UM10562

LPC408x/407x User manual

Rev. 3 — 12 March 2014 **User manual**

Document information

Info	Content
Keywords	ARM, ARM Cortex-M4, 32-bit, USB, Ethernet, LCD, CAN, I^2C, I^2S, Flash, EEPROM, Microcontroller
Abstract	LPC408x/407x user manual

Revision history

Rev	Date	Description
3	20140312	• Figure 16 “EMC block diagram” updated and CCLK renamed to EMCCLK throughout the chapter. • Update Section 9.11.1 “Mode register setup”. • Function SSP2_SCK added to pin P5[2]. See Table 75 and Table 90. • Function SSP2_SSEL added to pin P5[3]. See Table 75 and Table 90. • Updated the description of the ROM_LAT bit in Table 7 “Matrix Arbitration register (MATRIXARB - 0x400F C188) bit description”. This bit should be 1 for normal operation. • Figure 23 “Ethernet packet fields” updated with the correct order of octets in the MAC address. • Description of FLASHTIM bit values 0x11 and 0x100 corrected in Table 50 “Flash Accelerator Configuration register (FLASHCFG - address 0x400F C000) bit description”. • IRCCTRL register added. See Table 34 “IRC control register (IRCCTRL - address 0x400F C1A4) bit description”. • Incorrect reference to VREFN removed in Table 684 “D/A Pin Description” and Table 686 “D/A Converter Register (CR - address 0x4008 C000) bit description”. • Description of the SPIFI software library removed. The description is available on LPCWare.com. • Description of the SPIFI hardware and register interface added. See Section 15.7. • Description of the SLEW bit improved. See Table 83 “Type D IOCON registers bit description”.
2	20130306	• Added LQFP100. • Minor updates and corrections.
1	20120913	• Initial LPC408x/407x User manual version.

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 LPC408x/407x is an ARM Cortex-M4 based microcontroller for embedded applications requiring a high level of integration and low power dissipation.

The Cortex-M4 processor is a high-performance 32-bit processor with a 3-stage pipeline Harvard architecture with separate local instruction and data buses, as well as a third bus with slightly lower performance for peripherals. The Cortex-M4 uses the Thumb® instruction set, providing high code density and reduced program memory requirements. The Cortex-M4 CPU also includes an internal prefetch unit that supports speculative branches. The LPC408x/407x adds a specialized flash memory accelerator to give optimal performance when executing code from flash. The LPC408x/407x is targeted to operate at up to a 120 MHz CPU frequency under worst case commercial conditions.

The peripheral complement of the LPC408x/407x includes up to 512 kB of Flash memory, up to 96 kB of data memory, 4,032 bytes of EEPROM memory, an External Memory Controller for SDRAM and static memory access, an LCD panel controller, an Ethernet MAC, a high speed SPI flash memory interface (SPIFI), a General Purpose DMA controller, a USB device/host/OTG interface, 5 UARTs, 3 SSP controllers, 3 I^2C interfaces, an I^2S serial audio interface, a 2-channel CAN interface, an SD card interface, an 8 channel 12-bit ADC, a 10-bit DAC, analog comparators, a Motor Control PWM, a Quadrature Encoder Interface, 4 general purpose timers, a 6-output general purpose PWM, an ultra-low power RTC with separate battery supply and event monitor/recorder, a windowed watchdog timer, a CRC calculation engine, up to 165 general purpose I/O pins, and more.

See Section 41.2 “References” for additional documentation related to the LPC408x/407x parts.

1.2 Features

Refer to Section 1.4 for details of features for specific part numbers.

- Functional replacement for LPC23xx and 24xx family devices.
- ARM Cortex-M4 processor, running at frequencies of up to 120 MHz. The Cortex-M4 executes the Thumb®-2 instruction set for optimal performance and code size, including hardware division, single cycle multiply, and bit-field manipulation. A Memory Protection Unit (MPU) supporting eight regions is included.
- Cortex-M4 built-in Nested Vectored Interrupt Controller (NVIC).
- Cortex-M4 Floating Point Unit (FPU), supporting 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. The FPU is not available on LPC4074 devices.
- Up to 512 kB on-chip flash program memory with In-System Programming (ISP) and In-Application Programming (IAP) capabilities. The combination of an enhanced flash memory accelerator and location of the flash memory on the CPU local code/data bus provides high code performance from flash.
- Up to 96 kB on-chip SRAM includes:
  - Up to 64 kB of Main SRAM on the CPU code/data bus for high-performance CPU access.
  - Up to two 16 kB SRAM blocks with separate access paths for higher throughput. These SRAM blocks may be used for Ethernet, USB, LCD, and DMA memory, as well as for general purpose instruction and data storage.
  - Up to 4,032 bytes of on-chip EEPROM.
- External Memory Controller provides support for asynchronous static memory devices such as RAM, ROM and Flash up to 64 MB, as well as dynamic memories such as Single Data Rate SDRAM.
- Eight channel General Purpose DMA controller (GPDMA) on the AHB multilayer matrix that can be used with the SSP, I^2S, UART, SD/MMC, CRC engine, Analog-to-Digital and Digital-to-Analog converter peripherals, timer match signals, GPIO, and for memory-to-memory transfers.
- Multilayer AHB matrix interconnect provides a separate bus for each AHB master. AHB masters include the CPU, General Purpose DMA controller, Ethernet MAC, LCD controller, and the USB interface. This interconnect provides communication with no arbitration delays unless two masters attempt to access the same slave at the same time.
- Split APB bus allows for higher throughput with fewer stalls between the CPU and DMA. A single level of write buffering allows the CPU to continue without waiting for completion of APB writes if the APB was not already busy.
- LCD controller, supporting both Super-Twisted Nematic (STN) and Thin-Film Transistor (TFT) displays. The LCD controller is not available on LPC407x devices.
  - Dedicated DMA controller.
  - Selectable display resolution (up to 1024 × 768 pixels).

- Supports up to 24-bit true-color mode.

- Serial interfaces:
  - Ethernet MAC with MII/RMII interface and dedicated DMA controller.
  - USB 2.0 full-speed controller that can be configured for either device, Host, or OTG operation with an on-chip PHY for device and Host functions and a dedicated DMA controller. USB Host and OTG are not available on LPC4074 devices.
  - Five UARTs with fractional baud rate generation, internal FIFOs, IrDA, DMA support, and RS-485/EIA-485 support on most LPC408x/407x devices. UART1 also has a full set of modem handshaking signals. UART4 includes a synchronous mode and a smart card mode supporting ISO 7816-3. UART4 is not available on LPC4074 devices.
  - Three SSP controllers with FIFO and multi-protocol capabilities. The SSP interfaces can be used with the GPDMA controller.
  - Three enhanced I^2C-bus interfaces, one with an open-drain output supporting the full I^2C specification and Fast mode Plus with data rates of 1Mbit/s, two with standard port pins. Enhancements include multiple address recognition and monitor mode.
  - Two-channel CAN controller.
  - I^2S (Inter-IC Sound) interface for digital audio input or output, with fractional rate control. The I^2S interface can be used with the GPDMA. The I^2S interface supports 3-wire data transmit and receive or 4-wire combined transmit and receive connections, as well as master clock output.
  - SPIFI (SPI Flash Interface). This interface uses an SPI bus superset with 4 data lines to access off-chip Quad SPI Flash memory at a much higher rate than is possible using standard SPI or SSP interfaces. The SPIFI function allows memory mapping the contents of the off-chip SPI Flash memory such that it can be executed as if it were on-chip code memory. Supports SPI memories with 1 or 4 data lines.

