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588-2008 supports PTP transported over Ethernet. The Ethernet provides programmable support for both standards. It supports the following features:

- Supports both timestamp formats.
- Provides an option to take snapshot of all packets or only PTP type packets.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent.
- Provides an option to select the node to be a master or slave for ordinary and boundary clock.
- Identifies the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status.
- Provides an option to measure sub-second time in digital or binary format.

36.7.8.1 Clock types

The Ethernet controller supports the following clock types defined in the IEEE 1588-2008 standard:

- Ordinary clock
- Boundary clock
- End-to-end transparent clock
- Peer-to-peer transparent clock

36.7.8.1.1 Ordinary clock

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecommunications applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. The ordinary clock supports the following features:

- Send and receives PTP messages. The timestamp snapshot can be controlled as described in Table 806.
- Maintains the data sets such as timestamp values.

Table 866. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either one of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the control bit (TSVER2ENA) and selecting the snapshot mode in Table 806.

36.7.8.1.2 Boundary clock

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and are not forwarded. The PTP message type status given by the core (see.Section 36.7.8.7) helps you to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock.
- The local clock is common to all ports of the boundary clock.

Therefore, the features of the ordinary clock are also applicable to the boundary clock.

36.7.8.1.3 End-to-end transparent clock

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet inside the end-to-end transparent clock is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in Table 867.

You can take the snapshot by setting the snapshot select bits (SNAPTYPSEL) to 10 in Table 806.

Table 867. End-to-end transparent clock: PTP messages for which a snapshot is taken for transparent clock implementation

SYNC
FOLLOW_UP

36.7.8.1.4 Peer-to-peer transparent clock support

In this type of clock the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up messages with the link peer. Hence support for taking snapshot for the event messages related to Pdelay is added.

Table 868. End-to-end transparent clock: PTP messages for which a snapshot is taken for transparent clock implementation

SYNC
Pdelay_Req
Pdelay_Resp

The transparent clock corrects only the sync and follow-up message. As discussed earlier this can be achieved using the message status provided.

The type of clock to be implemented will be configurable through control register (see Table 806. To ensure that the snapshot is taken only for the messages indicated in the table for the corresponding clock type, the TSEVNTENA (enable timestamp snapshot for event messages) bit has to be set.

36.7.8.2 Delay request-response mechanism

The system or network is classified into master and slave nodes for distributing the timing/clock information. Figure 147 shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

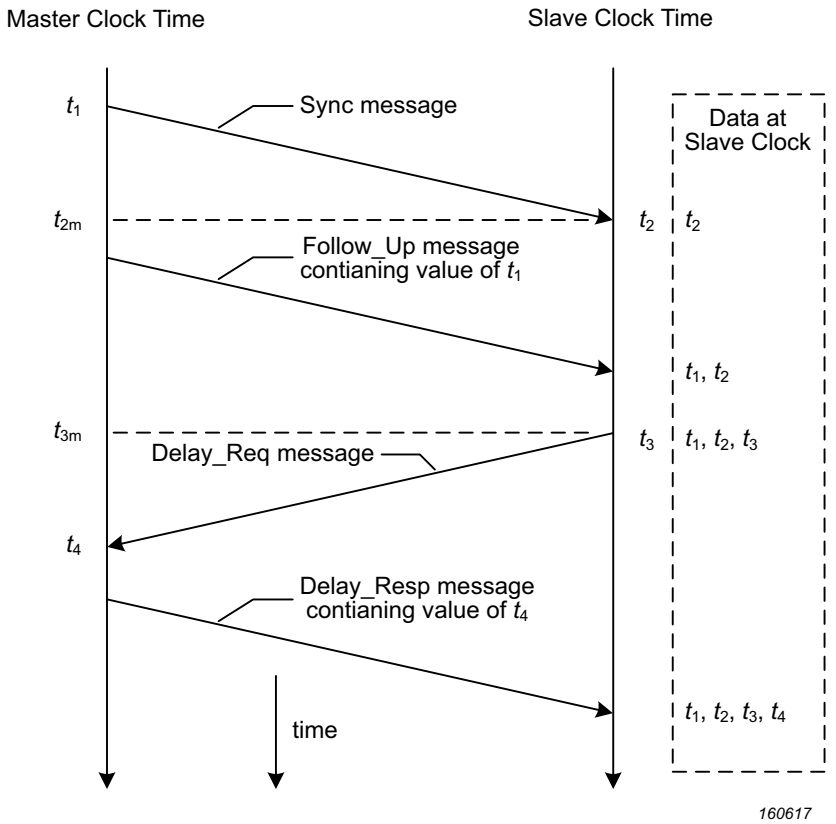


Fig 147. Networked time synchronization

The PTP uses the following process:

1. The master broadcasts the PTP sync messages to all its nodes. The sync message contains the master.s reference time information. The time at which this message leaves the master.s system is t1. This time must be captured, for Ethernet ports, at MII.
2. The slave receives the sync message and also captures the exact time, t2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t1 information for later use.
4. The slave sends a Delay_Req message to the master, noting the exact time, t3, at which this frame leaves the MII.
5. The master receives the message, capturing the exact time, t4, at which it enters its system.
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

36.7.8.3 Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports peer-to-peer PTP (Pdelay) message in addition to sync, delay request, follow-up, and delay response messages. Figure 148 shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

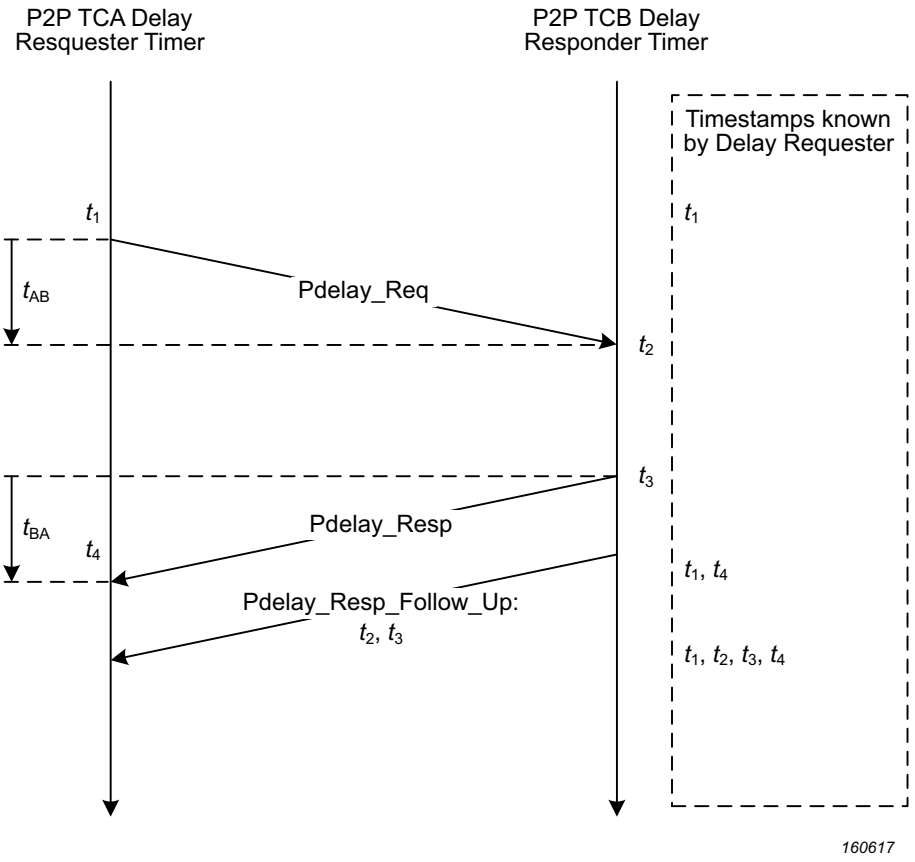


Fig 148. Propagation delay calculation in clocks supporting peer-to-peer path correction

The propagation delay is calculated in the following way:

1. Port-1 issues a Pdelay_Req message and generates a timestamp, t1, for the Pdelay_Req message.
2. Port-2 receives the Pdelay_Req message and generates a timestamp, t2, for this message.
3. Port-2 returns a Pdelay_Resp message and generates a timestamp, t3, for this message.

   To minimize errors because of any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The Port-2 returns any one of the following:

   - The difference between the timestamps t2 and t3 in the Pdelay_Resp message.
   - The difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message.
   - The timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.
4. Port-1 generates a timestamp, t4, on receiving the Pdelay_Resp message.
5. Port-1 uses all four timestamps to compute the mean link delay.

36.7.8.4 Frequency range of the reference timing clock

The timestamp information is transferred across asynchronous clock domains, that is, from MAC clock domain to application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of II and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes. Because the MII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends upon the operating mode and operating speed of the MAC as shown in Table 4-1.

Table 869. Minimum PTP clock frequency cycle

Mode	Minimum gap between two SFDs	Minimum PTP frequency
100-Mbps full-duplex operation	+ 24 clocks of min IFG + 16 clocks of preamble) 168 MII clocks (128 clocks for a 64-byte frame	(3 * PTP) + (4 * MII) ≤ 168 * MII that is, ~0.5 MHz ((168 – 4) * 40 ns ÷ 3 = 2,180 ns period)

36.7.8.5 PTP processing and control

Table 870 shows the common message header for the PTP messages. This format is taken from IEEE standard 1588-2008 (Revision of IEEE Std. 1588-2002).

Table 870. Message format defined in IEEE 1588-2008

Bits									
7	6	5	4	3	2	1	0	OCTETS	OFFSET
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField ([1])								1	32
logMessageInterva								1	33

[1] Field is used in version 1. In version 2, messageType field is used for detecting different message types.

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP frames:

- PTP Frames Over IPv4

- PTP Frames Over IPv6
- PTP Frames Over Ethernet

36.7.8.5.1 PTP frames over IPv4

Table 4-6 provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in Table 867.

Table 871. IPv4-UDP PTP frame fields required for control and status

Field matched	Octet position	Matched value	Description
MAC Frame Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed. 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-Primary multicast address: 224.0.1.129 PTP-Pdelay multicast address: 224.0.0.107
UDP Destination Port	36, 37	0x013F, 0x0140	0x013F – PTP event message ([1]) 0x0140 – PTP general messages
PTP control Field (IEEE version 1)	74	0x00/0x01/0x02/0x03/0x04	0x00 – SYNC, 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xB/ 0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

[1] PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only).

36.7.8.5.2 PTP Frames Over IPv6

Table 872 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in Table 867.

Table 872. IPv6-UDP PTP frame fields required for control and status

Field Matched	Octet Position	Matched Value	Description
MAC Frame Type	12, 13	0x86DD	IP datagram
IP version	14(bits 7:4)	0x6	IP version is IPv6
Layer 4 Protocol	20 ([1])	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP – Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP – Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57 ([1])	0x013F, 0x140	0x013F – PTP event message 0x0140 – PTP general messages
PTP Control Field (IEEE version 1)	93 ([1])	0x00/0x01/0x02/ 0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management (version1)
PTP Message Type Field (IEEE version 2)	74 ([1]) (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/ 0xB/0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	75 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

[1] The Extension Header is not defined for PTP packets.

36.7.8.5.3 PTP frames over Ethernet

Table 4-8 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of the IEEE 1588-2008 standard and the message format defined in Table 870.

Table 873. Ethernet PTP frame fields required for control and status

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address[1]	0-5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses[2]: 01-1B-19-00-00-00 01-80-C2-00-00-0E[3]
MAC Frame Type	12, 13	0x88F7	PTP Ethernet frame

Table 873. Ethernet PTP frame fields required for control and status

Field Matched	Octet Position	Matched Value	Description
PTP Control Field (IEEE version 1)	45	0x00/0x01/0x02/ 0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0/0x1/0x2/0x3/0x8/ 0x9/0xB/0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	14 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

[1] The address match of destination addresses (DA) programmed in MAC address 1 to 31 is used if the control bit 18 (TSENMACADDR: enable MAC address for PTP frame filtering) of the timestamp control register is set.

[2] IEEE standard 1588-2008, Annex F

[3] The Ethernet controller does not consider the PTP version 1 messages with peer delay multicast address (01-80-C2-00-00- 0E) as valid PTP messages.

36.7.8.6 Transmit path functions

The MAC captures a timestamp when the start frame delimiter (SFD) of a frame is sent on MII. The frames for which you want to capture timestamps are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether a timestamp should be captured for that frame.

The MAC does not process the transmitted frames to identify the PTP frames. You need to specify the frames for which you want to capture timestamps.

Use the control bits in the transmit descriptor (see Section 36.7.10.3). The MAC returns the timestamp to the software inside the corresponding transmit descriptor, thus connecting the timestamp automatically to the specific PTP frame. The 64-bit timestamp information is written to the TDES6 and TDES7 fields. The TDES7 field holds the 32 least significant bits of the timestamp.

36.7.8.7 Receive path functions

The MAC captures the timestamp of all frames received on the MII. The MAC does not process the received frames to identify the PTP frames in the default mode, that is, when the advanced timestamp feature is not selected.

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. You can control the snapshot of the time to be sent to the application by using the following options of the MAC_Timestamp_Control register.

- Enable snapshot for all packets.

- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp.
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet.
- Enable timestamp snapshot for the received packet for IPv4 or IPv6.
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP).
- Enable the node to be a master or slave and select the snapshot type This feature controls the type of messages for which snapshots are taken.

The DMA returns the timestamp to the software inside the corresponding receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in normal descriptor RDES1 and the snapshot of the timestamp is written in RDES0 and RDES1 fields of context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

36.7.8.8 IEEE 1588-2008 advanced timestamps

In addition to the basic timestamp features (see Section 36.7.8) the Ethernet controller supports the following advanced timestamp features defined in the IEEE 1588-2008 standard:

- Supports the IEEE 1588-2008 (version 2) timestamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end, and peer-to-peer.
- Provides an option to select the node to be a master or slave for ordinary and boundary clock.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to measure sub-second time in digital or binary format.

36.7.8.8.1 Reference timing source

The Ethernet controller supports the 48-bit seconds field reference timing sources featured in the IEEE 1588-2008 standard.

48-bit seconds field: The Ethernet controller supports 80-bit timestamping. The timestamp has the following fields:

- UInteger48 secondsField

  The seconds field is the integer portion of the timestamp in units of seconds and is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000 0000 0002.

- UInteger32 nanosecondsField

  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000 0001.

  The nanoseconds field supports the following two modes:

- Digital rollover mode: In digital rollover mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.
- Binary rollover mode: In binary rollover mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

You can set these modes by using the bit 9 (TSCTRLSSR) of Table 806.

When you select the advanced timestamp feature, the timestamp maintained in the core is still 64-bit wide. The overflow to the upper 16-bits of seconds register happens once in 130 years. You can read the values of the upper 16-bits of the seconds field from the CSR register.

36.7.8.9 System time register module

The 64-bit time is maintained in this module and updated using the input reference clock. This time is the source for taking snapshots (timestamps) of Ethernet frames being transmitted or received at the MII.

The system time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the timestamp update register (Table 806). For initialization, the system time counter is written with the value in the timestamp update registers, while for system time correction, the offset value is added to or subtracted from the system time.In the fine correction method, a slave clock's frequency drift with respect to the master clock (as defined in IEEE 1588) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP sync message intervals. In this method, an accumulator sums up the contents of the addend register, as shown in Figure 4-2. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

This algorithm is shown in Figure 149:

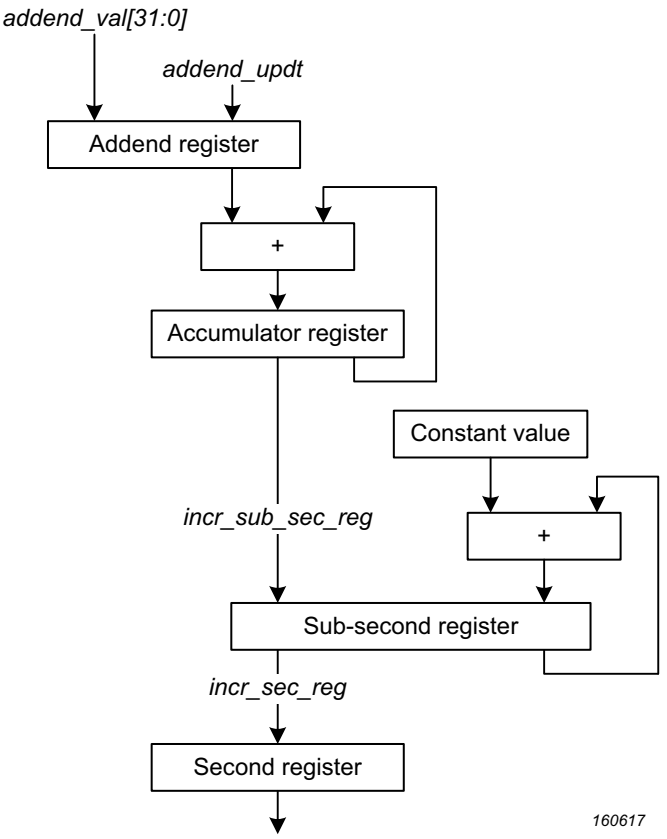


Fig 149. System update using fine method

The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock is, for example, 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Hence, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, to 65 MHz for example, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4. If the clock drifts higher, to 67 MHz for example, the addend register must be set to 0xBF0B 7672. When the clock drift is nil, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In Figure 149, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the sync messages and update the addend register accordingly.

Initially, the slave clock is set with FreqCompensationValue0 in the addend register. This value is:

FreqCompensationValue0 = 2^{32} / FreqDivisionRatio

The frequency division (FreqDivisionRatio) is the ratio of the reference clock frequency to the required clock frequency.

If MasterToSlaveDelay is initially assumed to be the same for consecutive sync messages, the algorithm described below must be applied. After a few sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time $MasterSyncTime_n$ the master sends the slave clock a sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as:

  $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$
- The master clock count for current sync cycle, $MasterClockCount_n$ is given by:

  $MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for sync cycles n and n - 1)
- The slave clock count for current sync cycle, $SlaveClockCount_n$ is given by:

  $SlaveClockCount_n = SlaveClockTimen_n - SlaveClockTime_{n-1}$
- The difference between master and slave clock counts for current sync cycle, $ClockDiffCount_n$ is given by:

  $ClockDiffCount_n = MasterClockCount_n - SlaveClockCount_n$
- The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is given by:

  $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$
- The frequency compensation value for addend register, $FreqCompensationValue_n$ is given by:

  $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions.

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more sync cycles.

36.7.9 DMA controller description

The DMA has independent transmit and receive engines and a CSR space. The transmit engine transfers data from system memory to the device port (MTL), while the receive engine transfers data from the device port to the system memory. The DMA controller uses descriptors to efficiently move data from source to destination with minimal host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the host CPU for situations such as frame transmit and receive transfer completion, and other normal/error conditions.

The DMA and the host driver communicate through two data structures:

- Control and status registers (CSR). See Section 36.6.
- Lists and data buffers. See Section 36.7.10.

The DMA transfers data frames received by the MAC to the receive buffer in the system memory, and transmit data frames from the transmit buffer in the system memory.

Descriptors that reside in the system memory act as pointers to these buffers. There are two descriptor lists; one for reception, and one for transmission.The DMA supports up to two Tx and two Rx descriptor lists (or DMA channels). The base address of each list is written to the respective Tx descriptor list address register(Table 845) and Rx descriptor list address register(Table 846). The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. This offset can be controlled by the DSL field of DMA_CH0_CTRL/DMA_CH1_CTRL registers. The number of descriptors in the list is programmed in the respective Tx (or Rx) descriptor ring length register. Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the list address register to create a descriptor ring. The descriptors lists resides in the host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptors. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when end-of-frame is detected. data chaining can be enabled or disabled.

Figure 150 shows the ring structure.

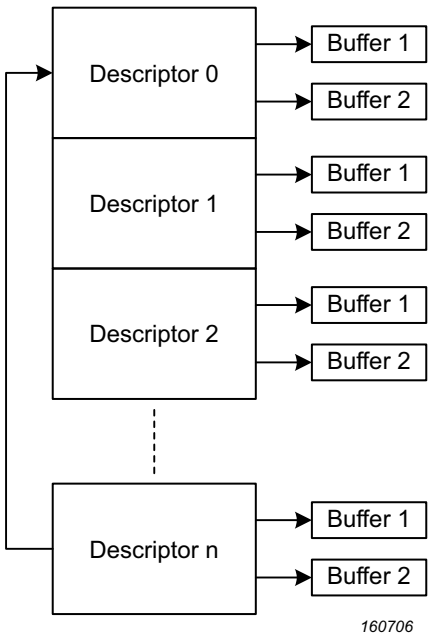


Fig 150. Ring structure

The transmit and receive engines enter the running state and attempt to acquire descriptors from the respective descriptor lists. The receive and transmit engines then begin processing receive and transmit operations. The transmit and receive processes are independent of each other and can be started or stopped separately.

36.7.9.1 Host bus burst access

The DMA attempts to execute fixed-length Burst transfers on the AHB master interface if configured to do so (FB bit of DMA system bus mode register 0). The maximum burst length is indicated and limited by the PBL field (DMA register 0[13:8]). The receive and transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 x 8/bus width) burst-size for the 16-bytes to be read.

The transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise (no fixed-length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL Receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the Receive FIFO. The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the AHB interface, then dummy transfers are performed in order to complete the fixed-burst. Otherwise (FB bit of DMA_ SYSBUS_MODE register Table 839 is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16), because the AHB interface does not support more than INCR16.

36.7.9.2 Host data buffer alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example: buffer read

If the transmit buffer address is 0x0000 0FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA reads five full words from address 0x0000 0FF0, but when transferring data to the MTL Tx Queue, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the MTL Transmit FIFO, unless it is the end-of-frame.

Example: buffer write

If the receive buffer address is 0x0000 0FF2 (for 64-bit data bus) and 16 bytes of a received frame need to be transferred, then the DMA writes 3 full words from address 0x0000 0FF0. But the first 2 bytes of first transfer and the last 6 bytes of the third transfer have dummy data.

36.7.9.3 Buffer size calculations

The DMA does not update the size fields in the transmit and receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES2) towards the MAC core. If a descriptor is marked as first (FS bit of TDES2 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES2), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. When the FS bit of a descriptor is set, the amount of valid data in a buffer is accurately indicated by the buffer size field (programmed in DMA channel receive control register) minus the data buffer pointer offset. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, the buffer may not be full (as indicated by the buffer size in bits 14:1 of receive control

register). To compute the amount of valid data in this final buffer, the driver must read the packet length (FL bits of RDES3[14:0]) and subtract the sum of the buffer sizes of the preceding buffers in this packet. The Rx DMA always transfers the start of next packet with a new descriptor.

Remark: Even when the start address of a receive buffer is not aligned to the system bus's data width, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the receive descriptor to have a 0x1002 offset. The receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset.

36.7.9.4 DMA arbiter

The arbiter inside the DMA module performs the arbitration between the transmit and receive channel accesses to the AHB host interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (DA bit of DMA mode register in Table 838 is reset), the arbiter allocates the data bus in the ratio set by the PR bits of DMA mode register Table 838, when both transmit and receive DMAs are requesting for access simultaneously. When the DA bit is set, the receive DMA always gets priority over the transmit DMA for data access by default. When the TXPR bit (bit 11 of DMA mode register, Table 838) is also set, then the transmit DMA gets priority over the receive DMA. See Table 838.

36.7.9.5 Transmission

36.7.9.5.1 TxDMA operation: Default (non-OSF) mode

The transmit DMA engine in default mode proceeds as follows:

1. The host sets up the transmit descriptor (TDES0-TDES3) and sets the OWN bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet frame data.
2. The application advances the descriptor tail pointer offset value of the transmit channel.
3. While in the run state, the DMA runs an arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended and bit 2 and bit 16 of status register of corresponding DMA channel are set and the Tx engine proceeds to step 11.
   - The descriptor is flagged as owned by the application (TDES3[31] = 0).
   - The descriptor tail pointer is equal to the current descriptor pointer in ring descriptor list mode.
   - An error condition occurs.
6. If the acquired descriptor is flagged as owned by DMA (TDES3[31] = 1), the DMA decodes the transmit data buffer address from the acquired descriptor.
7. The DMA fetches the transmit data from the host memory and transfers the data to the MTL for transmission.
8. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end-of-Ethernet-frame data is transferred to the MTL.
9. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value obtained from MTL is written to the transmit descriptor (TDES0 and TDES1) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES3). Because the OWN bit is cleared during this step, the host now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES0 and TDES1.
10. Bit 0 of status register of corresponding channel is set after completing transmission of a packet that has interrupt on completion (TDES2[31]) set in its last descriptor. The DMA engine returns to Step 3.
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the DMA CH0 transmit descriptor tail pointer Table 847 when it receives a transmit poll demand and the underflow interrupt status bit is cleared. If the application stopped the DMA by clearing bit 0 of transmit control register of corresponding DMA channel, the DMA enters the stop state.

Figure 151 shows the TxDMA transmission flow in default mode.

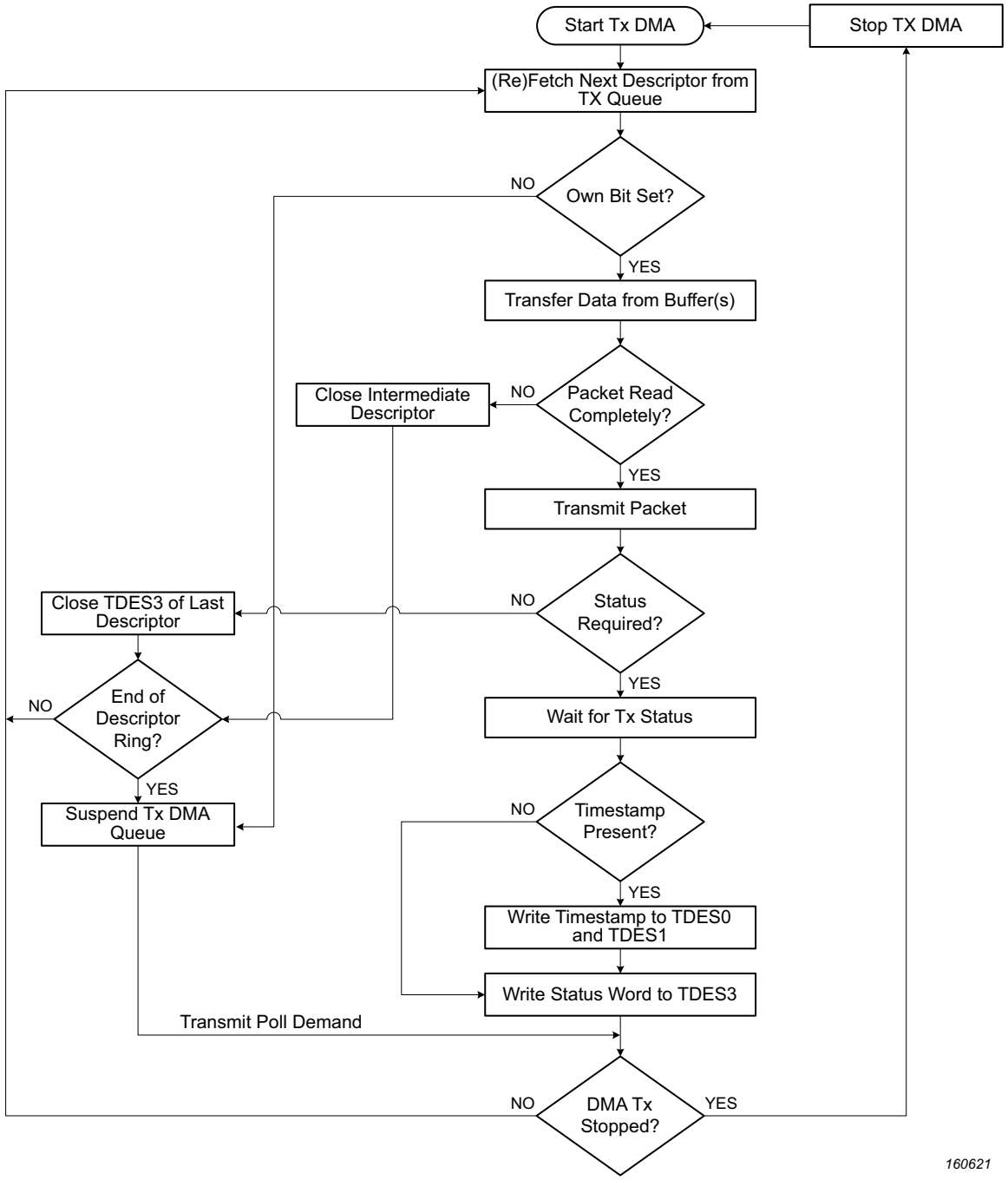


Fig 151. Transmit DMA operation in default mode

36.7.9.5.2 TxDMA operation: OSF mode

In the run state, if bit 4 is set in the transmit control register of corresponding DMA channel, the transmit process can simultaneously acquire two packets without closing the status descriptor of the first frame. As the transmit process finishes transferring the first

frame, it immediately polls the transmit descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In OSF mode, the run state transmit DMA operates in the following sequence:

1. The DMA operates as described in steps 1 to 7of the TxDMA (default mode).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into suspend mode and skips to Step 7.
4. The DMA fetches the transmit frame from the host memory and transfers the frame to the MTL until the end-of-frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's frame transmission status and timestamp. Once the status is available, the DMA writes the timestamp to TDES0 and TDES1, if such timestamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared OWN bit, to the corresponding TDES3, thus closing the descriptor. If timestamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3.
6. If enabled, the transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to Step 3 (when status is normal). If the previous transmission status shows an underflow error, the DMA goes into suspend mode (Step 7).
7. In suspend mode, if a pending status and timestamp are received from the MTL, the DMA writes the timestamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES3. It then sets relevant interrupts and returns to suspend mode. If no status is pending and the application stopped the DMA by clearing bit 0 of transmit control register of corresponding DMA channel, the DMA enters the stop state.
8. The DMA can exit suspend mode and enter the run state (go to Step 1 or Step 2 depending on pending status) only after receiving a transmit poll demand in transmit descriptor tail pointer register of corresponding channel.

Remark: As the DMA fetches the next descriptor in advance before closing the current descriptor, the descriptor chain should have more than 2 different descriptors for correct and proper operation.

The basic flow is described in Figure 152.

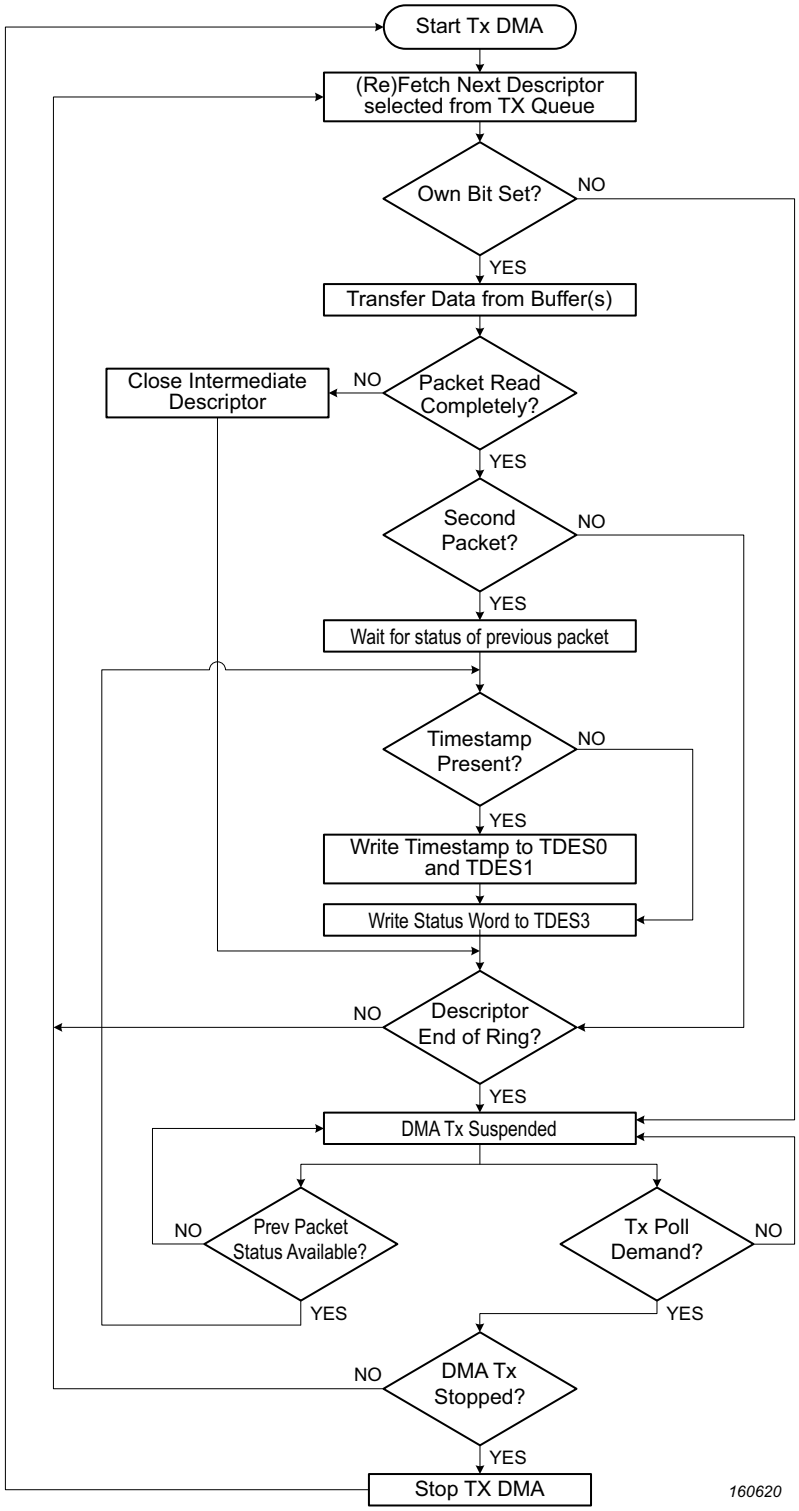


Fig 152. Transmit DMA operation in OSF mode

36.7.9.5.3 Timestamp Correction

According to the IEEE 1588 specification, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network. Because the reference timing source (the PTP clock clk_ptp_ref_i) is different from the MAC Tx or Rx clock, the captured timestamp must be corrected for latency issues because of synchronization. In addition, latency issues between the internal snapshot point and the recommended capture point (the boundary between the node and the network), must also be corrected.

Ingress Correction: In the receive side, the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (ingress correction value) programmed in the ingress correction register. The value that needs to be programmed in the ingress correction register is calculated as follows:

The timestamp correction because of synchronization is compensated by adding INGRESS_SYNC_CORR to the synchronized timestamp value as:

INGRESS_SYNC_CORR = - (2*PTP_CLK_PER)

The latency correction between the message timestamp point and the internal timestamp snapshot point is done by subtracting the latency value(INGRESS_LATENCY) with the captured timestamp as:

Ingress Correction = INGRESS_SYNC_CORR - INGRESS_LATENCY

Ingress correction is performed by programming the TSIC field in the MAC timestamp ingress correction register. The ingress correction is always negative and has a unit of nanoseconds. The value is represented in complement form as follows:

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns. It is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <ingress_correction_value> represented in binary. For example, if the required correction value is -5 ns, then the programmed value is 0xBB9A_C9FB. When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. It is represented by setting bit 31 to 1 and bits 30:0 containing 2^31 - <ingress_correction_value> represented in binary.

Egress correction:

In the transmit side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (egress correction value) programmed in the egress correction register. The value that needs to be programmed in the egress correction register is calculated as follows:

The timestamp correction because of synchronization is compensated by adding EGRESS_SYNC_CORR to the synchronized timestamp value as follows:

When enable one step timestamp feature is selected,

EGRESS_SYNC_CORR = (1*PTP_CLK_PER + 4*TX_CLK_PER)

Otherwise,

EGRESS_SYNC_CORR = -(2*PTP_CLK_PER)

The egress latency correction between the recommended capture point and the internal timestamp snapshot point is done by adding the latency value (EGRESS_LATENCY) with the captured timestamp as follows:

Egress Correction = EGRESS_SYNC_CORR + EGRESS_LATENCY

Egress correction is performed by programming the TSEC field in the MAC timestamp egress correction register. The egress correction can be positive or negative and has a unit of nanoseconds. Negative values are represented in complement form as follows:

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The value must not exceed 0x 3B9A_C9FF. If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <egress_correction_value> represented in binary.

For example, if the required correction value is -5 ns, then the programmed value should be 0xBB9A_C9FB

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 nS. If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The maximum value is 0x7FFF_FFFF. If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 2^31 - <egress_correction_value> represented in binary.

36.7.9.5.4 Transmit frame processing

The transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and type/len fields contain valid data. If the transmit descriptor indicates that the MAC core must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the first descriptor (TDES3[29]) and the last descriptor (TDES3[28]), respectively.

As transmission starts, the first descriptor must have (TDES3[29]) set. When this occurs, frame data transfers from the host buffer to the MTL Tx Queue. Concurrently, if the current frame has the last descriptor (TDES3[28]) clear, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES3[29] clear. If TDES3[28] is clear, it indicates an intermediary buffer. If TDES3[28] is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the transmit descriptor 3(TDES3) word of the descriptor that has the last descriptor bit set in transmit descriptor 3 (TDES3[28]). At this time, if interrupt on completion (TDES2[31]) was set, bit 0 of status register of corresponding DMA channel is set, the next descriptor is fetched, and the process repeats.

The actual frame transmission begins after the MTL Tx Queue has reached either a programmable transmit threshold (DMA operation mode register, bits 6:4), or a full frame is contained in the FIFO. There is also an option for store and forward mode (MTL operation mode register, bit 1). In this mode descriptors are released (OWN bit TDES0[31] clears) when the DMA finishes transferring the frame.

Remark: To ensure proper transmission of a frame and the next frame, you must specify a non-zero buffer size for the transmit descriptor that has the last descriptor (TDES3[28]) set.

36.7.9.5.5 Transmit polling suspended

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the host (TDES3[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a poll demand command by writing the tail pointer register.
- A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate transmit descriptor 3 (TDES3) bit is set.
- The DMA detects that the tail pointer is equal to the current descriptor closed by the it. To resume, the software driver must modify the tail pointer register.

If the second condition occur, bit 14 of status register of corresponding DMA channel Table 858) and transmit underflow bit of corresponding queue in MTL interrupt status Table 821) are set, and the information is written to transmit descriptor 0, causing the suspension. If the DMA goes into suspend state because of this condition, bit 15 and bit 2 of status register of corresponding DMA channel are set Table 858) are set.

In all three cases, the position in the transmit list is retained. The retained position is that of the descriptor following the last descriptor closed by the DMA.

The driver must explicitly issue a transmit poll demand command after rectifying the suspension cause.

36.7.9.5.6 Transmit channel arbitration

An arbiter provides access to multiple DMAs trying to access the bus interface unit (BIU). Figure 3-21 shows the arbitration process.

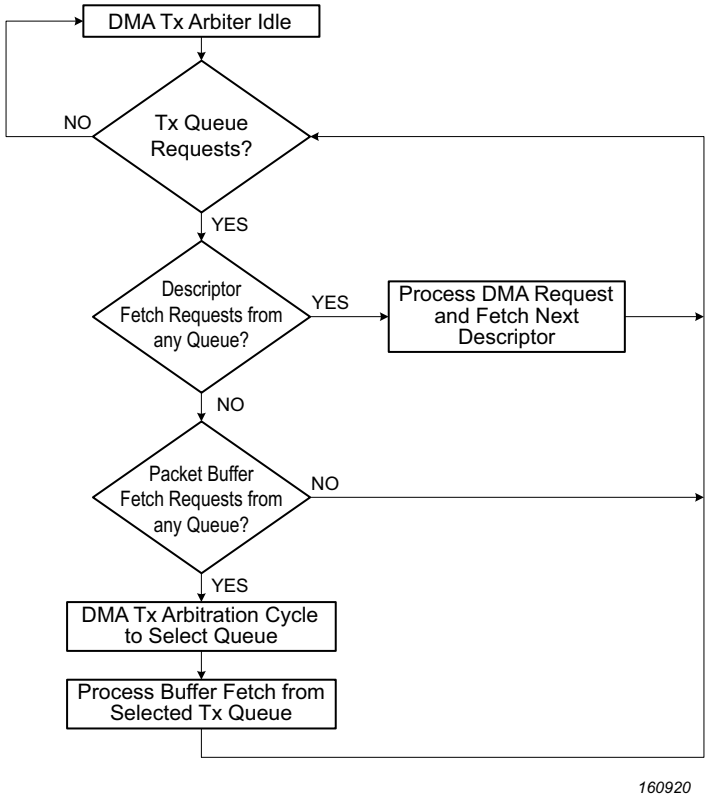


Fig 153. DMA transmit channel arbitration process

When there is any request in the Tx Queue, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests. Therefore, when there is a descriptor fetch request, the DMA arbiter acknowledges the DMA channel that is requesting for a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (channel 1 having priority over channel 0, channel 2 having priority over channel 1 and so on). For packet buffer fetches, the DMA arbiter uses the programmed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms:

- Weighted strict priority (WSP): In WSP arbitration mode, the arbiter first processes channel 7 (or the last enabled channel) and then channel 6, channel 5, and so on. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to channel7 (or last enabled channel). If channel 7 has no frames to transmit, the remaining weight is assigned to channel 6 and so on.
- Weighted round robin (WRR): In WRR arbitration mode, the arbiter first selects the channel with the highest weight programmed, and then the channel with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel gets equally distributed to all channels that have frames to transmit.

- Fixed priority (FP): In fixed priority mode, channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programmed in the transmit control register of a channel is ignored.

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

36.7.9.5.7 Reception

In the receive path, the DMA reads a packet from the MTL Rx Queue and writes it to the frame data buffers of the corresponding DMA channel.The receive DMA engine's reception sequence is shown in Figure 154 and proceeds as follows:

1. The host sets up receive descriptors (RDES0-RDES3) and sets the OWN bit (RDES3[31]). The host must set the correct value in the receive descriptor tail pointer register of corresponding DMA channel.
2. When bit 0 (SR bit) of receive control register of corresponding DMA channel Table 844 is set, the DMA enters the run state. While in the run state, the DMA polls the receive descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the suspend state and jumps to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor.
5. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
6. Incoming frames are processed and placed in the acquired descriptor's data buffers.
7. The DMA takes the status of the receive frame from the MTL and writes the status word to current descriptor with the OWN bit cleared and the last descriptor bit set.
8. The DMA writes the frame length to RDES3. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. If IEEE 1588 timestamp feature is enabled, the DMA stores the timestamp (if available).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3; otherwise, go to the suspend state (step 11).
11. The receive DMA exits the Suspend state when a receive poll demand is given and the application advances the receive tail pointer register of a channel. The engine proceeds to step 2 and refetches the next descriptor.

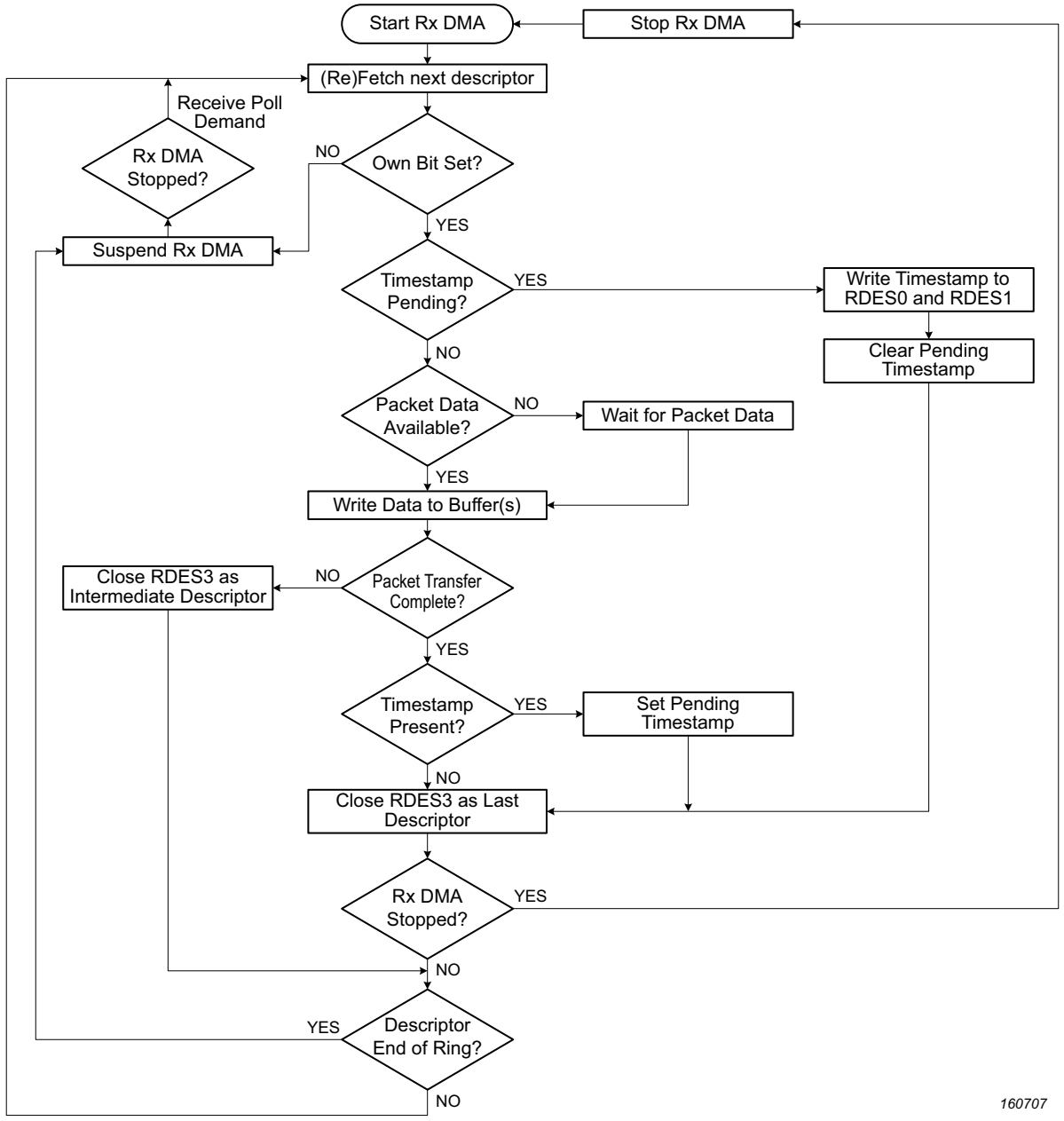


Fig 154. Receive DMA operation

36.7.9.5.8 Receive descriptor acquisition

The receive engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- The receive start/stop bit (receive control register of corresponding DMA channel Table 844) has been set immediately after being placed in the run state.
- The descriptor tail pointer register value is ahead of the current descriptor acquired by the Rx DMA.

- The controller has completed frame reception, but the current receive descriptor is not yet closed.
- A receive poll demand has been issued.

36.7.9.5.9 Receive frame processing

The MAC transfers the received frames to the host memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the Receive FIFO of MTL, or when the complete frame is written to the queue in store and forward mode.

If the frame fails the address filtering, it is dropped in the MAC block itself (unless receive all bit 31 is set in the MAC frame filter register; Table 779). Frames that are shorter than 64 bytes, because of collision or premature termination, can be purged from the MTL Rx Queue.

When the DMA application interface AHB becomes ready, it transfers the data and sets the following: If the packet fits in a single descriptor, the DMA sets both last descriptor (RDES3[28]) and first descriptor (RDES3[29]). If the packets fits into more than one descriptor, the DMA sets the first descriptor (RDES3[29]) to delimit the packet. The descriptors are released when the OWN (RDES[31]) bit is reset to 0, either as the data buffer fills up or as the last segment of the frame is transferred to the receive buffer. The received packets status is updated in the last descriptor.

If interrupt enabled on completion (RDES3[30]) bit is set in any of the descriptors between the first and last descriptor of the packet and bit 6 of interrupt enable register of corresponding DMA channel is set, the DMA sets bit 6 of status register of corresponding DMA channel. The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if bit 7 of interrupt enable register of corresponding DMA channel is set, the receive process sets bit 7 of status register of corresponding DMA channel Table 858 and then enters the suspend state. The position in the receive list is retained.

36.7.9.5.10 Interrupts

Interrupts can be generated as a result of various events. The DMA interrupt status register (Table 840) contains all the bits that might cause an interrupt. Table 851 contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, normal and abnormal, as described in DMA status channel status register (Table 858). The normal group is for events that happen during the normal transfer of packets (TI, RI, TBU) while the abnormal interrupt events are for error events. Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal is de-asserted. Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. For example, receive interrupt bit 6 of DMA channel status register Table 858 indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received.

Remark: The DMA interrupt status register (Table 840) is the (interrupt) status register. The interrupt pin is asserted because of any event in this status register only if the corresponding interrupt enable bit is set in DMA interrupt enable register (Table 851).

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the DMA interrupt status register (Table 840) for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in DMA status register.

36.7.9.5.11 Error response to DMA

For any data transfer initiated by a DMA channel, if the slave replies with an error response, that DMA stops all operations and updates the error bits and the fatal bus error bit in the status register of corresponding DMA channel (Table 858). That DMA controller can resume operation only after soft resetting or hard resetting the core and re-initializing the DMA. The rest of the DMA channels are not affected by such errors.

36.7.10 Ethernet descriptors

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The Ethernet block supports the following descriptors:

Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

Remark: Note: There is no limit for number of descriptors that can be used for a single packet.

36.7.10.1 Descriptor structure

Ethernet block supports the ring structure for DMA descriptor.

In Ring structure, descriptors are separated by the word, dword, or lword number programmed in the DSL field of the DMA channel control register Table 842. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

DMA channel transmit descriptor ring length register Table 849.

DMA channel receive descriptor ring length register Table 850.

The descriptor tail pointer register contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

Current descriptor pointer == descriptor tail pointer;

The DMA goes into the suspend mode when this condition occurs. The application must perform a write to the descriptor tail pointer register and update the tail pointer so that the following condition is true:

Current descriptor pointer < descriptor tail pointer;

The DMA automatically wraps around the base address when the end of ring is reached. See Figure 155

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to descriptor base address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to descriptor base address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

The DMA writes the header length in RDES2 of the first receive descriptor (RDES3[29] (FD bit) is set) for the packet. The packet length is written in RDES3 of the last receive descriptor (RDES3[28] (LD bit) set). The buffer length for the payload is set by the driver through the RBSZ field in the corresponding DMA channel register 2 (receive control register). The DMA fills receive buffers fully in all except the last descriptor.

The header length is taken to be the value based on the bits programmed in the MAC extended configuration register HDSMS field (bits 22:20).

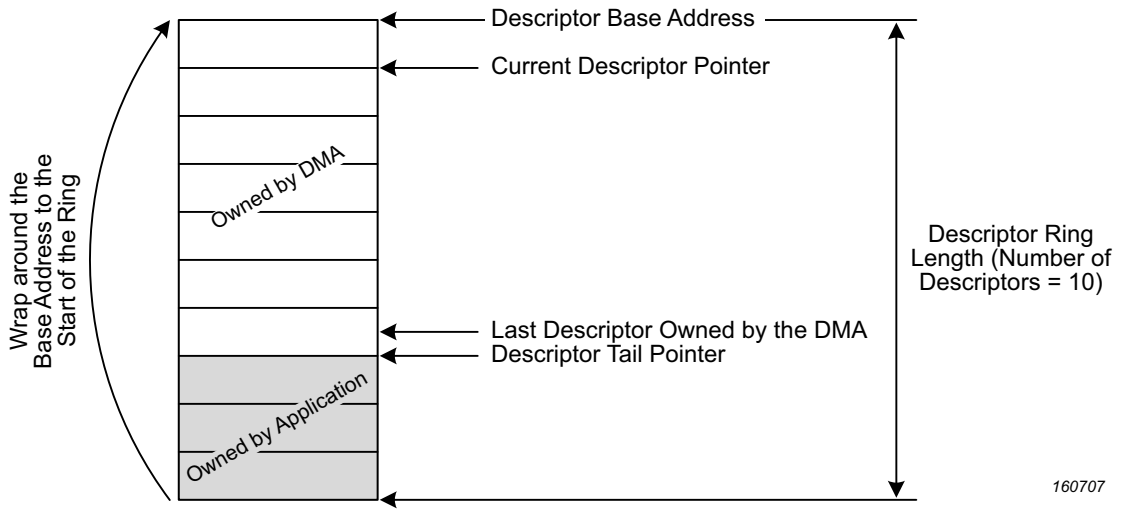


Fig 155. Descriptor ring

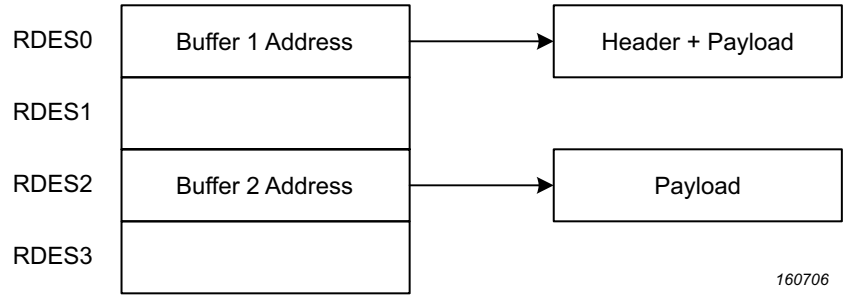


Fig 156. Descriptor structure

36.7.10.2 Descriptor endianness

The data bus can be configured for little-endian format.

36.7.10.3 Transmit descriptor

The DMA in Ethernet core requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The transmit normal descriptor has the following two formats: read format and write-back format.

36.7.10.3.1 Transmit normal descriptor (read format)

The data bus can be configured for either little-endian or big-endian format. Figure 157 shows the read format for a transmit normal descriptor.

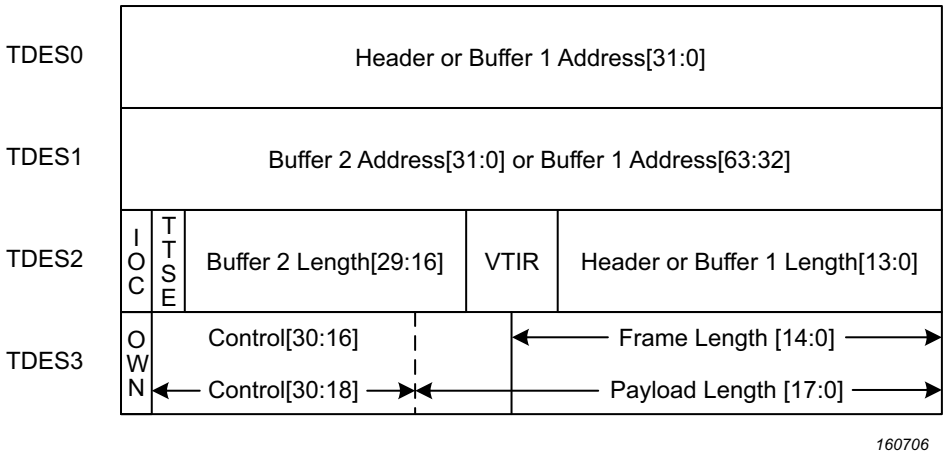


Fig 157. Transmitter descriptor read format

Table 874. Transmit descriptor word 0(TDES0)

Bit	Symbol	Description
31:0	BUF1AP	Buffer 1 address pointer or TSO header address pointer. These bits indicate the physical address of buffer 1. These bits indicate the TSO header address pointer when the following bits are set: TSE bit of TDES3 FD bit of TDES3

Table 875. Transmit descriptor word 1(TDES1)

Bit	Symbol	Description
31:0	BUF2AP	Buffer 2 or buffer 1 address pointer. This bit indicates the physical address of buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment. In 40- or 48-bit addressing mode, these bits indicate the most-significant 8- or 16- bits of the buffer 1 address pointer.

Table 876. Transmit descriptor word 2 (TDES2)

Bit	Symbol	Description
13:0	B1L	Buffer 1 length. This field is equal to buffer 1 length.
15:14	-	Reserved.
29:16	B2L	Buffer 2 length. The driver sets this field. When set, this field indicates buffer 2 length.
30	TTSE	Transmit timestamp enable. This bit enables the IEEE1588 timestamping for transmit packet referenced by the descriptor.
31	IOC	Interrupt on completion. This bit sets the TI bit in the DMA channel status register after the present packet has been transmitted.

Table 877. Transmit descriptor word 3 (TDES3)

Bit	Symbol	Description
14:0	FL	Frame length. This field is equal to the length of the frame to be transmitted in bytes. This field is equal to the total length of the frame to be transmitted: Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length
15	-	Reserved.
17:16	CIC	Checksum insertion control. These bits control the checksum calculation and insertion. The following list describes the bit encoding: 0x0: Checksum insertion disabled. 0x1: Only IP header checksum calculation and insertion are enabled. 0x2: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 0x3: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.
18	-	Reserved.

Table 877. Transmit descriptor word 3 (TDES3)

Bit	Symbol	Description
22:19	SLOTNUM	Slot number control bits in AV mode. These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field. DMA channel slot function control status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels.
25:23	-	Reserved.
27:26	CPC	CRC pad control. This field controls the CRC and pad insertion for Tx packet. This field is valid only when the first descriptor bit (TDES3[29]) is set. The following list describes the values of bits 27:26: 0x0: CRC and pad insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 0x1: CRC insertion (disable pad insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the transmit buffer, that is, the packet being transferred from the transmit buffer is of length greater than or equal to 60 bytes. 0x2: Disable CRC insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. 0x3: CRC replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer.
28	LD	Last descriptor. When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.
29	FD	First descriptor When this bit is set, it indicates that the buffer contains the first segment of a packet.
30	CTXT	Context type. This bit should be set to 0 for normal descriptor
31	OWN	Own bit. When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).

36.7.10.3.2 Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding transmit packet.