- Other peripherals:
  - SD card interface that also supports MMC cards. The SD card interface is not available on LPC4074 devices.
  - General Purpose I/O (GPIO) pins with configurable pull-up/down resistors, open drain mode, and repeater mode. All GPIOs are located on an AHB bus for fast access, and support Cortex-M4 bit-banding. GPIOs can be accessed by the General Purpose DMA Controller. Any pin of ports 0 and 2 can be used to generate an interrupt. There are 165 GPIOs on 208-pin packages, 141 GPIOs on 180-pin packages, and 109 GPIOs on 144-pin packages.
  - 12-bit Analog-to-Digital Converter (ADC) with input multiplexing among eight pins, conversion rates up to 400 kHz, and multiple result registers. The 12-bit ADC can be used with the GPDMA controller.
  - 10-bit Digital-to-Analog Converter (DAC) with dedicated conversion timer and DMA support.
  - Dual analog comparator with multiple selectable inputs, selectable internal reference voltages, and versatile interrupt generation. The comparators are not available on LPC4074 devices.

- Four general purpose timers/counters, with a total of eight capture inputs and ten compare outputs. Each timer block has an external count input. Specific timer events can be selected to generate DMA requests.
- One motor control PWM with support for three-phase motor control.
- Quadrature encoder interface that can monitor one external quadrature encoder. The QEI is not available on LPC4074 devices.
- Two standard PWM/timer blocks with external count input option.
- Real-Time Clock (RTC) with a separate power domain. The RTC is clocked by a dedicated RTC oscillator. The RTC block includes 20 bytes of battery-powered backup registers, allowing system status to be stored when the rest of the chip is powered off. Battery power can be supplied from a standard 3 V Lithium button cell. The RTC will continue working when the battery voltage drops to as low as 2.1 V. An RTC interrupt can wake up the CPU from any reduced power mode.
- Event Monitor/Recorder that can capture the RTC value when an event occurs on any of 3 inputs. The event identification and the time it occurred are stored in registers. The Event Monitor/Recorder is in the RTC power domain, and can therefore operate as long as there is RTC power.
- Windowed Watchdog Timer (WWDT). Windowed operation, dedicated internal oscillator, watchdog warning interrupt, and safety features.
- CRC Engine block can calculate a CRC on supplied data using 1 of 3 standard polynomials. The CRC engine can be used in conjunction with the DMA controller to generate a CRC without CPU involvement in the data transfer.
- Cortex-M4 system tick timer, including an external clock input option.

- Standard JTAG test/debug interface as well as Serial Wire Debug and Serial Wire Trace Port options.
- Emulation trace module supports real-time trace.
- Single 3.3 V power supply (2.4 V to 3.6 V). Temperature range of -40 °C to 85 °C.
- Four reduced power modes: Sleep, Deep-sleep, Power-down, and Deep Power-down.
- Power savings for operation at or below 100 MHz by reducing on-chip regulator output.
- Four external interrupt inputs configurable as edge/level sensitive. All pins on PORT0 and PORT2 can be used as edge sensitive interrupt sources.
- Non-maskable Interrupt (NMI) input.
- Clock output function that can reflect the main oscillator clock, IRC clock, RTC clock, CPU clock, USB clock, SPIFI clock, or the watchdog timer clock.
- The Wakeup Interrupt Controller (WIC) allows the CPU to automatically wake up from any priority interrupt that can occur while the clocks are stopped in deep sleep, Power-down, and Deep Power-down modes.
- Processor wake-up from Power-down mode via any interrupt able to operate during Power-down mode (includes external interrupts, RTC interrupt, USB activity, Ethernet wake-up interrupt, CAN bus activity, PORT0/2 pin interrupt, and NMI).
- Brownout detect with separate threshold for interrupt and forced reset.
- On-chip Power-On Reset (POR).
- On-chip crystal oscillator with an operating range of 1 MHz to 25 MHz.

- 12 MHz Internal RC oscillator (IRC) trimmed to 1% accuracy that can optionally be used as a system clock.
- An on-chip PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency crystal. May be run from the main oscillator or the internal RC oscillator.
- A second, dedicated PLL may be used for the USB and/or SPIFI interfaces in order to allow added flexibility for the Main PLL settings.
- Versatile pin function selection feature allows many possibilities for using on-chip peripheral functions.
- Boundary scan for simplified board testing.
- Unique device serial number for identification purposes.
- Available as 208-pin LQFP, 208-pin TFBGA, 180-pin TFBGA, 144-pin LQFP, 80-pin LQFP packages.

1.3 Applications

- **Communications**
  - Point-of-sale terminals, Web servers, multi-protocol bridges
- **Industrial/Medical**
  - Automation controllers, application control, robotic controls, HVAC, PLC, inverters, circuit breakers, medical scanning, security monitoring, motor drive, video intercom
- **Consumer/Appliance**
  - Audio, MP3 decoders, alarm systems, displays, printers, scanners, small appliances, fitness equipment
- **Automotive**
  - Aftermarket, car alarms, GPS/Fleet Monitor

1.4 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC4088			
LPC4088FBD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC4088FET208	TFBGA208	plastic thin fine-pitch ball grid array package; 208 balls; body 15 × 15 × 0.7 mm	SOT950-1
LPC4088FET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 × 12 × 0.8 mm	SOT570-2
LPC4088FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC4078			
LPC4078FBD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC4078FET208	TFBGA208	plastic thin fine-pitch ball grid array package; 208 balls; body 15 × 15 × 0.7 mm	SOT950-1
LPC4078FET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 × 12 × 0.8 mm	SOT570-2
LLPC4078FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC4078FBD80	LQFP80	plastic low-profile quad package; 80 leads; body 12 × 12 × 1.4 mm	SOT315-1
LPC4078FBD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC4076			
LPC4076FET180	TFBGA180	thin fine-pitch ball grid array package; 180 balls; body 12 × 12 × 0.8 mm	SOT570-2
LPC4076FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC4074			
LPC4074FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC4074FBD80	LQFP80	plastic low-profile quad package; 80 leads; body 12 × 12 × 1.4 mm	SOT315-1
LPC4072			
LPC4072FBD80	LQFP80	plastic low-profile quad package; 80 leads; body 12 × 12 × 1.4 mm	SOT315-1

1.4.1 Part options summary

Table 2. Ordering options

Type number	Flash (kB)	SRAM (kB)	EEPROM (B)	EMC bus width (bit)	LCD	Ethernet	USB	UART	QEI	SD/MMC	Comparator	FPU	Package
LPC4088													
LPC4088FBD208	512	96	4032	32	yes	yes	H/O/D	5	yes	yes	yes	yes	LQFP208
LPC4088FET208	512	96	4032	32	yes	yes	H/O/D	5	yes	yes	yes	yes	TFBGA208
LPC4088FET180	512	96	4032	16	yes	yes	H/O/D	5	yes	yes	yes	yes	TFBGA180
LPC4088FBD144	512	96	4032	8	yes	yes	H/O/D	5	yes	yes	yes	yes	LQFP144
LPC4078													
LPC4078FBD208	512	96	4032	32	no	yes	H/O/D	5	yes	yes	yes	yes	LQFP208

Table 2. Ordering options

Type number	Flash (kB)	SRAM (kB)	EEPROM (B)	EMC bus width (bit)	LCD	Ethernet	USB	UART	QEI	SD/MMC	Comparator	FPU	Package
LPC4078FET208	512	96	4032	32	no	yes	H/O/D	5	yes	yes	yes	yes	TFBGA208
LPC4078FET180	512	96	4032	16	no	yes	H/O/D	5	yes	yes	yes	yes	TFBGA180
LPC4078FBD144	512	96	4032	8	no	yes	H/O/D	5	yes	yes	yes	yes	LQFP144
LPC4078FBD100	512	96	4032	-	no	yes	H/O/D	5	yes	yes	yes	yes	LQFP100
LPC4078FBD80	512	96	4032	-	no	yes	H/O/D	5	yes	yes	yes	yes	LQFP80
LPC4076													
LPC4076FET180	256	80	2048	16	no	yes	H/O/D	5	yes	yes	yes	yes	TFBGA180
LPC4076FBD144	256	80	2048	8	no	yes	H/O/D	5	yes	yes	yes	yes	LQFP144
LPC4074													
LPC4074FBD144	128	40	2048	-	no	no	D	4	no	no	no	no	LQFP144
LPC4074FBD80	128	40	2048	-	no	no	D	4	no	no	no	no	BGA80
LPC4072													
LPC4072FBD80	64	24	2048	-	no	no	D	4	no	no	no	no	LQFP80