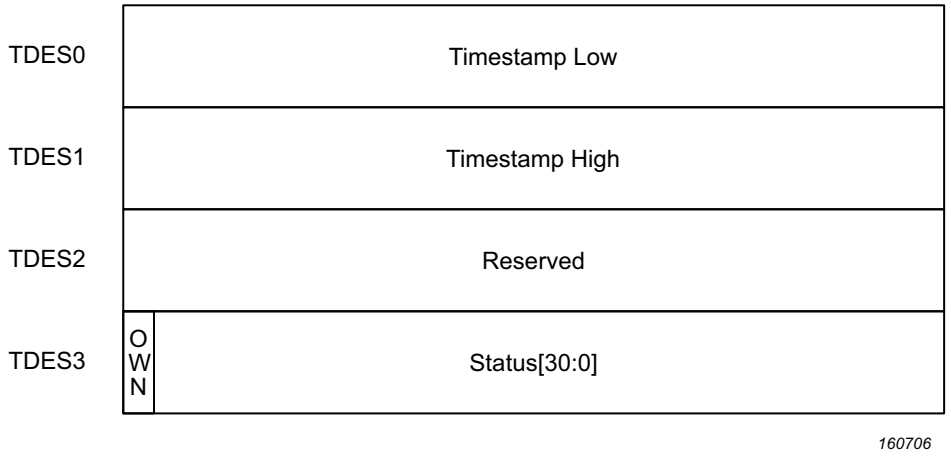


Fig 158. Transmitter descriptor write-back Format

36.7.10.3.3 TDES0 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 878. TDES0 normal descriptor (write-back Format)

Bit	Symbol	Description
31:0	TTSL	Transmit packet timestamp low. The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the last segment bit (LS) in the descriptor is set and the timestamp status (TTSS) bit is set.

36.7.10.3.4 TDES1 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 879. TDES1 normal descriptor (write-back format)

Bit	Symbol	Description
31:0	TTSH	Transmit packet timestamp high. The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the last segment bit (LS) in the descriptor is set and timestamp status (TTSS) bit is set.

36.7.10.3.5 TDES2 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 880. TDES2 normal descriptor (write-back format)

Bit	Symbol	Description
31:0	--	Reserved.

36.7.10.3.6 TDES3 normal descriptor (write-back format)

This format is only applicable to the last descriptor of a packet.

Table 881. TDES3 normal descriptor (write-back format)

Bit	Symbol	Description
0	IHE	IP header error. When IP header error is set, this bit indicates that the checksum offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet type field indicates an IPv4 payload.
1	DB	Deferred bit This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
2	UF	Underflow error. This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty transmit buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the MTL interrupt status register.
3	ED	Excessive deferral. This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the MAC configuration register.
7:4	CC	Collision count/ This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
8	EC	Excessive collision. This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC configuration register, this bit is set after first collision and the transmission of the packet is aborted.
9	LC	Late collision. This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including preamble in MII mode). This bit is not valid if underflow error is set.
10	NC	No carrier. This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
11	LOC	Loss of carrier. This bit indicates that loss of carrier occurred during packet transmission. This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
12	PCE	Payload checksum error. This bit indicates that the checksum offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the payload length field of the IP header or the MTL starting to forward the packet to the MAC transmitter in store and forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store and forward mode.
13	FF	Packet flushed. This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
14	JT	Jabber timeout. This bit indicates that the MAC transmitter has experienced a jabber time-out. This bit is set only when the JD bit of the MAC configuration register is not set.

Table 881. TDES3 normal descriptor (write-back format)

Bit	Symbol	Description
15	ES	Error summary. This bit indicates the logical OR of the following bits: • TDES3[0]: IP header error • TDES3[14]: jabber timeout • TDES3[13]: packet flush • TDES3[12]: payload checksum error • TDES3[11]: loss of carrier • TDES3[10]: no carrier • TDES3[9]: late collision • TDES3[8]: excessive collision • TDES3[3]: excessive deferral • TDES3[2]: underflow error
16	-	Reserved.
17	TTSS	Tx timestamp status. This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the transmit packet. This field is valid only when the last segment control bit (TDES3[28]) in a descriptor is set.
27:18	-	Reserved.
28	LD	Last descriptor. This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
29		First descriptor This bit indicates that the buffer contains the first segment of a packet.
30	CTXT	Context type. This bit should be set to 0 for normal descriptor.
31	OWN	Own bit. When this bit is set, it indicates that the Ethernet DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.

36.7.10.4 Receive descriptor

The DMA in Ethernet block attempts to read a descriptor only if the tail pointer is different from the base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the Rx FIFO in MTL becomes full and starts dropping packets.

All RX descriptors are prepared by the software and given to the DMA as “normal” descriptors with the content as shown in receive normal descriptor (read format). The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the Rx DMA will close the descriptor with the corresponding packet status. The format of this status is given in the “receive normal descriptor (write-back format)”.

In the receive descriptor (read format), if the buffer address field is all 0s, Ethernet block does not transfer data to that buffer and skips to the next buffer or next descriptor.

In the receive descriptor (read format), if the buffer address field is all 0s, Ethernet does not transfer data to that buffer and skips to the next buffer or next descriptor.

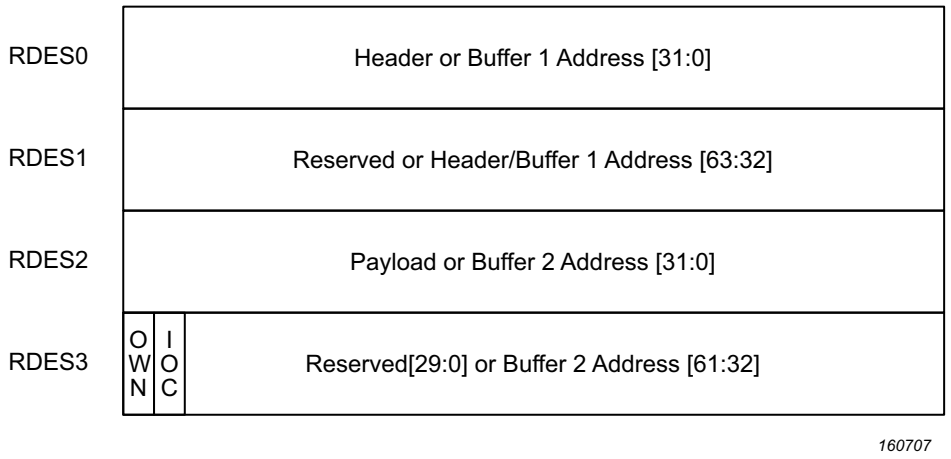


Fig 159. Receive normal descriptor (read format)

36.7.10.4.1 Receive normal descriptor (read format)

Table 882. RDES0 normal descriptor (read format)

Bit	Symbol	Description
31:0	BUF1AP	Header or buffer 1 address pointer. These bits indicate the physical address of buffer 1. The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a Write operation with RDES0[1:0] (or RDES0[2:0]/[3:0] in case of 64-/128-bit configuration) as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer.If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

Table 883. RDES1 normal descriptor (read format)

Bit	Symbol	Description
31:0	-	Reserved.

Table 884. RDES2 normal descriptor (read format)

Bit	Symbol	Description
31:0	BUF2AP	Buffer 2 address pointer. These bits indicate the physical address of buffer 2. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP[3:0 or 2:0 or 1:0] (corresponding to 128- or 64- or 32-bit data-bus) and writes to the complete location.

Table 885. RDES3 normal descriptor (read format)

Bit	Symbol	Description
23:0	-	Reserved.
24	BUF1V	Buffer 1 address valid. When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by buffer 1 address in RDES1 can be used by the DMA to write received packet data.
25	BUF2V	Buffer 1 address valid. When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by buffer 1 address in RDES1 can be used by the DMA to write received packet data.
29:26	-	Reserved.
30	IOC	Interrupt enabled on completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
31	OWN	Own bit. When this bit is set, it indicates that the Ethernet block's DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception. The buffers associated with the descriptor are full.

36.7.10.4.2 Receive normal descriptor (write-back format)

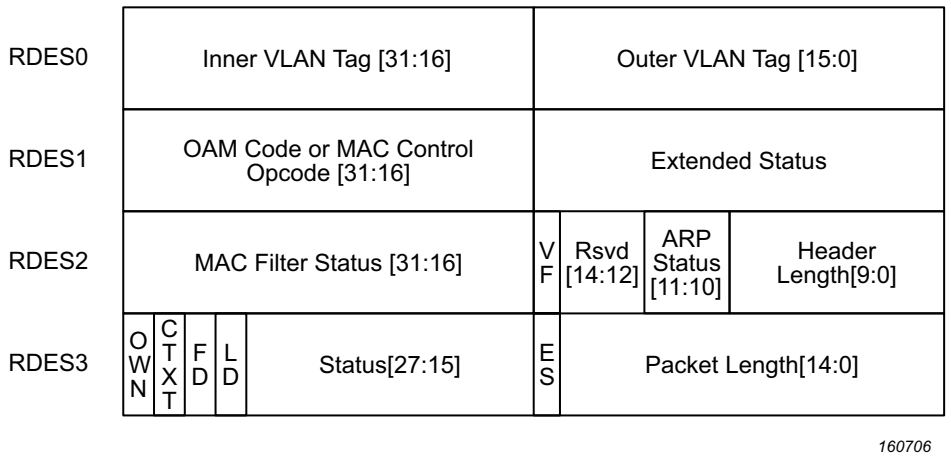


Fig 160. Receive descriptor write-back format

36.7.10.4.3 RDES0 normal descriptor (write-back format)

Table 886. TDES0 normal descriptor (write-back format)

Bit	Symbol	Description
31:0	-	Reserved.

36.7.10.4.4 RDES1 normal descriptor (write-back format)

The status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

Table 887. RDES1 normal descriptor (write-back format)

Bit	Symbol	Description
2:0	PT	Payload type. These bits indicate the type of payload encapsulated in the IP datagram processed by the receive checksum offload engine (COE): • 0x0: Unknown type or IP/AV payload not processed • 0x1: UDP • 0x2: TCP • 0x3: ICMP • 0x4: IGMP if IPV4 header present bit is set else Reserved • 0x5: AV untagged control packet • 0x6: AV tagged data packet • 0x7: AV tagged control packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 0x0.
3	IPHE	IP header error. When this bit is set, it indicates either of the following: • The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. • The IP datagram version is not consistent with the Ethernet type value. • Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
4	IPV4	IPV4 header present. This bit indicates that an IPV4 header is detected.
5	IPV6	IPv6 header present. This bit indicates that an IPV6 header is detected.
6	IPCB	IP checksum bypassed. This bit indicates that the checksum offload engine is bypassed.
7	IPCE	IP payload error. When this bit is set, it indicates either of the following: • The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. • The TCP, UDP, or ICMP segment length does not match the payload length value in the IP header field. • The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.

Table 887. RDES1 normal descriptor (write-back format) *…continued*

Bit	Symbol	Description
11:8	PMT	PTP message type. These bits are encoded to give the type of the message received: • 0x0: No PTP message received • 0x1: SYNC (all clock types) • 0x2: Follow_Up (all clock types) • 0x3: Delay Req (all clock types) • 0x4: Delay Resp (all clock types) • 0x5: Pdelay Req (in peer-to-peer transparent clock) • 0x6: Pdelay Resp (in peer-to-peer transparent clock) • 0x7: Pdelay Resp follow-up (in peer-to-peer transparent clock) • 0x8: Announce • 0x9: Management • 0xA: Signaling • 0xF: PTP packet with Reserved message type • Others: Reserved
12	PFT	PTP packet type. This bit indicates that the PTP message is sent directly over Ethernet.
13	PV	PTP version. This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format.
14	TSA	Timestamp available. When timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the last descriptor bit (RDES3[28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
15	TD	Timestamp dropped. This bit indicates that the timestamp was captured for this packet but it got dropped in the MTL Rx FIFO because of overflow
31:16	OPC	OAM sub-type code, or MAC control packet opcode OAM sub-type code. If bits 18:16 of RDES3 are set to 0x7, this field contains the OAM sub-type and code fields. MAC control packet opcode If bits 18:16 of RDES3 are set to 0x6, this field contains the MAC control packet opcode field.

36.7.10.4.5 RDES2 normal descriptor (write-back format)

Table 888. RDES2 normal descriptor (write-back format)

Bit	Symbol	Description
15:0	-	Reserved.
16	SAF	SA address filter fail. When this bit is set, it indicates that the packet failed the SA filter in the MAC.
17	DAF	Destination address filter fail. When this bit is set, it indicates that the packet failed the DA filter in the MAC.
18	-	Reserved. Always 0.
26:19	MADRM	MAC address match. When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset.
31:20	--	Reserved.

36.7.10.4.6 RDES3 normal descriptor (write-back format)

Table 889. RDES3 normal descriptor (write-back format)

Bit	Symbol	Description
14:0	PL	Packet length. These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and either the descriptor error (RDES3[13]) or overflow error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the last descriptor and error summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.
15	ES	Error summary. When this bit is set, it indicates the logical OR of the following bits: • RDES3[24]: CRC error • RDES3[19]: dribble error • RDES3[20]: receive error • RDES3[22]: watchdog timeout • RDES3[21]: overflow error • RDES3[23]: giant packet This field is valid only when the LD bit of RDES3 is set.
18:16	LT	Length/type field. This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: • 0x0: The packet is a length packet • 0x1: The packet is a type packet. • 0x3: The packet is a ARP request packet type • 0x4: The packet is a type packet with VLAN tag • 0x5: The packet is a type packet with double VLAN tag • 0x6: The packet is a MAC control packet type • 0x7: The packet is a OAM packet type • 0x2: Reserved
19	DE	Dribble bit error. When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII mode.
20	RE	Receive error. When this bit is set, it indicates that the gmii_rxer_i signal is asserted while the gmii_rxdv_i signal is asserted during packet reception. This error also includes carrier extension error in the MII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
21	OE	Overflow error. When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. **Note:** This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store and forward mode, all partial packets are dropped completely in Rx FIFO.
22	RWT	Receive watchdog timeout. When this bit is set, it indicates that the receive watchdog timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.

Table 889. RDES3 normal descriptor (write-back format)

Bit	Symbol	Description
23	GP	Giant packet. When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9,018 or 9,022 bytes if jumbo packet enable is set). Note: Giant packet indicates only the packet length. It does not cause any packet truncation.
24	CE	CRC error. When this bit is set, it indicates that a cyclic redundancy check (CRC) error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
25	RS0V	Receive status RDES0 valid. When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	Receive status RDES1 valid. When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
27	RS2V	Receive status RDES2 valid. When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
28	LD	Last descriptor. When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
29	FD	First descriptor. When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet.
30	CTXT	Receive context descriptor. When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0x0 to this bit for normal receive descriptor.
31	OWN	Own bit. When this bit is set, it indicates that the Ethernet block DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full

36.7.11 Programming

This section provides the instructions for initializing the DMA or MAC registers in the proper sequence.It contains the following sections:

- Initializing DMA
- Initializing MTL registers
- Initializing MAC
- Performing normal receive and transmit operation
- Stopping and starting transmission
- Programming guidelines for multi-channel multi-queuing
- Programming guidelines for MII link state transitions
- Programming guidelines for IEEE 1588 timestamping

- Programming guidelines for AV feature
- Programming guidelines for energy efficient Ethernet

36.7.11.1 Initializing DMA

Follow these steps to initialize the DMA controller:

1. Assert a software reset by setting bit 0 of DMA mode register Table 838. This resets all of the MAC internal registers and logic.
2. Wait for the completion of the reset process (poll bit 0 of the DMA mode register Table 838, which is only cleared after the reset operation is completed).
3. Program the following fields to initialize the DMA SYSBUS mode register Table 839:
   a. AAL
   b. Fixed burst or undefined burst
   c. Burst mode values for AHB bus interface.
4. Create a descriptor list for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (set bit 31 of descriptor TDES3/RDES3).
5. Program the transmit and receive ring length registers DMA_CH0_TXDESC_RING_LENGTH,DMA_CH1_TXDESC_RING_LENGTH and DMA_CH0_RXDESC_RING_LENGTH,DMA_CH1_RXDESC_RING_LENGTH. The ring length programmed must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor DMA_CH0_TXDESC_LIST_ADDR, DMA_CH1_TXDESC_LIST_ADDR and DMA_CH0_RXDESC_LIST_ADDR, DMA_CH1_RXDESC_LIST_ADDR. Also, program transmit and receive tail pointer registers indicating to the DMA about the available descriptors DMA_CH0_TXDESC_TAIL_PTR, DMA_CH1_TXDESC_TAIL_PTR and DMA_CH0_RXDESC_TAIL_PTR, DMA_CH1_RXDESC_TAIL_PTR.
7. Program the settings of the following registers for the parameters like maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSF in case of TxDMA, RBSZ in case of RxDMA etc. DMA_CH0_CTRL, DMA_CH1_CTRL DMA_CH0_TX_CTRL, DMA_CH1_TX_CTRL, DMA_CH0_RX_CTRL, DMA_CH1_RX_CTRL.
8. Enable the interrupts by programming the DMA_CH0_INT_EN, DMA_CH0_INT_EN registers.
9. Start the receive and transmit DMAs by setting SR (bit 0) of the DMA_CH0_RX_CTRL, DMA_CH1_RX_CTRL and ST (bit 0) of the DMA_CH0_RX_CTRL, DMA_CH1_RX_CTRL register.

36.7.11.2 Initializing MTL

Complete the following steps to initialize the MTL registers:

1. Program the Tx scheduling (SCHALG) and receive arbitration algorithm (RAA) fields in MTL operation mode register Table 820 to initialize the MTL operation in case of multiple Tx and Rx Queues.
2. Program the Rx Queue to DMA mapping in MTL Rx Queue and DMA channel mapping register Table 822.

3. Program the following fields to initialize the mode of operation in the MTL TxQ0 operation mode register Table 823
   a. Transmit store and forward (TSF) or transmit threshold control (TTC) in case of threshold mode
   b. Transmit Queue enable (TXQEN) to value 0x2 to enable Transmit Queue 0
   c. Transmit Queue Size (TQS).
4. Program the following fields to initialize the mode of operation in the MTL RxQ0 operation mode register Table 834:
   a. Receive store and forward (RSF) or RTC in case of threshold mode
   b. Flow control activation and deactivation thresholds for MTL Receive FIFO (RFA and RFD).
   c. Error packet and undersized good packet forwarding enable (FEP and FUP)
   d. Rx Queue size (RQS).
5. Repeat steps 3 and 4 for MTL Tx and Rx Queue 1.

36.7.11.3 Initializing MAC

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: MAC address high Table 804 and MAC address low Table 805 registers.
2. Program the following fields to set the appropriate filters for the incoming frames in the MAC frame filter register Table 779:
   a. Receive all
   b. Promiscuous mode
   c. Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the MAC transmit flow control register Table 782:
   a. Pause time and other pause frame control bits
   b. Transmit flow control bits.
   c. Flow control busy.
4. Program the MAC interrupt enable register Table 789, as required, and if applicable, for the configuration.
5. Program the appropriate fields in the MAC configuration register Table 777. For ex: Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in MAC configuration registers to start the MAC transmitter and receiver.

36.7.11.3.1 Host bus burst access

The DMA attempts to execute fixed-length burst transfers on the AHB master interface if configured to do so (FB bit of DMA register 0). The maximum burst length is indicated and limited by the PBL field (DMA register 0[13:8]). The receive and transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 x 8/bus width) burst-size for the 16-bytes to be read.

The transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise (no fixed-length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the receive FIFO. The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the AHB interface, then dummy transfers are performed in order to complete the fixed-burst. Otherwise (FB bit of DMA_SYS_BUS_MODE register Table 839 is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16), because the AHB interface does not support more than INCR16.

36.7.11.3.2 Host data buffer alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example: buffer read

If the transmit buffer address is 0x0000 0FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA reads five full words from address 0x00000FF0, but when transferring data to the MTL transmit FIFO, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the MTL transmit FIFO, unless it is the end-of-frame.

Example: buffer write

If the receive buffer address is 0x0000 0FF2 (for 64-bit data bus) and 16 bytes of a received frame need to be transferred, then the DMA writes 3 full words from address 0x0000 0FF0. But the first 2 bytes of first transfer and the last 6 bytes of the third transfer have dummy data.

36.7.11.3.3 Buffer size calculations

The DMA does not update the size fields in the transmit and receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the MAC core. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The receive DMA always transfers the start of next frame with a new descriptor.

Remark: Even when the start address of a receive buffer is not aligned to the system bus's data width, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the receive descriptor to have a 0x1002 offset. The receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset.

36.7.11.3.4 DMA arbiter

The arbiter inside the DMA module performs the arbitration between the transmit and receive channel accesses to the AHB master interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (DA bit of DMA_MODE register Table 838 (DMA_MODE register) is reset), the arbiter allocates the data bus in the ratio set by the PR bits of DMA_MODE register Table 838, when both transmit and receive DMAs are requesting for access simultaneously. When the DA bit is set, the receive DMA always gets priority over the transmit DMA for data access by default. When the TXPR bit (bit 11 of DMA_MODE register Table 838) is also set, then the transmit DMA gets priority over the receive DMA.

36.7.11.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the host (either transmit or receive).
2. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and writing the descriptor tail pointer to Tx/Rx tail pointer register (DMA_CH0_TXDESC_TAIL_PTR Table 847 / DMA_CH1_TXDESC_TAIL_PTR Table 847 and DMA_CH0_RXDESC_TAIL_PTR Table 848 / DMA_CH1_RXDESC_TAIL_PTR) Table 848.

   Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process DMA current host transmit register Table 854 and DMA current host receive register Table 855.
4. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (register DMA current host transmit buffer address register Table 856 and DMA current host receive buffer address register Table 857).

36.7.11.5 Stopping and Starting Transmission

Complete the following steps to pause the transmission for some time. The steps are provided for channel 0. Similar step can be used for other channels.

1. Disable the transmit DMA by clearing bit 0 (ST) of The DMA channel 0 transmit control Table 843 and The DMA channel 1 transmit control registers Table 843.
2. Wait for any previous frame transmissions to complete. It can be checked by reading the appropriate bits of MTL TxQ0 debug register Table 825 (TRCSTS is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing bit (RE) and bit 1(TE) of the MAC configuration register. Table 777.
4. Disable the receive DMA, after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of MTL TxQ0 debug register Table 825, PRXQ=0 and RXQSTS=00).
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in MTL TxQ0 debug register Table 825 and RXQSTS is 0 in MTL RxQ0 debug register Table 836).
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. The Software should change these parameters only when the MAC transmitter and receiver are not active. Similarly the DMA related configuration should not be changed when transmit and receive DMA are in active.

36.7.11.6 Programming guidelines for multi-channel multi-queuing

36.7.11.6.1 Transmit:

Complete the following steps for transmit.

1. Program the Tx Queue size in the TQS field of MTL TxQ0 operation mode register Table 823 if queue 0 is used or MTL TxQ1 operation mode register Table 823 if queue1 is used. Based on the value programmed in TQS field, the size of the queue is determined. In the transmit operation, the number of channels is equal to the number of the queues. Due to this reason, the channel to queue mapping is fixed.
2. For a queue to be used, the queue needs to be enabled in TXQEN in the corresponding MTL TxQ operation mode register (MTL TxQ0 operation mode register Table 823 or MTL TxQ1 operation mode register Table 823).
3. The scheduling method need to be programmed in SCHALG of MTL operation mode register Table 820.
4. Program the corresponding quantum weight register Table 828 as per the selected algorithm. In case of CBS algorithm in AVB queues, the MTL_TxQ1_ETS_Control, MTL_TxQ1_SendSlopeCredit, MTL_TxQ1_HiCredit and MTL_TxQ1_LoCredit registers also need to be programmed as required.These registers are not available for Q0.

36.7.11.6.2 Receive:

Complete the following steps for transmit.

1. Program the Rx Queue size in the RQS field of MTL RxQ operation mode register Table 834. Based on the value programmed in RQS field, the size of the queue is determined.
2. Enable both the Rx Queues 0 and 1 in the fields RXQ0EN and RXQ1EN in MAC Rx Queue control0 register Table 785 for AV.
3. The MAC routes the Rx packets to the Rx Queues based on following packet types:
   a. AV PTP Packets: Based on the programming of AVPTPQ in MAC Rx Queue control register Table 786.
   b. AV untagged control packets: Based on the programming of AVCPQ in MAC Rx Queue control 1 register Table 786.
4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping
   a. Program the RAA field of MTL operation mode register to select the arbitration algorithm to decide which RxQ is read out from the Rx FIFO memory.
   b. Program the MTL RxQ control register to decide the weights and the packet arbitration for each RxQ.
   c. Set corresponding DADMACH bit in MTL Rx Queue and DMA channel mapping register to select dynamic mapping of packets in each Rx Queue.
   d. In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of DCS field in the lowest MAC address register.

36.7.11.7 Programming guidelines for MII link state transitions

36.7.11.7.1 Transmit and receive clocks are running when the link is down

1. Disable the transmit DMA (if applicable) by clearing bit 0 (ST) of DMA channel control registers (both CH0 and CH1). or
2. Disable the MAC receiver by clearing bit 0(RE) of MAC configuration register.

3. Wait for any previous frame transmissions to complete. It can be checked this by reading the appropriate bits of MTL_TXQ0_DBG register (TRCSTS is not 01). or Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing bit 1(TE) of the MAC configuration register.
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in (MTL_TXQ0_DBG register and RXQSTS is 0 in MTL_RxQ0_DBG register).
6. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers.
7. Restart the operation by starting the Tx DMA, and then enabling the MAC transmitter and receiver. The Rx DMA not need to be disabled. As the receiver is disabled, the FIFO does not get any data in the Rx FIFO.

Complete the following steps when the link is down but the transmit and receive clocks are running: The steps are provided for channel 0.

36.7.11.7.2 Transmit and receive clocks are stopped when the link is down

Complete the following steps when the link is down but the transmit and receive clocks are stopped: The steps are provided for channel 0.

1. Disable the MAC transmitter and receiver by clearing bits RE and TE of MAC configuration register Table 777.This will not take effect immediately as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait for the completion of the transfer of any partial frame if any at time of stopping of transmit/receive clock. This can be checked by reading the MAC debug register Table 798 (should be all zero).Some old packets may still remain in the Tx FIFO as the MAC transmitter is stopped.
4. Read the PHY registers to know the latest operating mode and accordingly program the MAC registers.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

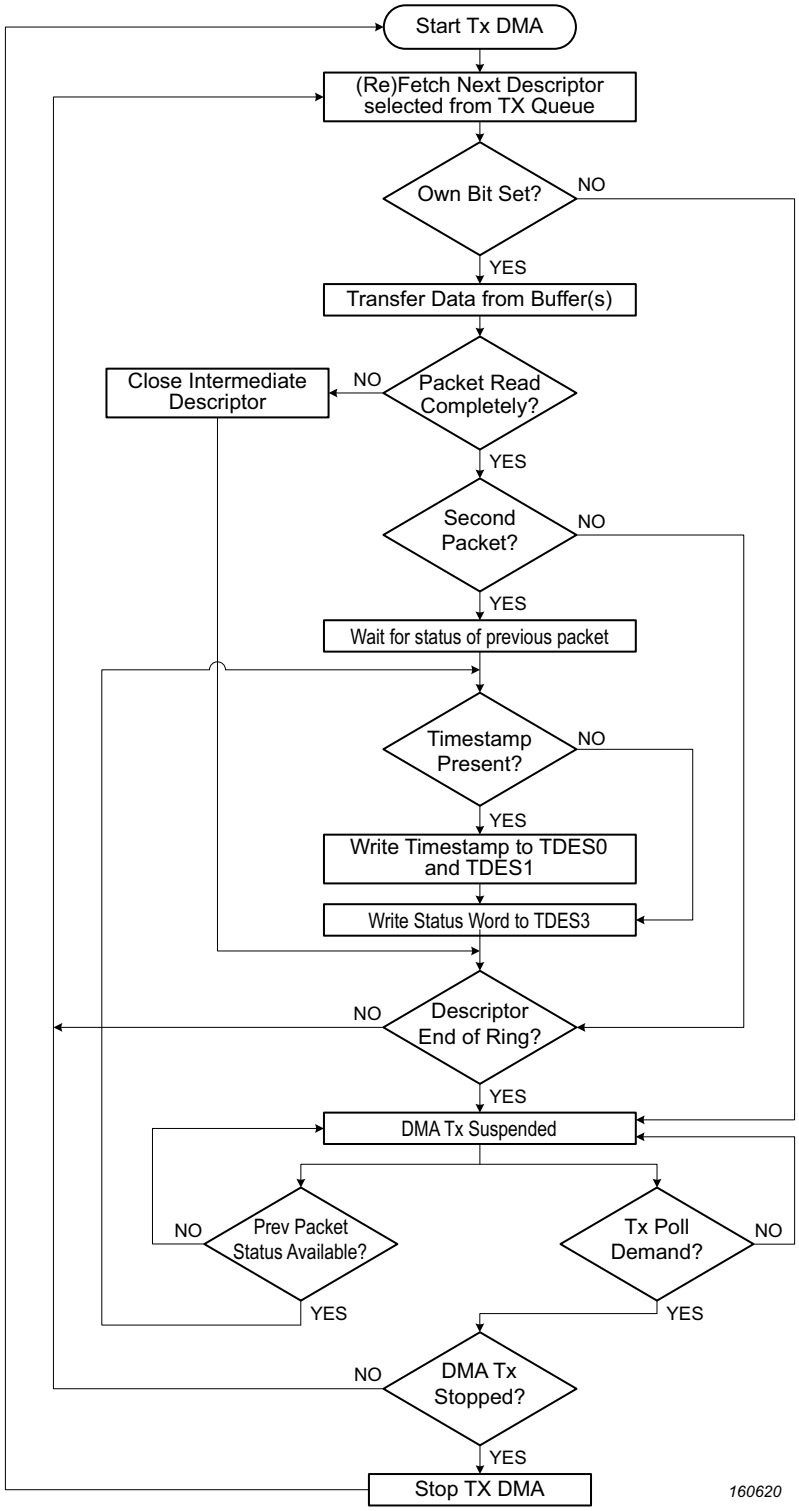


Fig 161. Transmit DMA operation in OSF mode

36.7.11.8 Programming guidelines for IEEE 1588 timestamping

36.7.11.8.1 Initialization guideline for system time generation

The timestamp feature can be enabled by setting bit 0 of the MAC timestamp control register Table 806. However, it is essential that the timestamp counter should be initialized after this bit is set. Complete the following steps during Ethernet initialization:

1. Mask the timestamp trigger interrupt by clearing the bit 16 of MAC interrupt enable register Table 789.
2. Set bit 0 of MAC timestamp control register Table 806 to enable timestamping.
3. Program MAC sub-second increment register Table 808 based on the PTP clock frequency.
4. If fine correction approach is used, program MAC timestamp addend register Table 813 and set bit 5 of MAC timestamp control register.
5. Poll the MAC timestamp control register until bit 5 is cleared.
6. Program bit 1 of MAC timestamp control register to select the fine update method (if required).
7. Program MAC system time seconds update register Table 811 and MAC system time nanoseconds update register Table 812 with the appropriate time value.
8. Set bit 2 in MAC timestamp control register. The timestamp counter starts operation as soon as it is initialized with the value written in the timestamp update registers.
9. Enable the MAC receiver and transmitter for proper timestamping.

If timestamp operation is disabled by clearing bit 0 of MAC timestamp control register, you need to repeat all these steps to restart the timestamp operation.

36.7.11.8.2 System time correction

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (MAC system time seconds update register Table 811 and MAC system time nanoseconds update register Table 812).
2. Set bit 3 (TSUPDT) of the MAC timestamp control register Table 806.The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm explained in Section 36.7.8.9 “System time register module”, calculate the rate by which you want to make the system time increments slower or faster.
2. Update the MAC timestamp addend register Table 813 with the new value and set bit 5 of the MAC timestamp control register Table 806.
3. Wait for the time for which you want the new value of the addend register to be active. You can do this by enabling the timestamp trigger interrupt after the system time reaches the target value.
4. Enable the timestamp interrupt in bit 12 of MAC interrupt enable register Table 789.

5. Set bit 4 in MAC timestamp control register Table 806.
6. When this trigger causes an interrupt, read MAC interrupt status register Table 788.
7. Reprogram MAC timestamp addend register with the old value and set bit 5 again Table 813.

36.7.11.9 Programming guidelines for AV feature

36.7.11.9.1 Initializing the DMA

Complete the following steps to initialize the DMA:

1. Assert a software reset by setting bit 0 in DMA mode register Table 838 to reset all Ethernet internal registers and logic.
2. Wait for the completion of the reset process. Poll bit 0 of the DMA mode register, which is cleared only after the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in DMA mode register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the transmit and receive descriptors. When OSF mode is used, at least two TX descriptors are required. For more information about descriptors, see Section 36.7.9 “DMA controller description”.
5. Make sure that the driver software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the transmit and receive ring length registers (DMA channel 0 and channel 1 transmit ring length register Table 849 and DMA channel 0 and channel 1 receive ring length register Table 850). The ring length programmed must be at least 4.
7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA channel 0 and channel 1 transmit list address register Table 845, DMA channel 0 and channel 1 receive list address register Table 846). In addition, you must program the transmit and receive tail pointer registers indicating to the DMA about the available descriptors (DMA channel 0 and channel transmit tail pointer register Table 847, DMA channel 0 and channel 1 receive tail pointer register Table 848).
8. Program the following fields to initialize the mode of operation in the MTL_TxQ0 operation mode register Table 823:
   a. Transmit store and forward (TSF)
   b. Transmit threshold control (TTC)
   c. Transmit Queue enable (TXQEN) to value 0x2 to enable Transmit Queue 0
   d. Transmit Queue size (TQS)
9. Enable the interrupts by programming the DMA channel 0 and channel 1 interrupt enable register Table 851.
10. Repeat steps 4 to 9 for both channels for AV feature.
11. Program the CBS control register, idleSlope (MTL_TXQ1_QNTM_WGHT), sendSlope, hiCredit, and loCredit registers of the AV queues. Follow step from Section 36.7.11.9.2 “Enabling slot number checking” before step 12.

12. Start the receive and transmit DMA by setting bit 0 of the DMA channel 0, channel 1 transmit control register and bit 0 of DMA channel 0, channel 1 receive control register.

36.7.11.9.2 Enabling slot number checking

The slot number check feature can be used to specify the intervals at which the DMA channels mapped to AV queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the Ethernet memory. Complete the following steps to enable the slot number checking:

1. Enable timestamping by following the steps described in Section 36.7.11.8.1 "Initialization guideline for system time generation".
2. Make sure that the SLOTNUM field (bits 22:19) of TDES3 normal descriptor (read format) contains a valid slot number. The current reference slot number can be read from the DMA channel 0 and channel 1 slot function control register Table 853.
3. Set bit 0 (ESC) of the slot function control and status register of a channel to enable the slot number checking Table 853.

36.7.11.9.3 Enabling average bits per slot reporting

The MTL TxQ ETS status register Table 827 of a particular AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. Enable timestamping by following the steps described in Section 36.7.11.8.1 "Initialization guideline for system time generation".
2. Program bits 6:4, SLC, of the corresponding MTL TxQ ETS control register of a channel with number of slots over which the average transmitted bits per slot need to be computed Table 826.
3. Enable bit 9 (ABPSSIE) of the corresponding MTL TxQ interrupt control status register Table 833 of a channel to generate the average bits per slot interrupt.
4. Read bits 16:0, ABS, from the corresponding MTL TxQ ETS status register Table 827 of a channel on each interrupt.

The frequency of this interrupt depends on the value programmed in the step 2. For example, when you program value 0 in the SLC field, the interrupt is generated every 125 microseconds.

When not required, you can disable this interrupt to stop the interrupt flooding. The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled.When high, bit 1 (ABPSIS) of the MTL TxQ ETS status register Table 827 indicates that a new value is updated in the ABS field.

36.7.11.10 Programming guidelines for energy efficient Ethernet

36.7.11.10.1 Entering and exiting the Tx LPI mode

Complete the following steps during Ethernet core initialization:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values.

2. Program the PHY registers through the MDIO interface (including the RX CLK stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode.)
3. Program bits 25:16 and bits 15:0 in MAC LPI Timer control register Table 794.
4. Read the link status of the PHY chip by using the MDIO interface and update bit 17 of MAC LPI control status register Table 793 accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Make sure that the driver software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the transmit and receive ring length registers (DMA channel 0 and channel 1 transmit ring length register Table 849 and DMA channel 0 and channel 1 receive ring length register Table 850). The ring length programmed must be at least 4.
7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA channel 0 and channel 1 transmit list address register Table 845, DMA channel 0 and channel 1 receive list address register Table 846). In addition, you must program the transmit and receive tail pointer registers indicating to the DMA about the available descriptors (DMA channel 0 and channel transmit tail pointer register Table 847, DMA channel 0 and channel 1 receive tail pointer register Table 848).
8. Program the following fields to initialize the mode of operation in the MTL_TxQ0 operation mode register Table 823:
   a. Transmit Store And Forward (TSF)
   b. Transmit Threshold Control (TTC)
   c. Transmit Queue enable (TXQEN) to value 0x2 to enable Transmit Queue 0
   d. Transmit Queue size (TQS)
9. Enable the interrupts by programming the DMA channel 0 and channel 1 interrupt enable register Table 851.
10. Repeat steps 4 through 9 for both channels for AV feature.
11. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues. Follow step from section “enable slot number checking” Section 36.7.11.9.2 before step 12.
12. Start the receive and transmit DMA by setting bit 0 of the DMA channel 0, channel 1 transmit control register and bit 0 of DMA channel 0, channel 1 receive control register.

36.7.11.10.2 Gating off the CSR clock in the LPI mode

The slot number check feature can be used to specify the intervals at which the DMA channels mapped to AV queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the Ethernet memory. Complete the following steps to enable the slot number checking:

1. Enable timestamping by following the steps described in Section 36.7.11.8.1 “Initialization guideline for system time generation”.
2. Make sure that the SLOTNUM field (bits 22:19) of TDES3 normal descriptor (read format) contains a valid slot number. The current reference slot number can be read from the DMA channel 0 and channel 1 slot function control register Table 853.

3. Set bit 0 (ESC) of the slot function control and status register of a channel to enable the slot number checking Table 853.

Gating off the CSR clock in the Rx LPI mode: The slot number check feature can be used to specify the intervals at which the DMA channels mapped to AV queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the Ethernet memory. Complete the following steps to enable the slot number checking:

1. Enable timestamping by following the steps described in Section 36.7.11.8.1 "Initialization guideline for system time generation".
2. Make sure that the SLOTNUM field (bits 22:19) of TDES3 normal descriptor (read format) contains a valid slot number. The current reference slot number can be read from the DMA channel 0 and channel 1 slot function control register Table 853.
3. Set bit 0 (ESC) of the slot function control and status register of a channel to enable the soot number checking Table 853.

Gating off the CSR clock in the Tx LPI mode: The slot number check feature can be used to specify the intervals at which the DMA channels mapped to AV queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the Ethernet memory. Complete the following steps to enable the slot number checking:

1. Enable timestamping by following the steps described in Section 36.7.11.8.1 "Initialization guideline for system time generation".
2. Make sure that the SLOTNUM field (bits 22:19) of TDES3 normal descriptor (read format) contains a valid slot number. The current reference slot number can be read from the DMA channel 0 and channel 1 slot function control register Table 853.
3. Set bit 0 (ESC) of the slot function control and status register of a channel to enable the slot number checking Table 853.

37.1 How to read this chapter

The USB full-speed controller is available on all LPC546xx devices.This chapter describes the device functionality of the controller.

37.2 Features

- USB2.0 full-speed device controller.
- Supports 10 physical (5 logical) endpoints including control endpoints.
- Supports single and double buffering.
- Each non-control endpoint supports bulk, interrupt, or isochronous endpoint types.
- Supports wake-up from deep-sleep mode on USB activity and remote wake-up.
- Supports SoftConnect internally.
- Supports Link Power Management (LPM).

37.3 Basic configuration

Initial configuration of the USB0 device controller:

- Pins: Configure the USB0 pins in the IOCON register block. See Table 256 and Section 37.5 “Pin description”.
- In the AHBCLKCTRL1 register, enable the clock to the USB0D device controller register interface (see Section 7.5.20).
- Power: Enable the power to the USB0 PHY by clearing the bit PDEN_USB0_PHY in the PDRUNCFG0 register (see Section 7.5.84).
- Port mode configuration: Enable port mode configuration by setting the USB0HSL host clock in the AHBCLKCTRL2 register. See Table 141. Check DEV_ENABLE bit 16 in Section 38.7.23 in PortMode register (offset 0x5C). Set bit 16 to 1 to enable the device controller on the USB0 port. Once configured, to save power, clear USB0HSL in the AHBCLKCTRL2 register (See Section 7.5.21).
- Reset: The USB0 device can be reset by toggling USB0D_RST, bit 25, in PRESETCTRL2 (See Section 7.5.10).
- Interrupt: The USB0 interrupt is connected to interrupt slot # 28 in the NVIC. The USB0_NEEDCLK signal is connected to slot # 27. See Section 6.3.1. Clear pending interrupts before enabling them.
- Configure the USB0 main clock (see Section 37.4.7).
- Configure the USB0 wake-up signal (see Section 37.7.6) if needed.

37.4 General description

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 1 ms frames. Each frame contains a Start-Of-Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 5 logical or 10 physical endpoints including control endpoint. There are four types of transfers defined for the endpoints:

- Control transfers are used to configure the device.
- Interrupt transfers are used for periodic data transfer.
- Bulk transfers are used when the latency of transfer is not critical.
- Isochronous transfers have guaranteed delivery time but no error correction.

For more information on the Universal Serial Bus, see the USB Implementers Forum website.

The USB0 device controller enables full-speed (12 Mb/s) data exchange with a USB host controller.

Figure 162 shows the block diagram of the USB0 device controller.

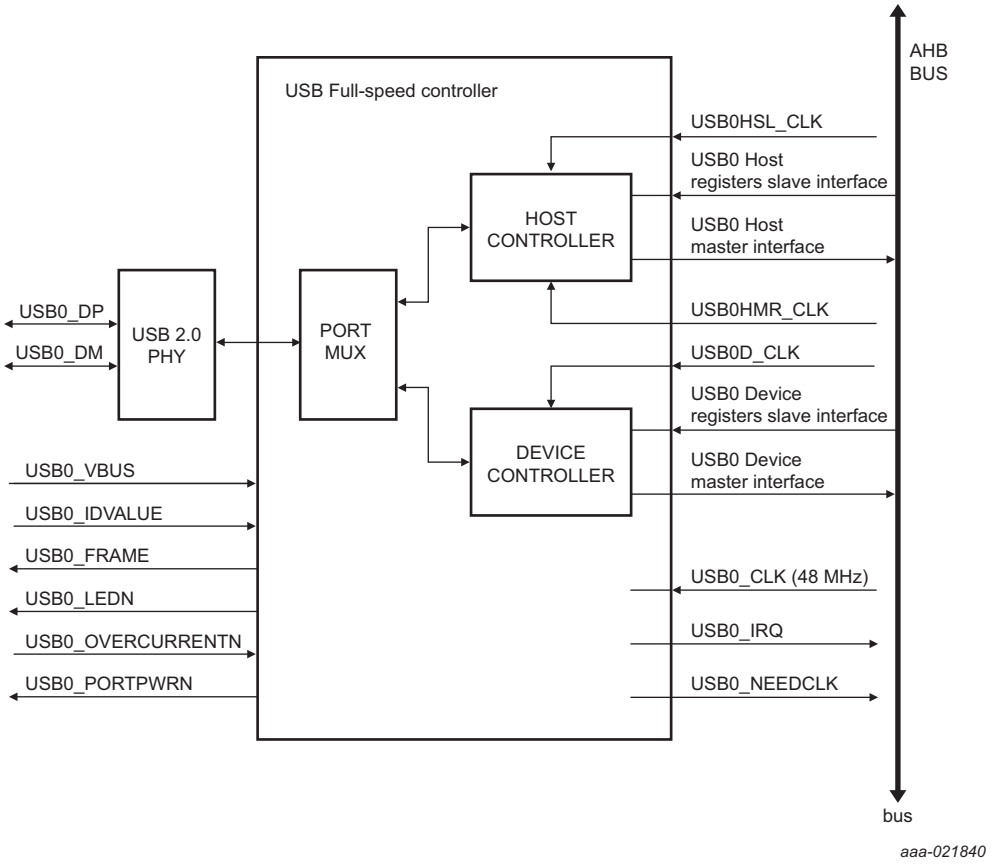


Fig 162. USB Full-speed host/device controller block diagram

37.4.1 USB0 software interface

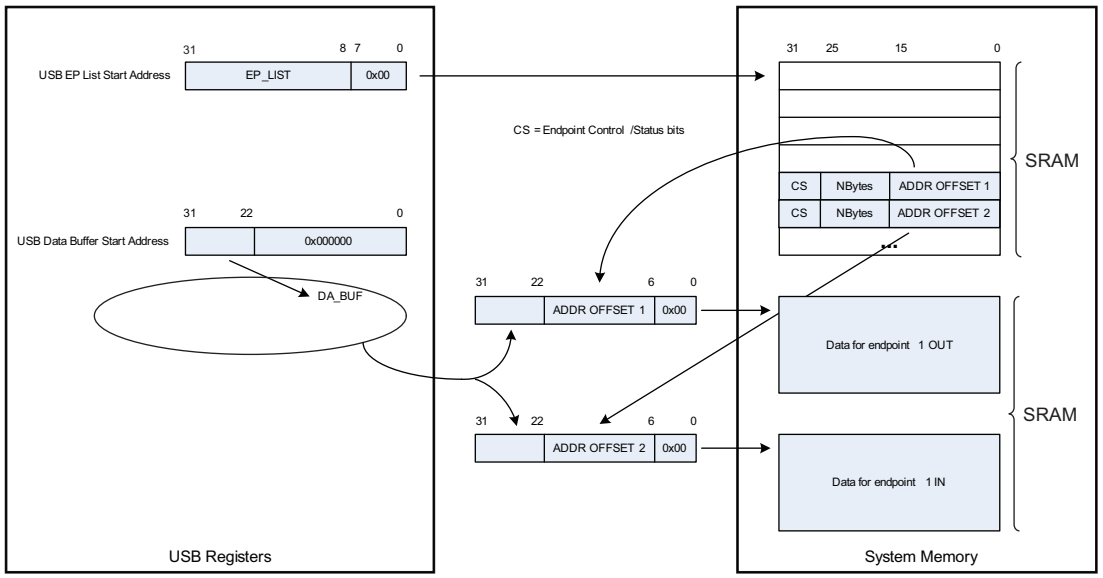


Fig 163. USB0 software interface

37.4.2 Fixed endpoint configuration

Table 890 shows the supported endpoint configurations. The packet size is configurable up to the maximum value for each type of end point.

Table 890. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
1	3	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
2	4	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
2	5	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
3	6	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
3	7	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
4	8	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
4	9	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes

37.4.3 Soft connect

The softConnect signal is implemented internally. An external pull-up resistor between USB_DP and VDD is not necessary. Software can control the pull-up by setting the DCON bit in the DEVCMDSTAT register. If the DCON bit is set to 1, the USB_DP line is pulled up to VDD through an internal 1.5 KOhm pull-up resistor.

37.4.4 Interrupts

The USB controller has two interrupt lines, a general USB interrupt (USB0) and a USB activity wake-up interrupt (USB0_NEEDCLK). See Table 88. A general interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting). See Section 37.6.9 and Section 37.6.10.

37.4.5 Suspend and resume

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100 mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
- A suspended device should draw a maximum of 500 μA.

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up if there is transmission from the host (host-initiated wake up). The USB controller also supports software initiated remote wake-up (device-initiated wake up). To initiate remote wake-up, the software on the device must enable all clocks and clear the DSUS in DEVCMDSTAT bit. This will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link Power Management defines an additional link power management state, L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (ON).

The assertion of the USB suspend signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for a suspended state.

If there is no activity for the next 2 ms, the USB0 NEEDCLK signal will go LOW. This indicates that the USB main clock can be switched off.

When any activity is detected on the USB bus, the USB NEEDCLK signal is activated. This process is fully combinatorial and therefore, no USB main clock is required to activate the USB NEEDCLK signal.

37.4.6 Frame toggle output

The USB0_FRAME output pin reflects the 1 kHz clock derived from the incoming Start of Frame tokens sent by the USB host.

37.4.7 Clocking

The USB0 device controller has the following clock connections:

- USB main clock: The USB main clock is a 48 MHz clock used for USB functions (see Section 7.5.35 and Section 7.5.55). If the FRO is used as the USB clock source, it can be configured to adjust automatically to the USB bus rate (see Section 7.5.77).
- CPU clock: The minimum frequency of the CPU clock is 12 MHz when the USB device controller is receiving or transmitting USB packets.

37.5 Pin description

The device controller can access one USB0 port.

Table 891. USB0 device pin description

Name	Port pin	IOCON function/Mode	Direction	Description
USB0_VBUS	PIO0_22/ PIO1_11/ PIO2_25/ PIO3_24	PIO0_22, function 7 PIO1_11, function 4 PIO2_25, function 2 PIO3_24, function 3 Mode: inactive	I	USB VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven HIGH internally.
USB0_DP	-	-	I/O	Positive differential data.
USB0_DM	-	-	I/O	Negative differential data.
USB0_IDVALUE	PIO0_26/ PIO2_12/ PIO4_11	PIO0_26, function 7 PIO2_12, function 3 PIO4_11, function 3 Mode: pull-up	I	A-device (host role) or B-device (peripheral role) indication. Enable this function when using a micro USB receptacle to identify whether a micro-A or micro-B plug is inserted. When enabled, the pull-up on the corresponding port pin should be enabled.
USB0_FRAME	PIO1_13/ PIO2_14/ PIO4_7	PIO1_13, function 5 PIO2_14, function 2 PIO4_7, function 4 Mode: inactive	O	1 kHz clock derived from the incoming Start of Frame tokens sent by the USB host. This is an optional function.
USB0_LEDN	PIO1_14/ PIO2_15/ PIO4_8	PIO1_14, function 5 PIO2_15, function 2 PIO4_8, function 4 Mode: inactive	O	USB connection indication. This is an optional function.
USB0_PORTPWRN	-	-	-	Host only function.
USB0_OVERCURRENTN	-	-	-	Host only function.

37.6 Register description

Table 892. Register overview: USB0 (base address: 0x4008 4000)

Name	Access	Offset	Description	Reset value	Section
DEVCMDSTAT	R/W	0x000	USB Device Command/Status register	0x00000800	37.6.1
INFO	R/W	0x004	USB Info register	0x05010000	37.6.2
EPLISTSTART	R/W	0x008	USB EP Command/Status List start address	0	37.6.3
DATABUFSTART	R/W	0x00C	USB Data buffer start address	0	37.6.4
LPM	R/W	0x010	USB Link Power Management register	0	37.6.5
EPSKIP	R/W	0x014	USB Endpoint skip	0	37.6.6
EPINUSE	R/W	0x018	USB Endpoint Buffer in use	0	37.6.7
EPBUFCFG	R/W	0x01C	USB Endpoint Buffer Configuration register	0	37.6.8
INTSTAT	R/W	0x020	USB interrupt status register	0	37.6.9
INTEN	R/W	0x024	USB interrupt enable register	0	37.6.10
INTSETSTAT	R/W	0x028	USB set interrupt status register	0	37.6.11
EPTOGGLE	R	0x034	USB Endpoint toggle register	0	37.6.12

37.6.1 USB0 Device Command/Status register

This register contains all the fields to control the behavior of the full-speed USB device.

Table 893. USB0 Device Command/Status register (DEVCMDSTAT, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value	Access
6:0	DEV_ADDR	-	USB device address. After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, the software must program the new address before completing the status phase of the SetAddress Control Request.	0	R/W
7	DEV_EN	-	USB device enable. If this bit is set, the hardware will start responding on packets for function address DEV_ADDR.	0	R/W
8	SETUP	-	SETUP token received. If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set, all received IN and OUT tokens will be NAKed by hardware. The software must clear this bit by writing a 1. If this bit is 0, the hardware will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by software.	0	R/W1C
9	FORCE_ NEEDCLK		Forces the NEEDCLK output to always be ON.	0	R/W
		0	USB_NEEDCLK has normal function.		
		1	USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.		
10	-	-	Reserved.	-	-
11	LPM_SUP		LPM Supported:	1	R/W
		0	LPM not supported.		
		1	LPM supported.		

Table 893. USB0 Device Command/Status register (DEVCMDSTAT, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value	Access
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
16	DCON	-	Device status - connect. The connect bit must be set by software to indicate that the device must signal a connect. The pull-up resistor on USB_DP will be enabled when this bit is set and the VBUSDEBOUNCED bit is one.	0	R/W
17	DSUS	-	Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device has not seen any activity on its upstream port for more than 3 ms. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1). When the device is not connected or not suspended, writing a 0 has no effect. Writing a 1 never has an effect.	0	R/W
18	-	-	Reserved.	-	-
19	LPM_SUS	-	Device status - LPM Suspend. This bit represents the current LPM suspend state. It is set to 1 by hardware when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 μs has elapsed. When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a 0 to this bit, the device will generate a remote walk-up. Software can only write a 0 to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to 1.	0	R/W
20	LPM_REWP	-	LPM Remote Wake-up Enabled by USB host. Hardware sets this bit to one when the bRemoteWake bit in the LPM extended token is set to 1. Hardware will reset this bit to 0 when it receives the host initiated LPM resume, when a remote wake-up is sent by the device or when a USB bus reset is received. Software can use this bit to check if the remote wake-up feature is enabled by the host for the LPM transaction.	0	RO
23:21	-	-	Reserved.	-	-
24	DCON_C	-	Device status - connect change. The Connect Change bit is set when the pull-up resistor of the device is disconnected because VBUS disappeared. The bit is reset by writing a 1 to it.	0	R/W1C

Table 893. USB0 Device Command/Status register (DEVCMDSTAT, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value	Access
25	DSUS_C	-	Device status - suspend change. The suspend change bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because: - The device goes in the suspended state - The device is disconnected - The device receives resume signaling on its upstream port. The bit is reset by writing a one to it.	0	R/W1C
26	DRES_C	-	Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a one to it.	0	R/W1C
27	-	-	Reserved.	-	-
28	VBUS_ DEBOUNCED	-	This bit indicates if VBUS is detected or not. The bit raises immediately when VBUS becomes high. It drops to 0 if VBUS is low for at least 3 ms. If this bit is high and the DCon bit is set, the hardware will enable the pull-up resistor to signal a connect.	0	RO
31:29	-	-	Reserved.	-	-

37.6.2 USB0 Info register

Table 894. USB0 Info register (INFO, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value	Access
10:0	FRAME_NR	-	Frame number. This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.	0	RO
14:11	ERR_CODE		The error code which last occurred:	0	R/W
		0x0	No error		
		0x1	PID encoding error		
		0x2	PID unknown		
		0x3	Packet unexpected		
		0x4	Token CRC error		
		0x5	Data CRC error		
		0x6	Time out		
		0x7	Babble		
		0x8	Truncated EOP		
		0x9	Sent/Received NAK		
		0xA	Sent Stall		
		0xB	Overrun		
		0xC	Sent empty packet		
		0xD	Bitstuff error		
		0xE	Sync error		
		0xF	Wrong data toggle		

Table 894. USB0 Info register (INFO, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value	Access
15	-	-	Reserved.	0	RO
23:16	Minrev	-	Minor revision	0x01	RO
31:24	Majrev	-	Major revision	0x05	RO

37.6.3 USB0 EP Command/Status List start address

This 32-bit register indicates the start address of the USB EP Command/Status List.

Only a subset of these bits is programmable by software. The 8 least-significant bits are hard coded to 0 because the list must start on a 256 byte boundary. Bits 31 to 8 can be programmed by software.

Table 895. USB0 EP Command/Status List start address (EPLISTSTART, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved	0	RO
31:8	EP_LIST	Start address of the USB EP Command/Status List.	0	R/W

37.6.4 USB0 Data buffer start address

This register indicates the page of the AHB address where the endpoint data can be located. The 22 LSBs are fixed to 0 so that the location resides on a 4 MB boundary. The start address of each individual endpoint's buffer is an offset to the Data buffer start address. The buffer address of the endpoint is set using the Address Offset field of the endpoint's corresponding entry in the Endpoint command/status list. See section Section 37.7.1.