1.5 Simplified block diagram

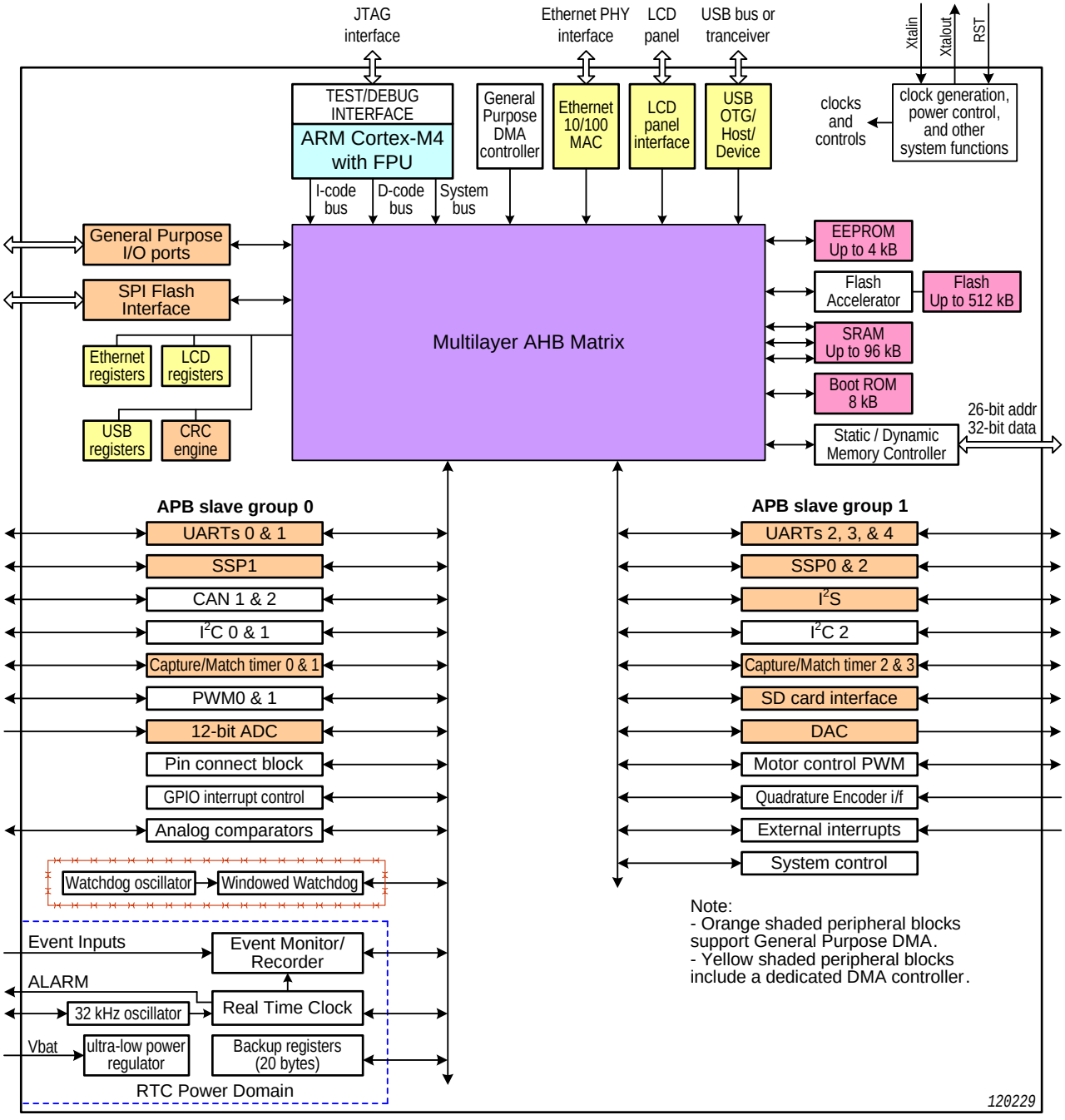


Fig 1. LPC408x/407x simplified block diagram

1.6 Architectural overview

The ARM Cortex-M4 includes three AHB-Lite buses, one system bus and the I-code and D-code buses which are faster and are used similarly to Tightly Coupled Memory interfaces: one bus dedicated for instruction fetch (I-code) and one bus for data access (D-code). The use of two core buses allows for simultaneous operations if concurrent operations target different devices.

The LPC408x/407x uses a multi-layer AHB matrix to connect the Cortex-M4 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. Details of the multilayer matrix connections are shown in Figure 2.

APB peripherals are connected to the CPU 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. The APB bus bridges are configured to buffer writes so that the CPU or DMA controller can write to APB devices without always waiting for APB write completion.

1.7 ARM Cortex-M4 processor

The ARM 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, and multiple core buses capable of simultaneous accesses.

Pipeline techniques are 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 Section 40.1.

1.8 On-chip flash memory system

The LPC408x/407x 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.

1.9 On-chip Static RAM

The LPC408x/407x contains up to 96 kB of on-chip static RAM memory. Up to 64 kB of SRAM, accessible by the CPU and the General Purpose DMA controller, is on a higher-speed bus. Up to 32 kB SRAM is provided in up to two additional 16 kB SRAM blocks for use primarily for peripheral data. When both SRAMs are present, they are situated on separate slave ports on the AHB multilayer matrix.

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.

1.10 On-chip EEPROM

The LPC408x/407x contains up to 4,032 bytes of on-chip EEPROM memory. The EEPROM is accessible only by the CPU.