Table 896. USB0 Data buffer start address (DATABUFSTART, offset 0x00C) bit description

Bit	Symbol	Description	Reset value	Access
21:0	-	The fixed portion of the data buffer start address.	0	RO
31:22	DA_BUF	Programmable portion of the data buffer start address.	0	R/W

37.6.5 USB0 Link Power Management register

Table 897. Link Power Management register (LPM, offset 0x010) bit description

Bit	Symbol	Description	Reset value	Access
3:0	HIRD_HW	Host Initiated Resume Duration - HW. This is the HIRD value from the last received LPM token	0	RO
7:4	HIRD_SW	Host Initiated Resume Duration - SW. This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.	0	R/W
8	DATA_PENDING	As long as this bit is set to 1 and LPM supported bit is set to 1, the hardware will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to 1 and this bit is 0, the hardware will return an ACK handshake on every LPM token it receives. If software has data still pending and LPM is supported, it must set this bit to 1.	0	R/W
31:9	-	Reserved	-	-

37.6.6 USB0 Endpoint skip

Table 898. USB0 Endpoint skip (EPSKIP, offset 0x014) bit description

Bit	Symbol	Description	Reset value	Access
9:0	SKIP	Endpoint skip: Writing 1 to one of these bits will indicate to hardware that it must deactivate the buffer assigned to this endpoint and return control back to the software. When hardware has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. Note: In case of double buffering, hardware will only clear the Active bit of the buffer indicated by the EPINUSE bit.	0	R/W
31:10	-	Reserved	-	-

37.6.7 USB0 Endpoint Buffer in use

Table 899. USB0 Endpoint Buffer in use (EPINUSE, offset 0x018) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	RO
9:2	BUF	Buffer in use: This register has one bit per physical endpoint: 0: HW is accessing buffer 0. 1: HW is accessing buffer 1.	0	R/W
31:10	-	Reserved	-	-

37.6.8 USB0 Endpoint Buffer Configuration

Table 900. USB0 Endpoint Buffer Configuration (EPBUFCFG, offset 0x01C) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	RO
9:2	BUF_SB	Buffer usage: This register has one bit per physical endpoint: 0: Single buffer. 1: Double buffer. If the bit is set to single buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the Active bit. If the bit is set to double buffer (1), hardware will toggle the EPINUSE bit when it clears the Active bit for the buffer.	0	R/W
31:10	-	Reserved	-	-

37.6.9 USB0 interrupt status register

Table 901. USB0 interrupt status register (INTSTAT, offset 0x020) bit description

Bit	Symbol	Description	Reset value	Access
0	EP0OUT	Interrupt status register bit for the Control EP0 OUT direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
1	EP0IN	Interrupt status register bit for the Control EP0 IN direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CI is set, this bit will also be set when a NAK is transmitted for the Control EP0 IN direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
2	EP1OUT	Interrupt status register bit for the EP1 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP1 OUT direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
3	EP1IN	Interrupt status register bit for the EP1 IN direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP1 IN direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
4	EP2OUT	Interrupt status register bit for the EP2 OUT direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP2 OUT direction. Software can clear this bit by writing a 1 to it.	0	R/W1C

Table 901. USB0 interrupt status register (INTSTAT, offset 0x020) bit description

Bit	Symbol	Description	Reset value	Access
5	EP2IN	Interrupt status register bit for the EP2 IN direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP2 IN direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
6	EP3OUT	Interrupt status register bit for the EP3 OUT direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP3 OUT direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
7	EP3IN	Interrupt status register bit for the EP3 IN direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP3 IN direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
8	EP4OUT	Interrupt status register bit for the EP4 OUT direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP4 OUT direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
9	EP4IN	Interrupt status register bit for the EP4 IN direction. This bit will be set if the corresponding Active bit is cleared by hardware. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP4 IN direction. Software can clear this bit by writing a 1 to it.	0	R/W1C
29:10	-	Reserved	-	-
30	FRAME_INT	Frame interrupt. This bit is set to 1 every millisecond when the VBUSDebounced bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a 1 to it.	0	R/W1C
31	DEV_INT	Device status interrupt. This bit is set by hardware when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a 1 to it.	0	R/W1C

37.6.10 USB0 interrupt enable register

Table 902. USB0 interrupt enable register (INTEN, offset 0x024) bit description

Bit	Symbol	Description	Reset value	Access
9:0	EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a hardware interrupt is generated on the interrupt line.	0	R/W
29:10	-	Reserved	-	-
30	FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a hardware interrupt is generated on the interrupt line.	0	R/W
31	DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a hardware interrupt is generated on the interrupt line.	0	R/W

37.6.11 USB0 set interrupt status register

Table 903. USB0 set interrupt status register (INTSETSTAT, offset 0x028) bit description

Bit	Symbol	Description	Reset value	Access
9:0	EP_SET_INT	If software writes a 1 to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
29:10	-	Reserved	-	-
30	FRAME_SET_INT	If software writes a 1 to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
31	DEV_SET_INT	If software writes a 1 to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W

37.6.12 USB0 Endpoint toggle

Table 904. USB0 Endpoint toggle (EPTOGGLE, offset 0x034) bit description

Bit	Symbol	Description	Reset value	Access
9:0	TOGGLE	Endpoint data toggle: This field indicates the current value of the data toggle for the corresponding endpoint.	0	R
31:10	-	Reserved	-	-

37.7 Functional description

37.7.1 Endpoint command/status list

Figure 164 gives an overview on how the Endpoint List is organized in memory. The USB EP command/status list start register points to the start of the list that contains all the endpoint information in memory. The order of the endpoints is fixed as shown in the picture.

USB EP Command/Status FIFO start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes										EP0 OUT Buffer Address Offset																	0x00
R	R	R	R	R	R	Reserved										SETUP bytes Buffer Address Offset																	0x04
A	R	S	TR	TV	R	EP0 IN Buffer NBytes										EP0 IN Buffer Address Offset																	0x08
R	R	R	R	R	R	Reserved										Reserved																	0x0C
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes										EP1 OUT Buffer 0 Address Offset																	0x10
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes										EP1 OUT Buffer 1 Address Offset																	0x14
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes										EP1 IN Buffer 0 Address Offset																	0x18
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes										EP1 IN Buffer 1 Address Offset																	0x1C
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes										EP2 OUT Buffer 0 Address Offset																	0x20
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes										EP2 OUT Buffer 1 Address Offset																	0x24
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes										EP2 IN Buffer 0 Address Offset																	0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes										EP2 IN Buffer 1 Address Offset																	0x2C
...																																
A	D	S	TR	RF TV	T	EP4 OUT Buffer 0 NBytes										EP4 OUT Buffer 0 Address Offset																	0x40
A	D	S	TR	RF TV	T	EP4 OUT Buffer 1 NBytes										EP4 OUT Buffer 1 Address Offset																	0x44
A	D	S	TR	RF TV	T	EP4 IN Buffer 0 NBytes										EP4 IN Buffer 0 Address Offset																	0x48
A	D	S	TR	RF TV	T	EP4 IN Buffer 1 NBytes										EP4 IN Buffer 1 Address Offset																	0x4C

Fig 164. Endpoint command/status list (see also Table 905)

Table 905. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	Active The buffer is enabled. Hardware can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to 1. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a 1 to the corresponding “skip” bit in the USB Endpoint skip register. Hardware can only write this bit to 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a 1 to the “skip” bit.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. If a USB token is received for an endpoint that has the disabled bit set, hardware will ignore the token and not return any data or handshake. When a bus reset is received, software must set the disable bit of all endpoints to 1. Software can only modify this bit when the Active bit is 0.
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The Active bit has always a higher priority than the Stall bit. This means that a Stall handshake is only sent when the Active bit is 0 and the stall bit is 1. Software can only modify this bit when the Active bit is 0.
TR	R/W	Toggle Reset When software sets this bit to 1, the hardware will set the toggle value equal to the value indicated in the “toggle value” (TV) bit. For the control endpoint 0, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to 0 when the endpoint is reset.
RF / TV	R/W	Rate Feedback mode / Toggle value For bulk endpoints and isochronous endpoints this bit is reserved and must be set to 0. For the control endpoint 0 this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. When the endpoint is used as an interrupt endpoint, it can be set to the following values. 0: Interrupt endpoint in ‘toggle mode’. 1: Interrupt endpoint in ‘rate feedback mode’. This means that the data toggle is fixed to 0 for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to 0.

Table 905. Endpoint command/status bit definitions

Symbol	Access	Description
T	R/W	Endpoint Type 0: Generic endpoint. The endpoint is configured as a bulk or interrupt endpoint. 1: Isochronous endpoint.
NBytes	R/W	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred. Note: If a short packet is received on an OUT endpoint, the Active bit will be cleared and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	R/W	Bits 21 to 6 of the buffer start address. The address offset is updated by hardware after each successful reception/transmission of a packet. Hardware increments the original value with the integer value when the packet size is divided by 64. Examples: • If an isochronous packet of 200 bytes is successfully received, the address offset is incremented by 3. • If a packet of 64 bytes is successfully received, the address offset is incremented by 1. • If a packet of less than 64 bytes is received, the address offset is not incremented.

Remark: When receiving a SETUP token for endpoint 0, the hardware will only read the SETUP bytes Buffer Address offset to know where it has to store the received SETUP bytes. The hardware will ignore all other fields. In case the SETUP stage contains more than eight bytes, it will only write the first eight bytes to memory. A USB compliant host must never send more than eight bytes during the SETUP stage.

For EP0 transfers, the hardware will do auto handshake as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer. See Figure 165 “Flowchart of control endpoint 0 - OUT direction”.

37.7.2 Control endpoint 0

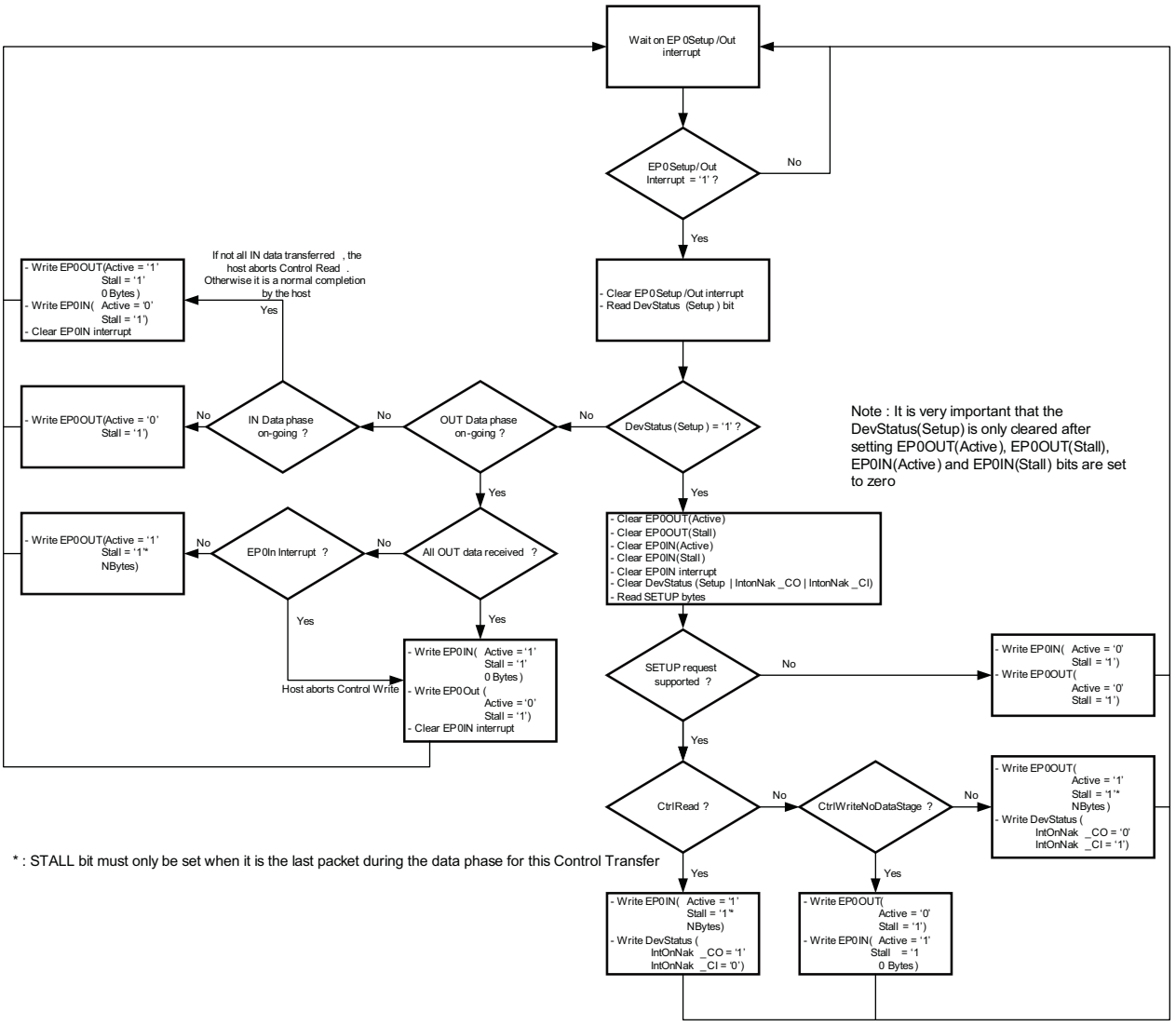
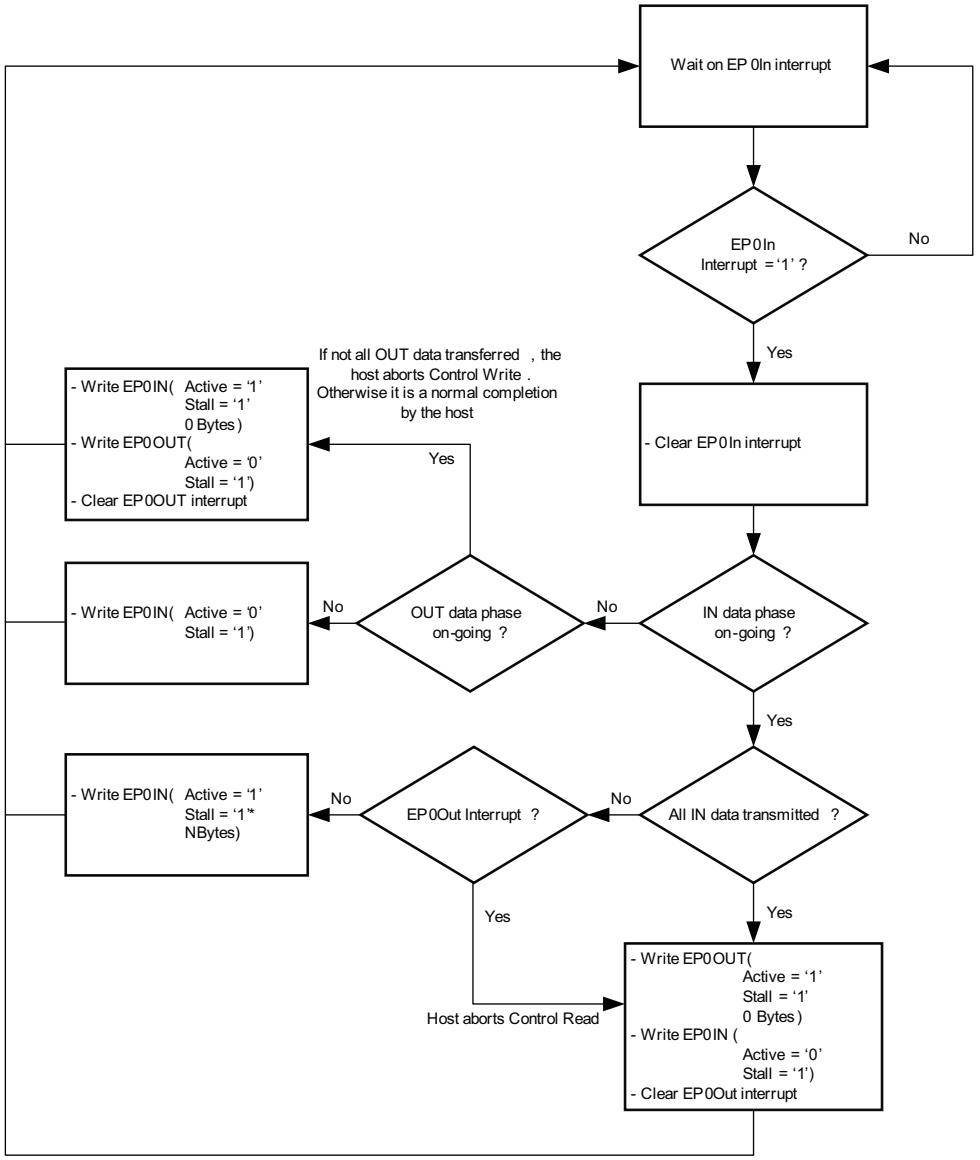


Fig 165. Flowchart of control endpoint 0 - OUT direction



* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 166. Flowchart of control endpoint 0 - IN direction

37.7.3 Generic endpoint: single buffering

To enable single buffering, software must set the corresponding "BUF_SB bit in the "USB EP Buffer Configuration" register" to 0. In the "USB EP Buffer in use" register, the software can indicate which buffer is used in this case.

When software wants to transfer data, it programs the different bits in the Endpoint command/status list entry for the desired endpoint and sets the Active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes goes to 0, hardware clears the Active bit and sets the corresponding endpoint interrupt status bit in INTSTAT.

Software must wait until hardware has cleared the Active bit to change the command/status bits in the Endpoint command/status list entry. This prevents hardware from overwriting a new value programmed by software with old values that were still cached.

If software wants to disable the Active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint SKIP bit in USB endpoint skip register (EPSKIP).

37.7.4 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding "USB EP Buffer Config" bit to 1. The "USB EP Buffer in use" register indicates which buffer will be used by hardware when the next token is received.

When hardware clears the Active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding "USB EP Buffer in use" bit.

37.7.5 Special cases

37.7.5.1 Use of the Active bit

The use of the Active bit is slightly different between OUT and IN endpoints.

When data must be received for the OUT endpoint, the software will set the Active bit to 1 and program the NBytes field to the maximum number of bytes it can receive.

When data must be transmitted for an IN endpoint, the software sets the Active bit to 1 and programs the NBytes field to the number of bytes that must be transmitted.

37.7.5.2 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0).
- The Active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to 1.

37.7.5.3 Clear Feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a Clear Feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be un-stalled and the toggle bit for that endpoint must be reset to 0. To do that the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double buffer mode, program the following:

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

37.7.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in Section 37.7.5.3 for every endpoint that will be used in this configuration.

For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to 1.

37.7.6 USB0 wake-up

37.7.6.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register (Section 37.6.1) to 0 (default) to enable automatic control of the USB0_NEEDCLK signal.
2. Wait until USB device is suspended by polling the DSUS bit in the DSVCMD_STAT register (DSUS = 1).
3. The USB0_NEEDCLK signal will be deasserted after another 2 ms. Poll the USB0CLKSTAT register until the USB0_NEEDCLK status bit is 0. See Section 7.5.66.
4. Clear pending USB activity/wake-up interrupt before enabling it. Enable the USB activity wake-up interrupt in the NVIC (# 27). See Section 6.4.1.
5. Set bit 1 in the USB0CLKCTRL register to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB0_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on the USB activity interrupt by enabling the USB0_NEEDCLK signal in the STARTER0 register (Section 7.5.90).
7. Enter deep-sleep mode via the power API (see Section 9.4.2).

The chip will automatically wake up and resume execution on USB activity.

37.7.6.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register to 0 (Section 37.6.1, default) to enable automatic control of the USB NEEDCLK signal.
2. When it is time to issue a remote wake-up, turn on the USB clock and enable the USB clock source.
3. Force the USB clock on by writing a 1 to bit FORCE_NEEDCLK (Section 37.6.1) in the DEVCMDSTAT register.
4. Write a 0 to the DSUS bit in the DSVCMD_STAT register.
5. Wait until the USB leaves the suspend state by polling the DSUS bit in the DSVCMD_STAT register (DSUS =0).
6. Clear the FORCE_NEEDCLK bit (Section 37.6.1, bit 0) in the DEVCMDSTAT to enable automatic USB clock control.

38.1 How to read this chapter

The USB full-speed controller is available on all LPC546xx devices. This chapter describes the host functionality of the controller.

38.2 Introduction

This section describes the host portion of the USB0 Full-speed controllerUSB 2.0

The USB is a four-wire bus that supports communication between a host and a number (up to 127) of peripherals. The host controller allocates the USB bandwidth to attached devices through a token based protocol. The bus supports hot plugging, un-plugging, and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host controller enables data exchange with various USB devices attached to the bus. It consists of register interface, serial interface engine, and DMA controller. The register interface complies to the OHCI specification.

Table 906. USB (OHCI) related acronyms and abbreviations used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-Performance Bus
DMA	Direct Memory Access
FS	Full Speed
LS	Low Speed
OHCI/OpenHCI	Open Host Controller Interface
USB	Universal Serial Bus

38.3 Features

- OHCI compliant.
- OpenHCI specifies the operation and interface of the USB Host Controller and SW Driver.
- The Host Controller has four USB states visible to the SW Driver:
  - USBOperational: Process Lists and generate SOF Tokens.
  - USBReset: Forces reset signaling on the bus, SOF disabled.
  - USBSuspend: Monitor USB for wake-up activity.
  - USBResume: Forces resume signaling on the bus.
- HCCA register points to Interrupt and Isochronous Descriptors List.
- ControlHeadED and BulkHeadED registers point to Control and Bulk Descriptors List.

38.4 Architecture

The architecture of the USB host controller is shown below in Figure 167.

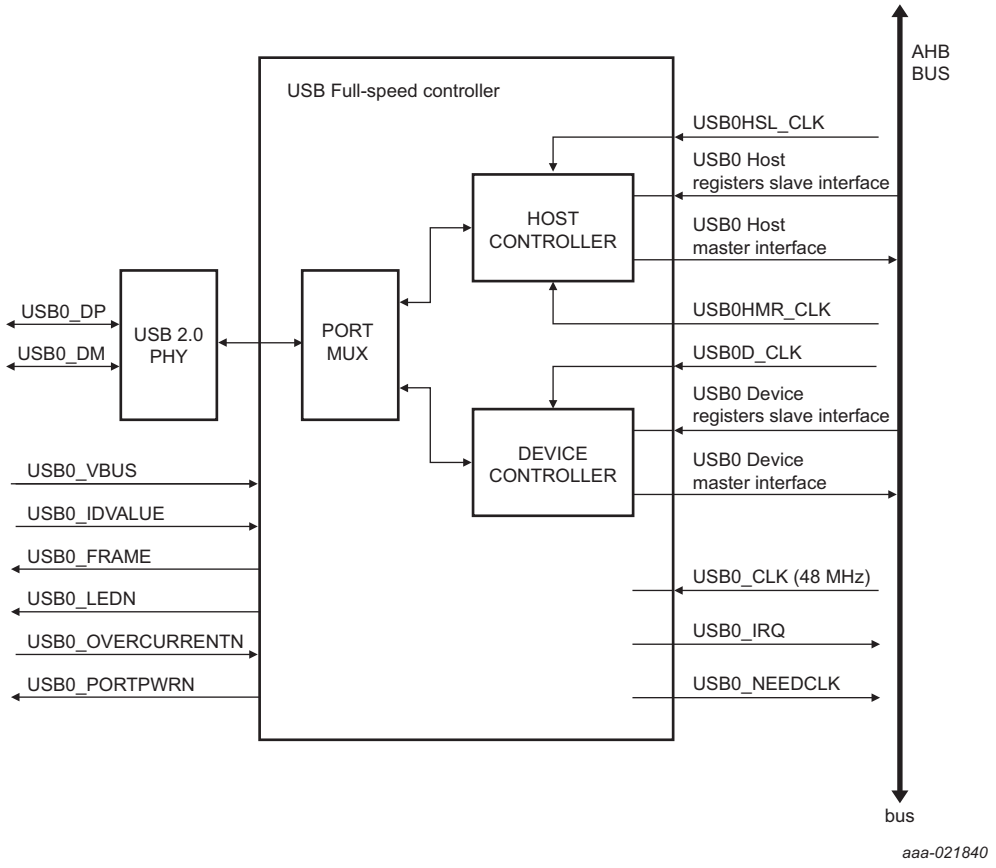


Fig 167. USB full-speed host/device controller block diagram

38.5 Basic configuration

The USB controller is configured using the following registers:

- Power: In the PDRUNCFG0 register (Table 216), set bit PDEN_USB0_PHY.

  Remark: On reset, the USB block is disabled (PDEN_USB0_PHY = 1).
- Clock: To have the full-speed USB operating, select either the System PLL, or USB PLL, or FRO clock output as the USB0 clock and the clock must be 48 MHz. The CPU clock must be configured to a minimum frequency of 12 MHz. See Section 7.5.35 and Section 7.5.55 for more details.
  In AHBCLKCTRL2, enable both the USB0 host master and host slave bits. See Section 7.5.21 for more details.
- Port control: Clear DEV_ENABLE bit in Port Mode register (Section 38.7.23) to ensure that the port is controlled by the USB0 host block. Set ID_EN to enable ID pin pull-up.
- Pins: See Table 907 “USB Host pin description” for more details.
- Reset: The USB0 Host AHB master and slave can be reset by toggling USB0HMR_RST (bit 16) and USB0HSL_RST (bit17) in PRESETCTRL2. See Section 7.5.11 for more details.

- Wake-up: Activity on the USB bus port can wake up the microcontroller from deep-sleep mode. See Section 38.6.2.1.
- Interrupts: The USB0_IRQ interrupt is connected to interrupt slot # 28 in the NVIC. The USB0_NEEDCLK signal is connected to slot # 27. See Section 6.3.1 for more details.

38.6 Interfaces

38.6.1 Pin description

Table 907. USB Host pin description

Pin name	Port pin	IOCON function, Mode	Direction	Description
USB0_DP	-	-	I/O	Positive differential data.
USB0_DM	-	-	I/O	Negative differential data.
USB0_IDVALUE	PIO0_26/ PIO2_12/ PIO4_11/	PIO0_26, function 7 PIO2_12, function 3 PIO4_11, function 3 Mode: pull-up	I	A-device (host role) or B-device (peripheral role) indication.
USB0_PORTPWRN	PIO1_3/ PIO1_12/ PIO2_14/ PIO4_7	PIO1_3, function 7 PIO1_12, function 4 PIO2_14, function 3 Mode: inactive	O	VBUS drive signal (towards external charge pump or power management unit).
USB0_OVERCURRENTN	PIO0_28/ PIO1_13/ /PIO2_15/ PIO4_8	PIO0_28, function 7 PIO1_13, function 4 PIO2_15, function 3 Mode: inactive	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry is required to detect over-current condition).
USB0_VBUS	-	-	-	Device only function.
USB0_FRAME	-	-	-	Device only function.
USB0_LEDN	-	-	-	Device only function.

38.6.2 Software interface

The software interface of the USB host block consists of a register view and the format definitions for the endpoint descriptors. See the OHCI specification for details on these two aspects. Section 38.7 shows the register map.

38.6.2.1 USB0 Host wake-up

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Send GET_STATUS command to the connected device and check if it is capable of REMOTE_WAKEUP. See "Get Status" command section in "USB Device Framework" in the USB 2.0 Specification.
2. Check HcRhPortStatus (offset 0x54) register to see if the device is connected. Set PSS to 1 to suspend the port and set HCFS bits to 11b in HcControl register (offset 0x04) to put the host controller into SUSPEND state and then set 3 ms delay for the device to suspend.
3. Set DRWE bit in HcRhStatus (offset 0x50) register to enable remote wake-up.
4. Poll the USB0CLKSTAT register until the HOST_NEED_CLKST bit is 0 (see Section 7.5.66).
5. Set POL_FS_HOST_CLK bit in the USB0CLKCTRL register to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB0 host NEEDCLK signal (see Section 7.5.65).
6. Enable the wake-up from deep-sleep mode on the USB activity interrupt by enabling the USB0_NEEDCLK signal in the STARTER0 register (Section 7.5.90).

7. Clear pending USB0 Activity Interrupt, USB0_NEEDCLK (Section 6.3.1) before enabling it.
8. Enter deep-sleep mode via power API (see Section 9.4.2). The chip will automatically wake-up and resume execution on USB activity.
9. Re-initialize the USB host controller after the USB wake-up interrupt is invoked.

38.7 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 908. Register overview: USB Host register address definitions (base address 0x400A 2000)

Name	Access	Offset	Description	Reset value	Section
HCREVISION	RO	0x00	BCD representation of the version of the HCI specification that is implemented by the Host Controller (HC).	0x10	38.7.1
HCCONTROL	R/W	0x04	Defines the operating modes of the HC.	0x0	38.7.2
HCCOMMANDSTATUS	R/W	0x08	This register is used to receive the commands from the Host Controller Driver (HCD). It also indicates the status of the HC.	0x0	38.7.3
HCINTERRUPTSTATUS	R/W	0x0C	Indicates the status on various events that cause hardware interrupts by setting the appropriate bits.	0x0	38.7.4
HCINTERRUPTENABLE	R/W	0x10	Controls the bits in the HcInterruptStatus register and indicates which events will generate a hardware interrupt.	0x0	38.7.5
HCINTERRUPTDISABLE	R/W	0x14	The bits in this register are used to disable corresponding bits in the HCInterruptStatus register and in turn disable that event leading to hardware interrupt.	0x0	38.7.6
HCHCCA	R/W	0x18	Contains the physical address of the host controller communication area.	0x0	38.7.7
HCPERIODCURRENTED	R	0x1C	Contains the physical address of the current isochronous or interrupt endpoint descriptor.	0x0	38.7.8
HCCONTROLHEADED	R/W	0x20	Contains the physical address of the first endpoint descriptor of the control list.	0x0	38.7.9
HCCONTROLCURRENTED	R/W	0x24	Contains the physical address of the current endpoint descriptor of the control list	0x0	38.7.10
HCBULKHEADED	R/W	0x28	Contains the physical address of the first endpoint descriptor of the bulk list.	0x0	38.7.11
HCBULKCURRENTED	R/W	0x2C	Contains the physical address of the current endpoint descriptor of the bulk list.	0x0	38.7.12
HCDONEHEAD	R	0x30	Contains the physical address of the last transfer descriptor added to the ‘Done’ queue.	0x0	38.7.13
HCFMINTERVAL	R/W	0x34	Defines the bit time interval in a frame and the full speed maximum packet size which would not cause an overrun.	0x2EDF	38.7.14
HCFMREMAINING	R	0x38	A 14-bit counter showing the bit time remaining in the current frame.	0x0	38.7.15

Table 908. Register overview: USB Host register address definitions (base address 0x400A 2000)

Name	Access	Offset	Description	Reset value	Section
HCFMNUMBER	R	0x3C	Contains a 16-bit counter and provides the timing reference among events happening in the HC and the HCD.	0x0	38.7.16
HCPERIODICSTART	R/W	0x40	Contains a programmable 14-bit value which determines the earliest time HC should start processing a periodic list.	0x0	38.7.17
HCLSTHRESHOLD	R/W	0x44	Contains 11-bit value which is used by the HC to determine whether to commit to transfer a maximum of 8-byte LS packet before EOF.	0x628	38.7.18
HCRHDESCRIPTORA	R/W	0x48	First of the two registers which describes the characteristics of the root hub.	0xFF000902	38.7.19
HCRHDESCRIPTORB	R/W	0x4C	Second of the two registers which describes the characteristics of the Root Hub.	0x60000	38.7.20
HCRHSTATUS	R/W	0x50	This register is divided into two parts. The lower D-word represents the hub status field and the upper word represents the hub status change field.	0x0	38.7.21
HCRHPORTSTATUS	R/W	0x54	Controls and reports the port events on a per-port basis.	0x0	38.7.22
PORTMODE	R/W	0x5C	Controls the port if it is attached to the host block or the device block.	0x0	38.7.23

38.7.1 Host controller revision register

Table 909. Host controller revision register (HCREVISION, offset 0x00) bit description

Bit	Symbol	Description	Reset value
7:0	REV	Revision. This read-only field contains the BCD representation of the version of the HCI specification that is implemented by this HC.	0x10
31:8	-	Reserved	-

38.7.2 Host controller control register

Table 910. Host controller control register (HCCONTROL, offset 0x04) bit description

Bit	Symbol	Description	Reset value
1:0	CBSR	ControlBulkServiceRatio. This specifies the service ratio between Control and Bulk EDs. Before processing any of the nonperiodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value.	0x0
	CBSR Value	**Number of Control EDs Over Bulk EDs Served**	
	0	1:1	
	1	2:1	
	2	3:1	
	3	4:1	
2	PLE	PeriodicListEnable. This bit is set to enable the processing of the periodic list in the next Frame. If cleared by HCD, processing of the periodic list does not occur after the next SOF. HC must check this bit before it starts processing the list.	0
3	IE	IsochronousEnable. This bit is used by HCD to enable/disable processing of isochronous EDs. While processing the periodic list in a Frame, HC checks the status of this bit when it finds an Isochronous ED (F=1). If set (enabled), HC continues processing the EDs. If cleared (disabled), HC halts processing of the periodic list (that contains only isochronous EDs) and begins processing the Bulk/Control lists. Setting this bit is guaranteed to take effect in the next Frame (not the current Frame).	0
4	CLE	ControlListEnable. This bit is set to enable the processing of the Control list in the next Frame. If cleared by HCD, processing of the Control list does not occur after the next SOF. HC must check this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcControlCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcControlCurrentED before re-enabling processing of the list.	0

Table 910. Host controller control register (HCCONTROL, offset 0x04) bit description

Bit	Symbol	Description	Reset value
5	BLE	BulkListEnable This bit is set to enable the processing of the Bulk list in the next Frame. If cleared by HCD, processing of the Bulk list does not occur after the next SOF. HC checks this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcBulkCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcBulkCurrentED before re-enabling processing of the list.	0
7:6	HCFS	HostControllerFunctionalState for USB 00b: USBRESET 01b: USBRESUME 10b: USBOPERATIONAL 11b: USBSUSPEND A transition to USBOPERATIONAL from another state causes SOFgeneration to begin 1 ms later. HCD may determine whether HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by HC only when in the USBSUSPEND state. HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the Root Hub and asserts subsequent reset signaling to downstream ports.	0x0
8	IR	InterruptRouting This bit determines the routing of interrupts generated by events registered in HcInterruptStatus. If clear, all interrupts are routed to the normal host bus interrupt mechanism. If set, interrupts are routed to the System Management Interrupt. HCD clears this bit upon a hardware reset, but it does not alter this bit upon a software reset. HCD uses this bit as a tag to indicate the ownership of HC.	0
9	RWC	RemoteWakeupConnected This bit indicates whether HC supports remote wake-up signaling. If remote wake-up is supported and used by the system it is the responsibility of system firmware to set this bit during POST. HC clears the bit upon a hardware reset but does not alter it upon a software reset. Remote wake-up signaling of the host system is host-bus-specific and is not described in this specification.	0
10	RWE	RemoteWakeupEnable This bit is used by HCD to enable or disable the remote wake-up feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus is set, a remote wake-up is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt.	0
31:11	-	Reserved	-

38.7.3 Host controller command status register

Table 911. Host controller command status register (HCCOMMANDSTATUS, offset 0x08) bit description

Bit	Symbol	Description	Reset value
0	HCR	HostControllerReset This bit is set by HCD to initiate a software reset of HC. Regardless of the functional state of HC, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise; For example, the InterruptRouting field of HcControl and no Host bus accesses are allowed. This bit is cleared by HC when the reset operation is completed. The reset operation must be completed within 10 μs. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.	0
1	CLF	ControlListFilled This bit is used to indicate whether there are any TDs on the Control list. It is set by HCD whenever it adds a TD to an ED in the Control list. When HC begins to process the head of the Control list, it checks CLF. As long as ControlListFilled is 0, HC will not start processing the Control list. If CF is 1, HC will start processing the Control list and will set ControlListFilled to 0. If HC finds a TD on the list, then HC will set ControlListFilled to 1 causing the Control list processing to continue. If no TD is found on the Control list, and if the HCD does not set ControlListFilled, then ControlListFilled will still be 0 when HC completes processing the Control list and Control list processing will stop.	0
2	BLF	BulkListFilled This bit is used to indicate whether there are any TDs on the Bulk list. It is set by HCD whenever it adds a TD to an ED in the Bulk list. When HC begins to process the head of the Bulk list, it checks BF. As long as BulkListFilled is 0, HC will not start processing the Bulk list. If BulkListFilled is 1, HC will start processing the Bulk list and will set BF to 0. If HC finds a TD on the list, then HC will set BulkListFilled to 1 causing the Bulk list processing to continue. If no TD is found on the Bulk list, and if HCD does not set BulkListFilled, then BulkListFilled will still be 0 when HC completes processing the Bulk list and Bulk list processing will stop.	0
3	OCR	OwnershipChangeRequest This bit is set by an OS HCD to request a change of control of the HC. When set, HC will set the OwnershipChange field in HcInterruptStatus. After the change over, this bit is cleared and remains so until the next request from OS HCD.	0
5:4	-	Reserved	-
7:6	SOC	SchedulingOverrunCount These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus has already been set. This is used by HCD to monitor any persistent scheduling problems.	0
31:8	-	Reserved	-

38.7.4 Host controller interrupt status register

The HC interrupt status register provides status on various events that cause hardware interrupts. When an event occurs, Host Controller sets the corresponding bit in this register. When a bit becomes set, a hardware interrupt is generated if the interrupt is

enabled in the HcInterruptEnable register (see Section 7.1.5) and the MasterInterruptEnable bit is set. The Host Controller Driver may clear specific bits in this register by writing 1 to bit positions to be cleared. The Host Controller Driver may not set any of these bits. The Host Controller will never clear the bit.

Table 912. Host controller interrupt status register (HCINTERRUPTSTATUS, offset 0x0C) bit description

Bit	Symbol	Description	Reset value
0	SO	SchedulingOverrun This bit is set when the USB schedule for the current Frame overruns and after the update of HccaFrameNumber. A scheduling overrun will also cause the SchedulingOverrunCount of HcCommandStatus to be incremented.	0
1	WDH	WritebackDoneHead This bit is set immediately after HC has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. HCD should only clear this bit after it has saved the content of HccaDoneHead.	0
2	SF	StartofFrame This bit is set by HC at each start of a frame and after the update of HccaFrameNumber. HC also generates a SOF token at the same time.	0
3	RD	ResumeDetected This bit is set when HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling causing this bit to be set. This bit is not set when HCD sets the USBRESUME state.	0
4	UE	UnrecoverableError This bit is set when HC detects a system error not related to USB. HC should not proceed with any processing nor signaling before the system error has been corrected. HCD clears this bit after HC has been reset.	0
5	FNO	FrameNumberOverflow This bit is set when the MSb of HcFmNumber (bit 15) changes value, from 0 to 1 or from 1 to 0, and after HccaFrameNumber has been updated	0
6	RHSC	RootHubStatusChange This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed.	0
9:7	-	Reserved	-
31:10	OC	OwnershipChange This bit is set by HC when HCD sets the OwnershipChangeRequest field in HcCommandStatus. This event, when unmasked, will always generate an System Management Interrupt (SMI) immediately. This bit is tied to 0b when the SMI pin is not implemented.	0

38.7.5 Host controller interrupt enable register

In the HcInterruptEnable register, each enable bit corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptEnable register is used to control which events generate a hardware interrupt. When a bit is set in the HcInterruptStatus register AND the corresponding bit in the HcInterruptEnable register is set AND the MasterInterruptEnable bit is set, then a hardware interrupt is requested on the host bus.

Writing a '1' to a bit in this register sets the corresponding bit, whereas writing a '0' to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Table 913. Host controller interrupt enable register (HCINTERRUPTENABLE, offset 0x10) bit description

Bit	Symbol	Value	Description	Reset value
0	SO		Scheduling Overrun interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
1	WDH		HcDoneHead Writeback interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
2	SF		Start of Frame interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
3	RD		Resume Detect interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
4	UE		Unrecoverable Error interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
5	FNO		Frame Number Overflow interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
6	RHSC		Root Hub Status Change interrupt.	0
		0	No effect.	
		1	Enables interrupt.	
29:7	-		Reserved.	-
30	OC		Ownership Change interrupt.	
		0	No effect.	
		1	Enables interrupt.	
31	MIE		Master Interrupt Enable. This is used by HCD as a Master Interrupt Enable. A 0 written to this field is ignored by HC. A 1 written to this field enables interrupt generation because of events specified in the other bits of this register.	0

38.7.6 Host controller interrupt disable register

Each disable bit in the HcInterruptDisable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptDisable register is coupled with the HcInterruptEnable register. Therefore, writing a 1 to a bit in this register clears the corresponding bit in the HcInterruptEnable register and writing a 0 to a bit in this register leaves the corresponding bit in the HcInterruptEnable register unchanged. On read, the current value of the HcInterruptEnable register is returned.

Table 914. Host controller interrupt disable register (HCINTERRUPTDISABLE, offset 0x14) bit description

Bit	Symbol	Value	Description	Reset value
0	SO		Scheduling Overrun interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
1	WDH		HcDoneHead Writeback interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
2	SF		Start of Frame interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
3	RD		Resume Detect interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
4	UE		Unrecoverable Error interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
5	FNO		Frame Number Overflow interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
6	RHSC		Root Hub Status Change interrupt.	0
		0	No effect.	
		1	Disables interrupt.	
29:7	-		Reserved.	-
30	OC		Ownership Change interrupt.	
		0	No effect.	
		1	Disables interrupt.	
31	MIE		A 0 written to this field is ignored by HC. A 1 written to this field disables interrupt generation due to events specified in the other bits of this register. This field is set after a hardware or software reset.	0

38.7.7 Host controller communication area register

Table 915. Host controller communication area register (HCHCCA, offset 0x18) bit description

Bit	Symbol	Description	Reset value
7:0	-	Reserved	0
31:8	HCCA	Base address of the Host Controller Communication Area.	-

38.7.8 Host controller period current ED register

The host controller period current ED register is used by the host controller to point to the head of one of the Periodic lists, which will be processed in the current Frame.

Table 916. Host controller period current ED register (HCPERIODCURRENTED, offset 0x1C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
31:4	PCED	The content of this register is updated by HC after a periodic ED is processed. HCD may read the content in determining which ED is currently being processed at the time of reading.	0

38.7.9 Host controller control head ED register

The Host controller control head ED register contains the physical address of the first Endpoint Descriptor of the Control list.

Table 917. Host controller control head ED register (HCCONTROLHEADED, offset 0x20) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
31:4	CHED	HC traverses the Control list starting with the HcControlHeadED pointer. The content is loaded from HCCA during the initialization of HC.	0

38.7.10 Host controller control current ED register

Table 918. Host controller control current ED register (HCCONTROLCURRENTED, offset 0x24) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
31:4	CCED	ControlCurrentED. This pointer is advanced to the next ED after serving the current one. HC continues to process the list from where it left off in the last Frame. When it reaches the end of the Control list, HC checks the ControlListFilled (CLF) bit in the HcCommandStatus. If set, it copies the content of HcControlHeadED to HcControlCurrentED and clears the bit. If not set, it does nothing. HCD is allowed to modify this register only when the ControlListEnable of HcControl is cleared. When set, HCD only reads the instantaneous value of this register. Initially, this is set to 0 to indicate the end of the Control list.	0

38.7.11 Host controller bulk head ED register

Table 919. Host controller bulk head ED register (HCBULKHEADED, offset 0x28) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
31:4	BHED	BulkHeadED HC traverses the bulk list starting with the HcBulkHeadED pointer. The content is loaded from HCCA during the initialization of HC.	0

38.7.12 Host controller bulk current ED register

Table 920. Host controller bulk current ED register (HCBULKCURRENTED, offset 0x2C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
31:4	BCED	BulkCurrentED This is advanced to the next ED after the HC has served the current one. HC continues to process the list from where it left off in the last Frame. When it reaches the end of the Bulk list, HC checks the ControlListFilled of HcControl. If set, it copies the content of HcBulkHeadED to HcBulkCurrentED and clears the bit. If it is not set, it does nothing. HCD is only allowed to modify this register when the BulkListEnable of HcControl is cleared. When set, the HCD only reads the instantaneous value of this register. This is initially set to 0 to indicate the end of the Bulk list.	0

38.7.13 Host controller done head register

Table 921. Host controller done head register (HCDONEHEAD, offset 0x30) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
31:4	DH	DoneHead When a TD is completed, HC writes the content of HcDoneHead to the NextTD field of the TD. HC then overwrites the content of HcDoneHead with the address of this TD. This is set to 0 whenever HC writes the content of this register to HCCA. It also sets the WritebackDoneHead of HcInterruptStatus.	0

38.7.14 Host controller frame interval register

Table 922. Host controller frame interval register (HCFMINTERVAL, offset 0x34) bit description

Bit	Symbol	Description	Reset value
13:0	FI	FrameInterval This specifies the interval between two consecutive SOFs in bit times. The nominal value is set to be 11,999. HCD should store the current value of this field before resetting HC. By setting the HostControllerReset field of HcCommandStatus, the HC resets this field to its nominal value. HCD may choose to restore the stored value on completion of the Reset sequence.	0x2EDF
15:14	-	Reserved	-
30:16	FSMPS	FSLargestDataPacket This field specifies a value which is loaded into the Largest Data Packet Counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the HCD.	-
31	FIT	FrameIntervalToggle HCD toggles this bit whenever it loads a new value to FrameInterval.	0

38.7.15 Host controller frame remaining register

The Host controller frame remaining register is a 14-bit down counter showing the bit time remaining in the current frame.

Table 923. Host controller frame remaining register (HCFMREMAINING, offset 0x38) bit description

Bit	Symbol	Description	Reset value
13:0	FR	FrameRemaining This counter is decremented at each bit time. When it reaches 0, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit time boundary. When entering the USBOPERATIONAL state, HC re-loads the content with the FrameInterval of HcFmInterval and uses the updated value from the next SOF.	2EDFh
30:14	-	Reserved	-
31	FRT	FrameRemainingToggle This bit is loaded from the FrameIntervalToggle field of HcFmInterval whenever FrameRemaining reaches 0. This bit is used by HCD for the synchronization between FrameInterval and FrameRemaining.	0

38.7.16 Host controller frame number register

The Host controller frame number register is a 16-bit counter. It provides a timing reference among events happening in the host controller and the host controller driver. The host controller driver may use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

Table 924. Host controller frame number register (HCFMNUMBER, offset 0x3C) bit description

Bit	Symbol	Description	Reset value
15:0	FN	FrameNumber This is incremented when HcFmRemaining is re-loaded. It will be rolled over to 0h after FFFH. When entering the USBOPERATIONAL state, this is incremented automatically. The content is written to HCCA after HC has incremented the FrameNumber at each frame boundary and sent a SOF but before HC reads the first ED in that Frame. After writing to HCCA, HC sets the StartofFrame in HcInterruptStatus.	0
31:16	-	Reserved	-

38.7.17 Host controller periodic start register

The Host controller periodic start register has a 14-bit programmable value that determines the earliest time when HC should start processing the periodic list.

Table 925. Host controller periodic start register (HCPERIODICSTART, offset 0x40) bit description

Bit	Symbol	Description	Reset value
13:0	PS	PeriodicStart After a hardware reset, this field is cleared and then set by HCD during the HC initialization. The value is calculated approximately as 10% off from HcFmInterval.. A typical value will be 3E67h. When HcFmRemaining reaches the value specified, processing of the periodic lists will have priority over Control/Bulk processing. HC will therefore start processing the Interrupt list after completing the current Control or Bulk transaction that is in progress.	0
31:11	-	Reserved	-

38.7.18 Host controller LS threshold register

The Host controller LS threshold register contains an 11-bit value used by the host controller to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF. The host controller and the host controller driver are not allowed to change this value.

Table 926. Host controller LS threshold register (HCLSTHRESHOLD, offset 0x44) bit description

Bit	Symbol	Description	Reset value
11:0	LST	LSThreshold This field contains a value which is compared to the FrameRemaining field prior to initiating a Low Speed transaction. The transaction is started only if FrameRemaining ≥ this field. The value is calculated by HCD with the consideration of transmission and setup overhead.	0x628
31:12	-	Reserved	-

38.7.19 Host controller root hub descriptor A register

The host controller root hub descriptor A register is the first register describing the characteristics of the Root Hub. Reset values are implementation specific. The descriptor length (11), descriptor type, and hub controller current (0) fields of the hub Class Descriptor are emulated by the HCD. All other fields are located in the HcRhDescriptorA and HcRhDescriptorB registers.

Table 927. Host controller root hub descriptor register (HCRHDESCRIPTORA offset 0x48) bit description

Bit	Symbol	Description	Reset value
7:0	NDP	NumberDownstreamPorts These bits specify the number of downstream ports supported by the root hub. It is implementation-specific. The minimum number of ports is 1. The maximum number of ports supported by OpenHCI is 15.	0x2
8	PSM	PowerSwitchingMode This bit is used to specify how the power switching of the root hub ports is controlled. It is implementation-specific. This field is only valid if the NoPowerSwitching field is cleared. 0: all ports are powered at the same time. 1: each port is powered individually. This mode allows portpower to be controlled by either the global switch or per-port switching. If the PortPowerControlMask bit is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, the port is controlled only by the global power switch (Set/ClearGlobalPower).	1
9	NPS	NoPowerSwitching These bits are used to specify whether power switching is supported or port are always powered. It is implementationspecific. When this bit is cleared, the PowerSwitchingMode specifies global or per-port switching. 0: ports are power switched. 1: ports are always powered on when the HC is powered on.	0
10	DT	DeviceType This bit specifies that the root hub is not a compound device. The root hub is not permitted to be a compound device. This field should always read/write 0.	0
11	OCPM	OverCurrentProtectionMode This bit describes how the overcurrent status for the root hub ports are reported. At reset, this fields should reflect the same mode as PowerSwitchingMode. This field is valid only if the NoOverCurrentProtection field is cleared. 0: over-current status is reported collectively for all downstream ports. 1: over-current status is reported on a per-port basis.	1
12	NOCP	NoOverCurrentProtection This bit describes how the overcurrent status for the root hub ports are reported. When this bit is cleared, the OverCurrentProtectionMode field specifies global or per-port reporting. 0: over-current status is reported collectively for all downstream ports. 1: no overcurrent protection supported.	0
23:13	-	Reserved	-
31:24	POTPGT	PowerOnToPowerGoodTime This byte specifies the duration the HCD has to wait before accessing a powered-on port of the root hub. It is implementation-specific. The unit of time is 2 ms. The duration is calculated as POTPGT * 2 ms.	0xF

38.7.20 Host controller root hub descriptor B register

The host controller root hub descriptor B register is the second register describing the characteristics of the root hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

Table 928. Host controller root hub descriptor register (HCRHDESCRIPTORB offset 0x4C) bit description

Bit	Symbol	Description	Reset value
15:0	DR	DeviceRemovable Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable. bit 0: reserved bit 1: device attached to port #1. bit 2: device attached to port #2. ..... bit 15: device attached to port #15.	0
31:16	PPCM	PortPowerControlMask Each bit indicates if a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode = 0), this field is not valid. bit 0: Reserved. bit 1: Ganged-power mask on port #1. bit 2: Ganged-power mask on port #2. ... bit15: Ganged-power mask on port #5.	0

38.7.21 Host controller root hub status register

The Host controller root hub status register is divided into two parts. The lower word of a Dword represents the hub status field and the upper word represents the hub status change field. Reserved bits should always be written 0.

Table 929. Host controller root hub status register (HCRHSTATUS register offset 0x50) bit description

Bit	Symbol	Description	Reset value
0	LPS	(read) LocalPowerStatus The Root Hub does not support the local power status feature; thus, this bit is always read as 0. (write) ClearGlobalPower In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a 0 has no effect.	0
1	OCI	OverCurrentIndicator This bit reports overcurrent conditions when the global reporting is implemented. When set, an overcurrent condition exists. When cleared, all power operations are normal. If per-port overcurrent protection is implementedthis bit is always 0.	0
14:2	-	Reserved	-

Table 929. Host controller root hub status register (HCRHSTATUS register offset 0x50) bit description

Bit	Symbol	Description	Reset value
15	DRWE	(read) DeviceRemoteWakeupEnable This bit enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 = ConnectStatusChange is not a remote wakeup event. 1 = ConnectStatusChange is a remote wakeup event. (write) SetRemoteWakeupEnable Writing a '1' sets DeviceRemoveWakeupEnable. Writing a 0 has no effect.	0
16	LPSC	(read) LocalPowerStatusChange The root hub does not support the local power status feature. Therefore, this bit is always read as 0. (write) SetGlobalPower In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a 0 has no effect.	0
17	OCIC	OverCurrentIndicatorChange This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a 1. Writing a 0 has no effect.	0
30:18	-	Reserved	-
31	CRWE	(write) ClearRemoteWakeupEnable Writing a 1 clears DeviceRemoveWakeupEnable. Writing a 0 has no effect.	0

38.7.22 Host controller root hub port status [1:NDP] register

Remark: In LPC546xx, Number Downstream Ports (NDP) = 1

The HcRhPortStatus[1:NDP] register is used to control and report port events on a per-port basis. NumberDownstreamPorts represents the number of HcRhPortStatus registers that are implemented in the hardware. The lower word is used to reflect the port status and the upper word reflects the status change bits. Some status bits are implemented with special write behavior. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes. Reserved bits should always be written 0.

Table 930. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) bit description

Bit	Symbol	Description	Reset value
0	CCS	(read) CurrentConnectStatus This bit reflects the current state of the downstream port. 0 = no device connected. 1 = device connected. (write) ClearPortEnable The HCD writes a 1 to this bit to clear the PortEnableStatus bit. Writing a 0 has no effect. The CurrentConnectStatus is not affected by any write. **Remark:** This bit is always read 1b when the attached device is nonremovable (DeviceRemoveable[NDP]).	0
1	PES	(read) PortEnableStatus This bit indicates whether the port is enabled or disabled. The root hub may clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error, such as, babble is detected. The change also causes PortEnabledStatusChange to be set. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable.This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. 0 = port is disabled. 1 = port is enabled. (write) SetPortEnable The HCD sets PortEnableStatus by writing a 1. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port.	0
2	PSS	(read) PortSuspendStatus This bit indicates the port is suspended or in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. 0 = port is not suspended. 1 = port is suspended. (write) SetPortSuspend The HCD sets the PortSuspendStatus bit by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus, instead, it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port.	0

Table 930. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) bit description *…continued*

Bit	Symbol	Description	Reset value
3	POCI	(read) PortOverCurrentIndicator This bit is only valid when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal. 0 = no overcurrent condition. 1 = overcurrent condition detected. (write) ClearSuspendStatus The HCD writes a 1 to initiate a resume. Writing a 0 has no effect. A resume is initiated only if PortSuspendStatus is set.	0
4	PRS	(read) PortResetStatus When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. 0 = port reset signal is not active. 1 = port reset signal is active. (write) SetPortReset The HCD sets the port reset signaling by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus, instead, sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port.	0
7:5	-	Reserved	-
8	PPS	(read) PortPowerStatus This bit reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared if an overcurrent condition is detected. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. The PowerSwitchingMode and PortPortControlMask[NDP] determine which power control switches are enabled. In global switching mode (PowerSwitchingMode=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PowerSwitchingMode=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. 0 = port power is off. 1 = port power is on. The HCD writes a 1 to set the PortPowerStatus bit. Writing a 0 has no effect.	0

Table 930. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) bit description *...continued*

Bit	Symbol	Description	Reset value
9	LSDA	(read) LowSpeedDeviceAttached This bit indicates the speed of the device attached to this port. When set, a Low Speed device is attached to this port. When clear, a Full Speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. 0 = full speed device attached. 1 = low speed device attached. (write) ClearPortPower The HCD clears the PortPowerStatus bit by writing a 1 to this bit. Writing a 0 has no effect.	0
15:10	-	Reserved	-
16	CSC	ConnectStatusChange This bit is set whenever a connect or disconnect event occurs. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status because these writes should not occur if the port is disconnected. 0 = no change in CurrentConnectStatus. 1 = change in CurrentConnectStatus. (write) ClearPortPower The HCD clears the PortPowerStatus bit by writing a 1 to this bit. Writing a 0 has no effect. **Remark:** If the DeviceRemovable[NDP] bit is set, this bit is set only after a root hub reset to inform the system that the device is attached.	0
17	PESC	PortEnableStatusChange This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = no change in PortEnableStatus. 1 = change in PortEnableStatus.	0
18	PSSC	PortSuspendStatusChange This bit is set when the full resume sequence is completed. This sequence includes the 20 ms K-state resume pulse. LS EOP, and 3 ms resychronization delay. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. This bit is also cleared when ResetStatusChange is set. 0 = resume is not completed. 1 = resume completed.	0

Table 930. Host controller root hub port status register (HCRHPORTSTATUS[1:NDP] register offset 0x54) bit description *...continued*

Bit	Symbol	Description	Reset value
19	OCIC	PortOverCurrentIndicatorChange This bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when root hub changes the PortOverCurrentIndicator bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = no change in PortOverCurrentIndicator. 1 = PortOverCurrentIndicator has changed.	0
20	PRSC	PortResetStatusChange This bit is set at the end of the 10 ms port reset signal. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 = port reset is not complete. 1 = port reset is complete.	0
31:21	-	Reserved	-

38.7.23 PortMode register

The port mode register controls the host or device role in addition to setting the polarity of the ID pin.

Table 931. Port Mode (PORTMODE, offset, 0x5C)

Bit	Symbol	Description	Reset value
0	ID	Port ID pin value. This bit indicates the value on the ID line of the port This field only contains a valid value some time after the ID_EN bit is set to 1.	0
7:1	-	Reserved	-
8	ID_EN	Port ID pin pull-up enable. If this bit is set, the pull-up on the ID line of the port will be enabled. This bit must be set to read a good value in the ID field.	0
15:9	-	Reserved	-
16	DEV_ENABLE	1: device 0: host	0
31:17	-	Reserved	-

38.8 USB Host Register Definitions

See the OHCI specification for more details on the OHCI registers.

39.1 How to read this chapter

The USB1 high-speed controller is available on selected LPC546xx devices.

The USB1 contains the USB RAM, which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB1 is not in use.

This chapter describes the device functionality of the controller.

39.2 Features

- USB2.0 high-speed device controller.
- Supports 12 physical (6 logical) endpoints including control endpoints.
- Supports single and double buffering.
- Each non-control endpoint supports bulk, interrupt, or isochronous endpoint types.
- Supports wake-up from deep-sleep mode on USB activity and remote wake-up.
- Supports A device/B device detection.
- Supports Link Power Management (LPM).

39.3 Basic configuration

Initial configuration of the USB1 device controller:

- Pins: Configure the USB1 pins in the IOCON register block. See Table 933.
- Clocks:
  - To have high-speed USB operating, the CPU clock must be configured to a minimum frequency of 60 MHz. Also, the external crystal oscillator and PLL must be configured to 48 MHz (USB clock input). See Section 7.5.36 and Section 7.5.56 for more details.
  - Set USB1D and USB1RAM bits in the AHBCLKCTRL2 register to enable USB device configuration. See Section 7.5.21.
- Power: Enable the following bits to power-up the USB device controller: See PDRUNCFG0 and PDRUNCFG1 register for more details (Section 7.5.84 and Section 7.5.85).
  - PDEN_VD2_ANA must be powered-up for external crystal oscillator to work.
  - PDEN_VD3 must be powered-up for USB PLL to work.
  - PDEN_VD5 must be powered-up for USB PHY to work.
  - PDEN_XTAL, PDEN_USB1_PLL, and PDEN_USB1_PHY must be powered-up.
- Port Control configuration:
  - Enable port control configuration by setting the USB1 host clock control in AHBCLKCTRL2 register. See Section 7.5.21.

- Check DEV_ENABLE bit 16 in PortMode register (offset 0x50) and set bit 16 to 1 to ensure that the port is routed to USB1 device controller. See Section 40.5.20 for more details.
- To save power, disable USB1 host clock control in SYSAHBCLKCTRL2 register.

- Reset:
  - The USB1 Device and RAM can be reset by toggling bit 5 (USB1D_RST) and bit 6 (USB1RAM_RST5) in PRESETCTRL2. See Section 7.5.11 for more details.

- Interrupt:
  - The USB1_IRQ interrupt is connected to interrupt slot # 47 in the NVIC. The USB1_NEEDCLK signal is connected to slot # 48. Clear pending interrupts before enabling them. See Section 6.3.1 for more details.
- Configure the USB1 wake-up signal (see Section 39.7.6) if necessary.

39.4 General description

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 125 μs frames. Each frame contains a Start Of Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 6 logical or 12 physical endpoints including control endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device.

Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the latency of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

The USB device controller enables high-speed (480 Mb/s) data exchange with a USB host controller.

Figure 168 shows the block diagram of the USB device controller.

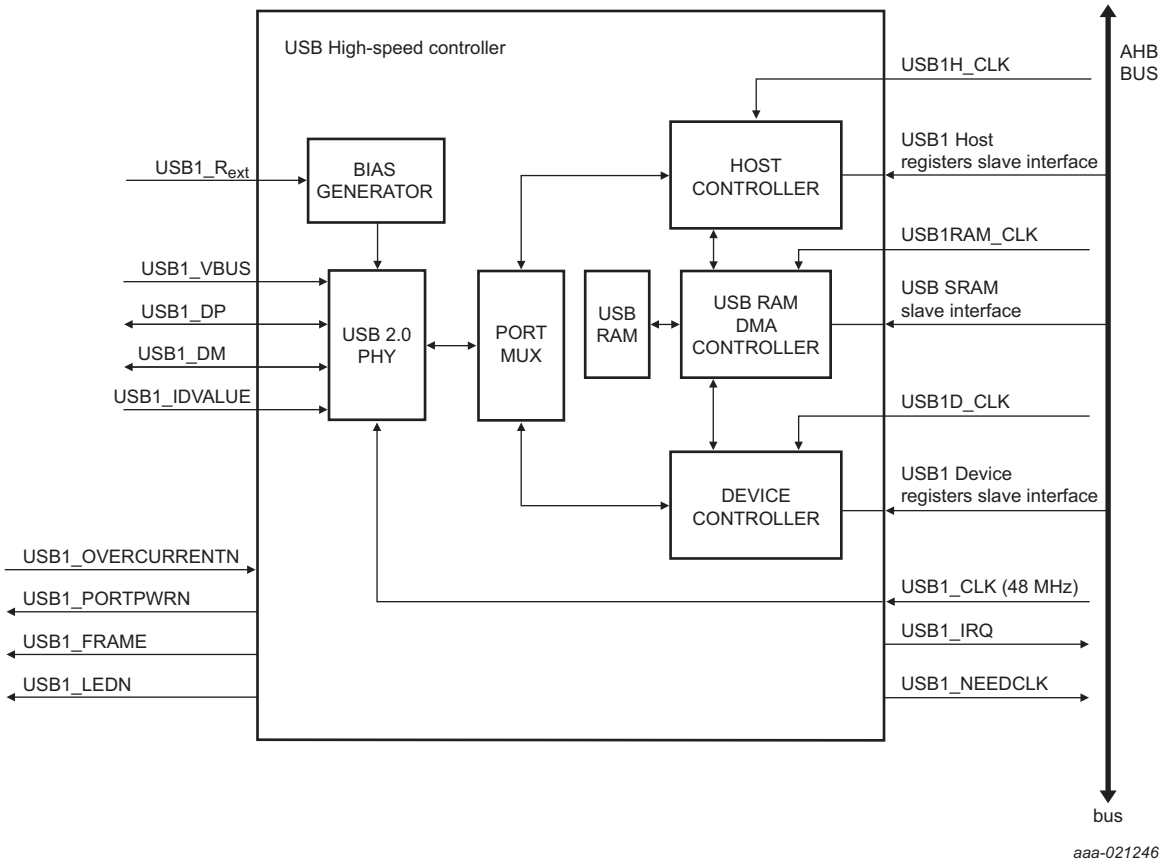


Fig 168. USB Host/Device controller block diagram

The USB device controller has a built-in analog transceiver (PHY). The USB PHY sends/receives the bi-directional USB1_DP and USB1_DM signals of the USB1 bus.

The PIE implements the high-speed USB protocol layer. It is completely hard-wired for speed and needs no software intervention. It handles transfer of data between the endpoint buffers in USB RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

39.4.1 USB1 software interface

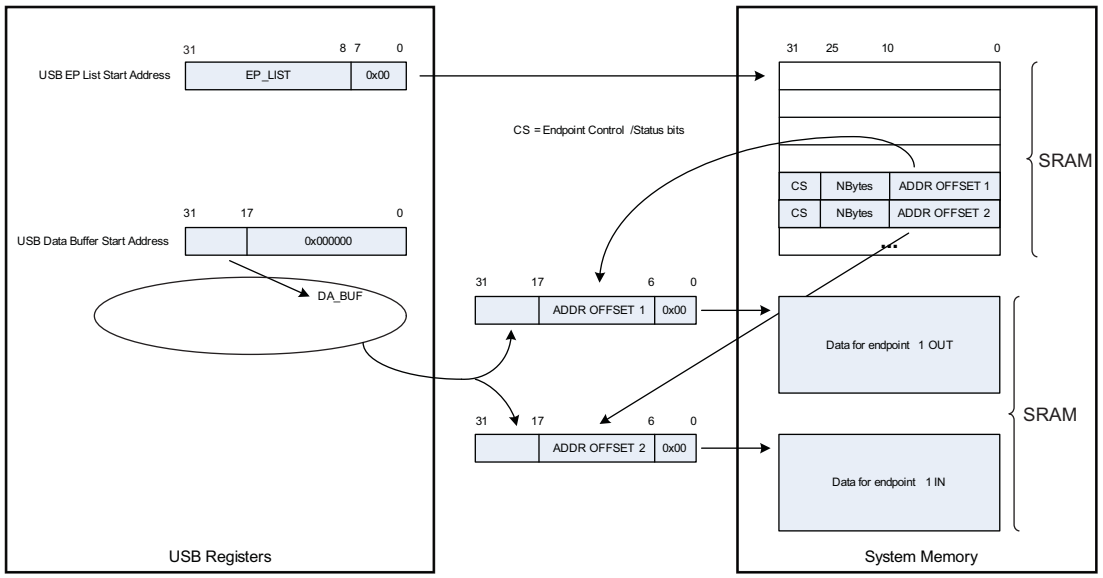


Fig 169. USB1 software interface

39.4.2 Fixed endpoint configuration

Table 932 shows the supported endpoint configurations. The packet size is configurable up to the maximum value shown in Table 932 for each type of end point.

Table 932. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
1	3	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
2	4	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
2	5	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
3	6	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
3	7	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
4	8	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
4	9	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
5	10	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
5	11	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes

39.4.3 Interrupts

The USB controller has two interrupt lines, a general USB interrupt (USB1_IRQ) and a USB activity wake-up interrupt (USB1_NEEDCLK). See Table 88. An general interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting). See Section 39.6.9 “USB1 interrupt status register” and Section 39.6.10 “USB1 interrupt enable register”.

39.4.4 Suspend and resume

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100 mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
- A suspended device should draw a maximum of 500 μA.

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up if there is transmission from the host (host-initiated wake-up). The USB controller also supports software initiated remote wake-up. To initiate remote wake-up, software on the device must enable all clocks and clear the suspend bit. This will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link Power Management defines an additional link power management state L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (On).

The assertion of USB suspend signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for suspend.

If there is no activity for the next 2 ms, the USB1 NEEDCLK signal will go low. This indicates that the USB main clock can be switched off.

When activity is detected on the USB bus, the USB NEEDCLK signal is activated. This process is fully combinatorial and hence no USB main clock is required to activate the USB NEEDCLK signal.

39.4.5 Clocking

The USB1 device controller has the following clock connections:

- USB main clock: The USB main clock is a 48 MHz clock used for USB functions (see Section 7.5.35 and Section 7.5.55).

  Remark: Unlike the USB0 full-speed controller, the FRO cannot be used as the USB high-speed clock source.
- AHB clock: The AHB system bus clock controls the USB device registers.

39.5 Pin description

Table 933. USB1 device pin description

Name	Port pin	IOCON function/Mode	Direction	Description
USB1_VBUS	-	-	I	VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven HIGH internally.
USB1_DP	-	-	I/O	Positive differential data.
USB1_DM	-	-	I/O	Negative differential data.
USB1_IDVALUE	-	-	I	A-device (host role) or B-device (peripheral role) indication. Pull-down on the ID pin (ID pin value 0) indicates connected peripheral is a USB host controller.
USB1_FRAME	PIO1_29/ PIO2_16/ PIO4_9		O	USB1 frame toggle signal.
USB1_LEDN	PIO1_30/ PIO2_17/ PIO4_10		O	USB1-configured LED indicator (active low).
USB1_AVSSC	-	-	-	USB1 analog 3.3 V ground.
USB1_REXT	-	-	-	USB1 analog signal for reference resistor, 12.4 kΩ +/-1 %.
USB1_AVDDC3V3	-	-	-	USB1 analog 3.3 V supply.
USB1_AVDDTX3V3	-	-	-	USB1 analog 3.3 V supply for line drivers.
USB1_AVSSTX3V3	-	-	-	USB1 analog ground for line drivers.
USB1_OVERCURRENTN	-	-	-	Host only function.
USB1_PORTPWRN	-	-	-	Host only function.

39.6 Register description

Table 934. Register overview: USB1 (base address: 0x4009 4000)

Name	Access	Offset	Description	Reset value	Section
DEVCMDSTAT	R/W	0x000	USB Device Command/Status register	0x800	39.6.1
INFO	RO	0x004	USB Info register	0x200 0000	39.6.2
EPLISTSTART	R/W	0x008	USB EP Command/Status List start address	0	39.6.3
DATABUFSTART	R/W	0x00C	USB Data buffer start address	0	39.6.4
LPM	R/W	0x010	USB Link Power Management register	0	39.6.5
EPSKIP	R/W	0x014	USB Endpoint skip	0	39.6.6
EPINUSE	R/W	0x018	USB Endpoint Buffer in use	0	39.6.7
EPBUFCFG	R/W	0x01C	USB Endpoint Buffer Configuration register	0	39.6.8
INTSTAT	R/W	0x020	USB interrupt status register	0	39.6.9
INTEN	R/W	0x024	USB interrupt enable register	0	39.6.10
INTSETSTAT	R/W	0x028	USB set interrupt status register	0	39.6.11
EPTOGGLE	RO	0x034	USB Endpoint toggle register	0	39.6.12

39.6.1 USB1 device command/status register

Table 935. USB1 Device Command/Status register (DEVCMDSTAT, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value	Access
6:0	DEV_ADDR		USB device address. After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.	0	R/W
7	DEV_EN		USB device enable. If this bit is set, the HW will start responding on packets for function address DEV_ADDR.	0	R/W
8	SETUP		SETUP token received. If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set all received IN and OUT tokens will be NAKed by HW. SW must clear this bit by writing a one. If this bit is 0, HW will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by SW.	0	R/W1C
9	FORCE_ NEEDCLK		Forces the NEEDCLK output to always be on:	0	R/W
		0	USB_NEEDCLK has normal function.		
		1	USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.		
10	FORCE_VBUS	0	If this bit is set to 1, the VBUS voltage indicators from the PHY are overruled. When this bit is set, the controller will consider the VBUS to be high and signal a connect when indicated by the other bits. When this bit is low, the real V_{BUS} indications are taken into account by the controller.	0	R/W
11	LPM_SUP		LPM Supported:	1	R/W
		0	LPM not supported.		
		1	LPM supported.		

Table 935. USB1 Device Command/Status register (DEVCMDSTAT, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value	Access
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP:	0	R/W
		0	Only acknowledged packets generate an interrupt.		
		1	Both acknowledged and NAKed packets generate interrupts.		
16	DCON		Device status - connect. The connect bit must be set by software to indicate that the device must signal a connect. The pull-up resistor on USB_DP will be enabled when this bit is set and the VBUSDEBOUNCED bit is one.	0	R/W
17	DSUS		Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device has not seen any activity on its upstream port for more than 3 ms. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1). When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect.	0	R/W
18	-		Reserved	0	RO
19	LPM_SUS		Device status - LPM Suspend. This bit represents the current LPM suspend state. It is set to 1 by hardware when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 μs has elapsed. When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a 0 to this bit, the device will generate a remote walk-up. Software can only write a 0 to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to 1.	0	R/W
21:20	R		Reserved	-	-
23:22	Speed		This field indicates the speed at which the device operates: 00b: reserved 01b: full-speed 10b: high-speed 11b: super-speed (reserved for future use)	01b	RO
24	DCON_C		Device status - connect change. The connect change bit is set when the pull-up resistor of the device is disconnected because VBUS disappeared. The bit is reset by writing a 1 to it.	0	R/W1C

Table 935. USB1 Device Command/Status register (DEVCMDSTAT, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value	Access
25	DSUS_C		Device status - suspend change. The suspend change bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because: - The device goes in the suspended state. - The device is disconnected. - The device receives resume signaling on its upstream port. The bit is reset by writing a one to it.	0	R/W1C
26	DRES_C		Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a 1 to it.	0	R/W1C
27	-		Reserved	0	RO
28	VBUS DEBOUNCED		This bit indicates if VBUS is detected or not. The bit raises immediately when VBUS becomes high. It drops to 0 if VBUS is low for at least 3 ms. If this bit is high and the DCon bit is set, the hardware will enable the pull-up resistor to signal a connect.	0	RO
31:29	PHY test mode		This field is written by firmware to put the PHY into a test mode as defined by the USB2.0 specification: 000b: Test mode disabled 001b: Test_J 010b: Test_K 011b: Test_SE0_NAK 100b: Test_Packet 101b: Test_Force_Enable 110b - 111b: reserved	0	R/W

39.6.2 USB1 info register

Table 936. USB1 Info register (INFO, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value	Access
10:0	FRAME_NR		Frame number. This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.	0	RO

Table 936. USB1 Info register (INFO, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value	Access
14:11	ERR_CODE		The error code which last occurred:	0	R/W
		0x0	No error		
		0x1	PID encoding error		
		0x2	PID unknown		
		0x3	Packet unexpected		
		0x4	Token CRC error		
		0x5	Data CRC error		
		0x6	Time out		
		0x7	Babble		
		0x8	Truncated EOP		
		0x9	Sent/Received NAK		
		0xA	Sent Stall		
		0xB	Overrun		
		0xC	Sent empty packet		
		0xD	Bitstuff error		
		0xE	Sync error		
		0xF	Wrong data toggle		
15	-		Reserved.	0	RO
23:16	Minrev	-	Minor revision	0x00	RO
31:24	Majrev	-	Major revision	0x02	RO

39.6.3 USB1 EP command/status list start address

This 32-bit register indicates the start address of the USB EP Command/Status List. The USB EP Command/Status List must be placed within the USB RAM address space.

Only a subset of these bits is programmable by software. The 8 least-significant bits are hard coded to 0 because the list must start on a 256 byte boundary. Bits 19 to 8 can be programmed by software. Bits 31:20 are hard coded to 0x401, the address of the USB RAM.

Table 937. USB1 EP Command/Status List start address (EPLISTSTART, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved	0	RO
19:8	EP_LIST_PRG	Programmable portion of the USB EP Command/Status List address.	0	R/W
31:20	EP_LIST_FIXED	Fixed portion of USB EP Command/Status List address.	0x401	RO

39.6.4 USB1 data buffer start address

This register indicates the page of the AHB address where the endpoint data is located. The endpoint data must be put in the USB RAM address space, hence the reset value of this register is the start address of the RAM, and should not be changed.

The start address of each individual endpoint's buffer is an offset to the Data buffer start address. The endpoint's buffer address is set using the Address Offset field of the endpoint's corresponding entry in the "Endpoint command/status list". See section Section 39.7.1 “Endpoint command/status list”.

Table 938. USB1 Data buffer start address (DATABUFSTART, offset 0x00C) bit description

Bit	Symbol	Description	Reset value	Access
17:0	-	The fixed portion of the data buffer start address.	0	RO
31:18	DA_BUF	Programmable portion of the data buffer start address.	0	R/W

39.6.5 USB1 link power management register

Table 939. Link Power Management register (LPM, offset 0x010) bit description

Bit	Symbol	Description	Reset value	Access
3:0	HIRD_HW	Host Initiated Resume Duration - HW. This is the HIRD value from the last received LPM token	0	RO
7:4	HIRD_SW	Host Initiated Resume Duration - SW. This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.	0	R/W
8	DATA_PENDING	As long as this bit is set to one and LPM supported bit is set to one, HW will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to one and this bit is 0, HW will return an ACK handshake on every LPM token it receives. If SW has still data pending and LPM is supported, it must set this bit to 1.	0	R/W
31:9	RESERVED	Reserved	0	RO

39.6.6 USB1 endpoint skip

Table 940. USB1 Endpoint skip (EPSKIP, offset 0x014) bit description

Bit	Symbol	Description	Reset value	Access
11:0	SKIP	Endpoint skip: Writing 1 to one of these bits, will indicate to HW that it must deactivate the buffer assigned to this endpoint and return control back to software. When HW has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. Note: In case of double buffering, HW will only clear the Active bit of the buffer indicated by the EPINUSE bit.	0	R/W
31:12	-	Reserved	0	R

39.6.7 USB1 endpoint buffer in use

Table 941. USB1 Endpoint Buffer in use (EPINUSE, offset 0x018) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	R
11:2	BUF	Buffer in use: This register has one bit per physical endpoint. 0: HW is accessing buffer 0. 1: HW is accessing buffer 1.	0	R/W
31:12	-	Reserved	0	R

39.6.8 USB1 endpoint buffer configuration

Table 942. USB1 Endpoint Buffer Configuration (EPBUFCFG, offset 0x01C) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.	0	R
11:2	BUF_SB	Buffer usage: This register has one bit per physical endpoint. 0: Single buffer 1: Double buffer If the bit is set to single buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the Active bit. If the bit is set to double buffer (1), HW will toggle the EPINUSE bit when it clears the Active bit for the buffer.	0	R/W
31:12	-	Reserved	0	R

39.6.9 USB1 interrupt status register

Table 943. USB1 interrupt status register (INTSTAT, offset 0x020) bit description

Bit	Symbol	Description	Reset value	Access
0	EP0OUT	Interrupt status register bit for the Control EP0 OUT direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
1	EP0IN	Interrupt status register bit for the Control EP0 IN direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CI is set, this bit will also be set when a NAK is transmitted for the Control EP0 IN direction. Software can clear this bit by writing a one to it.	0	R/W
2	EP1OUT	Interrupt status register bit for the EP1 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP1 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
3	EP1IN	Interrupt status register bit for the EP1 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP1 IN direction. Software can clear this bit by writing a one to it.	0	R/W
4	EP2OUT	Interrupt status register bit for the EP2 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP2 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
5	EP2IN	Interrupt status register bit for the EP2 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP2 IN direction. Software can clear this bit by writing a one to it.	0	R/W

Table 943. USB1 interrupt status register (INTSTAT, offset 0x020) bit description

Bit	Symbol	Description	Reset value	Access
6	EP3OUT	Interrupt status register bit for the EP3 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP3 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
7	EP3IN	Interrupt status register bit for the EP3 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP3 IN direction. Software can clear this bit by writing a one to it.	0	R/W
8	EP4OUT	Interrupt status register bit for the EP4 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP4 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
9	EP4IN	Interrupt status register bit for the EP4 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP4 IN direction. Software can clear this bit by writing a one to it.	0	R/W
10	EP5OUT	Interrupt status register bit for the EP5 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP5 OUT direction. Software can clear this bit by writing a one to it.	0	R/W
11	EP5IN	Interrupt status register bit for the EP5 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP5 IN direction. Software can clear this bit by writing a one to it.	0	R/W
29:12	-	Reserved	-	-
30	FRAME_INT	Frame interrupt. This bit is set to one every millisecond when the VbusDebounced bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a one to it.	0	R/W
31	DEV_INT	Device status interrupt. This bit is set by HW when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a one to it.	0	R/W

39.6.10 USB1 interrupt enable register

Table 944. USB1 interrupt enable register (INTEN, offset 0x024) bit description

Bit	Symbol	Description	Reset value	Access
11:0	EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.	0	R/W
29:12	-	Reserved	0	RO
30	FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.	0	R/W
31	DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.	0	R/W

39.6.11 USB1 set interrupt status register

Table 945. USB1 set interrupt status register (INTSETSTAT, offset 0x028) bit description

Bit	Symbol	Description	Reset value	Access
11:0	EP_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
29:12	-	Reserved	0	RO
30	FRAME_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W
31	DEV_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0	R/W

39.6.12 USB1 Endpoint toggle

Table 946. USB1 Endpoint toggle (EPTOGGLE, offset 0x034) bit description

Bit	Symbol	Description	Reset value	Access
29:0	TOGGLE	Endpoint data toggle: This field indicates the current value of the data toggle for the corresponding endpoint.	0	R
31:30	-	Reserved	0	R

39.7 Functional description

39.7.1 Endpoint command/status list

Figure 170 gives an overview on how the Endpoint List is organized in memory. The USB EP Command/Status List start register points to the start of the list that contains all the endpoint information in memory. The order of the endpoints is fixed as shown in the figure.

USB EP Command/Status FIFO start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes															EP0 OUT Buffer Address Offset												0x00
R	R	R	R	R	R	Reserved															SETUP Bytes Buffer Address Offset												0x04
A	R	S	TR	TV	R	EP0 IN Buffer NBytes															EP0 IN Buffer Address Offset												0x08
R	R	R	R	R	R	Reserved															Reserved												0x0C
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes															EP1 OUT Buffer 0 Address Offset												0x10
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes															EP1 OUT Buffer 1 Address Offset												0x14
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes															EP1 IN Buffer 0 Address Offset												0x18
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes															EP1 IN Buffer Address Offset												0x1C
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes															EP2 OUT Buffer 0 Address Offset												0x20
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes															EP2 OUT Buffer 1 Address Offset												0x24
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes															EP2 IN Buffer Address Offset												0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes															EP2 IN Buffer Address Offset												0x2C
...																																
A	D	S	TR	RF TV	T	EP5 OUT Buffer 0 NBytes															EP5 OUT Buffer 0 Address Offset												0x50
A	D	S	TR	RF TV	T	EP5 OUT Buffer 1 NBytes															EP5 OUT Buffer 1 Address Offset												0x54
A	D	S	TR	RF TV	T	EP5 IN Buffer 0 NBytes															EP5 IN Buffer 0 Address Offset												0x58
A	D	S	TR	RF TV	T	EP5 IN Buffer 1 NBytes															EP5 IN Buffer 1 Address Offset												0x5C

aaa-021724

Fig 170. Endpoint command/status list (see also Table 947)

Table 947. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	Active The buffer is enabled. HW can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to 1. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a one to the corresponding “skip” bit in the USB Endpoint skip register. Hardware can only write this bit to 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a one to the “skip” bit. If hardware receives a token for an endpoint that is not active, it will return the following handshake or data: Non-isochronous endpoint: NAK handshake is sent. Isochronous IN endpoint: empty data packet is sent. Isochronous OUT endpoint: received data is ignored and no handshake is sent.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. When a bus reset is received, firmware must set the disable bit of all endpoints to 1. Software can only modify this bit when the Active bit is 0.
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The Active bit has always higher priority than the Stall bit. This means that a Stall handshake is only sent when the Active bit is 0 and the stall bit is one. Software can only modify this bit when the Active bit is 0.
TR	R/W	Toggle Reset When software sets this bit to one, the HW will set the toggle value equal to the value indicated in the “toggle value” (TV) bit. For the control endpoint 0, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to 0 when the endpoint is reset.
RF / TV	R/W	Rate Feedback mode / Toggle value For the control endpoint 0 this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. For the non-control endpoints, this bit is used together with the T-bit to identify the type of endpoint When the endpoint type (T) is set to generic endpoint, this bit selects between bulk endpoint and interrupt endpoint in rate-feedback mode. 0: Bulk endpoint with maximum packet size of 512 bytes in HS mode and 64 bytes in FS mode 1: Interrupt endpoint in ‘rate feedback mode’. This means that the data toggle is fixed to 0 for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to 0. When the endpoint type (T) is set to periodic, this bit determines if the endpoint is interrupt or isochronous. 0: Isochronous endpoint (Max Packet Size is determined by the smallest value when comparing NBytes field with 1024). 1: Interrupt endpoint (Max Packet Size is determined by the smallest value when comparing NBytes field with 1024).

Table 947. Endpoint command/status bit definitions

Symbol	Access	Description
T	R/W	Endpoint Type 0: Generic endpoint. The endpoint is configured as a bulk or rate feedback interrupt endpoint. In case of an rate feedback interrupt endpoint, the Maximum Packet Size in High-Speed mode can only be maximum 512 bytes. 1: Periodic endpoint. The RF / TV bit determines if the endpoint is isochronous or interrupt.
NBytes	R/W	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred. **Remark:** If a short packet is received on an OUT endpoint, the Active bit clears and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	R/W	Bits 16 to 6 of the buffer start address. This address offset is updated by HW after each successful reception/transmission of a packet. HW increments the original value with the rounded up integer value when the packet size is divided by 64.E.g. if a packet of 200 bytes is successfully received, the Address Offset will be incremented by 4. Examples: If a packet of 64 bytes is successfully received, the Address Offset is incremented by 1. If a packet of less than 64 bytes is received, the Address Offset is also incremented by 1. If a packet with 0 bytes is received, the Address Offset is not incremented.

Remark: When receiving a SETUP token for endpoint 0, the HW will only read the SETUP bytes Buffer Address offset to know where it has to store the received SETUP bytes. HW will ignore all other fields. In case the SETUP stage contains more than 8 bytes, it will only write the first 8 bytes to memory. A USB compliant host must never send more than 8 bytes during the SETUP stage.

For EP0 transfers, the hardware will do auto handshake as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer. See Figure 171 “Flowchart of control endpoint 0 - OUT direction”.

39.7.2 Control endpoint 0

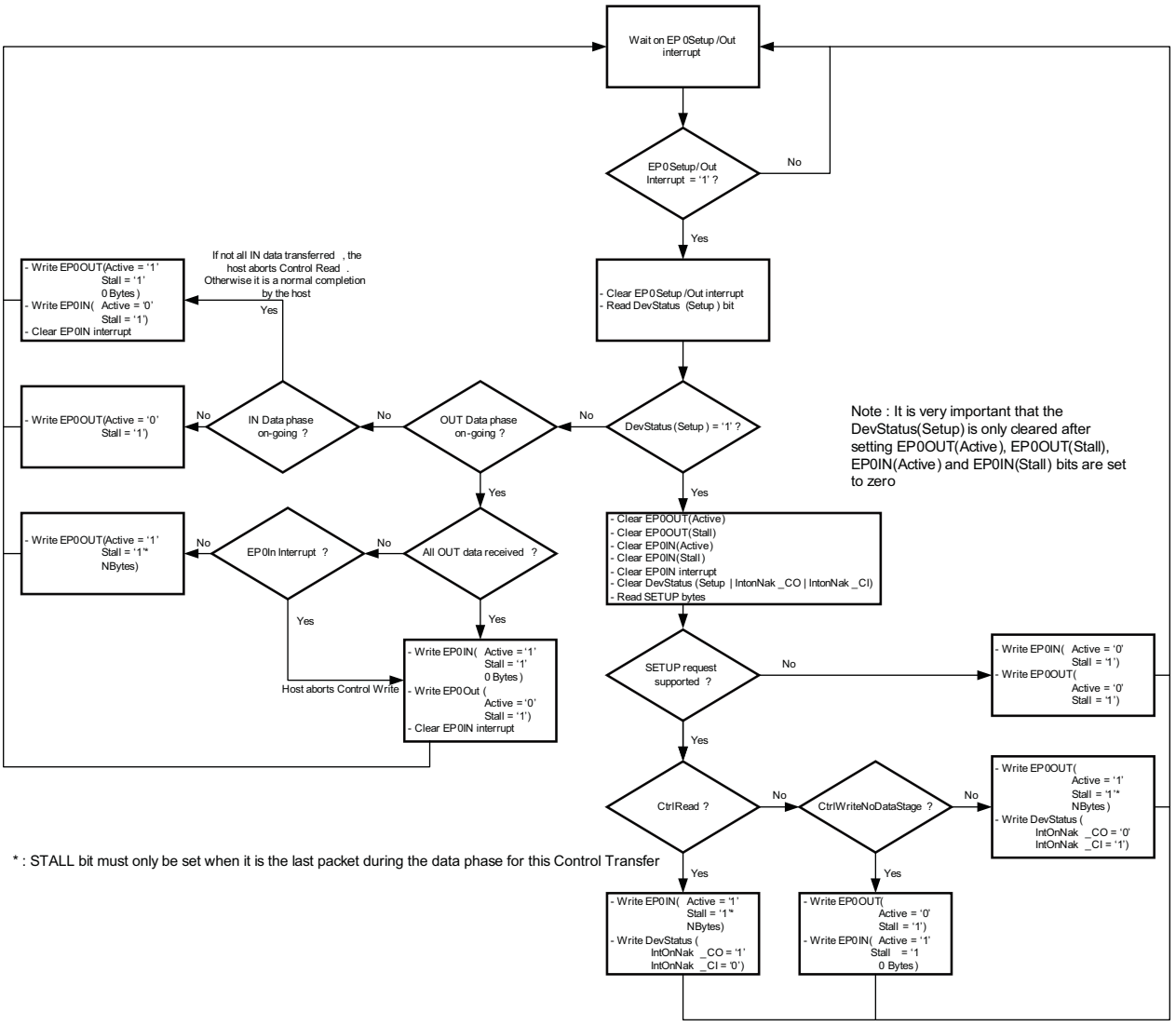
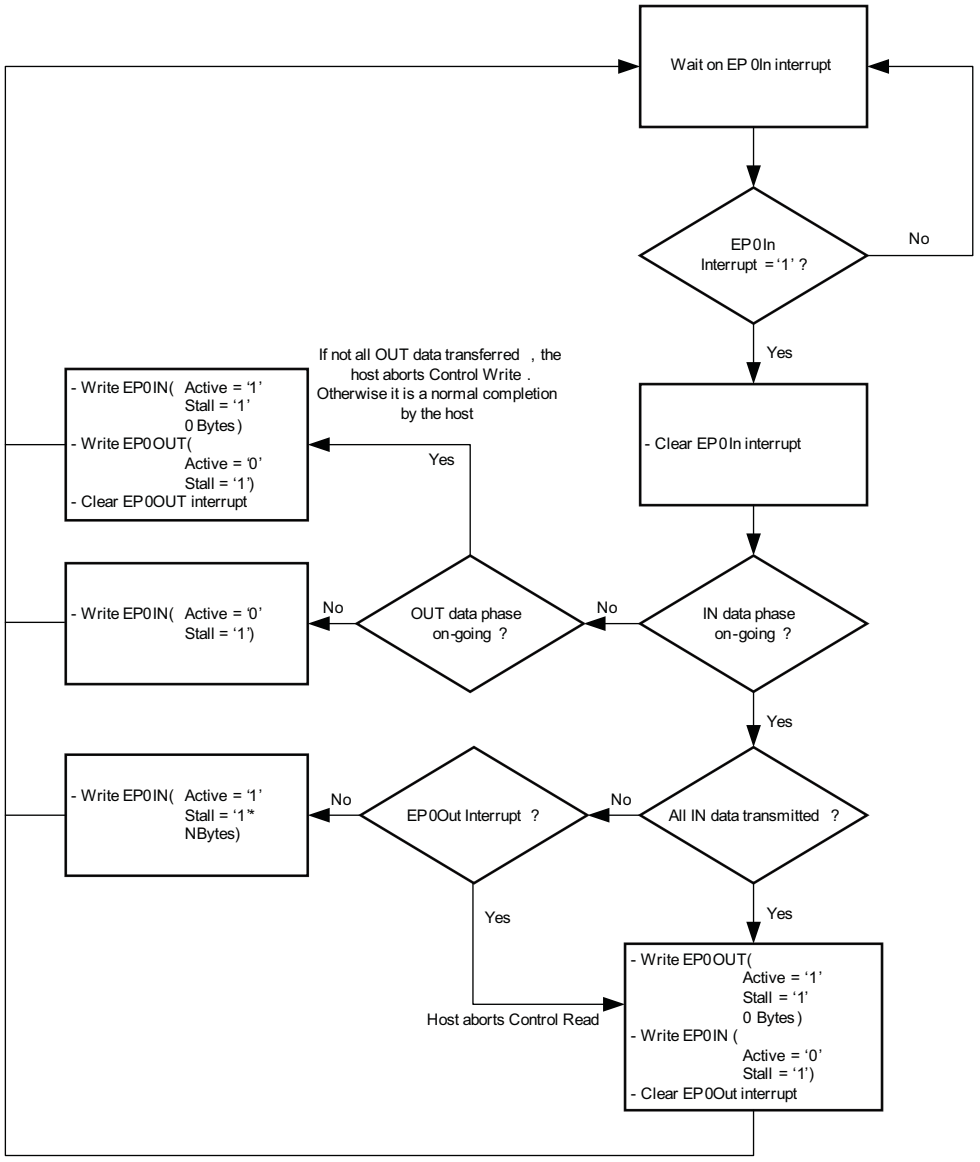


Fig 171. Flowchart of control endpoint 0 - OUT direction



* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 172. Flowchart of control endpoint 0 - IN direction

39.7.3 Generic endpoint: single buffering

To enable single buffering, software must set the corresponding "BUF_SB bit in the "USB EP Buffer Configuration" register" to 0. In the "USB EP Buffer in use" register, the software can indicate which buffer is used in this case.

When software wants to transfer data, it programs the different bits in the Endpoint command/status list entry for the desired endpoint and sets the Active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes goes to 0, hardware clears the Active bit and sets the corresponding endpoint interrupt status bit in INTSTAT.

Software must wait until hardware has cleared the Active bit to change the command/status bits in the Endpoint command/status list entry. This prevents hardware from overwriting a new value programmed by software with old values that were still cached.

If software wants to disable the Active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint SKIP bit in USB endpoint skip register (EPSKIP).

39.7.4 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding "USB EP Buffer Config" bit to 1. The "USB EP Buffer in use" register indicates which buffer will be used by hardware when the next token is received.

When hardware clears the Active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding "USB EP Buffer in use" bit.

39.7.5 Special cases

39.7.5.1 Use of the Active bit

The use of the Active bit is slightly different between OUT and IN endpoints.

When data must be received for the OUT endpoint, the software will set the Active bit to one and program the NBytes field to the maximum number of bytes it can receive.

When data must be transmitted for an IN endpoint, the software sets the Active bit to one and programs the NBytes field to the number of bytes that must be transmitted.

39.7.5.2 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0).
- The Active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to one.

39.7.5.3 Clear Feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a Clear Feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be un-stalled and the toggle bit for that endpoint must be reset back to 0. In order to do that the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double buffer mode, program the following:

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

39.7.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in Section 39.7.5.3 for every endpoint that will be used in this configuration.

For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to one.

39.7.6 USB1 wake-up

39.7.6.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake-up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register (Section 39.6.1) to 0 (default) to enable automatic control of the USB NEEDCLK signal.
2. Set PortMode register (Section 40.5.20) to enable DEV_ENABLE bit and then poll USB1CLKSTAT (Section 7.5.70) until USB1 host NEEDCLK goes low.
3. Poll the DSUS bit in the DEVCMDSTAT register (DSUS = 1) (Section 39.6.1) until the USB device is suspended. The USB1_NEEDCLK signal will be deasserted after another 2 ms. Poll the USB1CLKSTAT register until the USB1_NEEDCLK status bit is 0. (Section 7.5.70).
4. Clear pending USB Activity/wake-up interrupt before enabling it. Enable the USB activity wake-up interrupt in the NVIC (# 48). See Section 6.3.1.
5. Set bit 1 in the USB1CLKCTRL register to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB1_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on this interrupt by enabling the USB1_NEEDCLK signal in the STARTER1 register (Section 7.5.91, bit 16).
7. Set PortMode register (Section 40.5.20) to set the PHY to power down mode. Enable PortMode configuration by enabling USB1 host clock in AHBCLKCTRL2 register (Section 7.5.21). Set SW_CTRL_PDCOM to 1 to enable S/W control to the PHY. Set SW_PDCOM to 1 to put the PHY in power down mode. Once configured, to save power, disable PortMode configuration by disabling USB1 host clock in AHBCLKCTRL2 register.
8. Enter deep-sleep mode via the power API (Section 9.4.2). When power API is called, make sure PDEN_SRAM0, PDEN_USB_SRAM, PDEN_VD5, and PDEN_USB1PHY are left ON before going to deep-sleep mode.

The chip automatically wakes up and resumes execution on USB activity. Set PortMode register (Section 40.5.20) to set the PHY back to operational mode. Enable PortMode configuration by enabling USB1 host clock in AHBCLKCTRL2 register (Section 7.5.21). Set SW_CTRL_PDCOM to 1 to enable S/W control to the PHY. Set SW_PDCOM to 0 to put the PHY in operational mode. Once configured, to save power, disable PortMode configuration by disabling USB1 host clock in AHBCLKCTRL2 register.

39.7.6.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register to 0 (Section 39.6.1, default) to enable automatic control of the USB NEEDCLK signal.
2. Setup a wake-up source, for example, a PIN interrupt.
3. Force the USB clock on by writing a 1 to bit FORCE_NEEDCLK (Section 39.6.1) in the DEVCMDSTAT register.
4. Write a 0 to the DSUS bit in the DEVCMDSTAT register (Section 39.6.1).
5. Wait until the USB device leaves the suspend state by polling the DSUS bit in the DEVCMDSTAT register (DSUS =0).

40.1 How to read this chapter

The USB1 high-speed controller is available on selected LPC546xx devices.

The USB1 contains the USB RAM, which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB1 is not in use.

This chapter describes the host functionality of the controller.

40.2 Introduction

The USB1 high-speed controller provides a plug-and-play connection of peripheral devices to a host with three different data speeds: high-speed with a data rate of 480 Mbps, full-speed with a data rate of 12 Mbps, and low-speed with a data rate of 1.5 Mbps. Many portable devices can benefit from the ability to communicate to each other over the USB interface without intervention of a host PC.

40.2.1 Features

- Contains on-chip high-speed UTMI+ compliant transceiver (PHY).
- Supports all high-speed, full-speed, and low-speed USB-compliant peripherals.
- Complies with *Universal Serial Bus specification 2.0.*
- Supports a hardware/software interface similar to the *Enhanced Host Controller Interface* (EHCI) specification.
- Supports USB 2.0 extension LPM mode.
- Supports port power switching.
- Supports power management.
- Integrated DMA engine can be used together with the audio PLL for USB streaming applications.

40.2.2 Architecture

Figure 173 shows the architecture of the USB host controller.

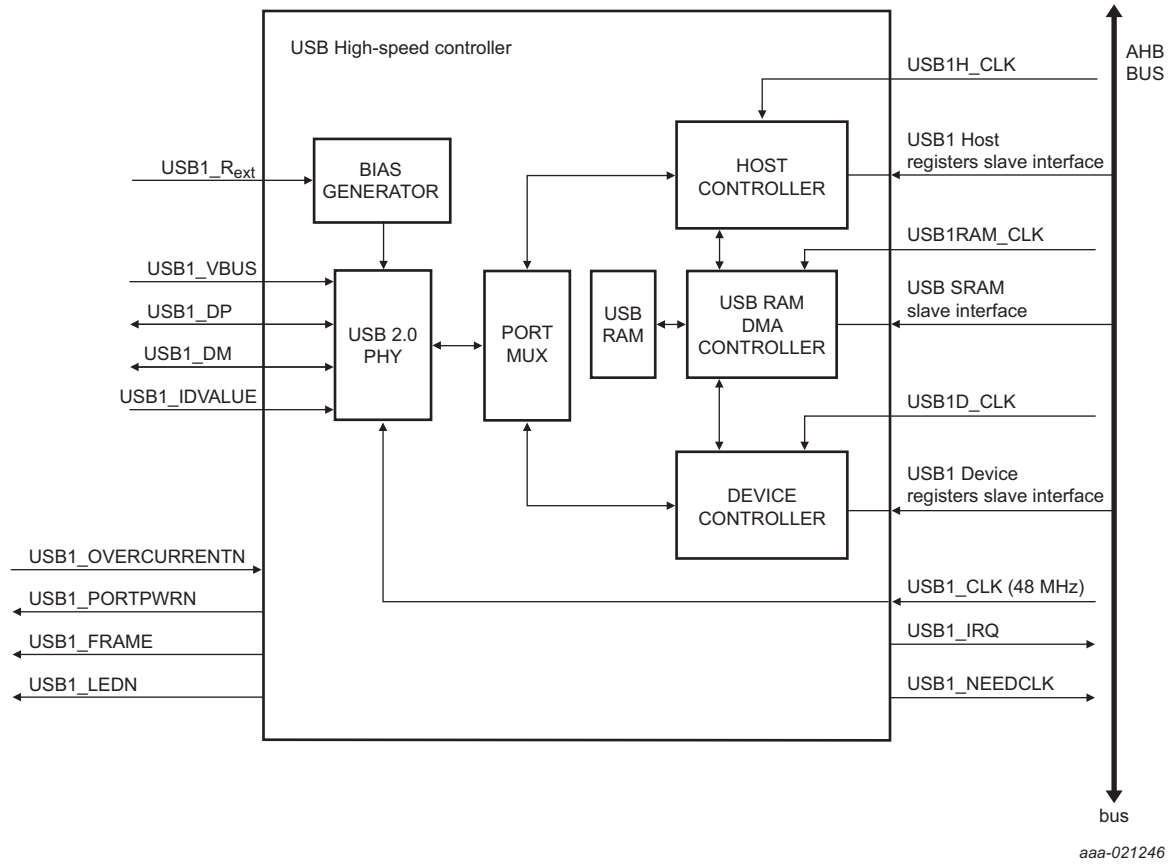


Fig 173. USB Host controller block diagram

40.3 Basic configuration

Initial configuration of the USB1 host controller:

Power:

- PDEN_VD2_ANA must be powered up for external crystal oscillator to work.
- PDEN_VD3 must be powered up for USB PLL to work.
- PDEN_VD5 must be powered up for USB PHY to work.
- PDEN_XTAL, PDEN_USB1_PLL, and PDEN_USB1_PHY must be powered up.
- See Section 7.5.84 and Section 7.5.85 for more details.

Clock:

To have high-speed USB operating, the CPU clock must be configured to a minimum frequency of 60 MHz. Also, the external crystal oscillator and PLL must be configured to 48 MHz (USB clock input). See Section 7.5.35 and Section 7.5.56 for more details.

Port Control configuration:

Set DEV_ENABLE bit to 0 in PortMode register (offset 0x50) to ensure that the port is routed to USB1 host controller. See Section 40.5.20.

Pins:

See Table 948 “USB Host pin description”.

Reset:

The USB1 Host and RAM can be reset by toggling bit 4 (USB1H_RST) and 6 (USB1RAM_RST) in PRESETCTRL2. See Section 7.5.9.

Interrupt:

The USB1 interrupt is connected to interrupt slot # 47 in the NVIC. The USB1_NEEDCLK interrupt is connected to slot # 48. See Section 6.3.1.

Remark: Software must ensure that there is a maximum of one outstanding PTD in the list for the same device address, endpoint number, endpoint direction combination. If this rule is violated, there is a risk that the USB HS host hardware will send the packets in the wrong order.

40.4 Interfaces

40.4.1 Pin description

Table 948 describes the USB host pins.

Table 948. USB Host pin description

Pin name	Port pin	IOCON function/Mode	Direction	Description
USB1_PORTPWRN	PIO1_2/ PIO1_29/ PIO2_16	7/4/3, pull-up	O	VBUS drive signal (towards external charge pump or power management unit); indicates that VBUS must be driven (active LOW). (Depending on the location of the pin you use, IOCON function may vary.)
USB1_OVERCURRENTN	PIO1_1/ PIO1_30/ PIO2_17	7/4/3, pull-up	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry required to detect over-current condition; depending on the location of the pin you use, IOCON function may vary).
USB1_DP	-	-	I/O	Positive differential data
USB1_DM	-	-	I/O	Negative differential data
USB1_IDVALUE	-	-	I	A-device (host role) or B-device (peripheral role) indication. Pull-down on the ID pin (ID pin value 0) indicates connected peripheral is a USB host controller.
USB1_AVSSC	-	-	-	USB1 analog 3.3 V ground.
USB1_REXT	-	-	-	USB1 analog signal for reference resistor, 12.4 kΩ +/-1 %.
USB1_AVDDC3V3	-	-	-	USB1 analog 3.3 V supply.
USB1_AVDDTX3V3	-	-	-	USB1 analog 3.3 V supply for line drivers.

Table 948. USB Host pin description

Pin name	Port pin	IOCON function/Mode	Direction	Description
USB1_AVSSTX3V3	-	-	-	USB1 analog ground for line drivers.
USB0_VBUS	-	-	-	Device only function.
USB0_FRAME	-	-	-	Device only function.
USB0_LEDN	-	-	-	Device only function.

40.4.2 Software interface

The AHB slave interface of the USB host must be used to access the registers and to configure the mode of the port (host mode or device mode). Figure 174 shows which part of the data is stored in registers and the location of the data in the USB RAM.

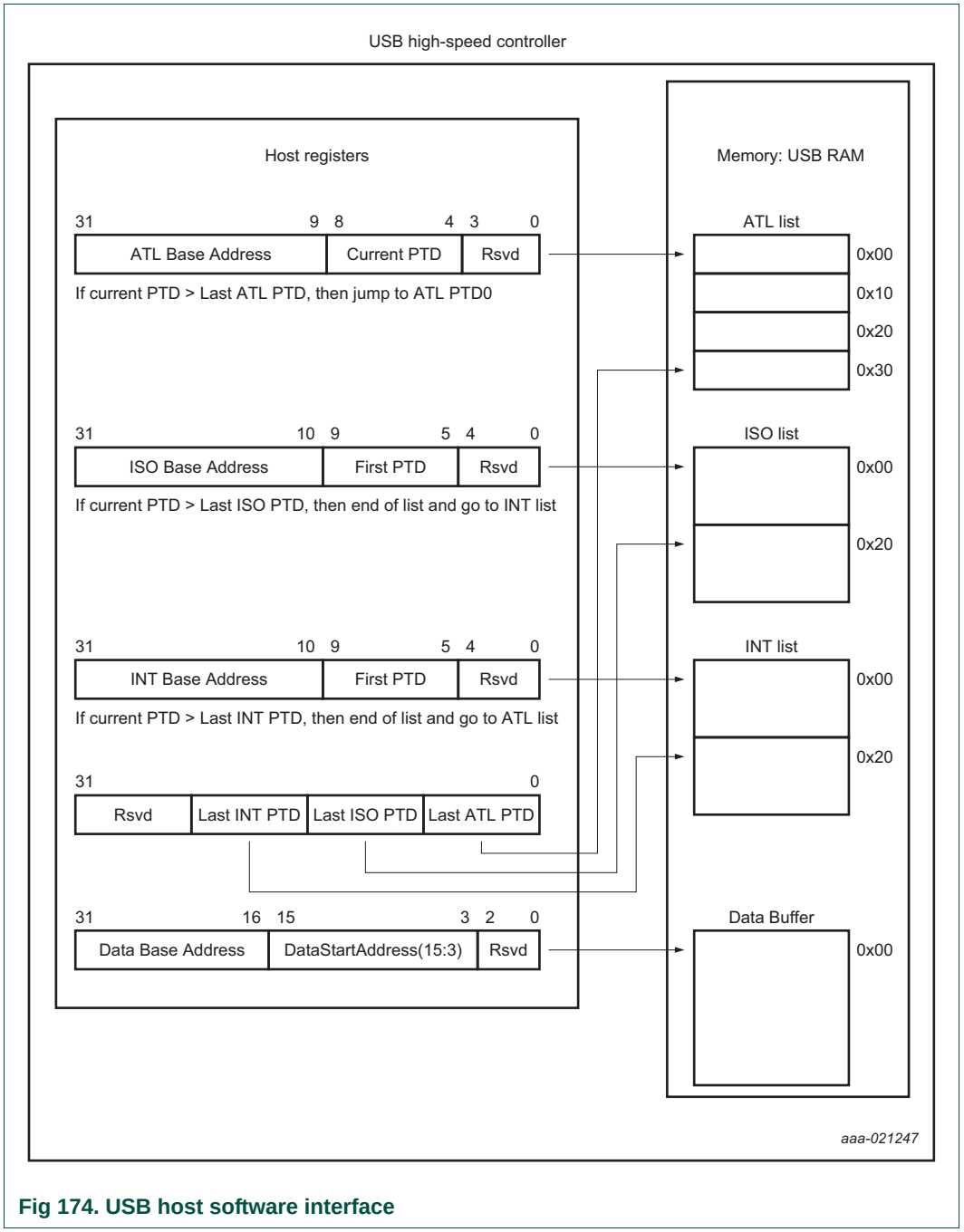


Fig 174. USB host software interface

40.5 Register description

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 949. Register overview: USB high-speed device controller (base address 0x400A 3000)

Name	Access	Offset	Description	Reset value [1]	Section
CAPLENGTH/CHIPID	RO	0x00	This register contains the offset value towards the start of the operational register space and the version number of the IP block.	0x01010010	40.5.1
HCSPARAMS	RO	0x04	Host Controller Structural Parameters.	0x00010011	40.5.2
HCCPARAMS	RO	0x08	Host Controller Capability Parameters.	0x00020006	40.5.3
FLADJ_FRINDEX	R/W	0x0C	Frame Length Adjustment.	0x00000020	40.5.4
ATLPTD	R/W	0x10	Memory base address where ATL PTD0 is stored.	0x00000000	40.5.5
ISOPTD	R/W	0x14	Memory base address where ISO PTD0 is stored.	0x00000000	40.5.6
INTPTD	R/W	0x18	Memory base address where INT PTD0 is stored.	0x00000000	40.5.7
DATAPAYLOAD	R/W	0x1C	Memory base address that indicates the start of the data payload buffers.	0x00000000	40.5.8
USBCMD	R/W	0x20	USB Command register.	0x00000000	40.5.9
USBSTS	R/W1C	0x24	USB Interrupt Status register.	0x00000000	40.5.10
USBINTR	R/W	0x28	USB Interrupt Enable register.	0x00000000	40.5.11
PORTSC1	R/W	0x2C	Port Status and Control register.	0x00000000	40.5.12
ATLPTD	R/W1C	0x30	Done map for each ATL PTD.	0x00000000	40.5.13
ATLPTD	R/W	0x34	Skip map for each ATL PTD.	0x00000000	40.5.14
ISOPTD	R/W1C	0x38	Done map for each ISO PTD.	0x00000000	40.5.15
ISOPTD	R/W	0x3C	Skip map for each ISO PTD.	0x00000000	40.5.16
INTPTD	R/W1C	0x40	Done map for each INT PTD.	0x00000000	40.5.17
INTPTD	R/W	0x44	Skip map for each INT PTD.	0x00000000	40.5.18
LASTPTD	R/W	0x48	Marks the last PTD in the list for ISO, INT and ATL.	0x00000000	40.5.19
PORTMODE	R/W	0x50	Controls the port if it is attached to the host block or the device block.	0x00040000	40.5.20

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

40.5.1 CAPLENGTH/CHIPID register

The CAPLENGTH/CHIPID register describes the capability length and the revision of the USB IP.

Table 950. Capability Length_Chip Identification register (CAPLENGTH_CHIPID, offset 0x00) bit description

Bits	Symbol	Description
7:0	CAPLENGTH	Capability Length: This is used as an offset. It is added to the register base to find the beginning of the operational register space.
15:8	R	Reserved
31:16	CHIPID	Chip identification: indicates major and minor revision of the IP: • [31:24] = Major revision • [23:16] = Minor revision Major revisions used: 0x01: USB2.0 high-speed host

40.5.2 HCSPARAMS register

The HCSPARAMS register describes the USB host port configuration.

Table 951. Host Controller Structural Parameters register (HCSPARAMS, offset 0x04) bit description

Bits	Symbol	Description
3:0	N_PORTS	This register specifies the number of physical downstream ports implemented on this host controller. This is fixed to 0x1 for this IP.
4	PPC	This field indicates whether the host controller implementation includes port power control. The value of this bit is controlled by the generic C_PORTPOWER_CONTROL.
15:5	R	Reserved
16	P_INDICATOR	This bit indicates whether the ports support port indicator control. The value of this bit is controlled by the generic C_PORT_INDICATORS.
31:17	R	Reserved

40.5.3 HCCPARAMS register

The HCSPARAMS register describes the USB host controller capability.

Table 952. Host Controller Capability Parameters (HCCPARAMS, offset 0x08) bit description

Bits	Symbol	Description
16:0	R	Reserved
17	LPMC	Link Power Management Capability. This indicates host controller support for the Link Power Management L1 state and associated PORTSC Suspend using L1, Suspend status and Device Address fields.
31:18	R	Reserved

40.5.4 FLADJ register (Address Offset = 0x0C)

The FLADJ register controls the SOF frame length timing and the frame index.

Table 953. Frame Length Adjustment (FLADJ, offset 0x0C) bit description

Bits	Symbol	Description
5:0	FLADJ	Frame Length Timing Value. Each decimal value change to this register corresponds to 16 high-speed bit times. The SOF cycle time (number of SOF counter clock periods to generate a SOF micro-frame length) is equal to 59488 + value in this field. The default value is decimal 32 (20h), which gives a SOF cycle time of 60000.
15:6	R	Reserved
29:16	FRINDEX	Frame Index: Bits 29 to16 in this register are used for the frame number field in the SOF packet. The value in this field is incremented by one every 125 μs (independent of the speed of the attached device). Software is only allowed to update this field when the Run/Stop bit is set to 0.
31:30	R	Reserved

40.5.5 ATL PTD BaseAddress register

The ATL PTD base address register configures the start address of the ATL list.

Table 954. ATL PTD Base Address (ATL PTD BaseAddress, offset 0x10) bit description

Bits	Symbol	Description
3:0	R	Reserved
8:4	ATL_CUR	This indicates the current PTD that is used by the hardware when it is processing the ATL list.
31:9	ATL_BASE	Base address to be used by the hardware to find the start of the ATL list.

Remark: The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

40.5.6 ISO PTD BaseAddress register

The ISO PTD base address register configures the start address of the ISO list.

Table 955. ISO PTD Base Address (ISO PTD BaseAddress, offset 0x14) bit description

Bits	Symbol	Description
4:0	R	Reserved
9:5	ISO_FIRST	This indicates the first PTD that is used by the hardware when it is processing the ISO list.
31:10	ISO_BASE	Base address to be used by the hardware to find the start of the ISO list.

Remark: The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

40.5.7 INT PTD BaseAddress register

The INT PTD base address register configures the start address of the INT list.

Table 956. INT PTD Base Address (INT PTD BaseAddress, offset 0x18) bit description

Bits	Symbol	Description
4:0	R	Reserved
9:5	INT_FIRST	This indicates the first PTD that is used by the hardware when it is processing the INT list.
31:10	INT_BASE	Base address to be used by the hardware to find the start of the INT list.

Remark: The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

40.5.8 Data Payload BaseAddress register

The data payload base address register configures the start address of the data payload list.

Table 957. Data Payload Base Address (Data Payload BaseAddress, offset 0x1C) bit description

Bits	Symbol	Description
15:0	R	Reserved
31:16	DAT_BASE	Base address to be used by the hardware to find the start of the data payload section. The hardware will only use the least significant bits required for addressing the correct location in RAM.

40.5.9 USBCMD register

The USB Command register controls the overall execution of the USB host scheduler.

Table 958. USB Command register (USBCMD, offset 0x20) bit description

Bits	Symbol	Description
0	RS	Run/Stop: 1b = Run. The host controller executes the schedule. 0b = Stop. If this bit is set to 1b, the USB clock will be always-on.
1	HCRESET	Host Controller Reset: This control bit is used by the software to reset the host controller
3:2	FLS	Frame List Size: This field specifies the size of the frame list. This field is used to control when the frame list roll over interrupt bit must be set. 00b 1024 elements 01b 512 elements 10b 256 elements 11b Reserved
6:4	R	Reserved
7	LHCR	Light Host Controller Reset: This bit allows the driver software to reset the host controller without affecting the state of the ports.
8	ATL_EN	ATL List enabled. When this bit is set, the hardware will process the ATL list.
9	ISO_EN	ISO List enabled. When this bit is set, the hardware will process the ISO list.
10	INT_EN	INT List enabled. When this bit is set, the hardware will process the INT list.

Table 958. USB Command register (USBCMD, offset 0x20) bit description

Bits	Symbol	Description
23:11	R	Reserved
27:24	HIRD	Host-Initiated Resume Duration. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from a LPM state (example, L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM token's bmAttributes field) on entry into a low-power state. The host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. A host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM token, specifically: a value 0000b equals 50us and each additional increment adds 75 μs. For example, 0001b equals 125 μs and a value 1111b equals 1175 μs.
31:28	R	Reserved

40.5.10 USBSTS register

The USB Interrupt Status register shows the interrupt status.

Table 959. USB Interrupt Status register (USBSTS, offset 0x24) bit description

Bits	Symbol	Description
1:0	R	Reserved
2	PCD	Port Change Detect: The host controller sets this bit to logic 1 when any port has a change bit transition from a 0 to a one or a Force Port Resume bit transition from a 0 to a 1 as a result of a J-K transition detected on a suspended port. Software must write a one to clear the bit.
3	FLR	Frame List Rollover: The host controller sets this bit to logic 1 when the frame list index rolls over its maximum value to 0. Software must write a one to clear the bit.
15:4	R	Reserved
16	ATL_IRQ	ATL IRQ: Indicates that an ATL PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No ATL PTD event occurred. 1 - ATL PTD event occurred. Software must write a one to clear the bit.
17	ISO_IRQ	ISO IRQ: Indicates that an ISO PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No ISO PTD event occurred. 1 - ISO PTD event occurred. Software must write a one to clear the bit.

Table 959. USB Interrupt Status register (USBSTS, offset 0x24) bit description

Bits	Symbol	Description
18	INT_IRQ	INT IRQ: Indicates that an INT PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0 - No INT PTD event occurred. 1 - INT PTD event occurred. Software must write a one to clear the bit.
19	SOF_IRQ	SOF interrupt: Every time when the host sends a Start of Frame token on the USB bus, this bit is set. Software must write a one to clear the bit.
31:20	R	Reserved

40.5.11 USBINTR register

The USB Interrupt Enable register enables or disables the interrupt. If the enable bit is set to one and the corresponding USBSTS bit is set to one, a hardware interrupt is generated.