1.11 Detailed block diagram

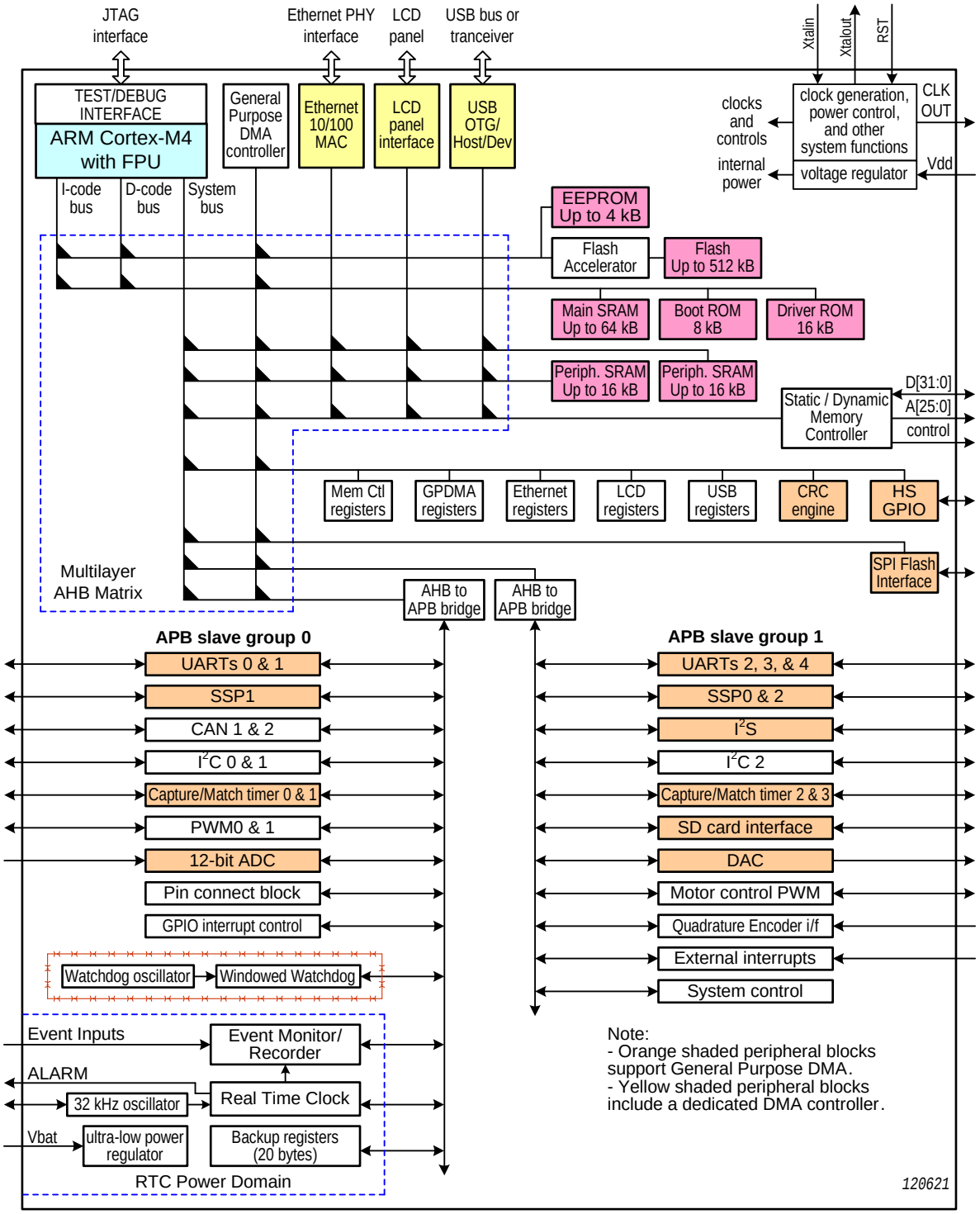


Fig 2. LPC408x/407x block diagram, CPU and buses

2.1 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 LPC408x/407x.

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	For devices with 512 kB of flash memory.
		0x0000 0000 - 0x0003 FFFF	For devices with 256 kB of flash memory.
		0x0000 0000 - 0x0001 FFFF	For devices with 128 kB of flash memory.
	On-chip SRAM	0x1000 0000 - 0x1000 FFFF	For devices with 64 kB of Main SRAM.
		0x1000 0000 - 0x1000 7FFF	For devices with 32 kB of Main SRAM.
	Boot ROM	0x1FFF 0000 - 0x1FFF 7FFF	8 kB Boot ROM with flash services.
	Driver ROM	0x1FFF 8000 - 0x1FFF 1FFF	16 kB Driver ROM
0x2000 0000 to 0x3FFF FFFF	On-chip SRAM (typically used for peripheral data)	0x2000 0000 - 0x2000 1FFF	Peripheral SRAM - bank 0 (first 8 kB)
		0x2000 2000 - 0x2000 3FFF	Peripheral SRAM - bank 0 (second 8 kB)
		0x2000 4000 - 0x2000 7FFF	Peripheral SRAM - bank 1 (16 kB)
	AHB peripherals	0x2008 0000 - 0x200B FFFF	See Section 2.3.1 for details
	SPIFI buffer space	0x2800 0000 - 0x28FF FFFF	SPIFI memory mapped access space
0x4000 0000 to 0x7FFF FFFF	APB Peripherals	0x4000 0000 - 0x4007 FFFF	APB0 Peripherals, up to 32 peripheral blocks of 16 kB each.
		0x4008 0000 - 0x400F FFFF	APB1 Peripherals, up to 32 peripheral blocks of 16 kB each.
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]
		0x9000 0000 - 0x93FF FFFF	Static memory chip select 1 (up to 64 MB)[2]
		0x9800 0000 - 0x9BFF FFFF	Static memory chip select 2 (up to 64 MB)
		0x9C00 0000 - 0x9FFF FFFF	Static memory chip select 3 (up to 64 MB)
		Four dynamic memory chip selects:	
		0xA000 0000 - 0xAFFF FFFF	Dynamic memory chip select 0 (up to 256MB)
		0xB000 0000 - 0xBFFF FFFF	Dynamic memory chip select 1 (up to 256MB)
		0xC000 0000 - 0xCFFF FFFF	Dynamic memory chip select 2 (up to 256MB)
		0xD000 0000 - 0xDFFF 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 SCS register bit 0 (Section 3.3.7.1).

[2] Can be up to 128 MB, upper address 0x97FF FFFF, if the address shift mode is enabled. See SCS register bit 0 (Section 3.3.7.1).

2.2 Memory maps

The LPC408x/407x incorporates several distinct memory regions, shown in the following figures. Figure 3 shows the overall map of the entire address space from the user program viewpoint following reset. The interrupt vector area supports address remapping, which is described later in this section.

Figure 3 and Table 5 show different views of the peripheral address space. The AHB peripheral area is 2 megabyte in size, and is divided to allow for up to 128 peripherals. The APB peripheral area is 1 megabyte in size and is divided to allow for up to 64 peripherals. Each peripheral of either type is allocated 16 kilobytes of space. This allows simplifying the address decoding for each peripheral.

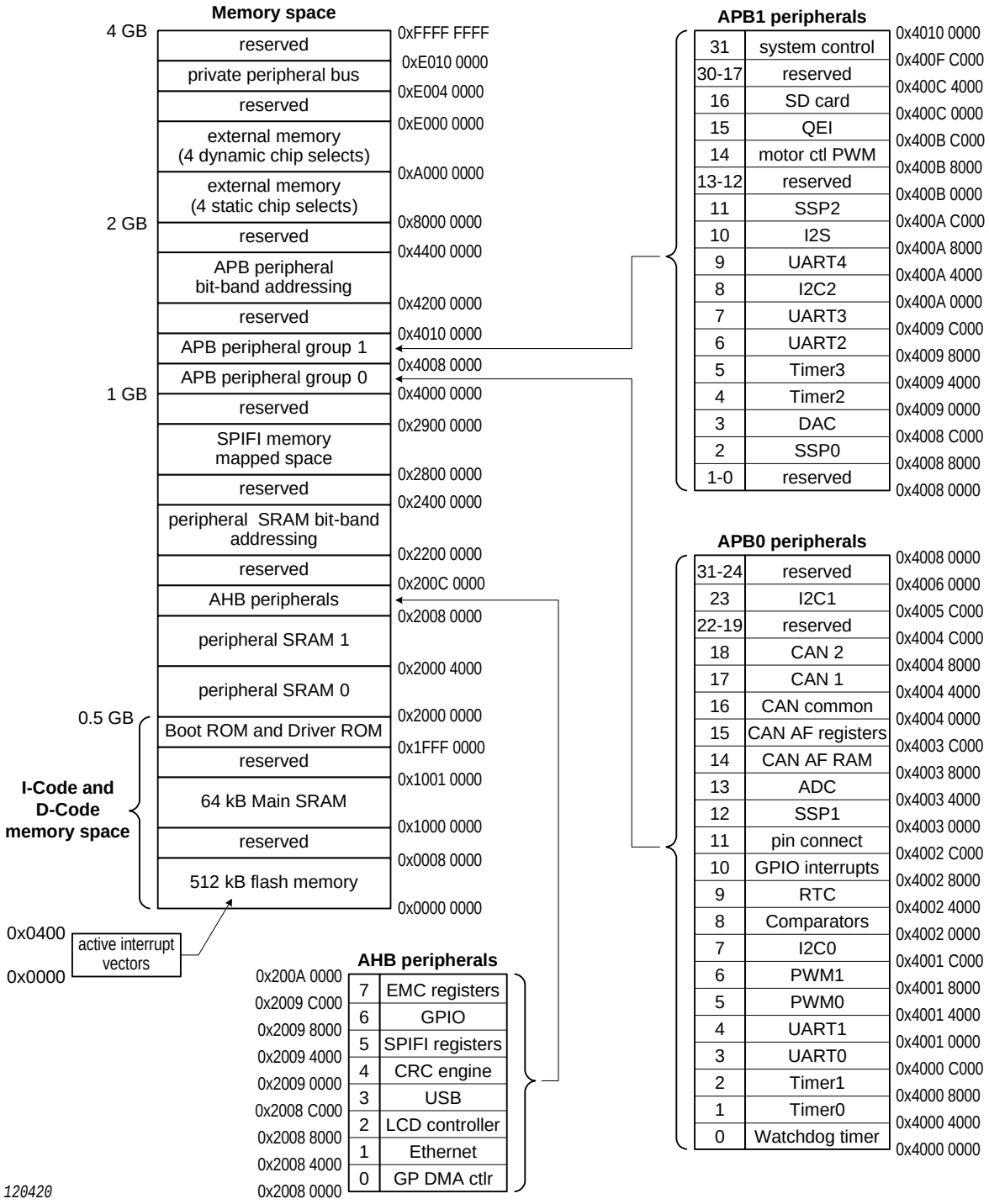


Fig 3. System memory map

2.3 On-chip peripherals

All peripheral register addresses are word aligned (to 32-bit boundaries) regardless of their size. This eliminates the need for byte lane mapping hardware that would be required to allow byte (8-bit) or half-word (16-bit) accesses to occur at smaller boundaries. An implication of this is that word and half-word registers must be accessed all at once. For example, it is not possible to read or write the upper byte of a word register separately.

2.3.1 AHB peripherals

The following table shows the addresses of peripheral functions that reside directly on the AHB bus matrix. Complete register descriptions may be found in the relevant chapters.

Table 4. AHB peripherals and base addresses

AHB peripheral	Address range	Peripheral name
0	0x2008 0000 to 0x2008 3FFF	General Purpose DMA controller
1	0x2008 4000 to 0x2008 7FFF	Ethernet MAC
2	0x2008 8000 to 0x2008 BFFF	LCD controller
3	0x2008 C000 to 0x2008 FFFF	USB interface
4	0x2009 0000 to 0x2009 3FFF	CRC engine
5	0x2009 4000 to 0x2009 7FFF	SPIFI
6	0x2009 8000 to 0x2009 BFFF	GPIO
7	0x2009 C000 to 0x2009 FFFF	External Memory Controller
8 to 15	0x200A 0000 to 0x200B FFFF	reserved

2.3.2 APB peripheral addresses

The following table shows the address maps of the 2 APB buses. APB peripherals do not use all of the 16 kB space allocated to them. Typically each device's registers are "aliased" or repeated at multiple locations within each 16 kB range.

Table 5. APB0 peripherals and base addresses

APB0 peripheral	Base address	Peripheral name
0	0x4000 0000	Watchdog Timer
1	0x4000 4000	Timer 0
2	0x4000 8000	Timer 1
3	0x4000 C000	UART0
4	0x4001 0000	UART1
5	0x4001 4000	PWM0
6	0x4001 8000	PWM1
7	0x4001 C000	I^2C0
8	0x4002 0000	Comparators
9	0x4002 4000	RTC and Event Monitor/Recorder
10	0x4002 8000	GPIO interrupts
11	0x4002 C000	Pin Connect Block
12	0x4003 0000	SSP1
13	0x4003 4000	ADC

Table 5. APB0 peripherals and base addresses

APB0 peripheral	Base address	Peripheral name
14	0x4003 8000	CAN Acceptance Filter RAM
15	0x4003 C000	CAN Acceptance Filter Registers
16	0x4004 0000	CAN Common Registers
17	0x4004 4000	CAN Controller 1
18	0x4004 8000	CAN Controller 2
19 to 22	0x4004 C000 to 0x4005 8000	reserved
23	0x4005 C000	I^2C1
24 to 31	0x4006 0000 to 0x4007 C000	reserved

Table 6. APB1 peripherals and base addresses

APB1 peripheral	Base address	Peripheral name
0 to 1	0x4008 0000 to 0x4008 4000	reserved
2	0x4008 8000	SSP0
3	0x4008 C000	DAC
4	0x4009 0000	Timer 2
5	0x4009 4000	Timer 3
6	0x4009 8000	UART2
7	0x4009 C000	UART3
8	0x400A 0000	I^2C2
9	0x400A 4000	UART4
10	0x400A 8000	I^2S
11	0x400A C000	SSP2
12 to 13	0x400B 0000 to 0x400B 4000	reserved
14	0x400B 8000	Motor control PWM
15	0x400B C000	Quadrature Encoder Interface
16	0x400C 0000	SD card interface
17 to 30	0x400D 0000 to 0x400F 8000	reserved
31	0x400F C000	System control

2.4 Memory re-mapping

The Cortex-M4 incorporates a mechanism that allows remapping the interrupt vector table to alternate locations in the memory map. This is controlled via the Vector Table Offset Register contained in the Cortex-M4. Refer to the NVIC description in Section 5.4 and to the ARM Cortex-M4 User Guide referred to in Section 40.1.

Boot ROM re-mapping

Following a hardware reset, the Boot ROM is temporarily mapped to address 0. This is normally transparent to the user. However, if execution is halted immediately after reset by a debugger, it should correct the mapping for the user. See Section 39.8.

2.5 AHB arbitration

The Multilayer AHB Matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. By default, the Cortex-M4 D-code bus has the highest priority, followed by the I-Code bus. All other masters share a lower priority.

The default priority can be altered by the user if care is taken. This may be particularly useful if the LCD interface is used and it has difficulty getting sufficient data.

2.5.1 Matrix Arbitration register

The Matrix Arbitration register provides the ability to change the default AHB Matrix arbitration priorities.

Table 7. Matrix Arbitration register (MATRIXARB - 0x400F C188) bit description

Bit	Symbol	Description	Reset value
1:0	PRI_ICODE	I-Code bus priority. Should be lower than PRI_DCODE for proper operation.	0x1
3:2	PRI_DCODE	D-Code bus priority.	0x3
5:4	PRI_SYS	System bus priority.	0
7:6	PRI_GPDMA	General Purpose DMA controller priority.	0
9:8	PRI_ETH	Ethernet DMA priority.	0
11:10	PRI_LCD	LCD DMA priority.	0
13:12	PRI_USB	USB DMA priority.	0
15:14	-	Reserved. Read value is undefined, only zero should be written.	NA
16	ROM_LAT	ROM latency select. Set to 1 by the boot code. Special notes when operating at above 60MHz: 1. When connecting the device for debugging, user should set this bit for proper operation if the debugger bypasses the boot code. 2. If the value of this register is altered by user code, this bit must not be inadvertently cleared.	1
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

The values used for the various priorities are 3 = highest, 0 = lowest.

An example of a way to give priority to the LCD DMA is to use the value 0x0000 0C09. The gives the LCD highest priority, D-code second priority, I-Code third priority, and all others lowest priority.

Where in the memory space code and various types of data are located can be managed to help minimize the need for arbitration and possible starvation of any of the bus masters, as well as a need for changing the default priorities. For instance, LCD refresh from off-chip memory connected to the EMC, while also executing off-chip code via the EMC can cause a great deal of arbitration.

3.1 Introduction

The system control block includes several system features and control registers for a number of functions that are not related to specific peripheral devices. These include:

- Chip Reset (see Section 3.4)
- Peripheral Reset control (see Section 3.5)
- Brown-Out Detection (see Section 3.6)
- External Interrupt Inputs (see Section 3.7)

Each type of function has its own registers if any are required and unneeded bits are defined as reserved in order to allow future expansion.