Table 960. USB Interrupt Enable register (USBINTR, offset 0x28) bit description

Bits	Symbol	Description
1:0	R	Reserved
2	PCDE	Port Change Detect Interrupt Enable: 1: enable 0: disable
3	FLRE	Frame List Rollover Interrupt Enable: 1: enable 0: disable
15:4	R	Reserved
16	ATL_IRQ_E	ATL IRQ Enable bit: 1: enable 0: disable
17	ISO_IRQ_E	ISO IRQ Enable bit: 1: enable 0: disable
18	INT_IRQ_E	INT IRQ Enable bit: 1: enable 0: disable
19	SOF_E	SOF Interrupt Enable bit: 1: enable 0: disable
31:20	R	Reserved

40.5.12 PORTSC1 register

The Port Status and Control register indicates the port status and configures the port operation.

Table 961. PORTSC1 register (PORTSC1, offset 0x2C) bit description

Bits	Symbol	Description
0	CCS	Current Connect Status: Logic 1 indicates a device is present on the port. Logic 0 indicates no device is present. This field is 0 if Port Power is 0.
1	CSC	Connect Status Change: Logic 1 means that the value of CCS has changed. Logic 0 means no change. This field is 0 if Port Power is 0. Software must write a logic 1 to clear the bit.
2	PED	Port Enabled/Disabled. Logic 1 means port enabled. Logic 0 means disabled. This field is 0 if Port Power is 0. Firmware can clear the bit to disable the port. Firmware cannot set the bit. This bit will be set at the end of a port reset sequence.
3	PEDC	Port Enabled/Disabled Change: Logic 1 means that the value of PED has changed. Logic 0 means no change. This field is 0 if Port Power is 0. Software must write a logic 1 to clear the bit.
4	OCA	Over-current active: Logic 1 means that this port has an over-current condition. This bit will automatically move from one to 0 when the over-current condition is removed.
5	OCC	Over-current change: Logic 1 means that the value of OCA has changed. Logic 0 means no change. Software must write a logic 1 to clear the bit.
6	FPR	Force Port Resume: Logic 1 means resume (K-state) detected or driven on the port. Logic 0 means no resume detected or driven on the port. The resume signaling is driven on the port as long as this bit remains a one. For legacy (L2) transitions, software must appropriately time the resume and set this bit to a 0 when the appropriate amount of time has elapsed. Software does not need to time resume signaling for L1 transactions as host controller hardware will automatically enforce the necessary timing and clear this bit when the port has fully resumed. Software can influence the amount of time the hardware will drive resume signaling during L1 exit via the HIRD field within the USBCMD register. This field is 0 if Port Power is 0.
7	SUSP	Suspend: Logic 1 means port is in the suspend state. Logic 0 means the port is not suspended. Software writes a logic 1 to this bit to put an enabled port in the L1 or L2 suspend state. Which suspend state the host controller attempts depends on the value of the Suspend Using L1 field. When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. If this bit is set to a one when a transaction is in progress then the blocking will not occur until the end of the current transaction. A write of 0 is ignored by the hardware. The hardware will unconditionally set this bit to 0 when: Software sets the Force Port Resume bit to 0. Software sets the Port Reset bit to a one. This field is 0 if Port Power is 0 or Current Connect Status is 0.
8	PR	Port Reset: Logic 1 means the port is in the reset state. Logic 0 means the port is not in reset. Software writes a logic 1 to indicate the start of the reset. SW writes a logic 0 to end the reset sequence. If the reset sequence on the USB bus is finished HW will clear the bit. SW should only check the PSPD field to know the speed of the attached device when the Port Reset bit is 0. This field is 0 if Port Power is 0.

Table 961. PORTSC1 register (PORTSC1, offset 0x2C) bit description

Bits	Symbol	Description
9	SUS_L1	Suspend using L1 0b = Suspend using L2 1b = Suspend using L1 When this bit is set to a 1 and a non-zero value is specified in the Device Address field, the host controller will generate an LPM Token to enter the L1 state whenever software writes a one to the Suspend bit, as well as L1 exit timing during any device or host-initiated resume. When set to 0 the host controller will employ the legacy (L2) suspend mechanism. Software should only set this bit when the device attached immediately downstream of this root port supports L1 transitions.
11:10	LS	Line Status: This field reflects the current logical levels of the DP (bit 11) and DM (bit 10) signal lines.
12	PP	Port Power: The function of this bit depends on the value of the Port Power Control (PPC) bit in the HCSPARAMS register. If PPC = 0b, this bit (PP) is read-only and will always be set to 1b. If PPC = 1b, this bit (PP) is RW, If the bit is set to 0, the port is not powered. If the bit is set to one the port is powered.
13	R	Reserved
15:14	PIC	Port Indicator Control : Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is logic 0. If P_INDICATOR is set to one, these bits will indicate the value of the port indicators: 00b: Port Indicators are off 01b: Amber 10b: Green 11b: Undefined This field is 0 if Port Power is 0.
19:16	PTC	Port Test Control: A non-zero value indicates that the port is operating in the test mode as indicated by the value. 0000b: Test mode not enabled 0001b: Test J_STATE 0010b: Test K_STATE 0011b: TEST SE0_NAK 0100b: Test_Packet 0101b: Test Force_Enable 0110b – 1111b: Reserved The reserved values should not be written by software.
21:20	PSPD	Port Speed: 00b: Low-speed 01b: Full-speed 10b: High-speed 11b: Reserved

Table 961. PORTSC1 register (PORTSC1, offset 0x2C) bit description

Bits	Symbol	Description
22	WOO	Wake on overcurrent enable: Writing this bit to a one enables the port to be sensitive to overcurrent conditions as wake-up events. This field is 0 if Port Power is 0.
24:23	SUS_STAT	These two bits are used by software to determine whether the most recent L1 suspend request was successful: 00b: Success – state transition was successful (ACK) 01b: Not Yet – Device was unable to enter the L1 state at this time (NYET) 10b: Not supported – Device does not support the L1 state (STALL) 11b: Timeout/Error – Device failed to respond or an error occurred. This field is updated by hardware immediately following the completion of an L1 transition request (via an LPM token). To avoid any race conditions with hardware, software should only consume the contents of this field when Suspend = 0b (port no longer in L1).
31:25	DEV_ADD	Device Address for LPM tokens. 7-bit USB device address that is used when sending an LPM token to the device attached to and immediately downstream of the associated root port. A value of 0 indicates no device is present or support for LPM feature is not present on this device.

40.5.13 ATL PTD Done Map register

The ATL PTD Done Map register represents a direct map of the done status of the 32 ATL PTDs.

Table 962. ATL PTD Done Map register (ATL_DONE, offset = 0x30) bit description

Bits	Symbol	Description
31:0	ATL_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

40.5.14 ATL PTD Skip Map register

Table 963. ATL PTD Skip Map register (ATL_SKIP, offset 0x34) bit description

Bits	Symbol	Description
31:0	ATL_SKIP	When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

40.5.15 ISO PTD Done Map register

The ISO PTD Done Map register represents a direct map of the done status of the 32 ISO PTDs.

Table 964. ISO PTD Done Map register (ISO_DONE, offset 0x38) bit description

Bits	Symbol	Description
31:0	ISO_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

40.5.16 ISO PTD Skip Map register

The ISO PTD Skip Map register represents a direct map of the done status of the 32 INT PTDs.

Table 965. ISO PTD Skip Map register (ISO_SKIP, offset 0x3C) bit description

Bits	Symbol	Description
31:0	ISO_SKIP	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

40.5.17 INT PTD Done Map register

The INT PTD Done Map register represents a direct map of the done status of the 32 INT PTDs.

Table 966. INT PTD Done Map register (INT_DONE, offset 0x40) bit description

Bits	Symbol	Description
31:0	INT_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

40.5.18 INT PTD Skip Map register

Table 967. INT PTD Skip Map register (INT_SKIP, offset 0x44) bit description

Bits	Symbol	Description
31:0	INT_SKIP	When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

40.5.19 Last PTD in use register

The Last PTD in use register indicates the last PTD in the ATL list.

Table 968. Last PTD in use register (LAST PTD, offset 0x48) bit description

Bits	Symbol	Description
4:0	ATL_LAST	If hardware has reached this PTD and the J bit is not set, it will go to PTD0 as the next PTD to be processed.
7:5	R	Reserved
12:8	ISO_LAST	This indicates the last PTD in the ISO list. If hardware has reached this PTD, it will continue with processing the INT list

Table 968. Last PTD in use register (LAST PTD, offset 0x48) bit description

Bits	Symbol	Description
15:13	R	Reserved
20:16	INT_LAST	This indicates the last PTD in the INT list. If hardware has reached this PTD, it will continue with processing the ATL list.
31:21	R	Reserved

40.5.20 Port Mode

The Port mode register controls the host or device role in addition to setting the polarity of the ID pin.

Table 969. Port Mode register (PortMode, offset 0x50) bit description

Bits	Symbol	Description
0	ID0	Port 0 ID pin value. This bit indicates the value on the ID line of port 0. This field only contains a valid value some time after the ID0_EN bit is set to one.
7:1	R	Reserved
8	ID0_EN	Port 0 ID pin pull-up enable. If this bit is set, the pull-up on the ID line of port 0 will be enabled. This bit must be set to read a good value in the ID0 field.
15:9	R	Reserved
16	DEV_ENABLE	If this bit is set to one, one of the ports will behave as a USB device. The DEV_ROUTE bit determines which port will be routed to the device block. The other port will be routed to the host block. If this bit is set to 0, both ports will be controlled by the USB host block.
17	R	Reserved
18	SW_CTRL_PDCOM	This bit indicates if the PHY power-down input is controlled by software or by hardware. 0b: hardware state machine controls PHY power-down. 1b: software controls PHY power-down by writing to SW_PDCOM bit.
19	SW_PDCOM	This bit is only used when SW_CTRL_PDCOM is set to 1b. When SW_CTRL_PDCOM is set to 1b, the software can directly control the PHY power-down bit by writing to this bit. 0b: PHY operational. 1b: PHY in power-down mode.
31:20	R	Reserved

40.6 USB PHY low-power operation

The USB PHY is put in low power mode if the Run/Stop bit is set to 0 and the USB port is in a state that allows putting the PHY in low-power mode.

A change on the status of the port will generate a wake-up event. An over current situation on the port will only generate a wake-up event if the WOO bit is set in PORTSC1 register. See Section 40.5.12. The PHY can be put in low-power mode during the following states:

Port state = disconnected and linestate indicates SE0

Port state = disabled and linestate indicates J-state

Port state = suspend and linestate indicates J-state

If the PHY is in low-power mode and there is a change on the linestate bits, the PHY is brought out of low-power mode. It remains active for at least 3 μs. If the change on linestate is only a glitch and not a valid port change, the PHY is put in low-power mode again after this time.

If the WOO bit is set in PORTSC1 register, the PHY will be started when there is an over current condition and the PHY is in low-power mode. The PHY will remain active as long as the over current condition remains.

40.7 Proprietary Transfer Descriptor (PTD)

The standard Enhanced Host Controller Interface (EHCI) data structures as described in the “Enhanced Host Controller Interface Specification for Universal Serial Bus Rev. 1.0” are optimized for the host controller.

The optimized form of EHCI data structures is necessary because the controller does not have an AHB master interface. Instead, the controller exclusively uses its internal USB RAM to store and manage these data structures.

The controller manages schedules in two lists: periodic and asynchronous. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic, and reduce hardware and software complexity. The USB host controller executes transactions for devices by using a simple shared-memory schedule. This schedule consists of data structures organized into three lists:

- qISO — Isochronous transfer
- qINTL — Interrupt transfer
- qATL — Asynchronous transfer; for the control and bulk transfers

The system software maintains two lists for the host controller: periodic and asynchronous.

The high speed host controller has a maximum of 32 ISO, 32 INTL, and 32 ATL PTDs. These PTDs are used as channels to transfer data from the shared memory to the USB bus. These channels are allocated and de-allocated on receiving the transfer from the core USB driver.

Multiple transfers are scheduled to the shared memory for various endpoints by traversing the next link pointer provided by endpoint data structures, until it reaches the end of the endpoint list. There are three endpoint lists: one for ISO endpoints, and the other for INTL and ATL endpoints. If the schedule is enabled, the host controller executes the ISO schedule, followed by the INTL schedule, and then the ATL schedule.

These lists are traversed and scheduled by the software according to the EHCI traversal rule. The host controller executes the scheduled ISO, INTL and ATL PTDs. The completion of a transfer is indicated to the software by the interrupt that can be grouped under various PTDs by using the AND or OR registers that are available for each schedule type: ISO, INTL and ATL. These registers are simple logic registers to decide the completion status of group and individual PTDs. When the logical conditions of the Done bit is true in the shared memory, it means that PTD has completed.

There are four types of interrupts in the high speed host controller: ISO, INTL, ATL and SOF. The NextPTD pointer is a feature that allows the high speed host controller to jump unused and skip PTDs. This will improve the PTD transversal latency time. The NextPTD pointer is not meant for same or single endpoint. The NextPTD works only in forward direction.

The NextPTD traversal rules defined by the high speed host controller are:

1. Start the PTD memory vertical traversal, considering the skip and LastPTD information.
2. If the current PTD is active and not done, perform the transaction.
3. Follow the NextPTD pointer as specified in bits 4 to 0 of DW4.
4. If combined with LastPTD, the LastPTD setting must be at a higher address than the NextPTD specified.
5. If combined with skip, the skip must not be set (logically) on the same position corresponding to NextPTD, pointed by the NextPTD pointer.
6. If PTD is set for skip, it will be neglected and the next vertical PTD will be considered.
7. If the skipped PTD already has a setting including a NextPTD pointer that will not be taken into consideration, the behavior will be the same as described in step 6.

Figure 175 shows the flowchart of the PTD scheduler.

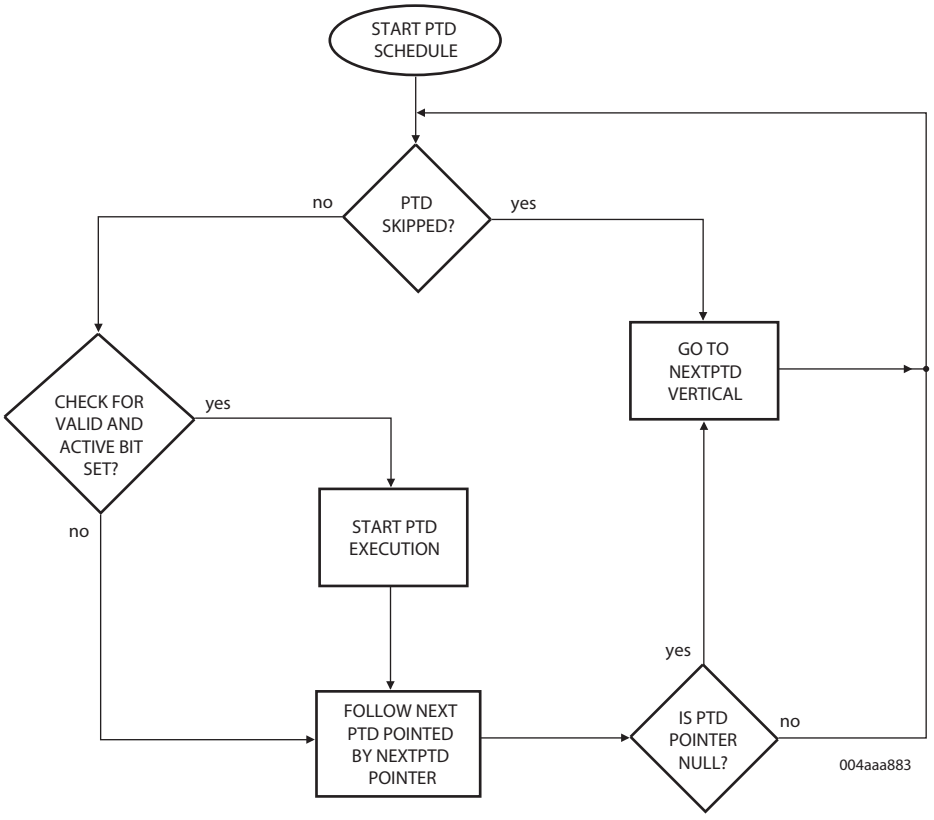


Fig 175. PTD scheduler flowchart

40.7.1 PTD on asynchronous list (qATL)

Table 970 shows the bit allocation of the bulk IN and OUT, asynchronous Transfer Descriptor (ATL PTD). This data structure is used for both regular HS/FS transactions and split transactions.

Table 970. PTD on asynchronous list (regular and split transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
HubAddress[6:0]							PortNumber[6:0]							SE [1:0]		RL[3:0]				S	DeviceAddress[6:0]							EP[3:0]				0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C

40.7.2 PTD on periodic list for regular transactions

Table 971 shows the bit allocation of the periodic IN and OUT, periodic Transfer Descriptor. This data structure is used for regular HS/FS transactions.

Table 971. PTD on periodic list (regular transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
R														SE [1:0]		RL[3:0]				S	DeviceAddress[6:0]							EP[3:0]			0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C
Status7 [2:0]			Status6 [2:0]			Status5 [2:0]			Status4 [2:0]			Status3 [2:0]			Status2 [2:0]		Status1 [2:0]			Status0 [2:0]			uSA[7:0]								0x10
ISO_IN_2[7:0]									ISO_IN_1[11:0]											ISO_IN_0[11:0]												0x14
ISO_IN_5 [3:0]				ISO_IN_4[11:0]												ISO_IN_3[11:0]														ISO_IN_2 [11:8]			0x18
ISO_IN_7[11:0]												ISO_IN_6[11:0]								ISO_IN_5[11:4]												0x1C

40.7.3 PTD on periodic list for split transactions

Table 972 shows the bit allocation of the periodic IN and OUT, periodic Transfer Descriptor. This data structure is used for split transactions on the periodic list.

Table 972. PTD on periodic list (split transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	TT_MPS_LEN[10:0] / MaxPacketLength[10:0]											uFrame[7:0]								J	R	NextPTDPointer [4:0]					V	0x00
HubAddress[6:0]							PortNumber[6:0]							SE[1:0]		RL[3:0]				S	DeviceAddress[6:0]							EP[3:0]			0x04
DataStartAddress[15:0]																I	NrBytesToTransfer[14:0]															0x08
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0]				EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]																		0x0C
Status7 [2:0]			Status6 [2:0]			Status5 [2:0]			Status4 [2:0]			Status3 [2:0]			Status2 [2:0]		Status1 [2:0]			Status0 [2:0]			uSA[7:0]								0x10

Table 972. PTD on periodic list (split transaction) *…continued*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
SP_ISO_IN_2[7:0]								SP_ISO_IN_1[7:0]								SP_ISO_IN_0[7:0] / S_Bytes[7:0]								uSCS[7:0]								0x14
SP_ISO_IN_6[7:0]								SP_ISO_IN_5[7:0]								SP_ISO_IN_4[7:0]								SP_ISO_IN_3[7:0]								0x18
R																								SP_ISO_IN_7[7:0]								0x1C

40.7.4 PTD bit definition

Table 973. PTD bit definition

Symbol	Access	Description
V	SW - sets HW - resets	Valid: 0b: This bit is deactivated when the entire PTD is executed (Active bit is cleared), or when a error is encountered (Halt bit is set). 1b: Software updates to one when there is payload to be sent or received. The current PTD is active.
NrBytesToTransfer[14:0]	SW - writes	Number of Bytes to Transfer: This field indicates the number of bytes that can be transferred by this data structure. It is used to indicate the depth of the DATA field (32 kB - 1).
MaxPacketLength[10:0] TT_MPS_Len[10:0]	SW - writes	Transaction Translator Maximum Packet Size Length: This field indicates the maximum number of bytes that can be sent per start split, depending on the number of total bytes needed. If the total bytes to be sent for the entire millisecond is greater than 188 bytes, this field should be set to 188 bytes for an OUT token and 192 bytes for an IN token. Otherwise, this field should be equal to the total bytes sent.
Mult[1:0]	SW - writes	Multiplier: This field is a multiplier used by the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution.
I	SW - writes	Interrupt on Complete: If this bit is set and the PTD is completed, the interrupt status bit of the corresponding list is set to one.
EP[3:0]	SW - writes	Endpoint: This is the USB address of the endpoint within the function.
DeviceAddress[6:0]	SW - writes	Device Address: This is the USB address of the function containing the endpoint that is referred to by this buffer.
S	SW - writes	This bit indicates whether a split transaction has to be executed. 0 – No split transaction – speed is same as port speed. 1 – Split transaction.
Token[1:0]	SW - writes	Token: Identifies the token Packet Identifier (PID) for this transaction. 00 – OUT 01 – IN 10 – SETUP 11 – Undefined

Table 973. PTD bit definition

Symbol	Access	Description
EPType[1:0]	SW - writes	Transaction type: 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt
SE[1:0]	SW - writes	This specifies the speed for a Control or Interrupt transaction to a device that is not high-speed: 00 – Full-speed 10 – Low-speed 01 or 11 – Undefined behavior For isochronous and bulk transactions this field must be set to 00. If a full-speed hub is connected to the port, the SE[1] bit indicates if a low-speed packet must be sent on a full-speed bus.
PortNumber[6:0]	SW - writes	Port Number: This indicates the port number of the hub.
HubAddress[6:0]	SW - writes	Hub Address: This indicates the hub address.
NextPTDPointer[4:0]	SW - writes	Next PTD Counter: Next PTD branching assigned by the PTDpointer.
J	SW - writes	Jump: 0: increment the PTD pointer. 1: enable the next PTD branching.
DataStartAddress[15:0]	SW - writes	Data Start Address: “DataStartAddress + DataPayload_BaseAddress” is the start address that points to the start of the data buffer that will be sent or received on or from the USB bus. The hardware does not update this field when the transfer is completed.
uFrame[7:0]		This field is only applicable for interrupt and isochronous endpoints. Interrupt endpoint: Bits 2 to 0 of this field together with the μsA field represent the polling rate. When the polling rate is ≤ 1 ms, bits 2 to 0 are set to 0. If the polling rate is greater than 1 ms, bits 2 to 0 define the polling rate and bits 7 to 3 define the frame number when the packet must be transmitted. Isochronous endpoint: Bits 2 to 0 — Don’t care. Bits 7 to 3 — Frame number at which this PTD will be sent.
NrBytesTransferred[14:0]	HW — writes	Number of Bytes Transferred: This field indicates the number of bytes sent or received for this transaction. If Mult[1:0] is greater than one, it is possible to store intermediate results in this field.
RL[3:0]	SW - writes	Reload: If RL is set to 0h, hardware ignores the NakCnt value. RL and NakCnt are set to the same value before a transaction.
NakCnt[3:0]	HW - writes SW - writes	NAK Counter: This field corresponds to the NakCnt field in TD. Software writes for the initial PTD launch. The V bit is reset if NakCnt decrements to zero and RL is a nonzero value. It reloads from RL if transaction is ACK-ed.
Cerr[1:0]	HW - writes SW - writes	Error Counter: This field corresponds to the Cerr[1:0] field in TD. The default value of this field is 0 for isochronous transactions. 00 — The transaction will not retry. 11 — The transaction will retry three times. The hardware will decrement these values.

Table 973. PTD bit definition

Symbol	Access	Description
DT	HW - updates SW - writes	Data Toggle: This bit is filled by software to start a PTD. If NrBytesToTransfer[14:0] is not complete, software needs to read this value and use this value to program the next PTD that will send data to the same endpoint.
P	SW - writes HW - updates	Ping: For high-speed transactions, this bit corresponds to the Ping state bit in the Status field of a TD. 0 — Ping is not set. 1 — Ping is set. For the first time, software sets the Ping bit to 0. For the successive asynchronous TD, software sets the bit in asynchronous TD based on the state of the bit for the previous asynchronous TD of the same transfer: The current asynchronous TD is completed with the Ping bit set. The next asynchronous TD will have its Ping bit set by the software.
SC	SW - writes 0 HW - updates	Start/Complete: 0 — Start split 1 — Complete split
X	HW - writes	Error: This bit corresponds to the Transaction Error bit in the Status field of iTD, siTD or TD. 0 — No PID error. 1 — If there are PID errors, this bit is set Active. The A and V bits are also set to inactive. This transaction is retried three times.
B	HW - writes	Babble: This bit corresponds to the Babble Detected bit in the Status field of iTD, siTD or TD. 1 — When babbling is detected, A and V are set to 0.
H	HW - writes	Halt: Set to a 1 by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this data structure. This can be caused by babble, the error counter counting down to 0, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to 0.
A	SW - sets	Active: This bit is the same as the Valid bit.
uSA[7:0]	SW - writes (0 $\rightarrow$1) HW - writes (1 $\rightarrow$ 0) after processing	This field is only used for periodic split transactions or if the port is enabled in HS mode. uSOF Active: When the frame number of bits uFrame[7:3] match the frame number of the USB bus, these bits are checked for 1 before they are sent for uSOF. For example: If uSA[7:0] = 1111 1111b: send ISO every uSOF of the entire millisecond. If uSA[7:0] = 0101 0101b: send ISO only on uSOF0, uSOF2, uSOF4, and SOF6.
Status0[2:0]	HW - writes	Isochronous IN or OUT status at uSOF0 Bit 2: Underrun Bit 1: Babble Bit 0: XactErr
Status1[2:0]	HW - writes	Isochronous IN or OUT status at uSOF1.
Status2[2:0]	HW - writes	Isochronous IN or OUT status at uSOF2.
Status3[2:0]	HW - writes	Isochronous IN or OUT status at uSOF3.
Status4[2:0]	HW - writes	Isochronous IN or OUT status at uSOF4.

Table 973. PTD bit definition

Symbol	Access	Description
Status5[2:0]	HW - writes	Isochronous IN or OUT status at uSOF5.
Status6[2:0]	HW - writes	Isochronous IN or OUT status at uSOF6.
Status7[2:0]	HW - writes	Isochronous IN or OUT status at uSOF7.
ISO_IN0[11:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
ISO_IN1[11:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
ISO_IN2[11:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 2 and frame number is correct.
ISO_IN3[11:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 3 and frame number is correct.
ISO_IN4[11:0]	HW - writes	Bytes received during uSOF4 if uSA[4] is set to 4 and frame number is correct.
ISO_IN5[11:0]	HW - writes	Bytes received during uSOF5 if uSA[5] is set to 5 and frame number is correct.
ISO_IN6[11:0]	HW - writes	Bytes received during uSOF6 if uSA[6] is set to 6and frame number is correct.
ISO_IN7[11:0]	HW - writes	Bytes received during uSOF7 if uSA[7] is set to 6and frame number is correct.
uSCS[7:0]	SW - writes (0 →1) HW - writes (1 → 0) after processing	All bits can be set to one for every transfer. It specifies which uSOF the complete split needs to be sent. Start split and complete split Active bits, uSA = 0000 0001b, uSCS = 0000 0100b, will cause SS to execute in uFrame0 and CS in uFrame2.
SP_ISO_IN0[7:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
SP_ISO_IN1[7:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
SP_ISO_IN2[7:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 1 and frame number is correct.
SP_ISO_IN3[7:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 1 and frame number is correct.
SP_ISO_IN4[7:0]	HW - writes	Bytes received during uSOF4, if uSA[4] is set to 1 and frame number is correct.
SP_ISO_IN5[7:0]	HW - writes	Bytes received during uSOF5, if uSA[5] is set to 1 and frame number is correct.
SP_ISO_IN6[7:0]	HW - writes	Bytes received during uSOF6, if uSA[6] is set to 1 and frame number is correct.
SP_ISO_IN7[7:0]	HW - writes	Bytes received during uSOF7, if uSA[7] is set to 1 and frame number is correct.
S_Bytes	HW - writes	This field is used by HW to store an intermediate value of number of bytes received while handling a complete-split for interrupt IN transfers.

40.7.4.1 Polling rate for Periodic transactions

Table 974 indicates how the hardware knows when to send a certain interrupt packet based on the polling rate. uFrame[2:0] defines the polling rate.

If set to 0 and the transaction is high-speed, the uSA determines during which uFrames a packet can be sent. If set to 0 and the transaction is a normal full-speed or low-speed, the packet will be sent during every frame. If it is set to 0 and the transaction is a full-speed or low-speed split transaction, the uSA and uCS fields will determine when to send a split transaction.

If uFrame[2:0] is different from 0 and the transaction is high-speed, the bits in uFrame[7:3] and the bits in uSA are used to know during which uFrames a packet must be sent.

If uFrame[2:0] is different from 0 and the transaction is full-speed or low-speed, the bits in uFrame[7:3] are used to know during which uFrames a packet must be sent. If uFrame[2:0] is different from 0 and the transaction is a full-speed or low-speed split transaction, the bits in uFrame[7:3] and the uSA and uCS fields will determine when to send a split transaction.

Table 974. Polling rate for periodic transactions

b	Rate	uFrame[2:0]	uFrame[7:3]	uSA[7:0]
1	1 uSOF	000b	Don’t care	1111 1111b
2	2 uSOF	000b	Don’t care	1010 1010b or 0101 0101b
3	4 uSOF	000b	Don’t care	Any 2 bits set
4	1 mS	000b	Don’t care	Any 1 bit set
5	2	001b	Bit 0 is compared with FRINDEX[3]	Any 1 bit set
6	4	010b	Bits[1:0] are compared with FRINDEX[4:3]	Any 1 bit set
7	8	011b	Bits[2:0] are compared with FRINDEX[5:3]	Any 1 bit set
8	16	100b	Bits[3:0] are compared with FRINDEX[6:3]	Any 1 bit set
9	32	101b	Bits[4:0] are compared with FRINDEX[7:3]	Any 1 bit set

41.1 How to read this chapter

The USB ROM driver routines are available on all LPC546xx devices.

USB on-chip drivers are provided via the USB Stack in SDK software package.

41.2 Features

- ROM-base USB drivers.
- Communication Device Class (CDC) device class.
- Human Interface Device (HID) device class.
- Mass Storage Device (MSC) class.
- Device Firmware Upgrade (DFU) class.

41.3 General description

The boot ROM contains a USB driver to simplify the USB application development. The USB driver implements the Communication Device Class (CDC), the Human Interface Device (HID), Mass Storage Device (MSC) device class, and Device Firmware Upgrade (DFU) class. The USB on-chip drivers support composite device.

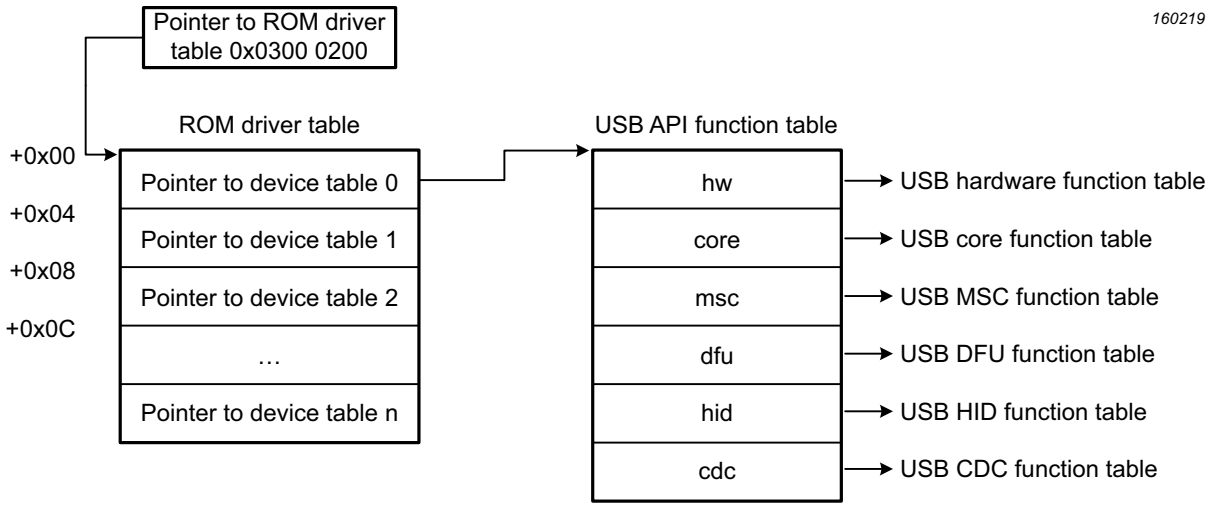


Fig 176. USB driver routines pointer structure

41.3.1 USB driver functions

The USB device driver ROM API consists of the following modules:

- Communication Device Class (CDC) function driver. This module contains an internal implementation of the USB CDC Class. User applications can use this class driver instead of implementing the CDC-ACM class manually via the low-level USBD_HW and USBD_Core APIs. This module is designed to simplify the user code by exposing only the required interface needed to interface with Devices using the USB CDC-ACM Class.
  - Communication Device Class function driver initialization parameter data structure (Table 1002 “USBD_CDC_INIT_PARAM class structure”).
  - CDC class API functions structure. This module exposes functions which interact directly with USB device controller hardware (Table 1001 “USBD_CDC_API class structure”).
- USB core layer
  - struct (Table 998 “_WB_T class structure”)
  - union (Table 975 “__WORD_BYTE class structure”)
  - struct (Table 976 “_BM_T class structure”)
  - struct (Table 989 “_REQUEST_TYPE class structure”)
  - struct (Table 996 “_USB_SETUP_PACKET class structure”)
  - struct (Table 992 “_USB_DEVICE_QUALIFIER_DESCRIPTOR class structure”)
  - struct USB device descriptor
  - struct (Table 992 “_USB_DEVICE_QUALIFIER_DESCRIPTOR class structure”)
  - struct USB configuration descriptor
  - struct (Table 994 “_USB_INTERFACE_DESCRIPTOR class structure”)
  - struct USB endpoint descriptor
  - struct (Table 997 “_USB_STRING_DESCRIPTOR class structure”)
  - struct (Table 990 “_USB_COMMON_DESCRIPTOR class structure”)
  - struct (Table 995 “_USB_OTHER_SPEED_CONFIGURATION class structure”)
  - USB descriptors data structure (Table 991 “_USB_CORE_DESCS_T class structure”)
  - USB device stack initialization parameter data structure (Table 1000 “USBD_API_INIT_PARAM class structure”).
  - USB device stack core API functions structure (Table 1003 “USBD_CORE_API class structure”).
- Device Firmware Upgrade (DFU) class function driver
  - DFU descriptors data structure (Table 1005 “USBD_DFU_INIT_PARAM class structure”).
  - DFU class API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 1004 “USBD_DFU_API class structure”).
- HID class function driver
  - struct (Table 984 “_HID_DESCRIPTOR class structure”).
  - struct (Table 986 “_HID_REPORT_T class structure”).
  - USB descriptors data structure (Table 1007 “USBD_HID_INIT_PARAM class structure”).

- HID class API functions structure. This structure contains pointers to all the functions exposed by the HID function driver module (Table 1008 “USBD_HW_API class structure”).

- USB device controller driver
  - Hardware API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 1008 “USBD_HW_API class structure”).
- Mass Storage Class (MSC) function driver
  - Mass Storage Class function driver initialization parameter data structure (Table 1010).
  - MSC class API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 1009).

41.3.2 Calling the USB device driver

A fixed location in ROM contains a pointer to the ROM driver table. The ROM driver table contains a pointer to the USB driver table. Pointers to the various USB driver functions are stored in this table. USB driver functions can be called by using a C structure. Figure 176 illustrates the pointer mechanism used to access the on-chip USB driver.

```
typedef struct USBD_API
{
const USBD_HW_API_T* hw;
const USBD_CORE_API_T* core;
const USBD_MSC_API_T* msc;
const USBD_DFU_API_T* dfu;
const USBD_HID_API_T* hid;
const USBD_CDC_API_T* cdc;
const uint32_t* reserved6;
const uint32_t version;
} USBD_API_T;
```

41.4 USB API

41.4.1 __WORD_BYTE

Table 975. __WORD_BYTE class structure

Member	Description
W	`uint16_t __WORD_BYTE::W` data member to do 16 bit access
WB	`WB_TWB_T __WORD_BYTE::WB` data member to do 8 bit access

41.4.2 _BM_T

Table 976. _BM_T class structure

Member	Description
Recipient	`uint8_t _BM_T::Recipient` Recipient type.
Type	`uint8_t _BM_T::Type` Request type.
Dir	`uint8_t _BM_T::Dir` Direction type.

41.4.3 _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR

Table 977. _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR class structure

Member	Description
bFunctionLength	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bFunctionLength`
bDescriptorType	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorType`
bDescriptorSubtype	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype`
bmCapabilities	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bmCapabilities`

41.4.4 _CDC_CALL_MANAGEMENT_DESCRIPTOR

Table 978. _CDC_CALL_MANAGEMENT_DESCRIPTOR class structure

Member	Description
bFunctionLength	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bFunctionLength`
bDescriptorType	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorType`
bDescriptorSubtype	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype`
bmCapabilities	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bmCapabilities`
bDataInterface	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDataInterface`

41.4.5 _CDC_HEADER_DESCRIPTOR

Table 979. _CDC_HEADER_DESCRIPTOR class structure

Member	Description
bFunctionLength	uint8_t _CDC_HEADER_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorSubtype
bcdCDC	uint16_t _CDC_HEADER_DESCRIPTOR::bcdCDC

41.4.6 _CDC_LINE_CODING

Table 980. _CDC_LINE_CODING class structure

Member	Description
dwDTERate	uint32_t _CDC_LINE_CODING::dwDTERate
bCharFormat	uint8_t _CDC_LINE_CODING::bCharFormat
bParityType	uint8_t _CDC_LINE_CODING::bParityType
bDataBits	uint8_t _CDC_LINE_CODING::bDataBits

41.4.7 _CDC_UNION_1SLAVE_DESCRIPTOR

Table 981. _CDC_UNION_1SLAVE_DESCRIPTOR class structure

Member	Description
sUnion	CDC_UNION_DESCRIPTORCDC_UNION_DESCRIPTOR _CDC_UNION_1SLAVE_DESCRIPTOR::sUnion
bSlaveInterfaces	uint8_t _CDC_UNION_1SLAVE_DESCRIPTOR::bSlaveInterfaces[1][1]

41.4.8 _CDC_UNION_DESCRIPTOR

Table 982. _CDC_UNION_DESCRIPTOR class structure

Member	Description
bFunctionLength	uint8_t _CDC_UNION_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_t _CDC_UNION_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_t _CDC_UNION_DESCRIPTOR::bDescriptorSubtype
bMasterInterface	uint8_t _CDC_UNION_DESCRIPTOR::bMasterInterface

41.4.9 _DFU_STATUS

Table 983. _DFU_STATUS class structure

Member	Description
bStatus	uint8_t _DFU_STATUS::bStatus
bwPollTimeout	uint8_t _DFU_STATUS::bwPollTimeout[3][3]
bState	uint8_t _DFU_STATUS::bState
iString	uint8_t _DFU_STATUS::iString

41.4.10 _HID_DESCRIPTOR

HID class-specific HID Descriptor.

Table 984. _HID_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _HID_DESCRIPTOR::bLength` Size of the descriptor, in bytes.
bDescriptorType	`uint8_t _HID_DESCRIPTOR::bDescriptorType` Type of HID descriptor.
bcdHID	`uint16_t _HID_DESCRIPTOR::bcdHID` BCD encoded version that the HID descriptor and device complies to.
bCountryCode	`uint8_t _HID_DESCRIPTOR::bCountryCode` Country code of the localized device, or zero if universal.
bNumDescriptors	`uint8_t _HID_DESCRIPTOR::bNumDescriptors` Total number of HID report descriptors for the interface.
DescriptorList	`PRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LISTPRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST _HID_DESCRIPTOR::DescriptorList[1][1]` Array of one or more descriptors

41.4.11 _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST

Table 985. _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST class structure

Member	Description
bDescriptorType	`uint8_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::bDescriptorType` Type of HID report.
wDescriptorLength	`uint16_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::wDescriptorLength` Length of the associated HID report descriptor, in bytes.

41.4.12 _HID_REPORT_T

HID report descriptor data structure.

Table 986. _HID_REPORT_T class structure

Member	Description
len	`uint16_t _HID_REPORT_T::len` Size of the report descriptor in bytes.
idle_time	`uint8_t _HID_REPORT_T::idle_time` This value is used by stack to respond to Set_Idle & GET_Idle requests for the specified report ID. The value of this field specified the rate at which duplicate reports are generated for the specified Report ID. For example, a device with two input reports could specify an idle rate of 20 milliseconds for report ID 1 and 500 milliseconds for report ID 2.
__pad	`uint8_t _HID_REPORT_T::__pad` Padding space.
desc	`uint8_t * _HID_REPORT_T::desc` Report descriptor.

41.4.13 _MSC_CBW

Table 987. _MSC_CBW class structure

Member	Description
dSignature	`uint32_t _MSC_CBW::dSignature`
dTag	`uint32_t _MSC_CBW::dTag`
dDataLength	`uint32_t _MSC_CBW::dDataLength`
bmFlags	`uint8_t _MSC_CBW::bmFlags`
bLUN	`uint8_t _MSC_CBW::bLUN`
bCBLength	`uint8_t _MSC_CBW::bCBLength`
CB	`uint8_t _MSC_CBW::CB[16][16]`

41.4.14 _MSC_CSW

Table 988. _MSC_CSW class structure

Member	Description
dSignature	`uint32_t _MSC_CSW::dSignature`
dTag	`uint32_t _MSC_CSW::dTag`
dDataResidue	`uint32_t _MSC_CSW::dDataResidue`
bStatus	`uint8_t _MSC_CSW::bStatus`

41.4.15 _REQUEST_TYPE

Table 989. _REQUEST_TYPE class structure

Member	Description
B	`uint8_t _REQUEST_TYPE::B` byte wide access member
BM	`BM_TBM_T _REQUEST_TYPE::BM` bitfield structure access member

41.4.16 _USB_COMMON_DESCRIPTOR

Table 990. _USB_COMMON_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_COMMON_DESCRIPTOR::bLength` Size of this descriptor in bytes
bDescriptorType	`uint8_t _USB_COMMON_DESCRIPTOR::bDescriptorType` Descriptor Type

41.4.17 _USB_CORE_DESCS_T

USB descriptors data structure.

Table 991. _USB_CORE_DESCS_T class structure

Member	Description
device_desc	`uint8_t * _USB_CORE_DESCS_T::device_desc` Pointer to USB device descriptor
string_desc	`uint8_t * _USB_CORE_DESCS_T::string_desc` Pointer to array of USB string descriptors
full_speed_desc	`uint8_t * _USB_CORE_DESCS_T::full_speed_desc` Pointer to USB device configuration descriptor when device is operating in full speed mode.
high_speed_desc	`uint8_t * _USB_CORE_DESCS_T::high_speed_desc` Pointer to USB device configuration descriptor when device is operating in high speed mode. For full-speed only implementation this pointer should be same as full_speed_desc.
device_qualifier	`uint8_t * _USB_CORE_DESCS_T::device_qualifier` Pointer to USB device qualifier descriptor. For full-speed only implementation this pointer should be set to null (0).

41.4.18 _USB_DEVICE_QUALIFIER_DESCRIPTOR

Table 992. _USB_DEVICE_QUALIFIER_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bLength` Size of descriptor
bDescriptorType	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDescriptorType` Device Qualifier Type
bcdUSB	`uint16_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bcdUSB` USB specification version number (e.g., 0200H for V2.00)
bDeviceClass	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceClass` Class Code
bDeviceSubClass	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceSubClass` SubClass Code
bDeviceProtocol	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceProtocol` Protocol Code
bMaxPacketSize0	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bMaxPacketSize0` Maximum packet size for other speed
bNumConfigurations	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bNumConfigurations` Number of Other-speed Configurations
bReserved	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bReserved` Reserved for future use, must be zero

41.4.19 _USB_DFU_FUNC_DESCRIPTOR

Table 993. _USB_DFU_FUNC_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_DFU_FUNC_DESCRIPTOR::bLength`
bDescriptorType	`uint8_t _USB_DFU_FUNC_DESCRIPTOR::bDescriptorType`
bmAttributes	`uint8_t _USB_DFU_FUNC_DESCRIPTOR::bmAttributes`
wDetachTimeOut	`uint16_t _USB_DFU_FUNC_DESCRIPTOR::wDetachTimeOut`
wTransferSize	`uint16_t _USB_DFU_FUNC_DESCRIPTOR::wTransferSize`
bcdDFUVersion	`uint16_t _USB_DFU_FUNC_DESCRIPTOR::bcdDFUVersion`

41.4.20 _USB_INTERFACE_DESCRIPTOR

Table 994. _USB_INTERFACE_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_INTERFACE_DESCRIPTOR::bLength` Size of this descriptor in bytes
bDescriptorType	`uint8_t _USB_INTERFACE_DESCRIPTOR::bDescriptorType` INTERFACE Descriptor Type
bInterfaceNumber	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceNumber` Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
bAlternateSetting	`uint8_t _USB_INTERFACE_DESCRIPTOR::bAlternateSetting` Value used to select this alternate setting for the interface identified in the prior field
bNumEndpoints	`uint8_t _USB_INTERFACE_DESCRIPTOR::bNumEndpoints` Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the Default Control Pipe.
bInterfaceClass	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceClass` Class code (assigned by the USB-IF).
bInterfaceSubClass	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceSubClass` Subclass code (assigned by the USB-IF).
bInterfaceProtocol	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceProtocol` Protocol code (assigned by the USB).
iInterface	`uint8_t _USB_INTERFACE_DESCRIPTOR::iInterface` Index of string descriptor describing this interface

41.4.21 _USB_OTHER_SPEED_CONFIGURATION

Table 995. _USB_OTHER_SPEED_CONFIGURATION class structure

Member	Description
bLength	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bLength` Size of descriptor
bDescriptorType	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bDescriptorType` Other_speed_Configuration Type
wTotalLength	`uint16_t _USB_OTHER_SPEED_CONFIGURATION::wTotalLength` Total length of data returned
bNumInterfaces	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bNumInterfaces` Number of interfaces supported by this speed configuration
bConfigurationValue	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bConfigurationValue` Value to use to select configuration
IConfiguration	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::IConfiguration` Index of string descriptor
bmAttributes	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bmAttributes` Same as Configuration descriptor
bMaxPower	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bMaxPower` Same as Configuration descriptor

41.4.22 _USB_SETUP_PACKET

Table 996. _USB_SETUP_PACKET class structure

Member	Description
bmRequestType	`REQUEST_TYPE _USB_SETUP_PACKET::bmRequestType` This bit-mapped field identifies the characteristics of the specific request. _BM_T.
bRequest	`uint8_t _USB_SETUP_PACKET::bRequest` This field specifies the particular request. The Type bits in the bmRequestType field modify the meaning of this field. USBD_REQUEST.
wValue	`WORD_BYTE _USB_SETUP_PACKET::wValue` Used to pass a parameter to the device, specific to the request.
wIndex	`WORD_BYTE _USB_SETUP_PACKET::wIndex` Used to pass a parameter to the device, specific to the request. The wIndex field is often used in requests to specify an endpoint or an interface.
wLength	`uint16_t _USB_SETUP_PACKET::wLength` This field specifies the length of the data transferred during the second phase of the control transfer.

41.4.23 _USB_STRING_DESCRIPTOR

Table 997. _USB_STRING_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_STRING_DESCRIPTOR::bLength` Size of this descriptor in bytes
bDescriptorType	`uint8_t _USB_STRING_DESCRIPTOR::bDescriptorType` STRING Descriptor Type
bString	`uint16_t _USB_STRING_DESCRIPTOR::bString` UNICODE encoded string

41.4.24 _WB_T

Table 998. _WB_T class structure

Member	Description
L	`uint8_t _WB_T::L` lower byte
H	`uint8_t _WB_T::H` upper byte

41.4.25 USBD_API

Main USBD API functions structure.This structure contains pointer to various USB Device stack's sub-module function tables. This structure is used as main entry point to access various methods (grouped in sub-modules) exposed by ROM based USB device stack.

Table 999. USBD_API class structure

Member	Description
hw	`const USBD_HW_API_T* USBD_API::hw` Pointer to function table which exposes functions which interact directly with USB device stack's core layer.
core	`const USBD_CORE_API_T* USBD_API::core` Pointer to function table which exposes functions which interact directly with USB device controller hardware.
msc	`const USBD_MSC_API_T* USBD_API::msc` Pointer to function table which exposes functions provided by MSC function driver module.
dfu	`const USBD_DFU_API_T* USBD_API::dfu` Pointer to function table which exposes functions provided by DFU function driver module.
hid	`const USBD_HID_API_T* USBD_API::hid` Pointer to function table which exposes functions provided by HID function driver module.

Table 999. USBD_API class structure

Member	Description
cdc	`const USBD_CDC_API_T* USBD_API::cdc` Pointer to function table which exposes functions provided by CDC-ACM function driver module.
reserved6	`const uint32_t* USBD_API::reserved6` Reserved for future function driver module.
version	`const uint32_t USBD_API::version` Version identifier of USB ROM stack. The version is defined as 0x0CHDMhCC where each nibble represents version number of the corresponding component. CC - 7:0 - 8bit core version number h - 11:8 - 4bit hardware interface version number M - 15:12 - 4bit MSC class module version number D - 19:16 - 4bit DFU class module version number H - 23:20 - 4bit HID class module version number C - 27:24 - 4bit CDC class module version number H - 31:28 - 4bit reserved

41.4.26 USBD_API_INIT_PARAM

USB device stack initialization parameter data structure.

Table 1000.USBD_API_INIT_PARAM class structure

Member	Description
usb_reg_base	`uint32_t USBD_API_INIT_PARAM::usb_reg_base` USB device controller's base register address.
mem_base	`uint32_t USBD_API_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 2048 byte boundary. **Remark:** When high_speed_capable is set to 1 (USB1 is used as the device), mem_base must use the USB SRAM region.
mem_size	`uint32_t USBD_API_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_HW_API::GetMemSize() routine.
max_num_ep	`uint8_t USBD_API_INIT_PARAM::max_num_ep` max number of endpoints supported by the USB device controller instance (specified by
high_speed_capable	`uint8_t USBD_API_INIT_PARAM::high_speed_capable` 0: for USB0 Full-speed only; 1: USB1 High speed capable.
pad0	`uint8_t USBD_API_INIT_PARAM::pad0[2]`
USB_Reset_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Reset_Event` Event for USB interface reset. This event fires when the USB host requests that the device reset its interface. This event fires after the control endpoint has been automatically configured by the library. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Suspend_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Suspend_Event` Event for USB suspend. This event fires when the USB host suspends the device by halting its transmission of Start Of Frame pulses to the device. This is generally hooked in order to move the device over to a low power state until the host wakes up the device. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will cause other system issues.

Table 1000.USBD_API_INIT_PARAM class structure

Member	Description
USB_Resume_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Resume_Event` Event for USB wake up or resume. This event fires when a the USB device interface is suspended and the host wakes up the device by supplying Start Of Frame pulses. This is generally hooked to pull the user application out of a low power state and back into normal operating mode. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will cause other system issues.
reserved_sbz	`USB_CB_T USBD_API_INIT_PARAM::reserved_sbz` Reserved parameter should be set to zero.
USB_SOF_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_SOF_Event` Event for USB Start Of Frame detection, when enabled. This event fires at the start of each USB frame, once per millisecond in full-speed mode or once per 125 microseconds in high-speed mode, and is synchronized to the USB bus. This event is time-critical; it is run once per millisecond (full-speed mode) and thus long handlers will significantly degrade device performance. This event should only be enabled when needed to reduce device wake-ups. This event is not normally active - it must be manually enabled and disabled via the USB interrupt register. **Remark:** This event is not normally active - it must be manually enabled and disabled via the USB interrupt register.
USB_WakeUpCfg	`USB_PARAM_CB_T USBD_API_INIT_PARAM::USB_WakeUpCfg` Event for remote wake-up configuration, when enabled. This event fires when the USB host request the device to configure itself for remote wake-up capability. The USB host sends this request to device which report remote wake-up capable in their device descriptors, before going to low-power state. The application layer should implement this callback if they have any special on board circuit to trigger remote wake up event. Also application can use this callback to differentiate the following SUSPEND event is caused by cable plug-out or host SUSPEND request. The device can wake-up host only after receiving this callback and remote wake-up feature is enabled by host. To signal remote wake-up the device has to generate resume signaling on bus by calling usapi.hw->WakeUp() routine. Parameters: 1. hUsb = Handle to the USB device stack. 2. param1 = When 0 - Clear the wake-up configuration, 1 - Enable the wake-up configuration. Returns: the call back should return ErrorCode_t type to indicate success or error condition.
USB_Power_Event	`USB_PARAM_CB_T USBD_API_INIT_PARAM::USB_Power_Event` Reserved parameter should be set to zero.
USB_Error_Event	`USB_PARAM_CB_T USBD_API_INIT_PARAM:USB_Error_Event` Event for error condition. This event fires when USB device controller detect an error condition in the system. Parameters: 1. hUsb = Handle to the USB device stack. 2. param1 = USB device interrupt status register. Returns: the call back should return ErrorCode_t type to indicate success or error condition.

Table 1000.USBD_API_INIT_PARAM class structure

Member	Description
USB_Configure_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Configure_Event` Event for USB configuration number changed. This event fires when a the USB host changes the selected configuration number. On receiving configuration change request from host, the stack enables/configures the endpoints needed by the new configuration before calling this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Interface_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Interface_Event` Event for USB interface setting changed. This event fires when a the USB host changes the interface setting to one of alternate interface settings. On receiving interface change request from host, the stack enables/configures the endpoints needed by the new alternate interface setting before calling this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Feature_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Feature_Event` Event for USB feature changed. This event fires when a the USB host send set/clear feature request. The stack handles this request for USB_FEATURE_REMOTE_WAKEUP, USB_FEATURE_TEST_MODE and USB_FEATURE_ENDPOINT_STALL features only. On receiving feature request from host, the stack handle the request appropriately and then calls this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
virt_to_phys	`uint32_t(* USBD_API_INIT_PARAM::virt_to_phys)(void *vaddr)` Reserved for future use. Should be set to zero.
cache_flush	`void(* USBD_API_INIT_PARAM::cache_flush)(uint32_t *start_adr, uint32_t *end_adr)` Reserved for future use. Should be set to zero.
lpm_setting	`void(* USBD_API_INIT_PARAM::lpm_setting` Configurable LPM setting, bit 31, 1 enable LPM mode, 0 disable LPM mode. Bits 15:0 is the value to be written to the LPM register. Bits 30:16 are reserved.
USB_ReqGetStringDesc	`ErrorCode_t(* USBD_API_INIT_PARAM::USB_ReqGetStringDesc)(USBD_HANDLE_T hUsb, USB_SETUP_PACKET *setupPacket, uint8_t **pD)` This callback is invoked by stack when get string descriptor request is received. If callback returns with pD pointing to NULL then the stack uses string_desc descriptor array provided during initialization to find the string. This interface provides mechanism to override the stack handler for string desriptor and helps in handling dis-contiguous string index descriptor requests. Parameters: 1. hUsb = Handle to the USB device stack. 2. setupPacket = Pointer to setup packet. 3. pD = Double Pointer to string descriptor which is updated in the function. Set the pointer to NULL for the stack to handle the request. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = Success. ERR_USBD_INVALID_REQ = If request is invalid.

41.4.27 USBD_CDC_API

CDC class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 1001.USBD_CDC_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_CDC_API::GetMemSize)(USBD_CDC_INIT_PARAM_T *param)` Function to determine the memory required by the CDC function driver module. This function is called by application layer before calling pUsbApi->CDC->Init(), to allocate memory used by CDC function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing CDC function driver module initialization parameters. Returns: the required memory size in bytes.
init	`ErrorCode_t(*ErrorCode_t USBD_CDC_API::init)(USBD_HANDLE_T hUsb, USBD_CDC_INIT_PARAM_T *param, USBD_HANDLE_T *phCDC)` Function to initialize CDC function driver module. This function is called by application layer to initialize CDC function driver module. hUsbHandle to the USB device stack. paramStructure containing CDC function driver module initialization parameters. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing CDC function driver module initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. ERR_API_INVALID_PARAM2 = Either CDC_Write() or CDC_Read() or CDC_Verify() callbacks are not defined. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

Table 1001.USBD_CDC_API class structure

Member	Description
SendNotification	`ErrorCode_t(*ErrorCode_t USBD_CDC_API::SendNotification)(USBD_HANDLE_T hCdc, uint8_t bNotification, uint16_t data)` Function to send CDC class notifications to host. This function is called by application layer to send CDC class notifications to host. See usbcdc11.pdf, section 6.3, Table 67 for various notification types the CDC device can send. **Remark:** The current version of the driver only supports following notifications allowed by ACM subclass: CDC_NOTIFICATION_NETWORK_CONNECTION, CDC_RESPONSE_AVAILABLE, CDC_NOTIFICATION_SERIAL_STATE. For all other notifications application should construct the notification buffer appropriately and call hw->USB_WriteEP() for interrupt endpoint associated with the interface. Parameters: 1. hCdc = Handle to CDC function driver. 2. bNotification = Notification type allowed by ACM subclass. Should be CDC_NOTIFICATION_NETWORK_CONNECTION, CDC_RESPONSE_AVAILABLE or CDC_NOTIFICATION_SERIAL_STATE. For all other types ERR_API_INVALID_PARAM2 is returned. See usbcdc11.pdf, section 3.6.2.1, table 5. 3. data = Data associated with notification. For CDC_NOTIFICATION_NETWORK_CONNECTION a non-zero data value is interpreted as connected state. For CDC_RESPONSE_AVAILABLE this parameter is ignored. For CDC_NOTIFICATION_SERIAL_STATE the data should use bitmap values defined in usbcdc11.pdf, section 6.3.5, Table 69. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success ERR_API_INVALID_PARAM2 = If unsupported notification type is passed.

41.4.28 USBD_CDC_INIT_PARAM

Communication Device Class function driver initialization parameter data structure.

Table 1002.USBD_CDC_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_CDC_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_CDC_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_CDC_API::GetMemSize() routine.
cif_intf_desc	`uint8_t * USBD_CDC_INIT_PARAM::cif_intf_desc` Pointer to the control interface descriptor within the descriptor array
dif_intf_desc	`uint8_t * USBD_CDC_INIT_PARAM::dif_intf_desc` Pointer to the data interface descriptor within the descriptor array
CIC_GetRequest	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_GetRequest)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length)` Communication Interface Class specific get request call-back function. This function is provided by the application software. This function gets called when host sends CIC management element get requests. **Remark:** Applications implementing Abstract Control Model subclass can set this param to NULL. As the default driver parses ACM requests and calls the individual ACM call-back routines defined in this structure. For all other subclasses this routine should be provided by the application. The setup packet data (pSetup) is passed to the call-back so that application can extract the CIC request type and other associated data. By default the stack will assign pBuffer pointer to EP0Buff allocated at init. The application code can directly write data into this buffer as long as data is less than 64 byte. If more data has to be sent then application code should update pBuffer pointer and length accordingly. Parameters: 1. hCdc = Handle to CDC function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing request data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 4. length = Amount of data to be sent back to host. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1002.USBD_CDC_INIT_PARAM class structure

Member	Description
CIC_SetRequest	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_SetRequest)(USBD_HANDLE_T hCdc, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` Communication Interface Class specific set request call-back function. This function is provided by the application software. This function gets called when host sends a CIC management element requests. **Remark:** Applications implementing Abstract Control Model subclass can set this param to NULL. As the default driver parses ACM requests and calls the individual ACM call-back routines defined in this structure. For all other subclasses this routine should be provided by the application. The setup packet data (pSetup) is passed to the call-back so that application can extract the CIC request type and other associated data. If a set request has data associated, then this call-back is called twice. (1) First when setup request is received, at this time application code could update pBuffer pointer to point to the intended destination. The length param is set to 0 so that application code knows this is first time. By default the stack will assign pBuffer pointer to EP0Buff allocated at init. Note, if data length is greater than 64 bytes and application code doesn't update pBuffer pointer the stack will send STALL condition to host. (2) Second when the data is received from the host. This time the length param is set with number of data bytes received. Parameters: 1. hCdc = Handle to CDC function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing request data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
CDC_BulkIN_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkIN_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Communication Device Class specific BULK IN endpoint handler. The application software should provide the BULK IN endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1002.USBD_CDC_INIT_PARAM class structure

Member	Description
CDC_BulkOUT_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkOUT_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Communication Device Class specific BULK OUT endpoint handler. The application software should provide the BULK OUT endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
SendEncpsCmd	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SendEncpsCmd)(USBD_HANDLE_T hCDC, uint8_t *buffer, uint16_t len)` Abstract control model(ACM) subclass specific SEND_ENCAPSULATED_COMMAND request call-back function. This function is provided by the application software. This function gets called when host sends a SEND_ENCAPSULATED_COMMAND set request. Parameters: 1. hCdc = Handle to CDC function driver. 2. buffer = Pointer to the command buffer. 3. len = Length of the command buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
GetEncpsResp	`ErrorCode_t(* USBD_CDC_INIT_PARAM::GetEncpsResp)(USBD_HANDLE_T hCDC, uint8_t **buffer, uint16_t *len)` Abstract control model(ACM) subclass specific GET_ENCAPSULATED_RESPONSE request call-back function. This function is provided by the application software. This function gets called when host sends a GET_ENCAPSULATED_RESPONSE request. Parameters: 1. hCdc = Handle to CDC function driver. 2. buffer = Pointer to a pointer of data buffer containing response data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 3. len = Amount of data to be sent back to host. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1002.USBD_CDC_INIT_PARAM class structure

Member	Description
SetCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t *buffer, uint16_t len)` Abstract control model(ACM) subclass specific SET_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a SET_COMM_FEATURE set request. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. 3. buffer = Pointer to the settings buffer for the specified communication feature. 4. len = Length of the request buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
GetCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::GetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t **pBuffer, uint16_t *len)` Abstract control model(ACM) subclass specific GET_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a GET_ENCAPSULATED_RESPONSE request. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. 3. buffer = Pointer to a pointer of data buffer containing current settings for the communication feature. Pointer-to-pointer is used to implement zero-copy buffers. 4. len = Amount of data to be sent back to host. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1002.USBD_CDC_INIT_PARAM class structure

Member	Description
ClrCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::ClrCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature)` Abstract control model(ACM) subclass specific CLEAR_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a CLEAR_COMM_FEATURE request. In the call-back the application should Clears the settings for a particular communication feature. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. See usbcdc11.pdf, section 6.2.4, Table 47. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
SetCtrlLineState	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCtrlLineState)(USBD_HANDLE_T hCDC, uint16_t state)` Abstract control model(ACM) subclass specific SET_CONTROL_LINE_STATE request call-back function. This function is provided by the application software. This function gets called when host sends a SET_CONTROL_LINE_STATE request. RS-232 signal used to tell the DCE device the DTE device is now present Parameters: 1. hCdc = Handle to CDC function driver. 2. state = The state value uses bitmap values defined the *USB CDC class specification document* published by usb.org. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
SendBreak	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SendBreak)(USBD_HANDLE_T hCDC, uint16_t mstime)` Abstract control model(ACM) subclass specific SEND_BREAK request call-back function. This function is provided by the application software. This function gets called when host sends a SEND_BREAK request. Parameters: 1. hCdc = Handle to CDC function driver. 2. mstime = Duration of Break signal in milliseconds. If mstime is FFFFh, then the application should send break until another SendBreak request is received with the wValue of 0000h. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1002.USBD_CDC_INIT_PARAM class structure

Member	Description
SetLineCode	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetLineCode)(USBD_HANDLE_T hCDC, CDC_LINE_CODING *line_coding)` Abstract control model(ACM) subclass specific SET_LINE_CODING request call-back function. This function is provided by the application software. This function gets called when host sends a SET_LINE_CODING request. The application should configure the device per DTE rate, stop-bits, parity, and number-of-character bits settings provided in command buffer. Parameters: 1. hCdc = Handle to CDC function driver. 2. line_coding = Pointer to the CDC_LINE_CODING command buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
CDC_InterruptEP_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_InterruptEP_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Communication Device Class specific INTERRUPT IN endpoint handler. The application software should provide the INT IN endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
CDC_Ep0_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user override-able function to replace the default CDC class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_CDC_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

41.4.29 USBD_CORE_API

USBD stack Core API functions structure.

Table 1003.USBD_CORE_API class structure

Member	Description
RegisterClassHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterClassHandler)(USBD_HANDLE_T hUsb, USB_EP_HANDLER_T pfn, void *data)` Function to register class specific EP0 event handler with USB device stack. The application layer uses this function when it has to register the custom class's EP0 handler. The stack calls all the registered class handlers on any EP0 event before going through default handling of the event. This gives the class handlers to implement class specific request handlers and also to override the default stack handling for a particular event targeted to the interface. Check USB_EP_HANDLER_T for more details on how the callback function should be implemented. Also application layer could use this function to register EP0 handler which responds to vendor specific requests. Parameters: 1. hUsb = Handle to the USB device stack. 2. pfn = Class specific EP0 handler function. 3. data = Pointer to the data which will be passed when callback function is called by the stack. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = The number of class handlers registered is greater than the number of handlers allowed by the stack.
RegisterEpHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterEpHandler)(USBD_HANDLE_T hUsb, uint32_t ep_index, USB_EP_HANDLER_T pfn, void *data)` Function to register interrupt/event handler for the requested endpoint with USB device stack. The application layer uses this function to register the custom class's EP0 handler. The stack calls all the registered class handlers on any EP0 event before going through default handling of the event. This gives the class handlers to implement class specific request handlers and also to override the default stack handling for a particular event targeted to the interface. Check USB_EP_HANDLER_T for more details on how the callback function should be implemented. Parameters: 1. hUsb = Handle to the USB device stack. 2. ep_index = Class specific EP0 handler function. 3. pfn = Class specific EP0 handler function. 4. data = Pointer to the data which will be passed when callback function is called by the stack. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = Too many endpoint handlers.

Table 1003.USBD_CORE_API class structure

Member	Description
SetupStage	`void(*void USBD_CORE_API::SetupStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in setup state. This function is called by USB stack and the application layer to set the EP0 state machine in setup state. This function will read the setup packet received from USB host into stack's buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
DataInStage	`void(*void USBD_CORE_API::DataInStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in data_in state. This function is called by USB stack and the application layer to set the EP0 state machine in data_in state. This function will write the data present in EP0Data buffer to EP0 FIFO for transmission to host. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
DataOutStage	`void(*void USBD_CORE_API::DataOutStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in data_out state. This function is called by USB stack and the application layer to set the EP0 state machine in data_out state. This function will read the control data (EP0 out packets) received from USB host into EP0Data buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
StatusInStage	`void(*void USBD_CORE_API::StatusInStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in status_in state. This function is called by USB stack and the application layer to set the EP0 state machine in status_in state. This function will send zero length IN packet on EP0 to host, indicating positive status. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.

Table 1003.USBD_CORE_API class structure

Member	Description
StatusOutStage	`void(*void USBD_CORE_API::StatusOutStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in status_out state. This function is called by USB stack and the application layer to set the EP0 state machine in status_out state. This function will read the zero length OUT packet received from USB host on EP0. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
StallEp0	`void(*void USBD_CORE_API::StallEp0)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in stall state. This function is called by USB stack and the application layer to generate STALL signalling on EP0 endpoint. This function will also reset the EP0Data buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.

41.4.30 USBD_DFU_API

DFU class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 1004.USBD_DFU_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_DFU_API::GetMemSize)(USBD_DFU_INIT_PARAM_T *param)` Function to determine the memory required by the DFU function driver module. This function is called by application layer before calling pUsbApi->dfu->Init(), to allocate memory used by DFU function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing DFU function driver module initialization parameters. Returns: the required memory size in bytes.

Table 1004.USBD_DFU_API class structure

Member	Description
init	`ErrorCode_t(*ErrorCode_t USBD_DFU_API::init)(USBD_HANDLE_T hUsb, USBD_DFU_INIT_PARAM_T *param, uint32_t init_state)` Function to initialize DFU function driver module. This function is called by application layer to initialize DFU function driver module. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing DFU function driver module initialization parameters. Returns:ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either DFU_Write() or DFU_Done() or DFU_Read() callbacks are not defined. 4. ERR_USBD_BAD_DESC = USB_DFU_DESCRIPTOR_TYPE is not defined immediately after interface descriptor.wTransferSize in descriptor doesn't match the value passed in param->wTransferSize.DFU_Detach() is not defined while USB_DFU_WILL_DETACH is set in DFU descriptor. 5. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed.

41.4.31 USBD_DFU_INIT_PARAM

USB descriptors data structure.

Table 1005.USBD_DFU_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_DFU_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_DFU_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_DFU_API::GetMemSize() routine.
wTransferSize	`uint16_t USBD_DFU_INIT_PARAM::wTransferSize` DFU transfer block size in number of bytes. This value should match the value set in DFU descriptor provided as part of the descriptor array (
pad	`uint16_t USBD_DFU_INIT_PARAM::pad`
intf_desc	`uint8_t * USBD_DFU_INIT_PARAM::intf_desc` Pointer to the DFU interface descriptor within the descriptor array (

Table 1005.USBD_DFU_INIT_PARAM class structure

Member	Description
DFU_Write	`uint8_t(*uint8_t(* USBD_DFU_INIT_PARAM::DFU_Write)(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout))(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout)` DFU Write callback function. This function is provided by the application software. This function gets called when host sends a write command. For application using zero-copy buffer scheme this function is called for the first time with Parameters: 1. block_num = Destination start address. 2. src = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. bwPollTimeout = Pointer to a 3 byte buffer which the callback implementer should fill with the amount of minimum time, in milliseconds, that the host should wait before sending a subsequent DFU_GETSTATUS request. 4. length = Number of bytes to be written. Returns: DFU_STATUS_ values defined in mw_usbd_dfu.h.
DFU_Read	`uint32_t(*uint32_t(* USBD_DFU_INIT_PARAM::DFU_Read)(uint32_t block_num, uint8_t **dst, uint32_t length))(uint32_t block_num, uint8_t **dst, uint32_t length)` DFU Read callback function. This function is provided by the application software. This function gets called when host sends a read command. Parameters: 1. block_num = Destination start address. 2. dst = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Amount of data copied to destination buffer. Returns: DFU_STATUS_ values defined in mw_usbd_dfu.h.
DFU_Done	`void(*USBD_DFU_INIT_PARAM::DFU_Done)(void)` DFU done callback function. This function is provided by the application software. This function gets called after download is finished. Returns: nothing.
DFU_Detach	`void(* USBD_DFU_INIT_PARAM::DFU_Detach)(USBD_HANDLE_T hUsb)` DFU detach callback function. This function is provided by the application software. This function gets called after USB_REQ_DFU_DETACH is received. Applications which set USB_DFU_WILL_DETACH bit in DFU descriptor should define this function. As part of this function application can call Connect() routine to disconnect and then connect back with host. For application which rely on WinUSB based host application should use this feature since USB reset can be invoked only by kernel drivers on Windows host. By implementing this feature host doesn't have to issue reset instead the device has to do it automatically by disconnect and connect procedure. hUsbHandle DFU control structure. Parameters: 1. hUsb = Handle DFU control structure. Returns: nothing.

Table 1005.USBD_DFU_INIT_PARAM class structure

Member	Description
DFU_Ep0_Hdlr	`ErrorCode_t(* USBD_DFU_INIT_PARAM::DFU_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default DFU class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_DFU_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
DFU_GetStatus	`uint32_t(* USBD_DFU_INIT_PARAM::DFU_GetStatus)(uint32_t *timeout, int32_t last)` Optional callback function to get the status of the previous write (download) transfer. This function is called when the stack needs to know the status of the current transfer. Parameters: 1. timeout = If non zero, then DFU device sends dfuDNBUSY status when last = 0, or will send dfuMANIFEST when last = 1. 2. last = When 1 indicates that the download is complete. Return: Should return the 'bstatus' value to be sent to host as part DFU_GETSTATUS request. Return values: DFU_STATUS_OK on success. DFU_STATUS_errXXXXXX on DFU standard error. DFU_STATUS_errVENDOR on vendor specific error.

41.4.32 USBD_HID_API

HID class API functions structure.This structure contains pointers to all the function exposed by HID function driver module.

Table 1006.USBD_HID_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_HID_API::GetMemSize)(USBD_HID_INIT_PARAM_T *param)` Function to determine the memory required by the HID function driver module. This function is called by application layer before calling pUsbApi->hid->Init(), to allocate memory used by HID function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing HID function driver module initialization parameters. Returns: the required memory size in bytes.

Table 1006.USBD_HID_API class structure

Member	Description
init	`ErrorCode_t(*ErrorCode_t USBD_HID_API::init)(USBD_HANDLE_T hUsb, USBD_HID_INIT_PARAM_T *param)` Function to initialize HID function driver module. This function is called by application layer to initialize HID function driver module. On successful initialization the function returns a handle to HID function driver module in passed param structure. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing HID function driver module initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. ERR_API_INVALID_PARAM2 = Either HID_GetReport() or HID_SetReport() callback are not defined. ERR_USBD_BAD_DESC = HID_HID_DESCRIPTOR_TYPE is not defined immediately after interface descriptor. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

41.4.33 USBD_HID_INIT_PARAM

USB descriptors data structure.

Table 1007.USBD_HID_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_HID_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_HID_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_HID_API::GetMemSize() routine.
max_reports	`uint8_t USBD_HID_INIT_PARAM::max_reports` Number of HID reports supported by this instance of HID class driver.
pad	`uint8_t USBD_HID_INIT_PARAM::pad[3]`
intf_desc	`uint8_t * USBD_HID_INIT_PARAM::intf_desc` Pointer to the HID interface descriptor within the descriptor array (
report_data	`USB_HID_REPORT_T *USB_HID_REPORT_T* USBD_HID_INIT_PARAM::report_data` Pointer to an array of HID report descriptor data structure (**Remark:** This array should be of global scope.

Table 1007.USBD_HID_INIT_PARAM class structure

Member	Description
HID_GetReport	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length)` HID get report callback function. This function is provided by the application software. This function gets called when host sends a HID_REQUEST_GET_REPORT request. The setup packet data (**Remark:** HID reports are sent via interrupt IN endpoint also. This function is called only when report request is received on control endpoint. Application should implement HID_EpIn_Hdlr to send reports to host via interrupt IN endpoint. Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing report data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
HID_SetReport	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` HID set report callback function. This function is provided by the application software. This function gets called when host sends a HID_REQUEST_SET_REPORT request. The setup packet data (Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing report data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1007.USBD_HID_INIT_PARAM class structure

Member	Description
HID_GetPhysDesc	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetPhysDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` Optional callback function to handle HID_GetPhysDesc request. The application software could provide this callback HID_GetPhysDesc handler to handle get physical descriptor requests sent by the host. When host requests Physical Descriptor set 0, application should return a special descriptor identifying the number of descriptor sets and their sizes. A Get_Descriptor request with the Physical Index equal to 1 should return the first Physical Descriptor set. A device could possibly have alternate uses for its items. These can be enumerated by issuing subsequent Get_Descriptor requests while incrementing the Descriptor Index. A device should return the last descriptor set to requests with an index greater than the last number defined in the HID descriptor. **Remark:** Applications which don't have physical descriptor should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuf = Pointer to a pointer of data buffer containing physical descriptor data. If the physical descriptor is in USB accessible memory area application could just update the pointer or else it should copy the descriptor to the address pointed by this pointer. 4. length = Amount of data copied to destination buffer or descriptor length. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
HID_SetIdle	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetIdle)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t idleTime)` Optional callback function to handle HID_REQUEST_SET_IDLE request. The application software could provide this callback to handle HID_REQUEST_SET_IDLE requests sent by the host. This callback is provided to applications to adjust timers associated with various reports, which are sent to host over interrupt endpoint. The setup packet data (**Remark:** Applications which don't send reports on Interrupt endpoint or don't have idle time between reports should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet recived from host. 3. idleTime = Idle time to be set for the specified report. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1007.USBD_HID_INIT_PARAM class structure

Member	Description
HID_SetProtocol	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetProtocol)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t protocol)` Optional callback function to handle HID_REQUEST_SET_PROTOCOL request. The application software could provide this callback to handle HID_REQUEST_SET_PROTOCOL requests sent by the host. This callback is provided to applications to adjust modes of their code between boot mode and report mode. **Remark:** Applications which don't support protocol modes should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet recived from host. 3. protocol = Protocol mode. 0 = Boot Protocol 1 = Report Protocol Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
HID_EpIn_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpIn_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Interrupt IN endpoint event handler. The application software could provide Interrupt IN endpoint event handler. Application which send reports to host on interrupt endpoint should provide an endpoint event handler through this data member. This data member is ignored if the interface descriptor Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Handle to HID function driver. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1007.USBD_HID_INIT_PARAM class structure

Member	Description
HID_EpOut_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpOut_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Interrupt OUT endpoint event handler. The application software could provide Interrupt OUT endpoint event handler. Application which receives reports from host on interrupt endpoint should provide an endpoint event handler through this data member. This data member is ignored if the interface descriptor Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Handle to HID function driver. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
HID_GetReportDesc	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReportDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` Optional user overridable function to replace the default HID_GetReportDesc handler. The application software could override the default HID_GetReportDesc handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_HID_API::Init() and also provide report data array Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

Table 1007.USBD_HID_INIT_PARAM class structure

Member	Description
HID_Ep0_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default HID class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.

41.4.34 USBD_HW_API

Hardware API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 1008.USBD_HW_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_HW_API::GetMemSize)(USBD_API_INIT_PARAM_T *param)` Function to determine the memory required by the USB device stack's DCD and core layers. This function is called by application layer before calling pUsbApi->hw->. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing USB device stack initialization parameters. Returns: the required memory size in bytes.
Init	`ErrorCode_t(*ErrorCode_t USBD_HW_API::Init)(USBD_HANDLE_T *phUsb, USB_CORE_DESCS_T *pDesc, USBD_API_INIT_PARAM_T *param)` Function to initialize USB device stack's DCD and core layers. This function is called by application layer to initialize USB hardware and core layers. On successful initialization the function returns a handle to USB device stack which should be passed to the rest of the functions. Parameters: 1. phUsb = Pointer to the USB device stack handle of type USBD_HANDLE_T. 2. param = Structure containing USB device stack initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK(0) = On success ERR_USBD_BAD_MEM_BUF(0x0004000b) = When insufficient memory buffer is passed or memory is not aligned on 2048 boundary.

Table 1008.USBD_HW_API class structure

Member	Description
Connect	`void(*void USBD_HW_API::Connect)(USBD_HANDLE_T hUsb, uint32_t con)` Function to make USB device visible/invisible on the USB bus. This function is called after the USB initialization. This function uses the soft connect feature to make the device visible on the USB bus. This function is called only after the application is ready to handle the USB data. The enumeration process is started by the host after the device detection. The driver handles the enumeration process according to the USB descriptors passed in the USB initialization function. Parameters: 1. hUsb = Handle to the USB device stack. 2. con = States whether to connect (1) or to disconnect (0). Returns: nothing.
ISR	`void(*void USBD_HW_API::ISR)(USBD_HANDLE_T hUsb)` Function to USB device controller interrupt events. When the user application is active the interrupt handlers are mapped in the user flash space. The user application must provide an interrupt handler for the USB interrupt and call this function in the interrupt handler routine. The driver interrupt handler takes appropriate action according to the data received on the USB bus. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
Reset	`void(*void USBD_HW_API::Reset)(USBD_HANDLE_T hUsb)` Function to Reset USB device stack and hardware controller. Reset USB device stack and hardware controller. Disables all endpoints except EP0. Clears all pending interrupts and resets endpoint transfer queues. This function is called internally by pUsbApi->hw->init() and from reset event. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
ForceFullSpeed	`void(*void USBD_HW_API::ForceFullSpeed)(USBD_HANDLE_T hUsb, uint32_t cfg)` **Remark:** This API does not apply to the LPC546xx device. Function to force high speed USB device to operate in full speed mode. This function is useful for testing the behavior of current device when connected to a full speed only hosts. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - set force full-speed or 0 - clear force full-speed. Returns: nothing.

Table 1008.USBD_HW_API class structure

Member	Description
WakeUpCfg	`void(*void USBD_HW_API::WakeUpCfg)(USBD_HANDLE_T hUsb, uint32_t cfg)` Function to configure USB device controller to walk-up host on remote events. This function is called by application layer to configure the USB device controller to wake up on remote events. It is recommended to call this function from users's USB_WakeUpCfg() callback routine registered with stack. **Remark:** User's USB_WakeUpCfg() is registered with stack by setting the USB_WakeUpCfg member of USBD_API_INIT_PARAM_T structure before calling pUsbApi->hw->Init() routine. Certain USB device controllers needed to keep some clocks always on to generate resume signaling through pUsbApi->hw->WakeUp(). This hook is provided to support such controllers. In most controllers cases this is an empty routine. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - Configure controller to wake on remote events or 0 - Configure controller not to wake on remote events. Returns: nothing.
SetAddress	`void(*void USBD_HW_API::SetAddress)(USBD_HANDLE_T hUsb, uint32_t adr)` Function to set USB address assigned by host in device controller hardware. This function is called automatically when USB_REQUEST_SET_ADDRESS request is received by the stack from USB host. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. adr = USB bus Address to which the device controller should respond. Usually assigned by the USB host. Returns: nothing.
Configure	`void(*void USBD_HW_API::Configure)(USBD_HANDLE_T hUsb, uint32_t cfg)` Function to configure device controller hardware with selected configuration. This function is called automatically when USB_REQUEST_SET_CONFIGURATION request is received by the stack from USB host. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = Configuration index. Returns: nothing.

Table 1008.USBD_HW_API class structure

Member	Description
ConfigEP	`void(*void USBD_HW_API::ConfigEP)(USBD_HANDLE_T hUsb, USB_ENDPOINT_DESCRIPTOR *pEPD)` Function to configure USB Endpoint according to descriptor. This function is called automatically when USB_REQUEST_SET_CONFIGURATION request is received by the stack from USB host. All the endpoints associated with the selected configuration are configured. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. pEPD = Endpoint descriptor structure defined in USB 2.0 specification. Returns: nothing.
DirCtrlEP	`void(*void USBD_HW_API::DirCtrlEP)(USBD_HANDLE_T hUsb, uint32_t dir)` Function to set direction for USB control endpoint EP0. This function is called automatically by the stack on need basis. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - Set EP0 in IN transfer mode 0 - Set EP0 in OUT transfer mode Returns: nothing.
EnableEP	`void(*void USBD_HW_API::EnableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to enable selected USB endpoint. This function enables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing. This function enables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number corresponding to the event as per USB specification. ie. An EP1_IN is represented by 0x81 number. For device events set this param to 0x0. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. 4. enable = 1 - enable event, 0 - disable event. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK(0) = - On success ERR_USBD_INVALID_REQ(0x00040001) = - Invalid event type.

Table 1008.USBD_HW_API class structure

Member	Description
DisableEP	`void(*void USBD_HW_API::DisableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to disable selected USB endpoint. This function disables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
ResetEP	`void(*void USBD_HW_API::ResetEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to reset selected USB endpoint. This function flushes the endpoint buffers and resets data toggle logic. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
SetStallEP	`void(*void USBD_HW_API::SetStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to STALL selected USB endpoint. Generates STALL signalling for requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
ClrStallEP	`void(*void USBD_HW_API::ClrStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to clear STALL state for the requested endpoint. This function clears STALL state for the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
SetTestMode	`ErrorCode_t(*ErrorCode_t USBD_HW_API::SetTestMode)(USBD_HANDLE_T hUsb, uint8_t mode)` Function to set high speed USB device controller in requested test mode. USB-IF requires the high speed device to be put in various test modes for electrical testing. This USB device stack calls this function whenever it receives USB_REQUEST_CLEAR_FEATURE request for USB_FEATURE_TEST_MODE. Users can put the device in test mode by directly calling this function. Returns ERR_USBD_INVALID_REQ when device controller is full-speed only. Parameters: 1. hUsb = Handle to the USB device stack. 2. mode = Test mode defined in USB 2.0 electrical testing specification. Returns ErrorCode_t type to indicate success or error condition.: Return values: LPC_OK(0) = - On success ERR_USBD_INVALID_REQ(0x00040001) = - Invalid test mode or Device controller is full-speed only.

Table 1008.USBD_HW_API class structure

Member	Description
ReadEP	`uint32_t(*uint32_t USBD_HW_API::ReadEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData)` Function to read data received on the requested endpoint. This function is called by USB stack and the application layer to read the data received on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer where data is to be copied. Returns: Returns the number of bytes copied to the buffer.
ReadReqEP	`uint32_t(*uint32_t USBD_HW_API::ReadReqEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t len)` **Remark:** This API does not apply to the LPC546xx device. Function to queue read request on the specified endpoint. This function is called by USB stack and the application layer to queue a read request on the specified endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer where data is to be copied. This buffer address should be accessible by USB DMA master. 4. len = Length of the buffer passed. Returns: the length of the requested buffer.
ReadSetupPkt	`uint32_t(*uint32_t USBD_HW_API::ReadSetupPkt)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t *pData)` Function to read setup packet data received on the requested endpoint. This function is called by USB stack and the application layer to read setup packet data received on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP0_IN is represented by 0x80 number. 3. pData = Pointer to the data buffer where data is to be copied. Returns: the number of bytes copied to the buffer.
WriteEP	`uint32_t(*uint32_t USBD_HW_API::WriteEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t cnt)` Function to write data to be sent on the requested endpoint. This function is called by USB stack and the application layer to send data on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer from where data is to be copied. 4. cnt = Number of bytes to write. Returns: the number of bytes written.

Table 1008.USBD_HW_API class structure

Member	Description
WakeUp	`void(*void USBD_HW_API::WakeUp)(USBD_HANDLE_T hUsb)` Function to generate resume signaling on bus for remote host wake-up. This function is called by application layer to remotely wake up host controller when system is in suspend state. Application should indicate this remote wake up capability by setting USB_CONFIG_REMOTE_WAKEUP in bmAttributes of Configuration Descriptor. Also this routine will generate resume signalling only if host enables USB_FEATURE_REMOTE_WAKEUP by sending SET_FEATURE request before suspending the bus. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
EnableEvent	`ErrorCode_t(* USBD_HW_API::EnableEvent)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t event_type, uint32_t enable)`

41.4.35 USBD_MSC_API

MSC class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 1009.USBD_MSC_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_MSC_API::GetMemSize)(USBD_MSC_INIT_PARAM_T *param)` Function to determine the memory required by the MSC function driver module. This function is called by application layer before calling pUsbApi->msc->Init(), to allocate memory used by MSC function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing MSC function driver module initialization parameters. Returns: the required memory size in bytes.
init	`ErrorCode_t(*ErrorCode_t USBD_MSC_API::init)(USBD_HANDLE_T hUsb, USBD_MSC_INIT_PARAM_T *param)` Function to initialize MSC function driver module. This function is called by application layer to initialize MSC function driver module. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing MSC function driver module initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. ERR_API_INVALID_PARAM2 = Either MSC_Write() or MSC_Read() or MSC_Verify() callbacks are not defined. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

41.4.36 USBD_MSC_INIT_PARAM

Mass Storage class function driver initialization parameter data structure.

Table 1010.USBD_MSC_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_MSC_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_MSC_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_MSC_API::GetMemSize() routine.
InquiryStr	`uint8_t * USBD_MSC_INIT_PARAM::InquiryStr` Pointer to the 28 character string. This string is sent in response to the SCSI Inquiry command. **Remark:** The data pointed by the pointer should be of global scope.
BlockCount	`uint32_t USBD_MSC_INIT_PARAM::BlockCount` Number of blocks present in the mass storage device
BlockSize	`uint32_t USBD_MSC_INIT_PARAM::BlockSize` Block size in number of bytes
MemorySize	`uint32_t USBD_MSC_INIT_PARAM::MemorySize` Memory size in number of bytes
intf_desc	`uint8_t * USBD_MSC_INIT_PARAM::intf_desc` Pointer to the interface descriptor within the descriptor array (
MSC_Write	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Write)(uint32_t offset, uint8_t **src, uint32_t length))(uint32_t offset, uint8_t **src, uint32_t length)` MSC Write callback function. This function is provided by the application software. This function gets called when host sends a write command. Parameters: 1. offset = Destination start address. 2. src = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Number of bytes to be written. Returns: nothing.

Table 1010.USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Read	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Read)(uint32_t offset, uint8_t **dst, uint32_t length))(uint32_t offset, uint8_t **dst, uint32_t length)` MSC Read callback function. This function is provided by the application software. This function gets called when host sends a read command. Parameters: 1. offset = Source start address. 2. dst = Pointer to a pointer to the source of data. The MSC function drivers implemented in stack are written with zero-copy model. Meaning the stack doesn't make an extra copy of buffer before writing/reading data from USB hardware FIFO. Hence the parameter is pointer to a pointer containing address buffer (uint8_t** dst). So that the user application can update the buffer pointer instead of copying data to address pointed by the parameter. /note The updated buffer address should be access able by USB DMA master. If user doesn't want to use zero-copy model, then the user should copy data to the address pointed by the passed buffer pointer parameter and shouldn't change the address value. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Number of bytes to be read. Returns: Nothing.
MSC_Verify	`ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Verify)(uint32_t offset, uint8_t buf[], uint32_t length)` MSC Verify callback function. This function is provided by the application software. This function gets called when host sends a verify command. The callback function should compare the buffer with the destination memory at the requested offset and Parameters: 1. offset = Destination start address. 2. buf = Buffer containing the data sent by the host. 3. length = Number of bytes to verify. Returns: ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = If data in the buffer matches the data at destination ERR_FAILED = At least one byte is different.
MSC_GetWriteBuf	`void(*void(* USBD_MSC_INIT_PARAM::MSC_GetWriteBuf)(uint32_t offset, uint8_t **buff_adr, uint32_t length))(uint32_t offset, uint8_t **buff_adr, uint32_t length)` Optional callback function to optimize MSC_Write buffer transfer. This function is provided by the application software. This function gets called when host sends SCSI_WRITE10/SCSI_WRITE12 command. The callback function should update the Parameters: 1. offset = Destination start address. 2. buf = Buffer containing the data sent by the host. 3. length = Number of bytes to write. Returns: nothing.

Table 1010.USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Ep0_Hdlr	`ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default MSC class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_MSC_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: LPC_OK = On success. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. ERR_USBD_xxx = For other error conditions.
MemorySize64	`uint64_t USBD_MSC_INIT_PARAM::MemorySize64` If mem_size is zero, MemorySize64 will be used and passed in as memory size in bytes.

42.1 How to read this chapter

The flash signature generator is available on all LPC546xx devices.

42.2 Features

- Controls hardware flash signature generation.
- Signature generated for the entire flash or for a specified address range.

42.3 General description

The flash signature generator can generate a flash signature value for a specified address range under software control. It can also generate a flash signature value for the entire flash memory by using an IAP function call (see Section 5.4.5.18 for details).

The flash module contains a built-in signature generator. This generator can produce a 128-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (for example, during programming). The signature generator can also be accessed via an IAP function call (Section 5.4.5.18) or ISP command (Section 5.6.11).

The address range for generating a signature must be aligned on flash-word boundaries, that is, 128-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature generation is complete. Code outside of the flash (for example, internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

42.4 Register description

Remark: To configure flash access times, use the FLASHCFG register in the SYSCON block. See Section 7.5.64.

Table 1011. Register overview: FMC (base address 0x4003 4000)

Name	Access	Offset	Description	Reset value	Reference
FCTR	R/W	0x000	Control register.	0x20005	42.4.1
FBWST	R/W	0x010	Wait state register.	0xC005	42.4.2
FMSSTART	R/W	0x020	Signature start address register.	0	42.4.3.1
FMSSTOP	R/W	0x024	Signature stop-address register.	0	42.4.3.2
FMSW0	RO	0x02C	Word 0 of 128-bit signature word.	-	42.4.4
FMSW1	RO	0x030	Word 1 of 128-bit signature word.	-	42.4.4
FMSW2	RO	0x034	Word 2 of 128-bit signature word.	-	42.4.4
FMSW3	RO	0x038	Word 3 of 128-bit signature word.	-	42.4.4
FMSTAT	RO	0xFE0	Signature generation status register.	0	42.4.5
FMSTATCLR	WO	0xFE8	Signature generation status clear register.	-	42.4.6

42.4.1 Flash control register

The FCTR register controls the read mode for signature generation. Bits 3 and 4 must have the indicated values in order for signature generation to work correctly.

Table 1012. Flash control register (FCTR, offset 0x000) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved: leave existing value unchanged.	0x5
3	FS_RD0	Value must be 0 for signature generation.	0x0
4	FS_RD1	Value must be 1 for signature generation.	0x0
31:5	-	Reserved: leave existing value unchanged.	-

42.4.2 Flash wait state register

The FBWST registers sets the wait states used for flash signature generation. Signature generation does not use the flash timing value programmed in the FLASHTIM field of the FLASHCFG register that is used for normal flash accesses. Typically, The FBWST wait state value can be set to the same value used for normal flash operation.

Table 1013. Flash wait state register bit description (FBWST, offset 0x02C)

Bit	Symbol	Description	Reset value
7:0	WAITSTATES	Wait states for signature generation.	0x05
31:8	-	Reserved: leave existing value unchanged.	0xC0

42.4.3 Signature generation address and control registers

These registers control automatic signature generation. A signature can be generated for any part of the flash memory contents. The address range to be used for generation is defined by writing the start address to the signature start address register (FMSSTART) and the stop address to the signature stop address register (FMSSTOP. The start and stop addresses must be aligned to 128-bit boundaries and can be derived by dividing the byte address by 16.

Signature generation is started by setting the SIG_START bit in the FMSSTOP register. Setting the SIG_START bit is typically combined with the signature stop address in a single write.

42.4.3.1 Flash module signature start address register

Table 1014.Flash module signature start register (FMSSTART, offset 0x020) bit description

Bit	Symbol	Description	Reset value
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

42.4.3.2 Flash module signature stop address register

Table 1015.Flash module signature stop register (FMSSTOP, offset 0x024) bit description

Bit	Symbol	Description	Reset value
16:0	STOP	Stop address for signature generation (the word specified by STOP is included in the address range). The address is in units of memory words, not bytes.	0
17	SIG_START	When this bit is written to 1, signature generation starts. At the end of signature generation, this bit is automatically cleared.	0
31:18	-	Reserved. Read value is undefined, only zero should be written.	NA

42.4.4 Flash module signature generation result registers

The signature generation result registers return the flash signature produced by the embedded signature generator. The 128-bit signature is reflected by the four registers FMSW0, FMSW1, FMSW2 and FMSW3.

The generated flash signature can be used to verify the flash memory contents. The generated signature can be compared with an expected signature and thus saves time and code space. The method for generating the signature is described in Section 42.5.1.

Table 1016.Flash module signature word 0 register bit description (FMSW0, offset 0x02C)

Bit	Symbol	Description	Reset value
31:0	SW0	Word 0 of 128-bit signature (bits 31 to 0).	-

Table 1017.Flash module signature word 1 register bit description (FMSW1, offset 0x030)

Bit	Symbol	Description	Reset value
31:0	SW1	Word 1 of 128-bit signature (bits 63 to 32).	-

Table 1018.Flash module signature word 2 register bit description (FMSW2, offset 0x034)

Bit	Symbol	Description	Reset value
31:0	SW2	Word 2 of 128-bit signature (bits 95 to 64).	-

Table 1019.Flash module signature word 3 register bit description (FMSW3, offset 0x038)

Bit	Symbol	Description	Reset value
31:0	SW3	Word 3 of 128-bit signature (bits 127 to 96).	-

42.4.5 Flash module signature status register

The read-only FMSTAT register provides a means of determining when signature generation has completed. Completion of signature generation can be checked by polling the SIG_DONE bit in FMSTAT. SIG_DONE should be cleared via the FMSTATCLR register before starting a signature generation operation, otherwise the status might indicate completion of a previous operation.

Table 1020.Flash module signature status register (FMSTAT, offset 0x0FE0) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	SIG_DONE	When 1, a previously started signature generation has completed. See FMSTATCLR register description for clearing this flag.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

42.4.6 Flash module signature status clear register

The FMSTATCLR register is used to clear the signature generation completion flag.

Table 1021.Flash module signature status clear register (FMSTATCLR, offset 0x0FE8) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	SIG_DONE_CLR	Writing a 1 to this bits clears the signature generation completion flag (SIG_DONE) in the FMSTAT register.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

42.5 Functional description

42.5.1 Algorithm and procedure for signature generation

42.5.1.1 Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

Reading of the flash memory for signature generation requires a specific setting of 2 bits in the FCTR register, and uses FBWST for wait state generation. FBWST settings are the same as for the FLASHTIM field of the FLASHCFG register (see Section 7.5.64).

The signature generation is started by writing a 1 to the SIG_START bit in the FMSSTOP register. Starting the signature generation is typically combined with defining the stop address, which is done in the STOP bits of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. A safe estimation for the duration of the signature generation is:

Duration = (WAITSTATES * tcy) + 3) x (FMSSTOP - FMSSTART + 1)

When signature generation is triggered via software, the duration is in AHB clock cycles, and tcy is the time in ns for one AHB clock. The SIG_DONE bit in FMSTAT can be polled by software to determine when signature generation is complete.

After signature generation, a 128-bit signature can be read from the FMSW0 to FMSW3 registers. The 128-bit signature reflects the corrected data read from the flash. The 128-bit signature reflects flash parity bits and check bit values.

42.5.1.2 Content verification

The signature as it is read from the FMSW0 register must be equal to the reference signature. The following pseudo-code shows the algorithm to derive the reference signature:

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOP
{
     FOR i = 0 TO 126
     {
          nextSign[i] = f_Q[address][i] XOR sign[i + 1]
     }
     nextSign[127] = f_Q[address][127] XOR sign[0] XOR sign[2] XOR sign[27] XOR
     sign[29]
     sign = nextSign
}
signature128 = sign
```

43.1 How to read this chapter

This chapter applies to all LPC546xx devices.

43.2 General Description

The Enhanced Code Read Protection (ECRP) is a mechanism that allows the user to enable different features in the security system. The features are specified using a combination of OTP and flash values. Some levels are only controlled by either flash or OTP, but the majority have dual control. The overlap allows higher security by specifying access via OTP bits, which once set cannot be changed while allowing customers who are less concerned about security the ability to change levels in the flash image.

The ECRP setting is determined by reading the ECRP value from the image selected for booting (Offset 0x20) and then masking it with the value read from OTP. ECRP can come from either of the image because dual boot is supported. The OTP ECRP bits which are set (enabling a restriction) will take precedence over equivalent ECRP function values in flash. Certain aspects of ECRP are only specified in OTP (that is, Mass Erase disable), while others are only specified in flash (that is, Sector Protection count).

For Dual Enhanced images, the ECRP setting is determined by reading the ECRP value from the bootable image sector. The bootable image is defined as the highest revision image that passes the required validation methods (see Chapter 5 “LPC546xx ISP and IAP” for details).

43.3 Feature controls

Table 1022 shows the various features of the ECRP and identifies the corresponding OTP override bit (where applicable). It also identifies the features that are only present in OTP. Features implemented in flash utilize a 2 bit inverted on / off scheme to prevent malicious tampering. Once a feature is enabled (it is set to most restrictive), the only way to change a Flash ECRP setting is to:

- Either erase the sector that contains the ECRP value and re-write the appropriate pattern and associated boot code, or,
- For dual-enhanced images in which only one image is present, a second bootable image with a more recent version identifier can be programmed with a new ECRP setting.

Features implemented in OTP generally use a single bit given the “one time programmable” nature of OTP. See Chapter 46 “LPC546xx One-Time Programmable (OTP) memory and API” for details.

Table 1022.ECRP feature controls

Bits	Definition	OTP
5:0	-	n/a
	SECTOR PROTECT: Number of Sectors to protect from ISP Erase/Write operations. The number of sectors protected is encoded as a subtractive number; it is subtractive so writing more 0s can only increase the number of protected sectors. It is encoded in 6 bits so that the number of sectors protected = 63 – (bits 5:0). Therefore, 0x3F=0 sectors protected against Erase/Write during ISP; 0x3E=1 sector (Sector 0) is protected; … 0x00=63 sectors are protected. In the case where there are dual images, only the active image sector protection count is considered. If the sector protection count exceeds the number of sectors remaining to the end of the Flash, the sectors protected will wrap around to the beginning of the Flash. For example, if 3 sectors are protected in the first image and 4 sectors are protected in the second image, and the second image starts at sector 10, then only sectors 10-13 are protected. To protect the entire flash, set the count to protect the number of sectors in the flash (not the number of sectors remaining). For dual image applications, if both image are to be protected it is recommended to protect the entire Flash. **Remark:** The number of sectors to be protected wraps around the end of the Flash to the beginning. For example, if the number of sectors available in the flash is 8, the image starts at sector 6, and number of sectors protected is 4, then sectors 6, 7, 0, 1 are the sectors that will be protected. **Remark:** Regardless of how many sectors are protected by this field, an erase request to erase the entire flash (0 to last sector) will always be completed successfully if the Mass Erase disable bit in the OTP is not programmed. **Remark:** Protected sectors are always protected from ISP Erase/Write commands. If IAP Erase/Write Protection bits (11:10) are set to Disable, then IAP Erase/Write commands will work regardless of the Sector Protect setting in this field. If IAP Erase/Write Protection bits (11:10) are set to ENABLE then the sectors protected by this field will be honored.	
9:6	Reserved	
11:10	IAP ERASE / WRITE PROTECTION: = 10 IAP Sector Erase/Write protection is disabled. = 01 IAP Sector Erase/Write protection is enabled. = 00 Error condition, which is treated as IAP Erase/Write protection enabled. = 11 Error condition, which is treated as IAP Erase/Write protection enabled. **Remark:** When enabled, the count from SECTOR PROTECT specifies the number of sectors to protect. Once enabled, the sector count for protected sectors can only be modified after first using Mass Erase to erase the entire Flash.	
13:12	ISP ENTRY from bootloader: = 10 Allow ISP entry via pins. = 01 Do not allow ISP entry via pins. = 00 00 Error condition, which is treated as do not allow ISP entry. = 11 Error condition, which is treated as do not allow ISP entry. See Chapter 46 “LPC546xx One-Time Programmable (OTP) memory and API”	
15:14	ISP ENTRY from IAP call: = 10 Allow ISP entry via IAP call. = 01 Do not allow ISP entry via IAP call. = 00 Error condition, which is treated as do not allow ISP entry. = 11 Error condition, which is treated as do not allow ISP entry.	

Table 1022.ECRP feature controls

Bits	Definition	OTP
17:16	SWD ENABLE = 10 Enable external access to the chip. This option enables SWD. Internal memory contents can be read or compared. Sector protection is still applicable. Mass erase is permitted if enabled via OTP. Code can be loaded to on-chip ram and executed (ISP Go cmd). = 01 Disable external access to chip. This option disables SWD. ISP commands are restricted to prevent code being loaded to ram and executed (disables ISP Go command). Memory contents cannot be read or compared (disables ISP compare memory). Sector protection is still applicable. Mass erase is permitted if enabled via OTP. = 00 Error condition, which is treated as Disable external access to chip. = 11 Error condition, which is treated as Disable external access to chip.	
31:18	Reserved	
-	ALLOW 0 in flash ECRP (offset 0x0000 0020). This feature controls how 0 is interpreted when read from the flash ECRP location. See Chapter 46 “LPC546xx One-Time Programmable (OTP) memory and API”.	OTP only
-	MASS ERASE DISABLE: This feature enables / disables ability to issue mass erase commands in ISP (both from bootloader and IAP). See Chapter 46 “LPC546xx One-Time Programmable (OTP) memory and API”. **Remark:** When Mass Erase is enabled, the Debug Mailbox is also enabled and allows a debugger to communicate with the bootloader to execute a Mass Erase. The Debug Mailbox is disabled if Mass Erase is disabled. The term Mass Erase can mean the following: • The MASS ERASE bit in OTP that allows/disallows the entire Flash to be erased. • The process of erasing the entire on-chip Flash. This device does not have a specific ‘Mass Erase’ command. When the range passed to the erase command includes the entire Flash, it is processed as a special ‘Mass Erase’ case.	OTP only

[1] 0xFFFFFFFF => Always least restrictive; 0x00000000 [ZERO PROTECT OTP NOT PROGRAMMED] => Least restrictive; 0x00000000 [ZERO PROTECT OTP PROGRAMMED] => Most restrictive

Example: To protect the first five sectors of the boot image, disable ISP and IAP calls using flash only settings.

OTP ECRP: no settings necessary

FLASH ECRP: 0x0001 543A

Example: To protect the first five sectors of the boot image, disable ISP and IAP calls using flash only settings.

OTP ECRP: no settings necessary

FLASH ECRP: 0x0001 643D

Remark: If SWD (17:16) is DISABLED, ISP Entry from ISP (15:14) is DISABLED, and ISP Entry from Bootloader (13:12) is DISABLED, the IAP Mass Erase can still be used to erase the entire Flash to recover a device.as long as the OTP MASS ERASE is ENABLED. This will require a programming interface to program the Flash once it is erased.

Table 1023.SWD feature

SWD_DISABLE_H (Bit 10)	SWD_DISABLE_L (Bit 3)	SWD Feature
0	0	Enable
0	1	Enable
1	0	Disable
1	1	Disable

44.1 How to read this chapter

The ADC controller is available on all LPC546xx devices. The number of ADC channels available is dependent on the package size.

44.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among up to 12 pins.
- Two configurable conversion sequences with independent triggers.
- Optional automatic high/low threshold comparison and “zero crossing” detection.
- Measurement range VREFN to VREFP (typically 3 V; not to exceed VDDA voltage level).
- 12-bit conversion rate of 5.0 MHz. Options for reduced resolution at higher conversion rates.
- Burst conversion mode for single or multiple inputs.
- Synchronous or asynchronous operation. Asynchronous operation maximizes flexibility in choosing the ADC clock frequency, Synchronous mode minimizes trigger latency and can eliminate uncertainty and jitter in response to a trigger.
- A temperature sensor is connected as an alternative input for ADC channel 0.

44.3 Basic configuration

Configure the ADC as follows:

- Clear the PDEN_ADC0 bit in the PDRUNCFG0 register (Section 7.5.84) to enable the analog portion of the ADC.
- Set the ADC0 bit in the AHBCLKCTRL0 register (Section 7.5.19) to enable the clock to the ADC0 register interface and the ADC0 clock. This must be done at least 20 μs after clearing the PDRUNCFG0 PDEN_ADC0 bit.
- Clear the ADC0 peripheral reset using the PRESETCTRL0 register (Section 7.5.9).
- The ADC block creates three interrupts which are connected to the NVIC: ADC0_SEQA, ADC0_SEQB, and ADC0_THCMP. The ADC0_THCMP interrupt at the NVIC combines ADC0_THCMP and ADC0_OVR conditions from the ADC as described in this chapter. See Table 88 for interrupt numbers. The sequence interrupts can also be configured as DMA triggers through the INPUT MUX (see Section 11.6.3) for each DMA channel and as inputs to the SCT.
- Use IOCON (Chapter 10 “LPC546xx I/O pin configuration (IOCON)”) to disable digital function on the pins that will be used as ADC inputs.

- Either calibration or a dummy conversion cycle is required after every reset or power cycle of the ADC. See Section 44.6.1. Note that the ADC may be power cycled in deep-sleep mode if it is not requested to stay on when these modes are invoked by the power_mode_configure API. To perform a calibration, use the ADC API.
- There are two options in the ADC CTRL register to clock ADC conversions:
  - Use the system clock to clock the ADC in synchronous mode. This option allows exact timing of triggers but requires a system clock of 80 MHz to obtain the full ADC conversion speed.
  - Use the ADC clock, determined by the ADCCLKSEL register (Section 7.5.34) and the ADCCLKDIV register (Section 7.5.54). Some clock sources are independent of the system clock, and may require extra time to synchronize ADC trigger inputs.

Configure the temperature sensor as follows (see Chapter 45 “LPC546xx Temperature sensor”):

- Clear the PDEN_TS bit in the PDRUNCFG0 register (Section 7.5.84) in order to enable the temperature sensor.
- Select the temperature sensor as source for channel 0 of the ADC by writing the value 3 to the INSEL register (see Section 44.6.2). In order to return ADC channel 0 to measuring its related device pin, write a 0 to the INSEL register.
- The digital temperature reading is available after an analog-to-digital conversion of ADC channel 0.

Remark: To convert the ADC conversion result into a temperature reading, see the specific device data sheet for calibration information.

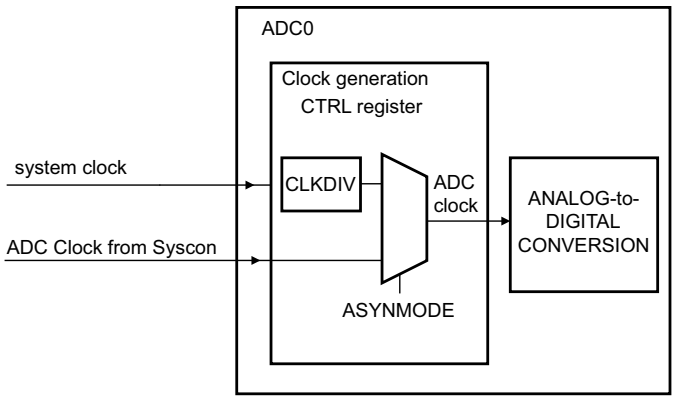


Fig 177. ADC clocking

44.4 Pin description

The ADC can measure the voltage on any of the input signals on the analog input channel. Digital signals must be disconnected from the ADC input pins when the ADC function is to be used by setting DIGIMODE = 0 on those pins in the related IOCON registers.

Warning: If the ADC is used, signal levels on analog input pins must not be above the level of V_{DDA} at any time. Otherwise, ADC readings will be invalid. If the ADC is not used in an application, then the pins associated with ADC inputs can be used as digital I/O pins.

The ADC pin triggers are movable (digital) functions. To use external pin triggers for ADC conversions, assign the pin triggers to a pin via IOCON. In addition to assigning the pin triggers to a pin, they must also be selected in the conversion sequence registers for each ADC conversion sequence defined.

The V_{REFP} and V_{REFN} pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x input voltage V_{IN})/(V_{REFP} - V_{REFN}). The result of an input voltage below V_{REFN} is 0, and the result of an input voltage above V_{REFP} is 4095 (0xFFF).

Analog Power and Ground should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. If the ADC is not used, V_{DDA} and V_{REFP} should be tied to V_{DD}, and V_{SSA} and V_{REFN} should be tied to V_{SS}.

Recommended IOCON settings are shown in Table 1026.

Table 1024.ADC common supply and reference pins

Pin	Description
VDDA	Analog supply voltage. V_{REFP} must not exceed the voltage level on V_{DDA}. This pin should be tied to V_{DD} (not left floating) if the ADC is not used. **Remark:** The supply voltage V_{DD} must be equal to V_{DDA}.
VSSA	Analog ground. This pin should be tied to V_{SS} (not left floating) if the ADC is not used.
VREFP	Positive reference voltage. To operate the ADC within specifications at the maximum sampling rate, ensure that V_{REFP} = V_{DDA}. This pin should be tied to V_{DD} (not left floating) if the ADC is not used. **Remark:** V_{REFP} is internally connected (not separately pinned) with V_{DDA} for some packages/part numbers.
VREFN	Negative reference voltage. The voltage level should typically be equal Vss and Vssa.This pin should be tied to V_{SS} (not left floating) if the ADC is not used. **Remark:** V_{REFN} is internally connected (not separately pinned) with V_{SSA} for some packages/part numbers.

Table 1025.ADC0 pin description

Function	Connect to	Description
ADC0_0	PIO0_10	Analog input channel 0.
ADC0_1	PIO0_11	Analog input channel 1.
ADC0_2	PIO0_12	Analog input channel 2.
ADC0_3	PIO0_15	Analog input channel 3.
ADC0_4	PIO0_16	Analog input channel 4.
ADC0_5	PIO0_31	Analog input channel 5.

Table 1025.ADC0 pin description

Function	Connect to	Description
ADC0_6	PIO1_0	Analog input channel 6.
ADC0_7	PIO2_0	Analog input channel 7.
ADC0_8	PIO2_1	Analog input channel 8.
ADC0_9	PIO3_21	Analog input channel 9.
ADC0_10	PIO3_22	Analog input channel 10.
ADC0_11	PIO0_23	Analog input channel 11.
ADC0_PINTRIG0	PINT0	ADC0 pin trigger input 0, from Pin Interrupt 0. Select in SEQA_CTRL or SEQB_CTRL register.
ADC0_PINTRIG1	PINT1	ADC0 pin trigger input 1, from Pin Interrupt 1. Select in SEQA_CTRL or SEQB_CTRL register.

Table 1026.Suggested ADC input pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
11	-	OD: Set to 0.	-
10	-	Not used, set to 0.	-
9	-	FILTEROFF: Set to 1.	-
8	-	DIGIMODE: Set to 0.	-
7	-	INVERT: Set to 0.	-
6	-	Not used, set to 0.	-
5:4	-	MODE: Set to 00.	-
3:0	-	FUNC: Select GPIO as the pin function.	-
General comment	Not applicable for ADC.	Only potential choice for ADC inputs.	Not applicable for ADC.

44.5 General description

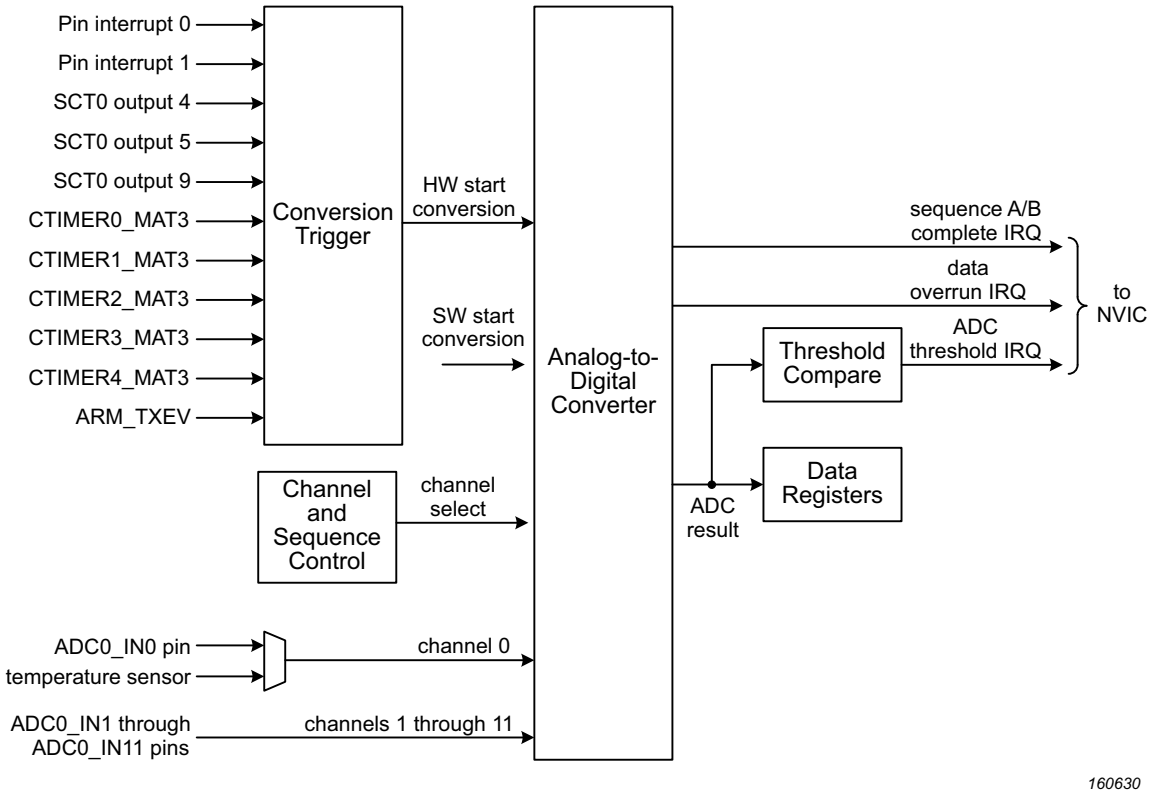


Fig 178. ADC block diagram

The ADC controller provides a great deal of flexibility in launching and controlling sequences of ADC conversions using the associated 12-bit, successive approximation ADC converter. ADC conversion sequences can be initiated under software control or in response to a selected hardware trigger.

Once the triggers are set up (software and hardware triggers can be mixed), the ADC runs through the pre-defined conversion sequences converting a set of samples whenever a trigger signal arrives until the sequence is disabled.

The ADC controller uses the system clock as a bus clock. The system clock or the asynchronous ADC clock (see Figure 177) can be used to create the ADC clock which drives the successive approximation process:

- In the synchronous operating mode, this ADC clock is derived from the system clock. In this mode, a programmable divider is included to scale the system clock to the maximum ADC clock rate of 80 MHz.
- In the asynchronous mode, an independent clock source is used as the ADC clock source without any further divider in the ADC. The maximum ADC clock rate is 80 MHz as well. In this mode, the ADC clock frequency must not exceed five times the system clock.

A fully accurate conversion requires 15 ADC clocks.

44.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 1027.Register overview: ADC (base address 0x400A 0000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	ADC Control register. Contains the clock divide value, resolution selection, sampling time selection, and mode controls.	0x600	44.6.1
INSEL	R/W	0x004	Input Select. Allows selection of the temperature sensor as an alternate input to ADC channel 0.	0	44.6.2
SEQA_CTRL	R/W	0x008	ADC Conversion Sequence-A control register: Controls triggering and channel selection for conversion sequence-A. Also specifies interrupt mode for sequence-A.	0	44.6.3
SEQB_CTRL	R/W	0x00C	ADC Conversion Sequence-B Control register: Controls triggering and channel selection for conversion sequence-B. Also specifies interrupt mode for sequence-B.	0	44.6.4
SEQA_GDAT	RO	0x010	ADC Sequence-A Global Data register. This register contains the result of the most recent ADC conversion performed under sequence-A.	-	44.6.5
SEQB_GDAT	RO	0x014	ADC Sequence-B Global Data register. This register contains the result of the most recent ADC conversion performed under sequence-B.	-	44.6.5
DAT0	RO	0x020	ADC Channel 0 Data register. This register contains the result of the most recent conversion completed on channel 0.	-	44.6.6
DAT1	RO	0x024	ADC Channel 1 Data register. This register contains the result of the most recent conversion completed on channel 1.	-	44.6.6
DAT2	RO	0x028	ADC Channel 2 Data register. This register contains the result of the most recent conversion completed on channel 2.	-	44.6.6
DAT3	RO	0x02C	ADC Channel 3 Data register. This register contains the result of the most recent conversion completed on channel 3.	-	44.6.6
DAT4	RO	0x030	ADC Channel 4 Data register. This register contains the result of the most recent conversion completed on channel 4.	-	44.6.6
DAT5	RO	0x034	ADC Channel 5 Data register. This register contains the result of the most recent conversion completed on channel 5.	-	44.6.6
DAT6	RO	0x038	ADC Channel 6 Data register. This register contains the result of the most recent conversion completed on channel 6.	-	44.6.6
DAT7	RO	0x03C	ADC Channel 7 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	44.6.6
DAT8	RO	0x040	ADC Channel 8 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	44.6.6
DAT9	RO	0x044	ADC Channel 9 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	44.6.6
DAT10	RO	0x048	ADC Channel 10 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	44.6.6
DAT11	RO	0x04C	ADC Channel 11 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	44.6.6

Table 1027.Register overview: ADC (base address 0x400A 0000)

Name	Access	Offset	Description	Reset value	Section
THR0_LOW	R/W	0x050	ADC Low Compare Threshold register 0: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0.	0x0	44.6.7
THR1_LOW	R/W	0x054	ADC Low Compare Threshold register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1.	0	44.6.7
THR0_HIGH	R/W	0x058	ADC High Compare Threshold register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0.	0	44.6.8
THR1_HIGH	R/W	0x05C	ADC High Compare Threshold register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1.	0	44.6.8
CHAN_ THRSEL	R/W	0x060	ADC Channel-Threshold Select register. Specifies which set of threshold compare registers are to be used for each channel	0	44.6.9
INTEN	R/W	0x064	ADC Interrupt Enable register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated.	0	44.6.10
FLAGS	RO	0x068	ADC Flags register. Contains the four interrupt/DMA trigger flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers).	0	44.6.11
STARTUP	R/W	0x06C	ADC Startup register (typically only used by the ADC API).	0	44.6.12
CALIB	R/W/RO	0x070	ADC Calibration register.	0	44.6.13

44.6.1 ADC Control register

This register specifies the clock divider value to be used to generate the ADC clock in synchronous mode and general operating mode bits including resolution and sampling time.

Table 1028.ADC Control register (CTRL, offset 0x0) bit description

Bit	Symbol	Value	Description	Reset value
7:0	CLKDIV		In synchronous mode only, the system clock is divided by this value plus one to produce the clock for the ADC converter, which should be less than or equal to 80 MHz. Typically, software should program the smallest value in this field that yields this maximum clock rate or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable. **Remark:** This field is ignored in the asynchronous operating mode.	0
8	ASYNMODE		Select clock mode.	0
		0	Synchronous mode. The ADC clock is derived from the system clock based on the divide value selected in the CLKDIV field. The ADC clock will be started in a controlled fashion in response to a trigger to eliminate any uncertainty in the launching of an ADC conversion in response to any synchronous (on-chip) trigger. In Synchronous mode with the SYNCBYPASS bit (in a sequence control register) set, sampling of the ADC input and start of conversion will initiate 2 system clocks after the leading edge of a (synchronous) trigger pulse.	
		1	Asynchronous mode. The ADC clock is based on the output of the ADC clock divider ADCCLKSEL in the SYSCON block. The maximum ADC clock frequency must not exceed 80 MHz.	
10:9	RESOL		The number of bits of ADC resolution. Accuracy can be reduced to achieve higher conversion rates. A single conversion (including one conversion in a burst or sequence) requires the selected number of bits of resolution plus 3 ADC clocks. **Remark:** This field must only be altered when the ADC is fully idle. Changing it during any kind of ADC operation may have unpredictable results. **Remark:** ADC clock frequencies for various resolutions must not exceed: - 5x the system clock rate for 12-bit resolution - 4.3x the system clock rate for 10-bit resolution - 3.6x the system clock for 8-bit resolution - 3x the bus clock rate for 6-bit resolution	0x3
		0x0	6-bit resolution. An ADC conversion requires 9 ADC clocks, plus any clocks specified by the TSAMP field.	
		0x1	8-bit resolution. An ADC conversion requires 11 ADC clocks, plus any clocks specified by the TSAMP field.	
		0x2	10-bit resolution. An ADC conversion requires 13 ADC clocks, plus any clocks specified by the TSAMP field.	
		0x3	12-bit resolution. An ADC conversion requires 15 ADC clocks, plus any clocks specified by the TSAMP field.	

Table 1028.ADC Control register (CTRL, offset 0x0) bit description

Bit	Symbol	Value	Description	Reset value
11	BYPASSCAL		Bypass Calibration. This bit may be set to avoid the need to calibrate if offset error is not a concern in the application.	0
		0	Calibrate. The stored calibration value will be applied to the ADC during conversions to compensated for offset error. A calibration cycle must be performed each time the chip is powered-up. Re-calibration may be warranted periodically - especially if operating conditions have changed.	
		1	Bypass calibration. Calibration is not utilized. Less time is required when enabling the ADC - particularly following chip power-up. Attempts to launch a calibration cycle are blocked when this bit is set.	
14:12	TSAMP		Sample Time. The default sampling period (TSAMP = “000”) at the start of each conversion is 2.5 ADC clock periods. Depending on a variety of factors, including operating conditions and the output impedance of the analog source, longer sampling times may be required. See Section 44.7.10. The TSAMP field specifies the number of additional ADC clock cycles, from zero to seven, by which the sample period will be extended. The total conversion time will increase by the same number of clocks. 000 - The sample period will be the default 2.5 ADC clocks. A complete conversion with 12-bits of accuracy will require 15 clocks. 001- The sample period will be extended by one ADC clock to a total of 3.5 clock periods. A complete 12-bit conversion will require 16 clocks. 010 - The sample period will be extended by two clocks to 4.5 ADC clock cycles. A complete 12-bit conversion will require 17 ADC clocks. ... 111 - The sample period will be extended by seven clocks to 9.5 ADC clock cycles. A complete 12-bit conversion will require 22 ADC clocks.	0
31:15			Reserved. Read value is undefined, only zero should be written.	0

44.6.2 Input Select register

This register allows the temperature sensor to be used as an alternate input for ADC channel 0.

Table 1029.Input Select register (INSEL, offset 0x04) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Selects the input source for channel 0. All other values are reserved.	0
		0x0	ADC0_IN0 function.	
		0x3	Internal temperature sensor.	
31:2			Reserved. Read value is undefined, only zero should be written.	0

44.6.3 ADC Conversion Sequence A Control register

There are two independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the A sequence and contains bits to allow software to initiate that conversion sequence.

Remark: Set the BURST and SEQU_ENA bits at the same time.

Table 1030.ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description

Bit	Symbol	Value	Description	Reset value
11:0	CHANNELS		Selects which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. **Remark:** This field can ONLY be changed while SEQA_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.	0x00
17:12	TRIGGER		Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. Writing all-zeroes (the default value) to this field will disable hardware triggering entirely for this sequence. See Table 1044. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
18	TRIGPOL		Select the polarity of the selected input trigger for this conversion sequence. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0
		0	Negative edge. A negative edge launches the conversion sequence on the selected trigger input.	
		1	Positive edge. A positive edge launches the conversion sequence on the selected trigger input.	
19	SYNCBYPASS		Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.	0
		0	Enable trigger synchronization. The hardware trigger bypass is not enabled.	
		1	Bypass trigger synchronization. The hardware trigger bypass is enabled.	
25:20	-		Reserved. Read value is undefined, only zero should be written.	N/A

Table 1030.ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
26	START		Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. **Remark:** This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.	0
27	BURST		Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence A triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. Note that a new sequence could begin just before BURST is cleared. For DMA triggering, clear this bit along with SEQA_ENA at the end of the DMA transfer.	0
28	SINGLESTEP		When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.	0
29	LOWPRIO		Set priority for sequence A. Setting this bit to a 1 will permit any enabled B sequence trigger (including a B sequence software start) to immediately interrupt sequence A and launch a B sequence in its place. The conversion currently in progress will be terminated. Sequence A that was interrupted will automatically resume after Sequence B completes. The conversion of the channel that was terminated is re-sampled and the conversion sequence resumes from that point. 0 High priority. Any B trigger that occurs while an A conversion sequence is active is ignored and lost.	0
		0	High priority. Sequence B will not interrupt sequence A. Any sequence B trigger that occurs while an A conversion sequence is active will be ignored and lost.	
		1	Low priority. Any enabled B sequence trigger (including a B sequence software start) will immediately interrupt sequence A and launch a B sequence in its place. If a conversion is currently in progress, it will be terminated. The A sequence that was interrupted will automatically resume after the B sequence is completed. The channel with the terminated conversion will be re-sampled and the conversion sequence will resume from that point.	

Table 1030.ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
30	MODE		Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQA_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below.	0
		0	End of conversion. The sequence A interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence A. This flag will mirror the DATAVALID bit in the SEQA_GDAT register. The OVERRUN bit in the SEQA_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled.	
		1	End of sequence. The sequence A interrupt/DMA trigger will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.	
31	SEQA_ENA		Sequence Enable. **Remark:** In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQA_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.	0
		0	Disabled. Sequence A is disabled. Sequence A triggers are ignored. If this bit is cleared while sequence A is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.	
		1	Enabled. Sequence A is enabled.	

44.6.4 ADC Conversion Sequence B Control register

There are two independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the B sequence, as well bits to allow software to initiate that conversion sequence.

Table 1031.ADC Conversion Sequence B Control register (SEQB_CTRL, offset 0x0C) bit description

Bit	Symbol	Value	Description	Reset value
11:0	CHANNELS		Selects which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. **Remark:** This field can ONLY be changed while SEQB_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.	0x00
17:12	TRIGGER		Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. Writing all-zeroes (the default value) to this field will disable hardware triggering entirely for this sequence. See Table 1044. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQB_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
18	TRIGPOL		Select the polarity of the selected input trigger for this conversion sequence. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQB_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0
		0	Negative edge. A negative edge launches the conversion sequence on the selected trigger input.	
		1	Positive edge. A positive edge launches the conversion sequence on the selected trigger input.	
19	SYNCBYPASS		Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.	0
		0	Enable synchronization. The hardware trigger bypass is not enabled.	
		1	Bypass synchronization. The hardware trigger bypass is enabled.	
25:20	-		Reserved. Read value is undefined, only zero should be written.	N/A

Table 1031.ADC Conversion Sequence B Control register (SEQB_CTRL, offset 0x0C) bit description

Bit	Symbol	Value	Description	Reset value
26	START		Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. **Remark:** This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.	0
27	BURST		Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence B triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated.	0
28	SINGLESTEP		When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.	0
29	-		Reserved. Read value is undefined, only zero should be written.	N/A
30	MODE		Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQB_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-B will be generated and which overrun conditions contribute to an overrun interrupt as described below.	0
		0	End of conversion. The sequence B interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence B. This flag will mirror the DATAVALID bit in the SEQB_GDAT register. The OVERRUN bit in the SEQB_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled.	
		1	End of sequence. The sequence B interrupt/DMA trigger will be set when the entire set of sequence-B conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQB_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.	
31	SEQB_ENA		Sequence Enable. **Remark:** In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQB_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.	0
		0	Disabled. Sequence B is disabled. Sequence B triggers are ignored. If this bit is cleared while sequence B is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.	
		1	Enabled. Sequence B is enabled.	

44.6.5 ADC Global Data register A and B

The ADC Global Data registers contain the result of the most recent ADC conversion completed under each conversion sequence.

Results of ADC conversions can be read in one of two ways. One is to use these ADC Global Data registers to read data from the ADC at the end of each ADC conversion. Another is to read the individual ADC Channel Data registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

The global registers are useful in conjunction with DMA operation - particularly when the channels selected for conversion are not sequential (hence the addresses of the individual result registers will not be sequential, making it difficult for the DMA engine to address them). For interrupt-driven code it will more likely be advantageous to wait for an entire sequence to complete and then retrieve the results from the individual channel registers.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQn_CTRL register since this will impact interrupt and overrun flag generation.

Table 1032.ADC Sequence A Global Data register (SEQA_GDAT, offset 0x10) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
15:4	RESULT	This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP. DATAVALID = 1 indicates that this result has not yet been read.	-
17:16	THCMP RANGE	Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH).	
19:18	THCMP CROSS	Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred.	
25:20	-	Reserved.	-
29:26	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.).	-
30	OVER RUN	This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQAA_CTRL) for the corresponding sequence is set to 0 (and if the overrun interrupt is enabled).	0
31	DATA VALID	This bit is set to 1 at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQA_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).	0

Table 1033.ADC Sequence B Global Data register (SEQB_GDAT, offset 0x14) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
15:4	RESULT	This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP. DATAVALID = 1 indicates that this result has not yet been read.	-
17:16	THCMP RANGE	Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH).	
19:18	THCMP CROSS	Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred.	
25:20	-	Reserved.	-
29:26	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.).	-
30	OVER RUN	This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQB_CTRL) for the corresponding sequence is set to 0 (and if the overrun interrupt is enabled).	0
31	DATA VALID	This bit is set to 1 at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQB_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).	0

44.6.6 ADC Channel Data registers 0 to 11

The ADC Channel Data registers hold the result of the last conversion completed for each ADC channel. They also include status bits to indicate when a conversion has been completed, when a data overrun has occurred, and where the most recent conversion fits relative to the range dictated by the high and low threshold registers.

Results of ADC conversion can be read in one of two ways. One is to use the ADC Global Data registers for each of the sequences to read data from the ADC at the end of each ADC conversion. Another is to use these individual ADC Channel Data registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQ_CTRL register since this will impact interrupt and overrun flag generation.

The information presented in the DAT registers always pertains to the most recent conversion completed on that channel regardless of what sequence requested the conversion or which trigger caused it.

The OVERRUN fields for each channel are also replicated in the FLAGS register.

Table 1034.ADC Data registers (DAT[0:11], offset [0x020:0x04C]) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
15:4	RESULT	This field contains the 12-bit ADC conversion result from the last conversion performed on this channel. This will be a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	-
17:16	THCMP RANGE	Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.	-

Table 1034.ADC Data registers (DAT[0:11], offset [0x020:0x04C]) bit description

Bit	Symbol	Description	Reset value
19:18	THCMP CROSS	Threshold Crossing Comparison result. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 0x2 = Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 0x3 = Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.	-
25:20	-	Reserved. Read value is undefined, only zero should be written.	-
29:26	CHANNEL	This field is hard-coded to contain the channel number that this particular register relates to (i.e. this field will contain 0b0000 for the DAT0 register, 0b0001 for the DAT1 register, etc)	-
30	OVER RUN	This bit will be set to a 1 if a new conversion on this channel completes and overwrites the previous contents of the RESULT field before it has been read, that is, while the DATA VALID bit is set. This bit is cleared, along with the DATA VALID bit, whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. This bit (in any of the 12 registers) will cause an overrun interrupt/DMA trigger to be asserted if the overrun interrupt is enabled. **Remark:** While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DATA VALID and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.	-
31	DATA VALID	This bit is set to 1 when an ADC conversion on this channel completes. This bit is cleared whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. **Remark:** While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DATA VALID and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.	-

44.6.7 ADC Compare Low Threshold registers 0 and 1

These registers set the LOW threshold levels against which ADC conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result LESS THAN this value on any channel will cause the THCMP_RANGE status bits for that channel to be set to 0b01. This result will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

If, for two successive conversions on a given channel, one result is below this threshold and the other is equal-to or above this threshold, than a threshold crossing has occurred. In this case the MSB of the THCMP_CROSS status bits will indicate that a threshold crossing has occurred and the LSB will indicate the direction of the crossing. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 1035.ADC Compare Low Threshold register 0 (THR0_LOW, offset 0x50) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRLOW	Low threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

Table 1036.ADC Compare Low Threshold register 1 (THR1_LOW, offset 0x54) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRLOW	Low threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

44.6.8 ADC Compare High Threshold registers 0 and 1

These registers set the HIGH threshold level against which ADC conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result greater than this value on any channel will cause the THCMP status bits for that channel to be set to 0b10. This result will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 1037.Compare High Threshold register0 (THR0_HIGH, offset 0x58) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRHIGH	High threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

Table 1038.Compare High Threshold register 1 (THR1_HIGH, offset 0x5C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRHIGH	High threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

44.6.9 ADC Channel Threshold Select register

For each channel, this register indicates which pair of threshold registers conversion results should be compared to.

Table 1039.ADC Channel Threshold Select register (CHAN_THRSEL, offset 0x60) bit description

Bit	Symbol	Value	Description	Reset value
0	CH0_THRSEL		Threshold select for channel 0.	0
		0	Threshold 0. Results for this channel will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers.	
		1	Threshold 1. Results for this channel will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers.	
1	CH1_THRSEL		Threshold select for channel 1. See description for channel 0.	0
2	CH2_THRSEL		Threshold select for channel 2. See description for channel 0.	0
3	CH3_THRSEL		Threshold select for channel 3. See description for channel 0.	0
4	CH4_THRSEL		Threshold select for channel 4. See description for channel 0.	0
5	CH5_THRSEL		Threshold select for channel 5. See description for channel 0.	0
6	CH6_THRSEL		Threshold select for channel 6. See description for channel 0.	0
7	CH7_THRSEL		Threshold select for channel 7. See description for channel 0.	0
8	CH8_THRSEL		Threshold select for channel 8. See description for channel 0.	0
9	CH9_THRSEL		Threshold select for channel 9. See description for channel 0.	0
10	CH10_THRSEL		Threshold select for channel 10. See description for channel 0.	0
11	CH11_THRSEL		Threshold select for channel 11. See description for channel 0.	0
31:12	-		Reserved. Read value is undefined, only zero should be written.	-

44.6.10 ADC Interrupt Enable register

There are four separate interrupt requests generated by the ADC: conversion, these are -complete or sequence-complete interrupts for each of the two sequences, a threshold-comparison out-of-range interrupt, and a data overrun interrupt. The two conversion/sequence-complete interrupts can also serve as DMA triggers. The threshold and data overrun interrupts share a slot in the NVIC.

These interrupts may be combined into one request on some chips if there is a limited number of interrupt slots. This register contains the interrupt-enable bits for each interrupt.

In this register, threshold events selected in the ADCMPINTENn bits are described as follows:

- Disabled: Threshold comparisons on channel n will not generate an ADC threshold-compare interrupt/DMA trigger.
- Outside threshold: A conversion result on channel n which is outside the range specified by the designated HIGH and LOW threshold registers will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.
- Crossing threshold: Detection of a threshold crossing on channel n will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.

Remark: Overrun and threshold-compare interrupts related to a particular channel will occur regardless of which sequence was in progress at the time the conversion was performed or what trigger caused the conversion.

Table 1040.ADC Interrupt Enable register (INTEN, offset 0x64) bit description

Bit	Symbol	Value	Description	Reset value
0	SEQA_INTEN		Sequence A interrupt enable.	0
		0	Disabled. The sequence A interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQA_CTRL register.	
1	SEQB_INTEN		Sequence B interrupt enable.	0
		0	Disabled. The sequence B interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence B interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence B, or upon completion of the entire B sequence of conversions, depending on the MODE bit in the SEQB_CTRL register.	
2	OVR_INTEN		Overrun interrupt enable.	0
		0	Disabled. The overrun interrupt is disabled.	
		1	Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 12 channel data registers will cause an overrun interrupt/DMA trigger. In addition, if the MODE bit for a particular sequence is 0, then an overrun in the global data register for that sequence will also cause this interrupt/DMA trigger to be asserted.	

Table 1040.ADC Interrupt Enable register (INTEN, offset 0x64) bit description

Bit	Symbol	Value	Description	Reset value
4:3	ADCMPINTEN0		Threshold comparison interrupt enable for channel 0.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
6:5	ADCMPINTEN1		Channel 1 threshold comparison interrupt enable. See description for channel 0.	00
8:7	ADCMPINTEN2		Channel 2 threshold comparison interrupt enable. See description for channel 0.	00
10:9	ADCMPINTEN3		Channel 3 threshold comparison interrupt enable. See description for channel 0.	00
12:11	ADCMPINTEN4		Channel 4 threshold comparison interrupt enable. See description for channel 0.	00
14:13	ADCMPINTEN5		Channel 5 threshold comparison interrupt enable. See description for channel 0.	00
16:15	ADCMPINTEN6		Channel 6 threshold comparison interrupt enable. See description for channel 0.	00
18:17	ADCMPINTEN7		Channel 7 threshold comparison interrupt enable. See description for channel 0.	00
20:19	ADCMPINTEN8		Channel 8 threshold comparison interrupt enable. See description for channel 0.	00
22:21	ADCMPINTEN9		Channel 9 threshold comparison interrupt enable. See description for channel 0.	00
24:23	ADCMPINTEN10		Channel 10 threshold comparison interrupt enable. See description for channel 0.	00
26:25	ADCMPINTEN11		Channel 21 threshold comparison interrupt enable. See description for channel 0.	00
31:27	-		Reserved. Read value is undefined, only zero should be written.	-

44.6.11 ADC Flags register

The ADC Flags status registers contains the four interrupt/DMA trigger flags along with the individual overrun flags that contribute to an overrun interrupt and the component threshold-comparison flags that contribute to that interrupt. Note that the threshold and overrun interrupts hare a slot in the NVIC.

The channel OVERRUN flags mirror those appearing in the individual DAT registers for each channel and indicate a data overrun in each of those registers.

Likewise, the SEQA_OVR and SEQB_OVR bits mirror the OVERRUN bits in the two global data registers (SEQA_GDAT and SEQB_GDAT).

Remark: The SEQn_INT conversion/sequence-complete flags also serve as DMA triggers.

Table 1041.ADC Flags register (FLAGS, offset 0x68) bit description

Bit	Symbol	Description	Reset value
0	THCMP0	Threshold comparison event on Channel 0. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0
1	THCMP1	Threshold comparison event on Channel 1. See description for channel 0.	0
2	THCMP2	Threshold comparison event on Channel 2. See description for channel 0.	0
3	THCMP3	Threshold comparison event on Channel 3. See description for channel 0.	0
4	THCMP4	Threshold comparison event on Channel 4. See description for channel 0.	0
5	THCMP5	Threshold comparison event on Channel 5. See description for channel 0.	0
6	THCMP6	Threshold comparison event on Channel 6. See description for channel 0.	0
7	THCMP7	Threshold comparison event on Channel 7. See description for channel 0.	0
8	THCMP8	Threshold comparison event on Channel 8. See description for channel 0.	0
9	THCMP9	Threshold comparison event on Channel 9. See description for channel 0.	0
10	THCMP10	Threshold comparison event on Channel 10. See description for channel 0.	0
11	THCMP11	Threshold comparison event on Channel 11. See description for channel 0.	0
12	OVERRUN0	Mirrors the OVERRRUN status flag from the result register for ADC channel 0	0
13	OVERRUN1	Mirrors the OVERRRUN status flag from the result register for ADC channel 1	0
14	OVERRUN2	Mirrors the OVERRRUN status flag from the result register for ADC channel 2	0
15	OVERRUN3	Mirrors the OVERRRUN status flag from the result register for ADC channel 3	0
16	OVERRUN4	Mirrors the OVERRRUN status flag from the result register for ADC channel 4	0
17	OVERRUN5	Mirrors the OVERRRUN status flag from the result register for ADC channel 5	0
18	OVERRUN6	Mirrors the OVERRRUN status flag from the result register for ADC channel 6	0
19	OVERRUN7	Mirrors the OVERRRUN status flag from the result register for ADC channel 7	0
20	OVERRUN8	Mirrors the OVERRRUN status flag from the result register for ADC channel 8	0
21	OVERRUN9	Mirrors the OVERRRUN status flag from the result register for ADC channel 9	0
22	OVERRUN10	Mirrors the OVERRRUN status flag from the result register for ADC channel 10	0
23	OVERRUN11	Mirrors the OVERRRUN status flag from the result register for ADC channel 11	0
24	SEQA_OVR	Mirrors the global OVERRUN status flag in the SEQA_GDAT register	0
25	SEQB_OVR	Mirrors the global OVERRUN status flag in the SEQB_GDAT register	0
27:26	-	Reserved.	-

Table 1041.ADC Flags register (FLAGS, offset 0x68) bit description

Bit	Symbol	Description	Reset value
28	SEQA_INT	Sequence A interrupt/DMA trigger. If the MODE bit in the SEQA_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQA_GDAT), which is set at the end of every ADC conversion performed as part of sequence A. It will be cleared automatically when the SEQA_GDAT register is read. If the MODE bit in the SEQA_CTRL register is 1, this flag will be set upon completion of an entire A sequence. In this case it must be cleared by writing a 1 to this SEQA_INT bit. This interrupt must be enabled in the INTEN register.	0
29	SEQB_INT	Sequence A interrupt/DMA trigger. If the MODE bit in the SEQB_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQB_GDAT), which is set at the end of every ADC conversion performed as part of sequence B. It will be cleared automatically when the SEQB_GDAT register is read. If the MODE bit in the SEQB_CTRL register is 1, this flag will be set upon completion of an entire B sequence. In this case it must be cleared by writing a 1 to this SEQB_INT bit. This interrupt must be enabled in the INTEN register.	0
30	THCMP_INT	Threshold Comparison Interrupt. This bit will be set if any of the THCMP flags in the lower bits of this register are set to 1 (due to an enabled out-of-range or threshold-crossing event on any channel). Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt. This bit will be cleared when all of the individual threshold flags are cleared via writing 1s to those bits.	0
31	OVR_INT	Overrun Interrupt flag. Any overrun bit in any of the individual channel data registers will cause this interrupt. In addition, if the MODE bit in either of the SEQn_CTRL registers is 0 then the OVERRUN bit in the corresponding SEQn_GDAT register will also cause this interrupt. This interrupt must be enabled in the INTEN register. This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.	0

44.6.12 ADC Startup register

This register is used exclusively when enabling the ADC, and typically only by the ADC API. This register should never be accessed during normal ADC operation. The ADC clock should be selected and running at full frequency prior to writing to this register.

Table 1042.ADC Startup register (STARTUP, offset 0x6C) bit description

Bit	Symbol	Description	Reset value
0	ADC_ENA	ADC Enable bit. This bit can only be set to a 1 by software. It is cleared automatically whenever the ADC is powered down. This bit must not be set until at least 10 microseconds after the ADC is powered up (typically by altering a system-level ADC power control bit).	0
1	ADC_INIT	ADC Initialization. After enabling the ADC (setting the ADC_ENA bit), the API routine will EITHER set this bit or the CALIB bit in the CALIB register, depending on whether or not calibration is required. Setting this bit will launch the “dummy” conversion cycle that is required if a calibration is not performed. It will also reload the stored calibration value from a previous calibration unless the BYPASSCAL bit is set. This bit should only be set AFTER the ADC_ENA bit is set and after the CALIREQD bit is tested to determine whether a calibration or an ADC dummy conversion cycle is required. It should not be set during the same write that sets the ADC_ENA bit. This bit can only be set to a 1 by software. It is cleared automatically when the “dummy” conversion cycle completes.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	0

44.6.13 ADC Calibration register

This register is used to perform ADC offset calibration. It will be used by the ADC API. It may also be written-to subsequently by user software to initiate re-calibration. **The maximum ADC clock frequency during calibration is 30 MHz**. If the operating ADC frequency exceeds this, a slower clock should be selected for calibration (eg. increasing the synchronous divided clock value). See Section 44.7.6.

Table 1043.ADC Calibration register (CALIB, offset 0x70) bit description

Bit	Symbol	Description	Reset value
0	CALIB	Calibration request. Setting this bit will launch an ADC calibration cycle. This bit can only be set to a 1 by software. It is cleared automatically when the calibration cycle completes.	0
1	CALREQD	Calibration required. This read-only bit indicates if calibration is required when enabling the ADC. CALREQD will be 1 if no calibration has been run since the chip was powered-up and if the BYPASSCAL bit in the CTRL register is low. The ADC API will test this bit to determine whether to initiate a calibration cycle or whether to set the ADC_INIT bit (in the STARTUP register) to launch the ADC initialization process which includes a “dummy” conversion cycle. Note: A “dummy” conversion cycle requires approximately 6 ADC clocks as opposed to 81 clocks required for calibration.	1
8:2	CALVALUE	Calibration Value. This read-only field displays the calibration value established during last calibration cycle. This value is not typically of any use to the user.	0
31:9	-	Reserved. Read value is undefined, only zero should be written.	0

44.7 Functional description

44.7.1 Conversion Sequences

A conversion sequence is a single pass through a series of ADC conversions performed on a selected set of ADC channels. Software can configure up to two independent conversion sequences, either of which can be triggered by software or by a transition on one of the hardware triggers. Each sequence can be triggered by a different hardware trigger. One of these conversion sequences is referred to as the A sequence and the other as the B sequence.

An optional single-step mode allows advancing through the channels of a sequence one at a time on each successive occurrence of a trigger.

The user can select whether a trigger on the B sequence can interrupt an already in-progress A sequence. The B sequence, however, can never be interrupted by an A trigger.

44.7.2 Hardware-triggered conversion

There are 11 hardware trigger sources available. Software can select among hardware triggers will launch each conversion sequence and it can specify the active edge for the selected trigger independently for each conversion sequence. Hardware triggering can also be independently disabled for each sequence, if not required.

For each conversion sequence, if a designated trigger event occurs, one single cycle through that conversion sequence will be launched unless:

- The BURST bit in the SEQn_CTRL register for this sequence is set to 1.
- The requested conversion sequence is already in progress.
- A set of conversions for the alternate conversion sequence is already in progress except in the case of a B trigger interrupting an A sequence if the A sequence is set to LOWPRIO.

If any of these conditions is true, the new trigger event will be ignored and will have no effect.

In addition, if the single-step bit for a sequence is set, each new trigger will cause a single conversion to be performed on the next channel in the sequence rather that launching a pass through the entire sequence.

If the A sequence is enabled to be interrupted (i.e. the LOWPRIO bit in the SEQA_CTRL register is set) and a B trigger occurs while an A sequence is in progress, then the following will occur:

- The ADC conversion which is currently in-progress will be aborted.
- The A sequence will be paused, and the B sequence will immediately commence.
- The interrupted A sequence will resume after the B sequence completes, beginning with the conversion that was aborted when the interruption occurred. The channel for that conversion will be re-sampled.

44.7.2.1 Avoiding spurious hardware triggers

Care should be taken to avoid generating a spurious trigger when writing to the SEQn_CTRL register to change the trigger selected for the sequence, switch the polarity of the selected trigger, or to enable the sequence for operation.

In general, the TRIGGER and TRIGPOL bits in the SEQn_ENA bit is should only be written when the sequence is disabled (while the SEQn_ENA bit = 0). The SEQn_ENA bit itself should only be set when the selected trigger input is in its INACTIVE state (as designated by the TRIGPOL bit). If this condition is not met, a trigger will be generated immediately upon enabling the sequence - even though no actual transition has occurred on the trigger input.

44.7.3 Software-triggered conversion

There are two ways that software can trigger a conversion sequence:

1. Start Bit: Setting the START bit in the corresponding SEQn_CTRL register. The response to this is identical to occurrence of a hardware trigger on that sequence. Specifically, one cycle of conversions through that conversion sequence will be immediately triggered except as indicated above.
2. Burst Mode: Set the BURST bit in the SEQn_CTRL register. As long as this bit is 1 the designated conversion sequence will be continuously and repetitively cycled through. Any new software or hardware trigger on this sequence will be ignored.

If a bursting A sequence is allowed to be interrupted (i.e. the LOWPRIO bit in its SEQA_CTRL register is set to 1) and a software or hardware trigger for the B sequence occurs, then the burst will be immediately interrupted and a B sequence will be initiated. The interrupted A sequence will resume continuous cycling, starting with the aborted conversion, after the alternate sequence has completed.

44.7.4 Interrupts

There are four interrupts that can be generated by the ADC:

- Conversion-Complete or Sequence-Complete interrupt for sequence A
- Conversion-Complete or Sequence-Complete interrupt for sequence B
- Threshold-Compare Out-of-Range Interrupt
- Data Overrun Interrupt

Any of these interrupt requests may be individually enabled or disabled in the INTEN register. Note that the threshold and overrun interrupts share a slot in the NVIC.

44.7.4.1 Conversion-Complete or Sequence-Complete interrupts

For each of the two sequences, an interrupt/DMA trigger can either be asserted at the end of each ADC conversion performed as part of that sequence or when the entire sequence of conversions is completed. The MODE bits in the SEQn_CTRL registers select between these alternative behaviors.

If the MODE bit for a sequence is 0 (conversion-complete mode), then the interrupt flag/DMA request for that sequence will reflect the state of the DATAVALID bit in the global data register (SEQn_GDAT) for that sequence. In this case, reading the SEQn_GDAT register will automatically clear the interrupt/DMA trigger.

If the MODE bit for the sequence is 1 (sequence-complete mode) then the interrupt flag/DMA request must be written-to by software to clear it (except when used as a DMA trigger, in which case it will be cleared in hardware by the DMA engine).

44.7.4.2 Threshold-Compare Out-of-Range Interrupt

Every conversion performed on any channel is automatically compared against a designated set of low and high threshold levels specified in the THRn_HIGH and THRn_LOW registers. The results of this comparison on any individual channel(s) can be enabled to cause a threshold-compare interrupt if that result was above or below the range specified by the two thresholds or, alternatively, if the result represented a crossing of the low threshold in either direction.

This flag must be cleared by a software write to clear the individual THCMP flags in the FLAGS register.

44.7.4.3 Data Overrun Interrupt

This interrupt/DMA trigger will be asserted if any of the OVERRUN bits in the individual channel data registers are set. In addition, the OVERRUN bits in the two sequence global data (SEQn_GDAT) registers will cause this interrupt/DMA trigger IF the MODE bit for that sequence is set to 0 (conversion-complete mode).

This flag will be cleared when the OVERRUN bit that caused it is cleared via reading the register containing it.

Note that the OVERRUN bits in the individual data registers are cleared when data related to that channel is read from either of the global data registers as well as when the individual data registers themselves are read.

44.7.5 Optional Operating Modes

There are three optional modes of ADC operation which may be selected in the CTRL register.

Four alternative ADC accuracy settings are available ranging from 12 bits down to 6 bits of resolution. Lowering the ADC resolution results in faster conversion times. A single ADC conversion (including one conversion in a burst or sequence) requires (resolution+3) ADC clocks when the minimum sampling period is selected. When reduced accuracy is selected, the unused LSBs of result data will automatically be forced to zero.

Two clocking modes are available, synchronous mode and asynchronous mode. The synchronous clocking mode uses the system clock in conjunction with an internal. programmable divider. The main advantage of this mode is determinism. The start of ADC sampling is always a fixed number of system clocks following any ADC trigger. The alternative asynchronous mode (on chips where this mode is supported) uses an independent clock source. In this mode the user has greater flexibility in selecting the ADC clock frequency to better achieve the maximum ADC conversion rate without restricting the clock rate for other peripherals. The penalty for using this mode may be longer latency and greater uncertainty in response to a hardware trigger.

44.7.6 Offset calibration and enabling the ADC

The A/D converter includes a built-in, self-calibration mode which can be used to minimize offset error. For applications where offset error is not a concern, calibration may be disabled by setting the BYPASSCAL bit in the CTRL register. If this bit is not set, a calibration cycle must be performed following chip power-up (including exit from deep-sleep or deep power-down mode) prior to using the ADC. The provided ADC API will automatically perform this required calibration.

Additional calibrations may be performed at any time by setting the CALIB bit in the CALIB register. Re-calibration is recommended if the temperature or voltage operating conditions have changed (including if the chip has been in a low-power mode for a considerable period of time). Re-calibration should also be performed if the ADC clock rate is changed.

A calibration cycle requires approximately 81 ADC clocks to complete. Normal ADC conversions cannot be launched, and the ADC Control register must not be written while calibration is in progress.

Remark: Enabling the ADC (following chip power-up, exit from any low-power mode, or when the ADC has been manually disabled) requires a specific start-up procedure. It is strongly recommended that the provided “Enable-ADC” API routine be used to enable or re-enable the ADC. The API routine will perform the necessary steps - including executing a calibration cycle if required.

Important: The ADC clock must be running at its full operating frequency prior to calling the API routine to enable the ADC. This means that the desired clocking mode and clock divide value (for sync mode) must be programmed into the CTRL register. If offset calibration is not desired, the BYPASSCAL bit in the CTRL register should also be set prior to calling the API routine to avoid wasting time on an unnecessary calibration cycle.

The ADC cannot be utilized until the startup routine has completed.

44.7.7 ADC vs. digital receiver

The analog ADC input must be selected via IOCON registers in order to get accurate voltage readings on the monitored pin. In the IOCON, the pull-up and pull-down resistors should be both disabled using the MODE bits. For a pin hosting an ADC input, it is not possible to have a have the digital function enabled and yet get valid ADC readings. Software must write a 0 to the DIGIMODE bit in the related IOCON register.

44.7.8 DMA control

The sequence A or sequence B conversion sequence complete interrupts may also be used to generate a DMA transfer trigger. To generate a DMA transfer the same conditions must be met as the conditions for generating an interrupt (see Section 44.7.4 and Section 44.6.10).

Remark: If DMA is used for a sequence, the corresponding sequence interrupt must be disabled in the INTEN register.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register. If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted. Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists.

44.7.9 ADC hardware trigger inputs

An analog-to-digital conversion can be initiated by a hardware trigger. The trigger can be selected independently for each of the two conversion sequences in the ADC SEQA_CTRL or SEQB_CTRL registers by programming the hardware trigger input # into the TRIGGER bits.

Related registers:

- Table 1030 “ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description”
- Table 1031 “ADC Conversion Sequence B Control register (SEQB_CTRL, offset 0x0C) bit description”

Table 1044.ADC0 hardware trigger inputs

Input #	Source	Description
0	None	Hardware Triggering Disabled
1	ADC0_PINTRIG0	Group GPIO input interrupt (GINT0/1)
2	ADC0_PINTRIG1	Group GPIO input interrupt (GINT0/1)
3	SCT0 output 4	Internal trigger from SCTimer/PWM
4	SCT0 output 5	Internal trigger from SCTimer/PWM
5	SCT0 output 9	Internal trigger from SCTimer/PWM
6	CTIMER0_MAT3	Internal trigger from Timer0
7	CTIMER1_MAT3	Internal trigger from Timer1
8	CTIMER2_MAT3	Internal trigger from Timer2
9	CTIMER3_MAT3	Internal trigger from Timer3
10	CTIMER4_MAT3	Internal trigger from Timer4
11	ARM_TXEV	Transmit Event output from CPU

44.7.10 Sample time

The analog input from the selected channel is sampled at the start of each new A/D conversion. The default (and shortest) duration of this sample period is 2.5 ADC clock cycles. Under some conditions, longer sample times may be required. A variety of factors including operating conditions, the ADC clock frequency, the selected ADC resolution, and the impedance of the analog source will influence the required sample period.

ADC channels 6 to 11 are somewhat slower than channels 0 to 5. Lower analog-source impedances may be required for these slow channels for a given sample period and set of operating conditions.

The following table provides guidelines for the required sample times. The “TSAMP” values displayed in the tables refer to the TSAMP field in the ADCTRL register. This value represents the number of additional clock cycles that must be added to the minimum 2.5-clock sample period in order to meet or exceed the required minimum sample time for

the maximum ADC clock rate of 80 MHz under worst-case operating conditions. At slower clock frequencies fewer sample clocks will be needed to achieve the required sample times.

Table 1045.Minimum required sample times

Selected ADC Resolution	Analog signal source impedance	Fast channels (ADC5:0)		Slow channels (ADC11:6)	
		Min. sample time	TSAMP field	Min. sample time	TSAMP field
12 bits	under 0.05k ohms	20 ns	0	43 ns	1
	0.05 to 0.1k ohms	23 ns	0	46 ns	1
	0.1K to 0.2k ohms	26 ns	0	50 ns	2
	0.2k to 0.5k ohms	31 ns	0	56 ns	2
	0.5k to 1.0k ohms	47 ns	1	74 ns	3
	1k to 5k ohms	75 ns	3	105 ns	6
10 bits	under 0.05k ohms	15 ns	0	35 ns	1
	0.05 to 0.1k ohms	18 ns	0	38 ns	1
	0.1K to 0.2k ohms	20 ns	0	40 ns	1
	0.2k to 0.5k ohms	24 ns	0	46 ns	1
	0.5k to 1.0k ohms	38 ns	1	61 ns	2
	1k to 5k ohms	62 ns	2	86 ns	4
8 bits	under 0.05k ohms	12 ns	0	27 ns	0
	0.05 to 0.1k ohms	13 ns	0	29 ns	0
	0.1K to 0.2k ohms	15 ns	0	32 ns	0
	0.2k to 0.5k ohms	19 ns	0	36 ns	1
	0.5k to 1.0k ohms	30 ns	0	48 ns	1
	1k to 5k ohms	48 ns	1	69 ns	3
6 bits	under 0.05k ohms	9 ns	0	20 ns	0
	0.05 to 0.1k ohms	10 ns	0	22 ns	0
	0.1K to 0.2k ohms	11 ns	0	23 ns	0
	0.2k to 0.5k ohms	13 ns	0	26 ns	0
	0.5k to 1.0k ohms	22 ns	0	36 ns	1
	1k to 5k ohms	36 ns	1	51 ns	2

44.8 Examples

44.8.1 Perform a single ADC conversion triggered by software

Remark: When ADC conversions are triggered by software only and hardware triggers are not used in the conversion sequence, follow these steps to avoid spurious conversions:

1. Before changing the trigger set-up, disable the conversion sequence by setting the SEQ_ENA bit to 0 in the SEQA_CTRL register.
2. Set the trigger source to an unused setting using the TRIGGER bits in the SEQA_CTRL register.
3. Set the TRIGPOL bit to 1 in the in the SEQA_CTRL register.

Once the sequence is enabled again, the ADC converts a sample whenever the START bit is written to.

The ADC converts an analog input signal VIN on the ADC0_[11:0] pins. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x VIN)/(VREFP - VREFN). The result of an input voltage below VREFN is 0, and the result of an input voltage above VREFP is 4095 (0xFFF).

To perform a single ADC conversion for ADC0 channel 1 using the analog signal on pin ADC0_1, follow these steps:

1. Enable the analog function on pin ADC0_1 via IOCON See Table 255 and Table 258.
2. Configure the system clock to be 50 MHz and select a CLKDIV value of 0 for a sampling rate of 50 MHz using the CTRL register. See Section 44.6.1 “ADC Control register”.
3. Select the synchronous mode in the CTRL register.
4. Select ADC channel 1 to perform the conversion by setting the CHANNELS bits to 0x2 in the SEQA_CTL register.
5. Set the TRIGPOL bit to 1 and the SEQA_ENA bit to 1 in the SEQA_CTRL register.
6. Set the START bit to 1 in the SEQA_CTRL register.
7. Read the RESULT bits in the DAT1 register for the conversion result.

44.8.2 Perform a sequence of conversions triggered by an external pin

The ADC can perform conversions on a sequence of selected channels. Each individual conversion of the sequence (single-step) or the entire sequence can be triggered by hardware. Hardware triggers are either a signal from an external pin or an internal signal. See Section 44.7.9.

To perform a single-step conversion on the first four channels of ADC0 triggered by rising edges on pin PIO1_0, follow these steps:

1. Enable the analog function on pin ADC0_0 to ADC0_3 via IOCON. See Table 255 and Table 258.
2. Configure PINT1 to respond to PIO1_0, see Chapter 12 for details.
3. Configure the system clock to be 80 MHz and select a CLKDIV value of 0 for a sampling rate of 80 MHz using the ADC CTRL register.
4. Select the synchronous mode in the CTRL register.
5. Select ADC channels 0 to 3 to perform the conversion by setting the CHANNELS bits to 0xF in the SEQA_CTRL register.
6. Select trigger PINT1 by writing 0x1 the TRIGGER bits in the SEQA_CTRL register.
7. To generate one interrupt at the end of the entire sequence, set the MODE bit to 1 in the SEQA_CTRL register.
8. Select single-step mode by setting the SINGLESTEP bit in the SEQA_CTRL register to 1.
9. Enable the Sequence A by setting the SEQA_ENA bit.

   A conversion on ADC0 channel 0 will be triggered whenever the pin PIO1_0 goes from LOW to HIGH. The conversion on the next channel (initially channel 1) is triggered on the next 0 to 1 transition of PINT1. The ADC0 interrupt is generated when the sequence has finished after four 0 to 1 transitions of PINT1.
10. Read the RESULT bits in the DAT0 to DAT3 registers for the conversion result.

45.1 How to read this chapter

The temperature sensor is available on all LPC546xx devices.

45.2 Features

- Linear temperature sensor.
- Sensor output internally connected to the ADC for temperature monitoring.

45.3 Basic configuration

- Enable the power to the temperature sensor by setting the TS_PD bit in the PDRUNCFG register. See Section 7.5.84.
- To monitor the temperature continually, select the temperature sensor as source for channel 0 of ADC0. See Chapter 44. The digital temperature reading is available after an analog-to-digital conversion.

Remark: To convert the ADC conversion result into a temperature reading, see the device data sheet for information on temperature calibration of the sensor.

45.3.1 Perform a single ADC conversion with the temperature sensor as ADC input

To perform a single ADC conversion for ADC0 channel 0 using the temperature sensor output:

1. Enable the temperature sensor output as input to ADC channel 0.
2. Configure the system clock and the ADC for operation.
3. Select the synchronous mode in the ADC CTRL register.
4. Select ADC channel 0 to perform the conversion by setting the CHANNELS bits to 0x1 in the SEQA_CTL register.
5. Set the START bit to 1 in the SEQA_CTRL register.
6. Read the RESULT bits in the DAT0 register for the conversion result.

45.4 Pin description

The temperature sensor has no configurable pins.

45.5 Register description

The temperature sensor has no configurable registers.

46.1 How to read this chapter

This chapter applies to all LPC546xx devices.

46.2 Features

- The OTP memory stores the following information:
  - User programmable ECRP settings.
  - User application specific data.
- Boot ROM API support for programming the OTP memory provided.

46.3 Basic configuration

- The clock to the OTP controller can be enabled using the OTP bit in AHBCLKCTRL2 register.
- The OTP controller is reset using the OTP_RST bit in PRESETCTRL2 register.

46.4 General description

The OTP memory contains a memory bank of 128 bits each. The OTP Bank contains 4 words: word 0 for ECRP, word 1 is reserved; programming will cause permanent part disable condition, words 2 and 3 can be used by user application for storing application specific options.

The virgin OTP state is all 0s. A 0 value can be programmed to 1, but once programmed, a 1 in any of the OTP bits is permanent and can never be changed. An API is provided to program data into the OTP banks.

46.5 OTP memory description

Table 1046.OTP content

OTP bank	Content	Software access	API
3	Word 0: Customer control data	Yes	otpProgramReg
3	Reserved	-	-
3	Words 2 to 3: General purpose data	Yes	otpProgramReg

Table 1047.OTP memory description (OTP base address 0x4001 5000)

OTP bank	Word	Access	Address offset	Size	Description	Reference
3	0	User programmable; initial state = 0	0x030	32 bit	Customer control data. See Table 1048.	-
3	1	-	0x034	32 bit	Reserved	-
3	2	User programmable; initial state = 0	0x038	32 bit	General purpose OTP memory, word 2	-
3	3	User programmable; initial state = 0	0x03C	32 bit	General purpose OTP memory, word 3	-

The word 0 register is used for storing the ECRP settings.

Table 1048.OTP memory bank 3, word 0 - Customer control data (address offset 0x030)

Bit	Symbol	Value	Description
2:0	-	-	Reserved
3	SWD_DISABLE_L		Bits 3 and 10 control the ability to enable and disable SWD. See Table 1049 “SWD feature”.
4	CRP_MASS_ERASE_DISABLE		This bit controls the ability to process mass erase commands in ISP mode and the debug mailbox.
		0	Enable
		1	Disable
5	IAP_PROTECTION_ENABLE		This bit controls the ability to enable checking for ECRP in IAP functions.
		0	Disable
		1	Enable
6	CRP_ISP_DISABLE_PIN		This bit controls the ability to enter ISP mode using the ISP pin.
		0	Enable
		1	Disable
7	CRP_ISP_DISABLE_IAP		This bit controls the ability to re-invoke ISP using IAP routines.
		0	Enable
		1	Disable
8	-	-	Reserved
9	CRP_ALLOW_ZERO		This bit determines how the flash ECRP policy is interpreted when the “ECRP” field present in user image is set to zero.
		0	All protections enabled by ECRP field will be set to lowest protection level. So, flash ECRP value is interpreted as SWD, ISP entry, IAP entry, write and erase of all flash sectors are allowed.
		1	All protections enabled by ECRP field will be set to highest protection level. So SWD, ISP entry, IAP entry, write and erase of all flash sectors are restricted.
		Remark: ROM enforces the protection policy based on combination of OTP ECRP bits and flash ECRP value.	
10	SWD_DISABLE_H		Bits 10 and 3 control the ability to enable and disable SWD. See Table 1049 “SWD feature”.

Table 1049.SWD feature

SWD_DISABLE_H (Bit 10)	SWD_DISABLE_L (Bit 3)	SWD Feature
0	0	Enable
0	1	Enable
1	0	Disable
1	1	Disable

46.6 OTP API

The OTP memory is controlled through a set of simple API calls located in the LPC546xx ROM.

The API calls to the ROM are performed by executing functions which are pointed to by pointer within the ROM driver table.

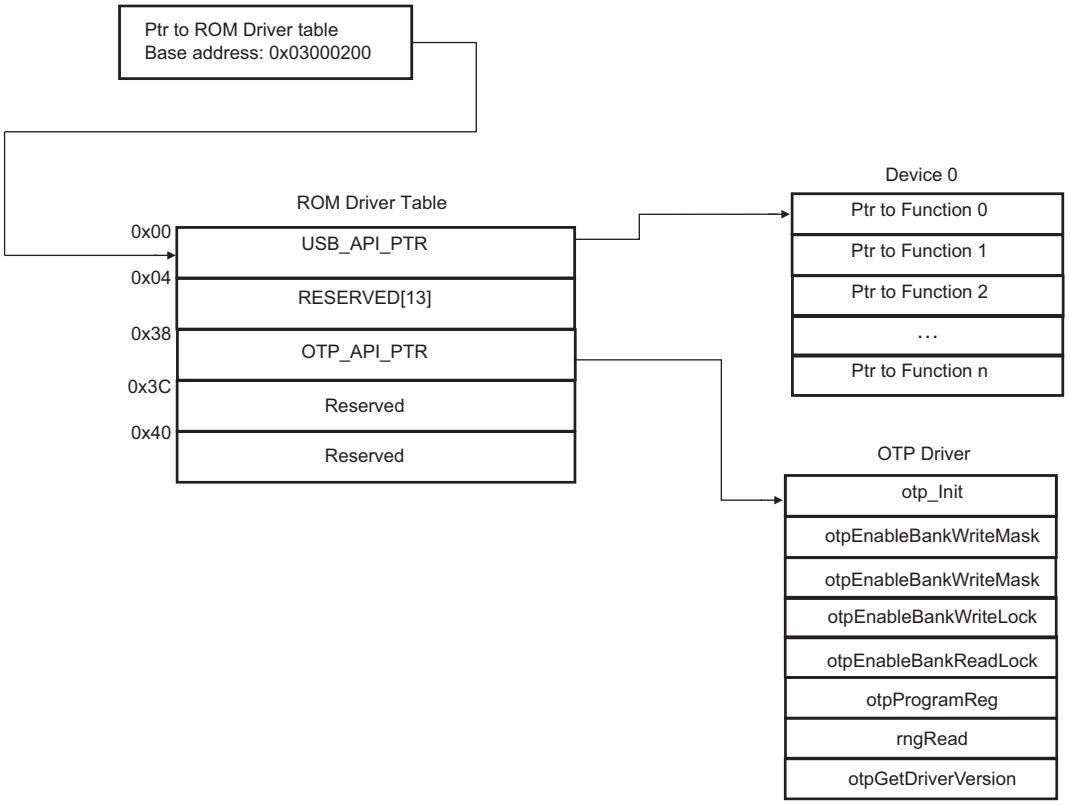


Fig 179. OTP driver pointer structure

Table 1050.ROM driver pointers (main API entry point 0x0300 0200)

API	Offset	Description
USB_API_PTR	0x00	Pointer to the USB API driver base.
RESERVED[13]	0x04	-
OTP_API_PTR	0x38	Pointer to the OTP API driver base.

46.6.1 OTP function allocation

To use the OTP API, the main system clock must be running from the 12 MHz clock. The OTP controller has critical timing values programmed so that running from clock speeds above 12 MHz will result in inconsistent fuse burning and will yield unpredictable results. The OTP programming range is 2.7 V ≤ V_{DD} ≤ 3.6 V and attempting to program below 2.7 V will result in unpredictable results and the part might enter an unrecoverable state.

See Section 46.5 “OTP memory description” for the parameters to use in the OTP functions in Table 1051.

Table 1051.OTP function allocation

Function	Offset	Description	Availability of API
			LPC546xx
otp_Init	0x00	Initializes OTP controller. Parameter - void Return - unsigned: see the general error codes.	Yes
otpEnableBankWriteMask	0x04	Unlock one or more OTP banks for write access. Parameter – unsigned Mask field that specifies which banks to lock. Return - unsigned: see the general error codes.	Yes
otpDisableBankWriteMask	0x08	Lock one or more OTP banks for write access. Parameter – unsigned Mask field that specifies which banks to unlock. bit 0 => bank0 bit 1 => bank1 bit 2 => bank2 bit 3 => bank3 bit 4:31 => Reserved Return- unsigned: see the general error codes.	Yes

Table 1051.OTP function allocation *...continued*

Function	Offset	Description	Availability of API
			LPC546xx
otpEnableBankWriteLock	0x0C	Locks or unlocks write access to a register of an OTP bank and the write lock. Parameter - unsigned OTP bank index: 0 => bank-0 1 => bank-1 2 => bank-2 3 => bank-3 Parameter - unsigned write enable mask: BIT 0 => When set enables write of WORD0 BIT 1 => When set enables write of WORD1 BIT 2 => When set enables write of WORD2 BIT 3 => When set enables write of WORD3 BIT 4:31 => Reserved Parameter - unsigned write disable mask: BIT 0 => When set disables write of WORD0 BIT 1 => When set disables write of WORD1 BIT 2 => When set disables write of WORD2 BIT 3 => When set disables write of WORD3 BIT 4:31 => Reserved Parameter - unsigned read/write lock: Non-zero value disables subsequent access modifications until Reset. Return- unsigned: see the general error codes.	Yes

Table 1051.OTP function allocation *...continued*

Function	Offset	Description	Availability of API
			LPC546xx
otpEnableBankReadLock	0x10	Locks or unlocks read access to a register of an OTP bank and the write lock. Parameter - unsigned OTP bank index: 0 => bank-0 1 => bank-1 2 => bank-2 3 => bank-3 Parameter - unsigned read enable mask: BIT 0 => When set enables read of WORD0 BIT 1 => When set enables read of WORD1 BIT 2 => When set enables read of WORD2 BIT 3 => When set enables read of WORD3 BIT 4:31 => Reserved Parameter - unsigned read disable mask: BIT 0 => When set disables read of WORD0 BIT 1 => When set disables read of WORD1 BIT 2 => When set disables read of WORD2 BIT 3 => When set disables read of WORD3 BIT 4:31 => Reserved Parameter - unsigned read/write lock 0. Access set cannot be modified till Reset. Parameter - unsigned read/write lock 1. Access set cannot be modified till Reset. Return- unsigned: see the general error codes.	Yes
otpProgramReg	0x14	Program a single register in an OTP bank. Parameter – unsigned OTP bank number Parameter – unsigned OTP register number Parameter – value to write Return - unsigned: see the general error codes **Remark:** otpEnableBankWriteLock API must be called to enable the appropriate register before calling otpProgramReg API function.	Yes
-	0x18	Reserved	-
-	0x1C	Reserved	-
-	0x20	Reserved	-
-	0x24	Reserved	-

Table 1051.OTP function allocation *...continued*

Function	Offset	Description	Availability of API
			LPC546xx
-	0x28	Reserved	-
rngRead	0x2C	Returns 32-bit number from hardware random number generator Parameter – void Return - unsigned random number	Yes
otpGetDriverVersion	0x30	Returns the version of the OTP driver in ROM Parameter – void Return - unsigned: driver version	Yes

Remark: The OTP has 4 banks (BANK0 to BANK3) and each bank has 4 32-bit registers (REG0 to REG3). For the mask parameters (enable or disable) in Table 1051, Bit0 corresponds to REG0, Bit1 corresponds to REG1 and so on. Since each bank has only 4 registers, Bit4 to Bit31 of the mask value are reserved. If a register is specified in both enable and disable masks then the disable mask takes priority.

Example 1: “otpEnableBankReadLock(0x3, 0x3, 0xC, 0x00)” will enable read access to BANK3-REG0, REG1 and disables access to BANK3-REG2, REG3. These access values can be changed by a subsequent call to “otpEnableBankReadLock”.

Example 2: “otpEnableBankReadLock (0x3, 0x6, 0x9, 0x01)” will enable read access to BANK3-REG1, REG2 and disable read access to BANK3-REG3, REG0 and since the write lock mask is also set, these permissions will be set and cannot be changed until reset.

Note: The disable mask takes priority if both enable and disable masks are set.

46.6.1.1 OTP API error codes