3.1.1 Summary of clocking and power control functions

This section describes the generation of the various clocks needed for device operation, and options of clock source selection, as well as power control and wake-up from reduced power modes. Functions described in the following subsections include:

- Oscillators (see Section 3.8)
- PLLs (see Section 3.10)
- Clock selection and dividers (see Section 3.11)
- Power control (see Section 3.12)
- Wake-up timer (see Section 3.13)
- External clock output (see Section 3.14)

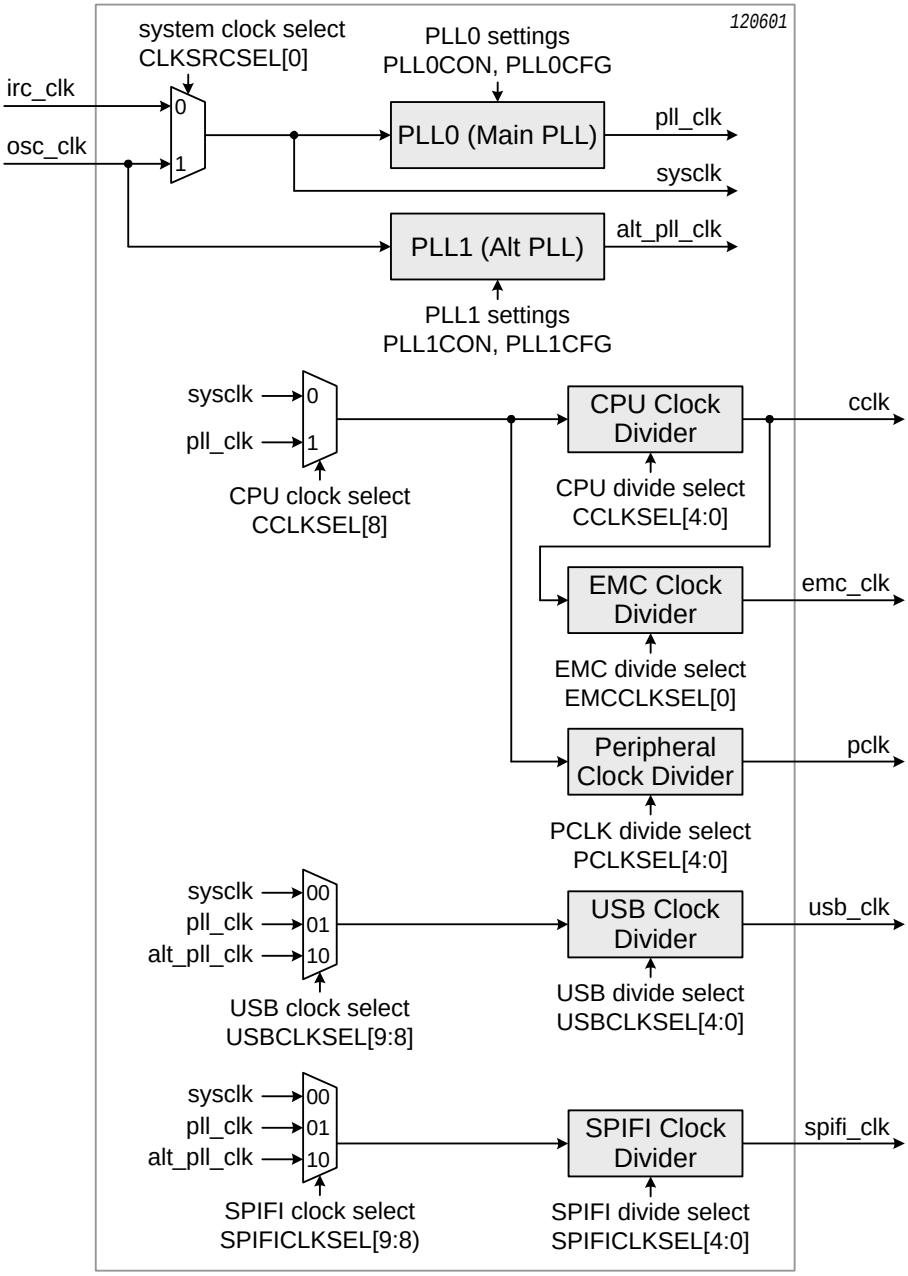


Fig 4. Clock generation

3.2 Pin description

Table 8 shows pins that are associated with System Control block functions.

Table 8. Pin summary

Pin name	Pin direction	Pin description
EINT0	Input	**External Interrupt Input 0** - An active low/high level or falling/rising edge general purpose interrupt input. This pin may be used to wake up the processor from Sleep, Deep-sleep, or Power-down modes.
EINT1	Input	**External Interrupt Input 1** - See the EINT0 description above.
EINT2	Input	**External Interrupt Input 2** - See the EINT0 description above.
EINT3	Input	**External Interrupt Input 3** - See the EINT0 description above.
$\overline{\text{RESET}}$	Input	**External Reset input** - A LOW on this pin resets the chip, causing I/O ports and peripherals to take on their default states, and the processor to begin execution at address 0x0000 0000.

3.3 Register description

All registers, regardless of size, are on word address boundaries. Details of the registers appear in the description of each function.

Table 9. Register overview: System control (base address 0x400F C000)

Name	Access	Address offset	Description	Reset value	Reference
PLL registers					3.3.1
PLLCON0:1	R/W	0x080; 0xA0	PLL0 and PLL1 Control registers	0	3.3.1.1
PLLCFG0:1	R/W	0x084; 0xA4	PLL0 and PLL1 Configuration registers	0	3.3.1.2
PLLSTAT0:1	RO	0x088; 0xA8	PLL0 and PLL1 Status registers	0	3.3.1.3
PLLFEED0:1	WO	0x08C; 0xAC	PLL0 and PLL1 Feed registers	NA	3.3.1.5
Power control					3.3.2
PCON	R/W	0x0C0	Power Control register	0	3.3.2.1
PCONP	R/W	0x0C4	Power Control for Peripherals	0x0408 829E	3.3.2.2
PCONP1	R/W	0x0C8	Power Control for Peripherals 1	0x8	3.3.2.2
PBOOST	R/W	0x1B0	Power boost register	0x3	3.3.2.3
Clock selection and divider registers					3.3.3
EMCCLKSEL	R/W	0x100	External Memory Controller Clock Selection register	0	3.3.3.1
CCLKSEL	R/W	0x104	CPU Clock Selection register	1	3.3.3.2
USBCLKSEL	R/W	0x108	USB Clock Selection register	0	3.3.3.3
CLKSRCSEL	R/W	0x10C	Clock Source Select Register	0	3.3.3.4
PCLKSEL	R/W	0x1A8	Peripheral Clock Selection register	0x04	3.3.3.5
SPIFICLKSEL	R/W	0x1B4	SPIFI Clock Selection register	0	3.3.3.6
External interrupts					3.3.4
EXTINT	R/W	0x140	External Interrupt Flag Register	0	3.3.4.1
EXTMODE	R/W	0x148	External Interrupt Mode register	0	3.3.4.2
EXTPOLAR	R/W	0x14C	External Interrupt Polarity Register	0	3.3.4.3
Device and peripheral reset					3.3.5
RSID	R/W	0x180	Reset Source Identification Register	see Table 28	3.3.5.1
RSTCON0	R/W	0x1CC	Individual peripheral reset control bits	0	3.3.5.2
RSTCON1	R/W	0x1D0	Individual peripheral reset control bits	0	3.3.5.3
EMC delay control and calibration					3.3.6
EMCDLYCTL	R/W	0x1DC	Values for the 4 programmable delays associated with SDRAM operation.	0x210	3.3.6.1
EMCCAL	R/W	0x1E0	Controls the calibration counter for programmable delays and returns the result value.	0x1F00	3.3.6.2
Miscellaneous system control registers					3.3.7
SCS	R/W	0x1A0	System Control and Status	0	3.3.7.1
IRCCTRL	R/W	0x1A4	IRC control	0x80	3.3.7.1
LCD_CFG	R/W	0x1B8	LCD Clock configuration register	0	3.3.7.3
CANSLEEPCLR	R/W	0x110	Allows clearing the current CAN channel sleep state as well as reading back that state.	0	3.3.7.4

Table 9. Register overview: System control (base address 0x400F C000)

Name	Access	Address offset	Description	Reset value	Reference
CANWAKEFLAGS	R/W	0x114	Indicates the wake-up state of the CAN channels.	0	3.3.7.5
USBINTST	R/W	0x1C0	USB Interrupt Status	0x8000 0000	3.3.7.6
DMACREQSEL	R/W	0x1C4	Selects between alternative requests on DMA channels 0 through 7 and 10 through 15.	0	3.3.7.7
CLKOUTCFG	R/W	0x1C8	Clock Output Configuration register	0	3.3.7.8

3.3.1 PLL registers

3.3.1.1 PLL Control registers

The PLLCON registers contains the bits that enable and connect each PLL. Enabling a PLL allows it to attempt to lock to the current settings of the multiplier and divider values. Changes to a PLLCON register do not take effect until a correct PLL feed sequence has been given for that PLL (see Section 3.3.1.5 and Section 3.3.1.2).

Table 10. PLL Control registers (PLLCON[0:1] - addresses 0x400F C080 (PLLCON0) and 0x400F C0A0 (PLLCON1)) bit description

Bit	Symbol	Description	Reset value
0	PLLE	PLL Enable. When 1, and after a valid PLL feed, this bit will activate the related PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 12.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	NA

Each PLL must be set up, enabled, and lock established before it may be used as a clock source. The hardware does not insure that the PLL is locked before it is selected nor does it automatically disconnect the PLL if lock is lost during operation.

3.3.1.2 PLL Configuration registers

The PLLCFG register contains the PLL multiplier and divider values. Changes to the PLLCFG register do not take effect until a correct PLL feed sequence has been given (see Section 3.3.1.5). Calculations for the PLL frequency, and multiplier and divider values are found in Section 3.10.5.

Table 11. PLL Configuration registers (PLLCFG[0:1] - addresses 0x400F C084 (PLLCFG0) and 0x400F C0A4 (PLLCFG1)) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations. The value stored here is the M value minus 1. **Note:** For details on selecting the right value for MSEL see Section 3.10.4.	0
6:5	PSEL	PLL 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 **Note:** For details on selecting the right value for PSEL see Section 3.10.4.	0
31:7	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.1.3 PLL Status registers

The read-only PLLSTAT register provides the actual PLL parameters that are in effect at the time it is read, as well as the PLL status. PLLSTAT may disagree with values found in PLLCON and PLLCFG because changes to those registers do not take effect until a proper PLL feed has occurred (see Section 3.3.1.5 “PLL Feed registers”).

Table 12. PLL Status registers (PLLSTAT[0:1] - addresses 0x400F C088 (PLLSTAT0) and 0x400F C0A8 (PLLSTAT1)) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL	Read-back for the PLL Multiplier value. This is the value currently used by the related PLL.	0
6:5	PSEL	Read-back for the PLL Divider value. This is the value currently used by the related PLL.	0
7	-	Reserved. The value read from a reserved bit is not defined.	NA
8	PLLE_STAT	Read-back for the PLL Enable bit. When 1, the related PLL is currently activated. When 0, the related PLL is turned off. This bit is automatically cleared when Power-down mode is activated.	0
9	-	Reserved. The value read from a reserved bit is not defined.	NA
10	PLOCK	Reflects the PLL Lock status. When 0, the related PLL is not locked. When 1, the related PLL is locked onto the requested frequency.	0
31:11	-	Reserved. The value read from a reserved bit is not defined.	NA

3.3.1.4 PLL Interrupts: PLOCK0 and PLOCK1

The PLOCK bit in the PLLSTAT register reflects the lock status of the related PLL1. When the PLL is first enabled, or when its parameters are changed, the PLL requires some time to establish lock under the new conditions. The related PLOCK bit can be monitored to determine when the PLL may be connected for use.

Each PLOCK bit is connected to the interrupt controller. This allows for software to turn on the PLL and continue with other functions without having to wait for the PLL to achieve lock. When the interrupt occurs, the PLL may be selected as a clock source, and the interrupt disabled. PLOCK0 and PLOCK1 appear as exception numbers 32 and 48 respectively in Table 51. Note that each PLOCK bit remains asserted whenever the related PLL is locked, so if the interrupt is used, the interrupt service routine must disable the interrupt prior to exiting.

3.3.1.5 PLL Feed registers

A correct feed sequence must be written to the related PLLFEED register in order for changes to the related PLLCON and PLLCFG registers to take effect. The feed sequence is:

1. Write the value 0xAA to PLLFEED.
2. Write the value 0x55 to PLLFEED.

The two writes must be in the correct sequence, and there must be no other register access in the same address space (0x400F C000 to 0x400F FFFF) between them. Because of this, it may be necessary to disable interrupts for the duration of the PLL feed operation, if there is a possibility that an interrupt service routine could write to another

register in that space. If either of the feed values is incorrect, or one of the previously mentioned conditions is not met, any changes to the PLLCON or PLLCFG register will not become effective.

Table 13. PLL Feed registers (PLLFEED[0:1] - addresses 0x400F C08C (PLLFEED0) and 0x400F C0AC (PLLFEED1)) bit description

Bit	Symbol	Description	Reset value
7:0	PLLFEED	The PLL feed sequence must be written to this register in order for the related PLL's configuration and control register changes to take effect.	0x00
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.2 Power control

3.3.2.1 Power Mode Control register

Controls for some reduced power modes and other power related controls are contained in the PCON register, as described in Table 14.

Table 14. Power Mode Control register (PCON - address 0x400F C0C0) bit description

Bit	Symbol	Description	Reset value
0	PM0	Power mode control bit 0. This bit controls entry to the Power-down mode. See Section 3.3.2.1.1 below for details.	0
1	PM1	Power mode control bit 1. This bit controls entry to the Deep Power-down mode. See Section 3.3.2.1.1 below for details.	0
2	BODRPM	Brown-Out Reduced Power Mode. When BODRPM is 1, the Brown-Out Detect circuitry will be turned off when chip Power-down mode or Deep Sleep mode is entered, resulting in a further reduction in power usage. However, the possibility of using Brown-Out Detect as a wake-up source from the reduced power mode will be lost. When 0, the Brown-Out Detect function remains active during Power-down and Deep Sleep modes. See the System Control Block chapter for details of Brown-Out detection.	0
3	BOGD	Brown-Out Global Disable. When BOGD is 1, the Brown-Out Detect circuitry is fully disabled at all times, and does not consume power. When 0, the Brown-Out Detect circuitry is enabled. See the System Control Block chapter for details of Brown-Out detection. **Note:** the Brown-Out Reset Disable (BORD, in this register) and the Brown-Out Interrupt (see Section 5.1) must be disabled when software changes the value of this bit.	0
4	BORD	Brown-Out Reset Disable. When BORD is 1, the BOD will not reset the device when the $V_{DD(REG)(3V3)}$ voltage dips goes below the BOD reset trip level. The Brown-Out interrupt is not affected. When BORD is 0, the BOD reset is enabled. See the Section 3.6 for details of Brown-Out detection.	0
7:3	-	Reserved. Read value is undefined, only zero should be written.	NA
8	SMFLAG	Sleep Mode entry flag. Set when the Sleep mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][2]
9	DSFLAG	Deep Sleep entry flag. Set when the Deep Sleep mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][2]
10	PDFLAG	Power-down entry flag. Set when the Power-down mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][2]
11	DPDFLAG	Deep Power-down entry flag. Set when the Deep Power-down mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][3]
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

[1] Only one of these flags will be valid at a specific time.

[2] Hardware reset value only for a power-up of core power or by a brownout detect event.

[3] Hardware reset value only for a power-up event on Vbat.

3.3.2.1.1 Encoding of Reduced Power Modes

The PM1and PM0 bits in PCON allow entering reduced power modes as needed. The encoding of these bits allows backward compatibility with devices that previously only supported Sleep and Power-down modes. Table 15 below shows the encoding for the three reduced power modes.