```
ERR_OTP_BASE = 0x00070000,
/*0x00070001*/ ERR_OTP_WR_ENABLE_INVALID = ERR_OTP_BASE + 1
/*0x00070002*/ ERR_OTP_SOME_BITS_ALREADY_PROGRAMMED
/*0x00070003*/ ERR_OTP_ALL_DATA_OR_MASK_ZERO
/*0x00070004*/ ERR_OTP_WRITE_ACCESS_LOCKED
/*0x00070005*/ ERR_OTP_READ_DATA_MISMATCH
/*0x00070006*/ ERR_OTP_USB_ID_ENABLED
/*0x00070007*/ ERR_OTP_ETH_MAC_ENABLED
/*0x00070009*/ ERR_OTP_ILLEGAL_BANK
/*0x0007000A*/ ERR_OTP_SHUFFLER_CONFIG_NOT_VALID
/*0x0007000B*/ ERR_OTP_SHUFFLER_NOT_ENABLED
/*0x0007000B*/ ERR_OTP_SHUFFLER_CAN_ONLY_PROG_SINGLE_KEY
/*0x0007000C*/ ERR_OTP_ILLEGAL_PROGRAM_DATA
/*0x0007000D*/ ERR_OTP_READ_ACCESS_LOCKED
/*Returned on success*/ LPC_OK = 0
```

47.1 How to read this chapter

The EEPROM is available on all LPC546xx devices.

47.2 Features

- 16,384 Byte EEPROM.
- Access via memory on the AHB bus.
- Less than 3 ms erase / program time.
- Endurance of > 100 k erase / program cycles.

47.3 Basic configuration

The EEPROM is configured using the following registers:

1. Power: The EEPROM is enabled after a device reset, but may be turned off if it is not needed. To save power, see Section 7.5.85 “Power Configuration register 1”.
2. Clock: Clock for EEPROM must be set up before it can be used.
3. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. See Section 6.4.2 “Interrupt Set-Enable Register 1”.

47.4 General description

The EEPROM can be read and written/erased. A write operation involves two steps. The first step is writing a minimum of 1 word (4 bytes) to a maximum of 32 words (128 bytes) to the desired page in the 16 kB EEPROM address space at address 0x4010 8000. Step two is an erase/program of that page into non-volatile memory. There are 128 pages within the 16 kB EEPROM address space. The last page contains the EEPROM initialization data and is not writable. Note that data written to a page cannot be read back from the page address until the data have been programmed into non-volatile memory.

47.5 Register description

Table 1052.Register overview: EEPROM (base address 0x4001 4000)

Name	Access	Offset	Description	Reset value[1]	Section
CMD	R/W	0x000	EEPROM command register	0	47.5.1.1
RWSTATE	R/W	0x008	EEPROM read wait state register	0x0000 0E07	47.5.1.2
AUTOPROG	R/W	0x00C	EEPROM auto programming register	0	47.5.1.3
WSTATE	R/W	0x010	EEPROM wait state register	0x0004 0802	47.5.1.4
CLKDIV	R/W	0x014	EEPROM clock divider register	0x0000 0063	47.5.1.5
PWRDWN	R/W	0x018	EEPROM power-down register	0	47.5.1.6
INTENCLR	W	0xFD8	EEPROM interrupt enable clear	0	47.5.2.1
INTENSET	W	0xFDC	EEPROM interrupt enable set	0	47.5.2.2
INTSTAT	R	0xFE0	EEPROM interrupt status	0	47.5.2.3
INTEN	R	0xFE4	EEPROM interrupt enable	0	47.5.2.4
INTSTATCLR	W	0xFE8	EEPROM interrupt status clear	0	47.5.2.5
INTSTATSET	W	0xFEC	EEPROM interrupt status set	0	47.5.2.6

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

47.5.1 EEPROM control registers

47.5.1.1 EEPROM command register

The EEPROM command register is used to trigger an erase/program operation on the EEPROM device.

Table 1053.EEPROM command register (CMD, address 0x4001 4000) bit description

Bits	Symbol	Description	Reset value
2:0	CMD	Command. Read data shows the last command executed on the EEPROM. 110 = erase/program page All other values are reserved.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.1.2 EEPROM read wait state register

The EEPROM has no awareness of absolute time, while for EEPROM operations several minimum absolute timing constraints have to be met. Therefore the EEPROM can only derive time from its clock by frequency division.

The EEPROM read wait state register is used to define wait states for the AHB read operation in functional mode only.

Program the wait state fields to appropriate values in this wait state register for EEPROM operation. The fields are -1 encoded, so programming zero will result in a one cycle wait state.

The register contains two fields, each representing a minimum duration of a phase of the EEPROM read. The appropriate values for each field are determined as follows:

(waitstates +1) x 12 MHz ≥ duration

Table 1054.EEPROM read wait state register (RWSTATE, address 0x4001 4008) bit description

Bits	Symbol	Description	Reset value
7:0	RPHASE2	Wait states 2 (minus 1 encoded). The number of system clock periods to meet the read operations TRPHASE2 duration.	0x07
15:8	RPHASE1	Wait states 1 (minus 1 encoded). The number of system clock periods to meet a duration equal to TRPHASE1.	0x0E
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.1.3 EEPROM auto programming register

The auto programming register allows the user to let the controller start an erase/program cycle automatically after AHB writes without the need to program the CMD register after the page register has been written.

Table 1055.EEPROM auto programming register (AUTOPROG, address 0x4001 400C) bit description

Bits	Symbol	Description	Reset value
1:0	AUTOPROG	Auto programming mode: 00 = auto programming off 01 = erase/program cycle is triggered after 1 word is written 10 = erase/program cycle is triggered after a write to AHB address ending with .....1111100 (last word of a page)	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.1.4 EEPROM write wait state register

The EEPROM has no awareness of absolute time, while for EEPROM operations several minimum absolute timing constraints have to be met. Therefore the EEPROM can only derive time from its clock by frequency division.

Program the wait state fields to appropriate values in this wait state register for EEPROM operation. The fields are -1 encoded so programming zero will result in a one cycle wait state.

The fields in the WAITSTATE register represent a minimum duration of a phase of the EEPROM operation. The appropriate wait state values for each fields are determined as follows:

(waitstates +1) x 12 MHz ≥ duration

The delays for the write and erase/program operations are combined to simplify the software interface. Timing for write and erase/program operations is almost identical.

Table 1056.EEPROM write wait state register (WSTATE, address 0x4001 4010) bit description

Bits	Symbol	Description	Reset value
7:0	PHASE3	Wait states for phase 3 (minus 1 encoded). The number of system clock periods to meet a duration equal to TPHASE3.	0x2
15:8	PHASE2	Wait states for phase 2 (minus 1 encoded). The number of system clock periods to meet a duration equal to TPHASE2.	0x8
23:16	PHASE1	Wait states for phase 1 (minus 1 encoded). The number of system clock periods to meet a duration equal to TPHASE1.	0x4
30:24	-	Reserved. Read value is undefined, only zero should be written.	NA
31	LCK_PARWEP	Lock timing parameters for write, erase and program operation 0 = WSTATE and CLKDIV registers have R/W access 1 = WSTATE and CLKDIV registers have R only access	0

47.5.1.5 EEPROM clock divider register

The EEPROM device requires a 1500 kHz clock. The nominal value of the frequency is 1500 kHz, the lower limit is 800 kHz, the maximum limit is 1600 kHz.

This clock is generated by dividing the system bus clock. The clock divider register contains the division factor.

If the division factor is 0, the clock is be IDLE to save power.

$$\frac{cclk}{CLKDIV + 1} \approx 1500kHz$$

Table 1057.EEPROM clock divider register (CLKDIV, address 0x4001 4014) bit description

Bits	Symbol	Description	Reset value
15:0	CLKDIV	Division factor (minus 1 encoded).	-
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.1.6 EEPROM power down register

Use the EEPROM power down register to put the EEPROM device in power down mode.

Do not put the EEPROM in power-down mode during a pending EEPROM operation. After clearing this bit any EEPROM operation has to be suspended for 100 μs while the EEPROM wakes up.

Table 1058.EEPROM power down/DCM register (PWRDWN, address 0x4001 4018) bit description

Bits	Symbol	Description	Reset value
0	PWRDWN	Power down mode bit. 0 = not in power down mode. 1 = power down mode.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.2 Interrupt registers

These registers control interrupts from the EEPROM. An interrupt occurs at the end of a program operation.

47.5.2.1 Interrupt enable clear register

Table 1059.Interrupt enable clear register (INTENCLR, address 0x4001 4FD8) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	PROG_CLR_EN	Clear program operation finished interrupt enable bit for EEPROM. 0 = leave corresponding bit unchanged. 1 = clear corresponding bit.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.2.2 Interrupt enable set register

Table 1060.Interrupt enable set register (INTENSET, address 0x4001 4FDC) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	PROG_SET_EN	Set program operation finished interrupt enable bit for EEPROM device 1. 0 = leave corresponding bit unchanged. 1 = set corresponding bit.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.2.3 Interrupt status register

Table 1061.Interrupt status register (INTSTAT, address 0x4001 4FE0) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. The value read from a reserved bit is not defined.	NA
2	END_OF_PROG	EEPROM program operation finished interrupt status bit. Bit is: set when an EEPROM write operation has finished OR when one is written to the corresponding bit of the INTSTATSET register. cleared when one is written to the corresponding bit of the INTSTATCLR register.	0
31:3	-	Reserved. The value read from a reserved bit is not defined.	NA

47.5.2.4 Interrupt enable register

Table 1062.Interrupt enable register (INTEN, address 0x4001 4FE4) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. The value read from a reserved bit is not defined.	NA
2	EE_PROG_DONE	EEPROM program operation finished interrupt enable bit. Bit is: set when one is written in the corresponding bit of the INTENSET register. cleared when one is written to the corresponding bit of the INTENCLR register.	0
31:3	-	Reserved. The value read from a reserved bit is not defined.	NA

47.5.2.5 Interrupt status clear register

Table 1063.Interrupt status clear register (INTSTATCLR, address 0x4001 4FE8) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	PROG_CLR_ST	Clear program operation finished interrupt status bit for EEPROM device. 0 = leave corresponding bit unchanged. 1 = clear corresponding bit.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

47.5.2.6 Interrupt status set

Table 1064.Interrupt status set register (INTSTATSET, address 0x4001 4FEC) bit description

Bits	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	PROG_SET_ST	Set program operation finished interrupt status bit for EEPROM device. 0 = leave corresponding bit unchanged. 1 = set corresponding bit.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

47.6 Functional description

47.6.1 Initialization

At power-up, the reset should be applied for at least 100us. However a normal reset (not at power-up) period is only 40 ns.

When reset is de-asserted, the EEPROM controller performs an initialization sequence, to set the EEPROM trim values. It first reads the trim values from the first word (containing trimming information) of the last (protected) page of the EEPROM. The controller, then, uses these values to set the EEPROM device trim settings.

During the EEPROM reset and initialization any started AHB/APB transfer will stall the bus.

47.6.2 EEPROM operations

An EEPROM device cannot be programmed directly. Writing data to it and the actual erase/program of the memory are two separate steps.

1. Write 1 word (4 bytes) to 32 words (128 bytes) to the desired page in the 16 kB EEPROM address space. There are 128 pages in the 16 kB EEPROM address space. Page 1 begins at the first address of the EEPROM address space (EEPROM_START = 0x4010 8000). Page 2 begins at EEPROM_START + 128, page 3 begins at EEPROM_START + 256, etc. Writes to a page cannot cross a 128 page boundary.

   Remark: Before reading this data from the EEPROM or writing to another page, program the contents of the page register into the EEPROM.
2. Program the data into the EEPROM with the erase/program command issued via the EEPROM CMD register on the APB.

47.6.2.1 Writing and erase/programming

Writing an EEPROM page is achieved by AHB writes.

Partial page writes are allowed, and the EEPROM will only take locations where a word has been written for the erase/program cycle. Any word in a page can be written in any order, but every word in the page may only be written once.

The address of the AHB transfer prior to the erase/program cycle will determine the page address for the erase/program cycle.

The following modes are supported for the write and erase/programming operation:

- AUTOPROG = 00: When the page has been written (fully or partially), the data has to be programmed into non-volatile memory. The erase/program cycle is triggered by writing 0x6 to the CMD register. The page that is programmed is determined by the address of the last AHB transfer. Therefore it is advised to perform a page write with the page address and to prevent AHB reads between page register writes and the erase/program trigger.
- AUTOPROG = 01: Writing a single word to the page starts the erase/program cycle automatically. This mode is useful store small data items (word) on random locations.

- AUTOPROG = 10: Writing to the last word of the page register (AHB address ending with 1111100) automatically starts the erase/program cycle. This mode is useful to store multiple full pages of data.

During programming, the EEPROM is not available for other operations. To prevent undesired loss in performance which would be caused by stalling the bus, the EEPROM instead generates an error for AHB read/writes and APB writes when programming is busy. In order to prevent the error response, the program operation finished interrupt can be enabled or the interrupt status bit can be polled.

48.1 How to read this chapter

The CRC engine is available on all LPC546xx devices.

48.2 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1's complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Supports CPU PIO back-to-back transfer.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

48.3 Basic configuration

Set the CRC bit in the AHBCLKCTRL0 register (Table 139) to enable the clock to the CRC engine.

48.4 Pin description

The CRC engine has no configurable pins.

48.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

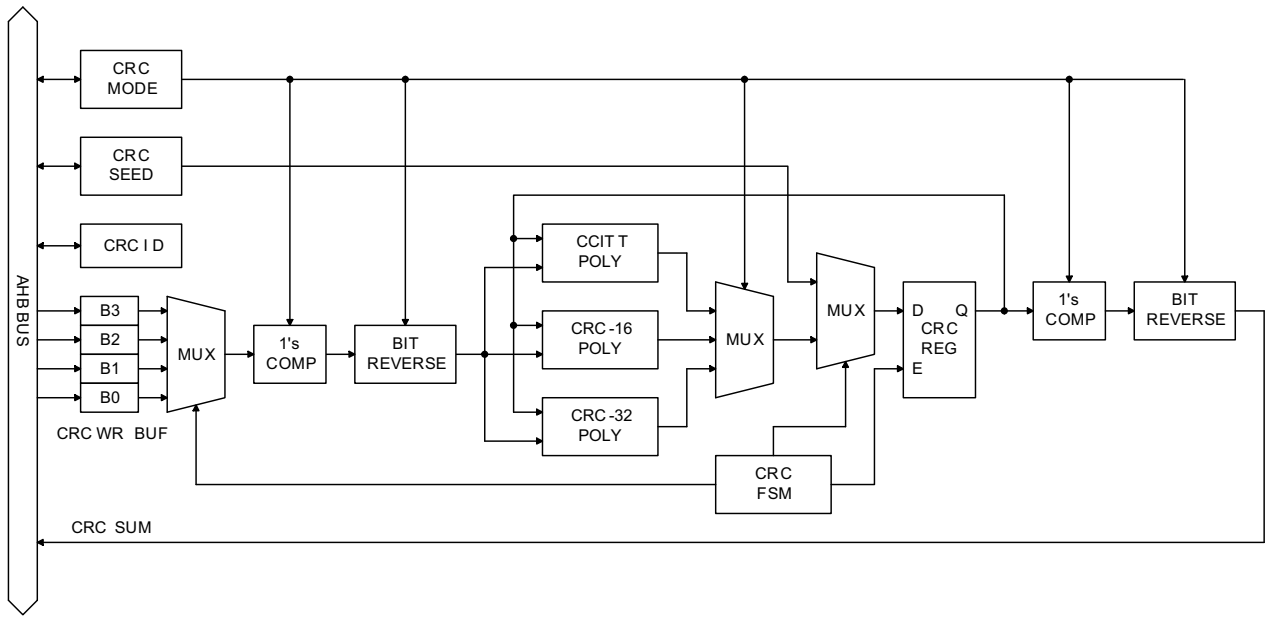


Fig 180. CRC block diagram

48.6 Register description

Table 1065.Register overview: CRC engine (base address 0x4009 5000)

Name	Access	Offset	Description	Reset value	Section
MODE	R/W	0x000	CRC mode register	0x0000 0000	48.6.1
SEED	R/W	0x004	CRC seed register	0x0000 FFFF	48.6.2
SUM	RO	0x008	CRC checksum register	0x0000 FFFF	48.6.3
WR_DATA	WO	0x008	CRC data register	-	48.6.4

48.6.1 CRC mode register

Table 1066.CRC mode register (MODE, offset 0x000) bit description

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynomial: 1X = CRC-32 polynomial 01 = CRC-16 polynomial 00 = CRC-CCITT polynomial	00
2	BIT_RVS_WR	Data bit order: 1 = Bit order reverse for CRC_WR_DATA (per byte) 0 = No bit order reverse for CRC_WR_DATA (per byte)	0
3	CMPL_WR	Data complement: 1 = 1's complement for CRC_WR_DATA 0 = No 1's complement for CRC_WR_DATA	0
4	BIT_RVS_SUM	CRC sum bit order: 1 = Bit order reverse for CRC_SUM 0 = No bit order reverse for CRC_SUM	0
5	CMPL_SUM	CRC sum complement: 1 = 1's complement for CRC_SUM 0 = No 1's complement for CRC_SUM	0
31:6	Reserved	Always 0 when read	0x0000000

48.6.2 CRC seed register

Table 1067.CRC seed register (SEED, offset 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load CRC seed value to CRC_SUM register with selected bit order and 1's complement pre-processes. **Remark:** A write access to this register will overrule the CRC calculation in progresses.	0x0000 FFFF

48.6.3 CRC checksum register

This register is a Read-only register containing the most recent checksum. The read request to this register is automatically delayed by a finite number of wait states until the results are valid and the checksum computation is complete.

Table 1068.CRC checksum register (SUM, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SUM	The most recent CRC sum can be read through this register with selected bit order and 1’s complement post-processes.	0x0000 FFFF

48.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 1069.CRC data register (WR_DATA, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be taken to perform CRC calculation with selected bit order and 1’s complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.	-

48.7 Functional description

The following sections describe the register settings for each supported CRC standard:

48.7.1 CRC-CCITT set-up

Polynomial = $x^{16} + x^{12} + x^5 + 1$

Seed Value = 0xFFFF

Bit order reverse for data input: NO

1's complement for data input: NO

Bit order reverse for CRC sum: NO

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0000

CRC_SEED = 0x0000 FFFF

48.7.2 CRC-16 set-up

Polynomial = $x^{16} + x^{15} + x^2 + 1$

Seed Value = 0x0000

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0015

CRC_SEED = 0x0000 0000

48.7.3 CRC-32 set-up

Polynomial = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Seed Value = 0xFFFF FFFF

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: YES

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

49.1 How to read this chapter

Debug functionality is available on all LPC546xx devices.

49.2 Features

- Supports ARM Serial Wire Debug mode for the Cortex-M4.
- Trace port provides Cortex-M4 CPU instruction trace capability. Output via a Serial Wire Viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: the Cortex-M4 includes 6 instruction breakpoints that can also be used to remap instruction addresses for code patches. Two literal comparators that can also be used to remap addresses for patches to literal values.
- Watchpoints: the Cortex-M4 includes 4 data watchpoints that can also be used as triggers.
- Supports JTAG boundary scan.
- Instrumentation Trace Macrocell allows additional software controlled trace for the Cortex-M4.

49.3 Basic configuration

The serial wire debug pins are enabled by default. The JTAG pins for boundary scan are selected by hardware after a reset.

49.4 Pin description

The tables below indicate the various pin functions related to debug. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Table 1070.Serial Wire Debug pin description

Function	Type	Connect to	Description
SWCLK	In	PIO0_11	Serial Wire **Clock.** This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). This pin is pulled up internally.
SWDIO	I/O	PIO0_12	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the part. This pin is pulled up internally.
SWO	Out	PIO0_10 or PIO0_8	**Serial Wire Output.** The SWO pin optionally provides data from the ITM for an external debug tool to evaluate.

Table 1071.Parallel trace pin description

Function	Type	Connect to	Description
TRACECLK	In	P1_0, P3_12, P5_5	**ClockTrace.** This pin provides the sample clock for trace data on the TRACEDATA pins when tracing is enabled by an external debug tool.
TRACEDATA[0]	Out	P0_31, P3_10, P5_6	**TRACEDATA[3:0].** Output Trace Data bits 3 to 0.These pins provide ETM trace data when tracing is enabled by an external debug tool. The debug tool can then interpret the compressed information and make it available to the user.
TRACEDATA[1]	Out	P0_30, P3_13, P5_7	
TRACEDATA[2]	Out	P0_29, P3_14, P5_8	
TRACEDATA[3]	Out	P0_28, P3_11, P5_9	

The JTAG boundary pin functions are selected by hardware at reset. See Section 49.6.3.

The following setup is required to enable SWO output on GPIO PIO0-15 (FUNC2) or PIO1_1 (FUNC2):

1. Write 0x00000001 to TRACECLKDIV (0x40000304). Enables the Trace divider.
2. If the clock to the IOCON block is not already enabled, write 0x00002000 to AHBCLKCTRLSET[0] (0x40000220). The clock must be enabled in order to access any IOCON registers.

Table 1072.JTAG boundary scan pin description

Function	Type	Connect to	Description
TCK	In	P0_3	**JTAG Test Clock.** This pin is the clock for JTAG boundary scan when the RESET pin is LOW.
TMS	In	P0_4	JTAG **Test Mode Select.** The TMS pin selects the next state in the TAP state machine. This pin includes an internal pull-up and is used for JTAG boundary scan when the RESET pin is LOW.
TDI	In	P0_5	JTAG **Test Data In.** This is the serial data input for the shift register. This pin includes an internal pull-up and is used for JTAG boundary scan when the RESET pin is LOW.
TDO	OuT	P0_6	JTAG **Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. This pin is used for JTAG boundary scan when the RESET pin is LOW.
TRST	In	P0_2	JTAG **Test Reset.** The TRST pin can be used to reset the test logic within the debug logic. This pin includes an internal pull-up and is used for JTAG boundary scan when the RESET pin is LOW.

49.5 General description

Serial wire debug functions are integrated into each CPU, with up to four breakpoints and two watchpoints. Boundary scan is also available.

Trace on the Cortex-M4 is supported via the Serial Wire Output.

49.6 Functional description

49.6.1 Debug limitations

Important: Due to limitations of the CPU, the part cannot wake up in the usual manner from deep-sleep mode during debugging.

The debug mode changes the way in which reduced power modes work internal to the CPU. Therefore power measurements should not be made while debugging, power consumption is higher than during normal operation.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

49.6.2 Debug connections for SWD

For debugging purposes, it is useful to provide access to the ISP entry pins (see Chapter 3 “LPC546xx Boot process”). The ISP entry pins can be used to recover the part from configurations which would disable the SWD port such as improper PLL configuration, assigning another function to the SWD pins via IOCON, entry into deep power-down mode out of reset, etc. The ISP entry pins can be used for other functions such as GPIO but should not be held LOW on power-up or reset.

Internal and external SWD connections are shown in Figure 181 and Figure 182.

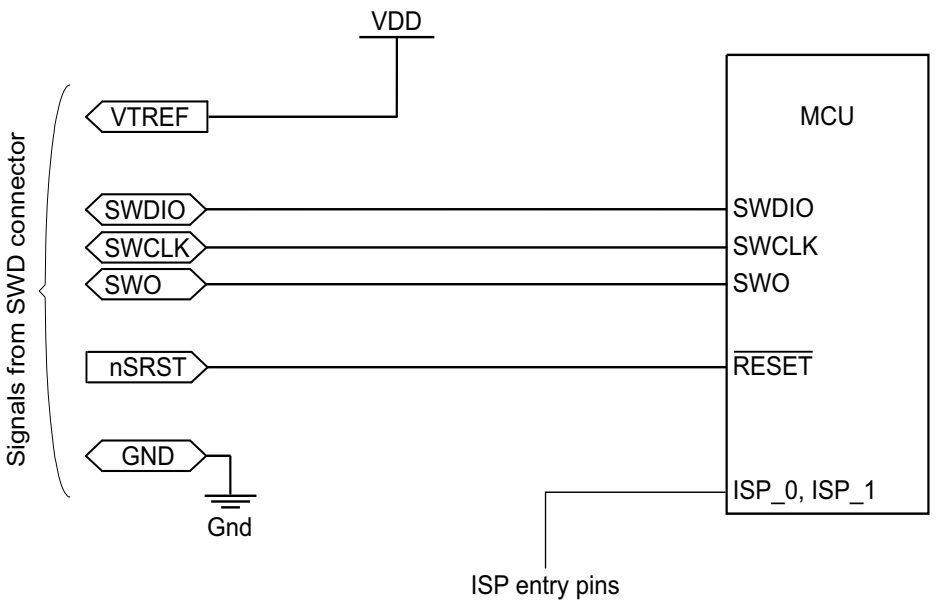


Fig 181. Connecting the SWD pins to a standard SWD connector

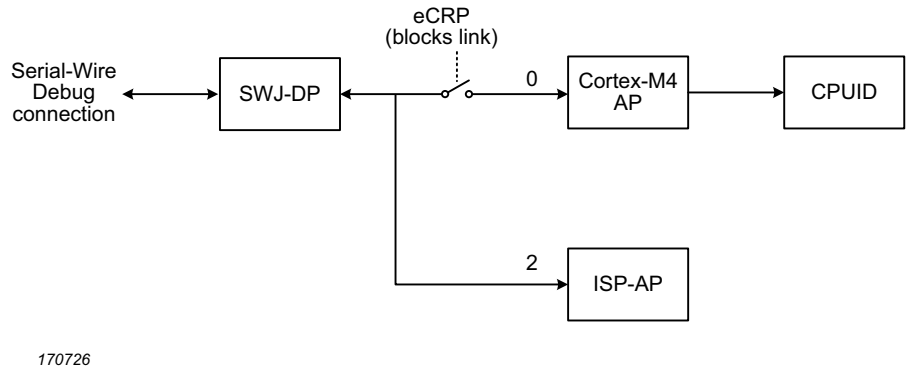


Fig 182. Serial Wire Debug internal connections

49.6.3 Boundary scan

The $\overline{\text{RESET}}$ pin selects between the test TAP controller for JTAG boundary scan ($\overline{\text{RESET}}$ = LOW) and the ARM SWD debug port TAP controller ($\overline{\text{RESET}}$ = HIGH). The ARM SWD debug port is disabled while the part is in reset. A LOW on the $\overline{\text{TRST}}$ pin resets the test TAP controller.

Remark: Boundary scan operations should not be started until 250 μs after POR. The test TAP must be reset after the boundary scan and left in either TLR or RTO state. Boundary scan is not affected by Code Read Protection.

Remark: POR, BOD reset, or a LOW on the $\overline{\text{TRST}}$ pin puts the test TAP controller in the Test-Logic Reset state. The first TCK clock while $\overline{\text{RESET}}$ = HIGH places the test TAP in Run-Test Idle mode.

49.6.4 In-System Programming Access Port (ISP-AP)

The ISP-AP is essentially a register-based communication port that may be accessed by both the CPU and the device debug port.

This port is used to implement certain commands that can operate even when the device has been programmed to the highest Code Read Protection level (ECRP). The ISP-AP is active whenever the device is attached to a debugger.

49.6.4.1 Resynchronization request

Communication with the ISP-AP is initiated by the debugger. The debugger first sets the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW, or by driving the actual reset pin of the device if it is able to do so.

49.6.4.2 Acknowledgement of resynchronization request

After requesting a resynchronization and resetting the device, the debugger reads the CSW register. This stalls the debugger if the device has not yet completed the resynchronization process. The debugger can repeat this process until it is able to read the CSW and find a 0 there.

49.6.4.3 Return phase

Following the initial resynchronization, communication by the debugger to the device is in the form of 32-bit writes to the REQUEST register. The debugger can read the result in the RETURN register. The debugger polls the RETURN register in the same manner as it polled the CSW following a resynchronization request.

49.6.4.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW. Once such an error occurs, the debugger will need to start with a new resynchronization request in order to clear the error flag.

49.6.4.5 Register description

The registers in the ISP-AP are show below. These registers are readable by the CPU and are intended primarily to allow on-chip ROM routines to implement requests from an external debugger.

Table 1073.Register overview: ISP-AP (base address 0x4009 C000)

Name	Access	Offset	Description	Reset value	Section
CSW	R/W	0x000	Command and status word.	0	49.6.4.5.1
REQUEST	R/W	0x004	Request from the debugger to the device.	0	49.6.4.5.2
RETURN	R/W	0x008	Return value from the device to the debugger.	0	49.6.4.5.3
ID	RO	0x0FC	Identification register.	0x002A 0000	49.6.4.5.4

49.6.4.5.1 Command and Status Word register

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Table 1074.Command and Status Word register (CSW, offset 0x000) bit description

Bit	Symbol	Description	Reset value
0	RESYNCH_REQ	The debugger sets this bit to requests a re-synchronization.	0
1	REQ_PENDING	A request is pending for the debugger: a value is waiting to be read from the REQUEST register.	0
2	DBG_OR_ERR	When 1, a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before it was read by the device.	0
3	AHB_OR_ERR	When 1, an AHB overrun has occurred: a RETURN value has been overwritten by the device before it was read by the debugger.	0
4	SOFT_RESET	This bit is write-only by the device and resets the ISP-AP.	0
5	CHIP_RESET_REQ	This write -only bit causes the device (but not the ISP-AP) to be reset.	0
31:6	-	Reserved	-

49.6.4.5.2 Request value register

The REQUEST register is used by a debugger to send action requests to the device.

Table 1075.Request value register (REQUEST, offset 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	REQUEST	Request value. Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication.	0

49.6.4.5.3 Return value register

The RETURN register provides any response from the device to the debugger.

Table 1076.Return value register (RETURN, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	RETURN	Return value. This is any response from the device to the debugger. If no new data is present, a debugger read will be stalled until new data is available.	0

49.6.4.5.4 Identification register

The ID register provides an identification of the ISP-AP interface.

Table 1077.Identification register (ID, offset 0x0FC) bit description

Bit	Symbol	Description	Reset value
31:0	ID	Identification value.	0x002A 0000

49.6.4.6 ISP-AP commands

Commands for the ISP-AP are listed below.

Table 1078.ISP-AP commands

Name	Command code	Description
Enter ISP-AP	1	Cause the device to enter ISP-AP command mode. This must be done prior to sending other commands.
Bulk Erase	2	Erase the entire on-chip flash memory.
Query ECRP Level	3	Queries the Code Read Protection (ECRP) level currently in force on the device.
Exit ISP_AP	4	Cause the device to exit ISP-AP command mode. The Device returns to normal mode.

49.6.4.7 ISP-AP return codes

Return codes for ISP-AP commands are listed below.

Table 1079.Register overview: ISP-AP return codes

Return code	Description
0x0000 0000	Command succeeded. Applies to commands other than “Query ECRP level”.
0x0010 0001	Debug mode not entered. This is returned if other commands are sent prior to the “Enter ISP-AP” command.
0x0010 0002	Command not recognized. A command was received other than the ones defined above.
0x0010 0003	Command failed.

49.7 Debug configuration

49.7.1 Cortex-M4

- Six instruction breakpoints that can also be used to remap instruction addresses for code patches. Two data comparators that can be used to remap addresses for patches to literal values.
- Four data Watchpoints.
- Instrumentation Trace Macrocell allows additional software controlled trace capability.

50.1 How to read this chapter

The Secure Hash Algorithm (SHA) block is available only in the LPC54628 device.

50.2 Features

- Performs SHA-1 and SHA-2(256) based hashing.
- Used with HMAC to support a challenge/response or to validate a message.

50.3 Basic configuration

- The priority for the SHA block can be set in PRI_SHA bit in AHBMATPRIO register. See Table 121.
- The SHA block can be reset using the SHA_RST bit in PRESETCTRL2 register. See Table 131.
- The clock to the SHA block can be controlled using the SHA bit in AHBCLKCTRL2 register. See Table 141.

50.4 General description

The secure devices provide on-chip hash support to perform SHA-1 and SHA-2 with 256-bit digest (SHA-256). Hashing is a way to reduce arbitrarily large messages and code images to a relatively small fixed size “unique” number called a digest. The SHA-1 Hash produces a 160 bit digest (5 words), and the SHA-256 hash produces a 256 bit digest (8 words).The digest has two important aspects:

The SHA hardware consists of these components:

- Even a small change to the input message will cause a major change in the digest output. Therefore, for a given input message/image there is only one digest.
- There is no predictable way to modify one input to result in a specific digest. a message cannot be added, inserted, or modified to get the same hash in any direct way.

These two properties make it useful for verifying a message is valid, whether corrupted intentionally or unintentionally.

Hashing processes blocks of 512 bits (16 words) at a time and performs the SHA-1 hashing in 80 clock cycles per block or SHA-256 hashing in 64 clock cycles per block. As many blocks as needed may be processed. The last block must be formatted per the SHA model:

1. The last data must be 447 bits or less. If more, then an extra block must be created.
2. After the last bit of data, a 1 bit is appended.
3. Then, as many 0 bits are appended to take it to 448 bits long (so, 0 or more).

4. Finally, the last 64 bits contain the length of the whole message, in bits, formatted as a word.

For example, if a message is an exact multiple of 512 bits, an extra block needs to be created. The first bit of the last block will be a 1 followed by 447 zeroes. The remaining 64 bits will contain the length of the whole message including the last block.

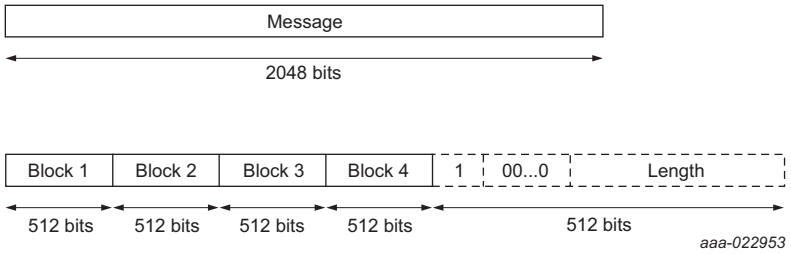


Fig 183. HASH data block

The ARM processor uses little-endian and therefore, the Hash block reverses the bytes in the words written to the data register to big-endian format. This is because a Hash is on bytes, so a string such as “abcd”, when read as a word by the processor (or DMA) is reversed into “dcba”. When the input data is provided in little-endian format, the Hash block swaps them to process correctly.

50.5 Register description

Table 1080.Register overview: SHA (base address 0x400A 4000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register	0x0	50.5.1
STATUS	R/W1C	0x004	Status register	0x0	50.5.2
INTENSET	R/W	0x008	Interrupt Enable register	0x0	50.5.3
INTENCLR	W1C	0x00C	Interrupt Clear register	-	50.5.4
MEMCTRL	R/W	0x010	Memory Control register	0x0	50.5.5
MEMADDR	R/W	0x014	Memory Address register	0x0	50.5.6
INDATA	WO	0x020	Input Data register	0x0	50.5.7
ALIAS0	WO	0x024	Alias0 register	-	50.5.7
ALIAS1	WO	0x28	Alias1 register	-	50.5.7
ALIAS2	WO	0x2C	Alias2 register	-	50.5.7
ALIAS3	WO	0x30	Alias3 register	-	50.5.7
ALIAS4	WO	0x34	Alias4 register	-	50.5.7
ALIAS5	WO	0x38	Alias5 register	-	50.5.7
ALIAS6	WO	0x3C	Alias6 register	-	50.5.7
DIGEST0	RO	0x040	Digest 0 register	0x0	50.5.8
DIGEST1	RO	0x44	Digest register	0x0	50.5.8
DIGEST2	RO	0x48	Digest 2 register	0x0	50.5.8
DIGEST3	RO	0x4C	Digest 3 register	0x0	50.5.8
DIGEST4	RO	0x50	Digest 4 register	0x0	50.5.8

Table 1080.Register overview: SHA (base address 0x400A 4000)

Name	Access	Offset	Description	Reset value	Section
DIGEST5	RO	0x54	Digest 5 register	0x0	50.5.8
DIGEST6	RO	0x58	Digest 6 register	0x0	50.5.8
DIGEST7	RO	0x5C	Digest 7 register	0x0	50.5.8

50.5.1 Control register

The control register is used to configure the SHA block. The SHA block is enabled when the MODE bit is selected to SHA-1 or SHA-256. The NEW bit field is written to 1, before the data can be loaded into INDATA (or its aliases, or both) register.

Table 1081.Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MODE		This field is used to select the operational mode of SHA block	0x0
		0x0	Disabled	
		0x1	SHA-1 is enabled	
		0x2	SHA-256 is enabled	
		0x3	Reserved	
3:2	-	-	Reserved	-
4	NEW		When this bit is set, a new hash operation is started. It automatically self-clears in one clock cycle. **Remark:** The WAITING bit in Status register gets cleared for one cycle during initialization.	0x0
7:5	-	-	Reserved	-
8	DMA		When this bit is set, the DMA is used to fill INDATA. The SHA block requests for 16 words from the DMA and then processes the Hash after which it sends the request again. The processor is interrupted when DMA length expires.	0x0
		0	DMA disabled	
		1	DMA enabled	
31:9	-	-	Reserved	-

50.5.2 Status register

The Status register shows when the SHA block is waiting for data and when the results are available (which may be partial). These bits correspond to both interrupts and DMA (in the case of data).

Table 1082.Status register (STATUS, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
0	WAITING		This field indicates if the block is waiting for more data to process.	0x0
		0	Not waiting for data	
		1	Waiting for data	

Table 1082.Status register (STATUS, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
1	DIGEST		This field indicates if a DIGEST is ready and waiting and there is no active next block that has already started. This bit is cleared when any data is written to INDATA or ALIAS registers, when NEW bit is written, or when the SHA block is disabled.	0x0
		0	Digest is not ready.	
		1	Digest is ready. Application may read it or may write more data.	
2	ERROR		This field indicates if an error has occurred. This error could be because of an attempted overrun (INDATA was written when it was not appropriate) or an AHB bus error. The COUNT bit in MEMCTRL register is used to indicate the block where the AHB bus error has occurred.	-
		0	No error.	
		1	An error occurred.	
31:3	-	-	Reserved	-

50.5.3 Interrupt Enable Register

The Interrupt Enable register is used to enable interrupt sources that cause processor interrupts.

Table 1083.Interrupt Enable register (INTENSET, offset 0x00B) bit description

Bit	Symbol	Value	Description	Reset value
0	WAITING		This field indicates if interrupt should be enabled when waiting for input data. The interrupt is cleared when any data is written to INDATA or ALIAS registers.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
1	DIGEST		This field indicates if interrupt is generated when Digest is ready (completed a Hash or completed a full sequence). The interrupt is cleared when any data is written to INDATA or ALIAS registers, when NEW bit is written, or when the SHA block is disabled.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
2	ERROR		This field indicates if interrupt is generated on an ERROR (as defined in STAT register)	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
31:3	-	-	Reserved	

50.5.4 Interrupt Clear register

The Interrupt Clear register is used to clear the interrupt mask enabled by the INTENSET register.

Table 1084.Interrupt Clear register (INTENCLR, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
0	WAITING	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
1	DIGEST	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
2	ERROR	Writing a 1 clears the interrupt enabled by the INTENSET register.	0
31:3	-	Reserved	-

50.5.5 Memory Control register

The Memory Control Register (MEMCTRL) allows setting up the SHA block to be the AHB bus master to read memory for hashing. It can be used to read 512-bit blocks from Internal Flash, SRAM0, SRAMX, and SPIFI. The starting location must be word aligned and the length may be up to 128 KB.

Table 1085.Memory Control register (MEMCTRL, offset 0x010) bit description

Bit	Symbol	Value	Description	Reset value
0	MASTER		This field is used to enable SHA block as AHB bus master.	0
		0	SHA block is not AHB bus master and the DMA or Interrupt based model is used with INDATA.	
		1	Enables SHA block as AHB bus master. DMA and INDATA should not be used.	
26:16	COUNT		This field indicates the number of 512-bit blocks to copy starting at MEMADDR. This register will decrement after each block is copied, ending in 0. The DIGEST interrupt will occur when it reaches 0. If a bus error occurs, it will stop with this field set to indicate the block that failed.	0
		0	Done. Nothing to process.	
		0x1	One 512-bit block to hash.	
		0x2	Two 512-bit blocks to hash.	
		0x3	Three 512-bit blocks to hash. The maximum number of 512-bit blocks that can be processed is 2047 blocks.	
31:27	-	-	Reserved	

50.5.6 Memory Address register

The Memory Address Register (MEMADDR) holds the base address for MEMCTRL. It must only point to valid locations in Internal Flash, SRAM0, SRAMX, or SPIFI based on the LPC54S6xx device used and must be word aligned.

Table 1086.Memory Address register (MEMADDR, offset 0x014) bit description

Bit	Symbol	Description	Reset value
31:0	BASEADDR	Address base to start copying from, word aligned (so bits 1:0 must be zero). This field will advance as it processes the words. If it fails with a bus error, the register will contain the failing word.	0

50.5.7 Input Data and ALIAS registers

The INDATA and its ALIAS registers are used for writing the 16 words per hash. The aliases exist so the processor can use STM (store multiple). The DMA would only write to INDATA.

Table 1087.Input Data register (INDATA, offset 0x020) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	0

Table 1088.Alias 0 register (ALIAS0, offset 0x024) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

Table 1089.Alias 1 register (ALIAS1, offset 0x028) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

Table 1090.Alias 2 register (ALIAS2, offset 0x02C) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

Table 1091.Alias 3 register (ALIAS3, offset 0x030) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

Table 1092.Alias 4 register (ALIAS4, offset 0x034) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

Table 1093.Alias 5 register (ALIAS5, offset 0x038) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

Table 1094.Alias 6register (ALIAS6, offset 0x03C) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	In this field the next word is written in little-endian format. The SHA block swaps the word to the required big-endian format.	-

50.5.8 DIGEST register

The DIGEST registers contain the 160 bits or 256 bits digest, depending on SHA-1 or SHA-256 respectively. The SHA hardware writes to the DIGEST registers in word format, therefore, endianness should be considered when sending or comparing. The first 5 DIGEST registers are populated for SHA-1and all 8 DIGEST registers are populated for SHA-256. If SHA-1 is used, DIGEST [0:4] are populated and if SHA-256 is used, DIGEST [0:7] are populated.

Table 1095.DIGEST 0 register (DIGEST0, offset 0x040) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1096.DIGEST 1 register (DIGEST1, offset 0x044) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1097.DIGEST 2 register (DIGEST2, offset 0x048) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1098.DIGEST 3 register (DIGEST3, offset 0x04C) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1099.DIGEST 4 register (DIGEST4, offset 0x050) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1100.DIGEST 5 register (DIGEST5, offset 0x054) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1101.DIGEST 6 register (DIGEST6, offset 0x058) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

Table 1102.DIGEST 7 register (DIGEST7, offset 0x05C) bit description

Bit	Symbol	Description	Reset value
31:0	DIGEST	This field contains one word of the Digest.	0

50.6 Functional description

To perform hashing, select one of the three possible ways to get the input data into the SHA block:

- Using Cortex-M4 with interrupts:

- The WAITING and ERROR interrupts are configured in the INTENSET register.
- When status of the WAITING interrupt in STAT register is 1, the block is loaded by copying the 16 words using INDATA and ALIAS registers.
- If more blocks need to be loaded, then the WAITING interrupt bit is retained. After the last block is loaded, the DIGEST interrupt is enabled.

- Using the DMA:
  - The DMA is configured for up to 1 K words (64, 512-bit blocks) to be read from Internal Flash, SRAM0, SRAMX, or SPIFI. The SHA peripheral will control the DMA to feed its data as fast as it can. See Chapter 15 “LPC546xx DMA controller” for configuring the DMA.
  - The SHA block is double buffered and therefore, will allow loading another 16 words while processing the previous words. This pipelined method allows continuous processing of input data.
  - An interrupt is used to notify the processor when the DMA transfer is complete. The interrupt service routine can enable the DIGEST interrupt and ERROR interrupt to process the results. Or, it can configure the DMA for more data if needed.
  - If the last block is to be constructed separately, then either the DMA can move those 16 words or the processor can do so via interrupts.

- Using AHB Master (when available):
  - The SHA peripheral is enabled as AHB bus master. The memory location to read from SRAM0, SRAMX, or internal Flash and the number of blocks to read is configured using the MEMCTRL register.
  - The DIGEST and ERROR interrupts are enabled in INTENSET register. The interrupts will not occur until the last block is completed and the digest is computed.
  - If the last block is to be constructed separately, then the interrupt service routine may load the constructed last block (or use the DMA) and it will be interrupted when the DIGEST is ready.

50.6.1 Performance of SHA Block

The SHA block contains two message buffers which can be loaded by CPU, DMA or AHB bus master. The performance of the block depends on the memory from where the input data is fetched (Internal Flash, SRAM0, SRAMX or SPIFI) and activity on the system bus.

50.6.1.1 Input data loaded by CPU

The Cortex-M4 core writes 16 words to start Hashing. The block uses INDATA and ALIAS registers to support write operation to contiguous locations. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The processor can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data. If the Flash wait states are less, this mode supports continuous hashing to be performed without stall.

50.6.1.2 Input data loaded by DM

The DMA loads the 16 words based on requests. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The DMA can request and load the next 16 words during the time when the Hash operation is being performed on the previous loaded data. If 0 wait state is used, this mode support continuous hashing without stall.

50.6.1.3 Input data loaded by AHB bus master

The AHB bus master loads 16 words from internal Flash, SRAM0, SRAMX, or SPIFI. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The AHB master can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data. This mode can perform continuous hashing using SHA-256 if Flash wait states is set to 3 and using SHA-1 if Flash wait states is set to 4.

50.6.2 Initialization

To setup the SHA block:

1. Take the SHA block out of reset mode using the SHA_RST bit and enable the clock to SHA peripheral in the AHBCLKCTRL2 register. Chapter 7 “LPC546xx System configuration (SYSCON)” .

   Remark: The SHA peripheral only uses the main AHB clock, so no special clocking or scaling is needed.
2. Select SHA-1 or SHA-256 mode using the CTRL register.
3. To start a new Hash write 1 to the NEW bit field in CTRL register. This bit automatically self-clears.
4. To input data into the SHA block, when using:
   - CPU: Write to INTENSET register to enable the WAITING and ERROR interrupts.
   - DMA:
   - Configure the DMA. See Section X.Y for DMA setup.
   - Enable the DMA interrupt so the application knows when DMA transfer is done.
   - Set the DMA bit in the CTRL register.
5. AHB Master:
   - Enable the DIGEST and ERROR interrupts using INTENSET register.
   - Write to the MEMADDR register with the offset in Internal Flash, SRAMX, SRAM0, or SPIFI.
   - Write to the MEMCTRL register to enable the SHA block as AHB bus master using the MASTER bit and write the number of 512-bit blocks to process in the COUNT field.

50.6.3 Interrupt Service Routine (ISR)

50.6.3.1 ISR when using CPU

When using CPU to load data into the SHA block, the algorithm for ISR is:

- If the ERROR bit is set in STAT register, there is an issue with the application code since ERROR means overrun.
- If the WAITING bit is set in STAT register, write 16 words into the SHA peripheral. The fastest method is by using structure copy as shown below. If there are no more blocks after this, clear the WAITING interrupt using INTENCLR register and then set DIGEST using INTENSET.
- The fastest copy is usually:

```
struct HASH_W {unsigned v[8];} *src, *dst;
src = (struct HASH_W * )memory_to_read_from; //use location in InternalFlash or
      SRAM0 or SRAMX
dst = (struct HASH_W * )HASH0_INDATA; // indata and aliases
dst[0] = src[0]; // 1st 8
dst[1] = src[1]; // 2nd 8
```

- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

50.6.3.2 ISR when using DMA

When the DMA is used for loading the data into the SHA peripheral, the ISR algorithm is:

- If all the input data are loaded into the SHA block, enable the DIGEST interrupt in the INTENSET register to generate an interrupt when the DIGEST is ready.
- If the last block needs to be manually loaded, write the 16 words now, or use the CPU ISR described in Section 50.6.3.1to do the one block.
- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

  The ERROR bit is always 0 in this case because an error is not possible.

50.6.3.3 ISR for AHB Master

The ISR for AHB Master is only for DIGEST or ERROR. An ERROR would be a bus error, so the algorithm is:

- If ERROR is set in the STAT register, there is AHB master bus fault. The COUNT field in the MEMCTRL register indicates which block it was processing and the MEMADDR register indicates which memory location it was on when the error occurred.
- If the DIGEST bit is set in STAT register, the DIGEST is ready and so process the Digest register (for example, copy) and clear the interrupt using INTENCLR register.

51.1 How to read this chapter

The Random Number Generator (RNG) is available on all LPC546xx devices.

51.2 Features

- True Random Number Generator.
- Random Number Generator accessible through API call.
- FIPS 140-1/2, NIST, DieHard compliant.

51.3 Basic configuration

- The RNG block can be powered or powered down using the PDEN_RNG bit in PDRUNCFG1 register. See Table 217.
- The clock to the RNG block is controlled by the RNG bit in AHBCLKCTRL2 register. See Table 141.
- The RNG AHB bus interface can be reset using the RNG_RST bit in PRESETCTRL2 register. See Table 131.

51.4 General description

Random number generators are used for cryptographic, modeling and simulation application which employ keys that must be generated in a random fashion. The LPC546xx devices contain a True Random Number Generator.

51.5 Random Number Generator API

The Random Number Generator is accessed through an API call located in the ROM. The ROM driver table contains a pointer to the RNG driver table (Table 1103). The main API entry point for ROM driver pointers is 0x0300 0238.

Table 1103. RNG API call

Function	Offset relative to the API entry point	Description
rngRead	0x2C	Brief: Returns a 32 bit random number from hardware Random Number Generator. Parameter - void. Return- unsigned Random number.

51.6 Entropy of the generated random numbers

The quality of randomness (entropy) of the numbers generated by the Random Number Generator relies on the initial states of internal logic. If a 128 bit or 256 bit random number is required, it is not recommended to concatenate several words of 32 bits to form the number. For example, if two 128 bit words are concatenated, the hardware RNG will not provide 2 times 128 bits of entropy.

Table 1104 shows the entropy distribution that is supported.

Table 1104. Entropy distribution

Number of 128 bit words	Entropy
1	128
2	256

To constitute one 128 bit number, a 32 bit random number is read, then the next 32 numbers are read but not used. The next 32 bit number is read and used and so on. Thus 32 32-bit random numbers are skipped between two 32-bit numbers that are used.

52.1 ARM Cortex-M4 Details

ARM Limited publishes the document “Cortex™-M4 Devices Generic User Guide”, which is available on their website at:

- For the online manual, go to “infocenter.arm.com”, then search for “cortex-m4 user guide”. This will bring up links to chapters of the user guide.
- There are links at the bottom of user guide chapters to download a PDF file of the user guide.

This section of this manual describes the Cortex-M4 implementation options and other distinctions that apply for the LPC546xx devices.

52.1.1 Cortex-M4 implementation options

The Cortex-M4 CPU provides a number of implementation options. These are given below for the LPC546xx.

- The MPU is included for the Cortex-M4. The MPU provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis.
- The FPU is included for the Cortex-M4. The FPU supports single-precision floating-point computation functionality in compliance with the ANSI/IEEE Standard 754-2008. The FPU provides add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also performs a variety of conversions between fixed-point, floating-point, and integer data formats.
- 46 interrupts are implemented for the Cortex-M4. Not all interrupts are available on all part numbers.
- 3 interrupt priority bits are implemented on the Cortex-M4.
- Sleep mode power-saving: NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M4. Details of reduced power modes and wake-up possibilities can be found in Chapter 9.
- Reset of the Cortex-M4 resets the CPU register bank.
- Memory features: The memory map for LPC546xx devices is shown in Section 2.1.3.
- Bit banding is included on the Cortex-M4. APB peripherals are located in bit-band space.

In addition, there are debug and trace options, see Chapter 49.

53.1 Abbreviations

Table 1105. Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
AVB	Audio Video Bridging
BOD	BrownOut Detection
Boot	At power-up or chip reset, any method of importing code from an external source to execute from on-chip SRAM, or code executed in place from the external memory.
BSDL	Boundary-Scan Description Language
CAN FD	Controller Area Network Flexible Data Rate
CRC	Cyclic Redundancy Check
DCC	Debug Communication Channel
DMA	Direct Memory Access
eCRP	Extended Configurable Restrictions and Protections
EMC	External Memory Controller
Ethernet AVB	Ethernet Audio Video Bridging
FIFO	First-In-First-Out
FMC	Flash Memory Controller
FRO oscillator	Internal Free-Running Oscillator, tuned to the factory specified frequency.
GPIO	General Purpose Input/Output
I2C	Inter-IC Control bus
I2C or IIC	Inter-Integrated Circuit bus
IAP	In-Application Programming
I2S	Inter-IC Sound or Integrated Interchip Sound. A serial audio data communication method.
IrDA	Infrared Data Association
ISP	In-System Programming. These are methods of programming any on-chip flash on a blank or previously programmed device.
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
LIN	Local Interconnect Network
NVIC	Nested Vectored Interrupt Controller
PDM	Pulse Density Modulation. This is the data format used by the digital microphone inputs.
PLL	Phase-Locked Loop
POR	Power-On Reset
PWM	Pulse Width Modulator

Table 1105. Abbreviations *…continued*

Acronym	Description
RAM	Random Access Memory
SDIO	Secure Digital Input Output
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SWD	Serial-Wire Debug
TAP	Test Access Port
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
VAD	Voice Activity Detect

53.2 References

[1] **Cortex-M4 TRM —** ARM Cortex-M4 Processor Technical Reference Manual

[2] **AN11538 —** AN11538 application note and code bundle (SCT cookbook)

[3] **UM10204 —** I^2C-bus specification and user manual

53.3 Legal information

53.3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

53.3.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

53.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^2C-bus — logo is a trademark of NXP B.V.

53.4 Tables

53.5 Figures

continued >>

53.6 Contents

Chapter 1: LPC546xx Introductory information

Chapter 2: LPC546xx Memory map

Chapter 3: LPC546xx Boot process

Chapter 4: LPC546xx FRO API ROM routine

Chapter 5: LPC546xx ISP and IAP

Chapter 6: LPC546xx Nested Vectored Interrupt Controller (NVIC)

Chapter 7: LPC546xx System configuration (SYSCON)

Chapter 8: LPC546xx Power management

Chapter 9: LPC546xx Power profiles/Power control API

Chapter 10: LPC546xx I/O pin configuration (IOCON)

Chapter 11: LPC546xx Input multiplexing (INPUT MUX)

Chapter 12: LPC546xx Pin interrupt and pattern match (PINT)

Chapter 13: LPC546xx General Purpose I/O (GPIO)

Chapter 14: LPC546xx Group GPIO input interrupt (GINT0/1)

Chapter 15: LPC546xx DMA controller

Chapter 16: LPC546xx SCTimer/PWM

Chapter 17: LPC546xx Standard counter/timers (CTIMER0 - 4)

Chapter 18: LPC546xx Windowed Watchdog Timer (WWDT)

Chapter 19: LPC546xx Real-Time Clock (RTC)

Chapter 20: LPC546xx Multi-Rate Timer (MRT)

Chapter 21: LPC546xx Repetitive Interrupt Timer (RIT)

Chapter 22: LPC546xx CPU system tick timer (SYSTICK)

Chapter 23: LPC546xx Micro-tick Timer (UTICK)

Chapter 24: LPC546xx Flexcomm Interface serial communication

Chapter 25: LPC546xx USARTs

Chapter 26: LPC546xx Serial Peripheral Interfaces (SPI)

Chapter 27: LPC546xx I²C-bus interfaces

Chapter 28: LPC546xx I²S interface

Chapter 29: LPC546xx Audio subsystem

Chapter 30: LPC546xx SD/MMC and SDIO interface

Chapter 31: LPC546xx Smart Card Interface

Chapter 32: LPC546xx SPI flash interface (SPIFI)

Chapter 33: LPC546xx External Memory Controller (EMC)

Chapter 34: LPC546xx LCD controller

Chapter 35: LPC546xx Controller Area Network Flexible Data

Chapter 36: LPC546xx Ethernet

Chapter 37: LPC546xx USB0 Full-speed Device Controller

Chapter 38: LPC546xx USB0 Full-speed Host controller

Chapter 39: LPC546xx USB1 High-speed Device Controller

Chapter 40: LPC546xx USB1 High-speed Host Controller

Chapter 41: LPC546xx USB ROM API

Chapter 42: LPC546xx Flash signature generator

Chapter 43: LPC546xx Enhanced Code Read Protection (ECRP)

Chapter 44: LPC546xx 12-bit ADC controller (ADC)

Chapter 45: LPC546xx Temperature sensor

Chapter 46: LPC546xx One-Time Programmable (OTP) memory and API

Chapter 47: LPC546xx EEPROM memory

Chapter 48: LPC546xx CRC engine

Chapter 49: LPC546xx Serial Wire Debug (SWD)

Chapter 50: LPC546xx Secure Hash Algorithm

Chapter 51: LPC546xx Random Number Generator (RNG)

Chapter 52: ARM Cortex Appendix

Chapter 53: Supplementary information

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 9 November 2017
Document identifier: UM10912