Table 15. Encoding of reduced power modes

PM1, PM0	Description
00	Execution of WFI or WFE enters either Sleep or Deep Sleep mode as defined by the SLEEPDEEP bit in the Cortex-M4 System Control Register.
01	Execution of WFI or WFE enters Power-down mode if the SLEEPDEEP bit in the Cortex-M4 System Control Register is 1.
10	Reserved, this setting should not be used.
11	Execution of WFI or WFE enters Deep Power-down mode if the SLEEPDEEP bit in the Cortex-M4 System Control Register is 1.

3.3.2.2 Power Control for Peripherals registers

The PCONP registers allow turning off selected peripheral functions for the purpose of saving power. This is accomplished by gating off the clock source to the specified peripheral blocks. A few peripheral functions cannot be turned off (i.e. the Watchdog timer and the System Control block).

Some peripherals, particularly those that include analog functions, may consume power that is not clock dependent. These peripherals may contain a separate disable control that turns off additional circuitry to reduce power. When this is the case, the peripheral should be disabled internally first, then turned off using PCONP, in order to get the greatest power savings. Information on peripheral specific power saving features may be found in the chapter describing that peripheral.

Each bit in PCONP controls one peripheral as shown in Table 16.

If a peripheral control bit is 1, that peripheral is enabled. If a peripheral control bit is 0, that peripheral's clock is disabled (gated off) to conserve power. For example if bit 19 is 1, the I^2C1 interface is enabled. If bit 19 is 0, the I^2C1 interface is disabled.

Important: valid data reads from a peripheral register and valid data writes to a peripheral register are possible only if that peripheral is enabled in the PCONP register!

Table 16. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description

Bit	Symbol	Description	Reset value
0	PCLCD	LCD controller power/clock control bit.	0
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	PCPWM0	PWM0 power/clock control bit.	0
6	PCPWM1	PWM1 power/clock control bit.	0
7	PCI2C0	I^2C0 interface power/clock control bit.	1
8	PCUART4	UART4 power/clock control bit.	0
9	PCRTC	RTC and Event Monitor/Recorder power/clock control bit.	1
10	PCSSP1	SSP 1 interface power/clock control bit.	0
11	PCEMC	External Memory Controller power/clock control bit.	0
12	PCADC	A/D converter (ADC) power/clock control bit. **Note:** Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before attempting to set PDN.	0
13	PCCAN1	CAN Controller 1 power/clock control bit.	0
14	PCCAN2	CAN Controller 2 power/clock control bit.	0
15	PCGPIO	Power/clock control bit for IOCON, GPIO, and GPIO interrupts.	1
16	PCSPIFI	SPI Flash Interface power/clock control bit.	0
17	PCMCPWM	Motor Control PWM power/clock control bit.	0
18	PCQEI	Quadrature Encoder Interface power/clock control bit.	0
19	PCI2C1	I^2C1 interface power/clock control bit.	1
20	PCSSP2	SSP2 interface power/clock control bit.	0
21	PCSSP0	SSP0 interface power/clock control bit.	0
22	PCTIM2	Timer 2 power/clock control bit.	0
23	PCTIM3	Timer 3 power/clock control bit.	0
24	PCUART2	UART 2 power/clock control bit.	0
25	PCUART3	UART 3 power/clock control bit.	0
26	PCI2C2	I^2C interface 2 power/clock control bit.	1
27	PCI2S	I^2S interface power/clock control bit.	0
28	PCSDC	SD Card interface power/clock control bit.	0
29	PCGPDMA	GPDMA function power/clock control bit.	0
30	PCENET	Ethernet block power/clock control bit.	0
31	PCUSB	USB interface power/clock control bit.	0

Table 17. Power Control for Peripherals register (PCONP1 - address 0x400F C0C8) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	NA
3	PCCMP	Comparator power/clock control bit.	1
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

Note that the DAC peripheral does not have a control bit in PCONP. To enable the DAC, its output must be selected to appear on the related pin, P0[26], by configuring the relevant IOCON register. See Section 7.4.1.

3.3.2.3 Power Boost control register

The Power Boost control register allows choosing between high-speed operation above 100 MHz, or power savings when operation is at 100 MHz or lower, by controlling the output of the main on-chip regulator. The boost feature is turned on when user code is first executed following reset. It can then be turned off by user code if the CPU clock rate will always be at or below 100 MHz, thus saving power that is only needed for operation above 100 MHz. Details are show in Table 18.

Table 18. Power Boost control register (PBOOST - address 0x400F C1B0) bit description

Bit	Symbol	Description	Reset value
1:0	Boost	Boost control bits. 00 : Boost is off, operation must be below 100 MHz. 11 : Boost is on, operation up to 120 MHz is supported. Other values are not allowed.	0x3
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.3 Clock selection and divider registers

3.3.3.1 EMC Clock Selection register

The EMCCLKSEL register controls division of the clock before it is used by the EMC. The EMC uses the same base clock as the CPU and the APB peripherals. The EMC clock can be the same as the CPU clock, or half that. This is intended to be used primarily when the CPU is running faster than the external bus can support.

Table 19. EMC Clock Selection register (EMCCLKSEL - address 0x400F C100) bit description

Bit	Symbol	Value	Description	Reset value
0	EMCDIV		Selects the EMC clock rate relative to the CPU clock.	0
		0	The EMC uses the same clock as the CPU.	
		1	The EMC uses a clock at half the rate of the CPU.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.3.2 CPU Clock Selection register

The CCLKSEL register controls selection of the clock used as the Main PLL input, and also controls the division of the PLL0 output before it is used by the CPU. When PLL0 is bypassed, the division may be by 1. When PLL0 is running, the output must be divided in order to bring the CPU clock frequency (CCLK) within operating limits. A 5-bit divider allows a range of options, including slowing CPU operation to a low rate for temporary power savings without turning off PLL0. Note that the CPU clock rate should not be set lower than the peripheral clock rate.

The two clock sources that may be chosen to drive PLL0 and ultimately the CPU and on-chip peripheral devices are the main oscillator and the Internal RC oscillator.

The clock source selection for PLL0 can only be changed safely when PLL0 is not being used. For a detailed description of how to change the clock source in a system using PLL0 see Section 3.10.7 “PLL configuration examples”.

Note the following restrictions regarding the choice of clock sources:

- The IRC oscillator should not be used (via PLL0) as the clock source for the USB subsystem.
- The IRC oscillator should not be used (via PLL0) as the clock source for the CAN controllers if the CAN baud rate is higher than 100 kbit/s.

Table 20. CPU Clock Selection register (CCLKSEL - address 0x400F C104) bit description

Bit	Symbol	Value	Description	Reset value
4:0	CCLKDIV		Selects the divide value for creating the CPU clock (CCLK) from the selected clock source. 0 = The divider is turned off., no clock will be provided to the CPU. This setting should typically not be used, the CPU will be halted and a reset will be required to restore operation. 1 = The input clock is divided by 1 to produce the CPU clock. 2 = The input clock is divided by 2 to produce the CPU clock. 3 = The input clock is divided by 3 to produce the CPU clock. ... 31 = The input clock is divided by 31 to produce the CPU clock.	0x01
7:5	-		Reserved. Read value is undefined, only zero should be written.	NA
8	CCLKSEL		Selects the input clock for the CPU clock divider.	0
		0	Sysclk is used as the input to the CPU clock divider.	
		1	The output of the Main PLL is used as the input to the CPU clock divider.	
31:9	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.3.3 USB Clock Selection register

The USBCLKSEL register controls selection of the clock used for the USB subsystem, and also controls the division of the that clock before it is used by the USB. The output of the selected PLL must be divided in order to bring the USB clock frequency to 48 MHz with a 50% duty cycle. A divider allows obtaining the correct USB clock from any even multiple of 48 MHz (i.e. any multiple of 96 MHz) within the PLL operating range.

Remark: A clock derived from the Internal RC oscillator should not be used to clock the USB subsystem.

Table 21. USB Clock Selection register (USBCLKSEL - address 0x400F C108) bit description

Bit	Symbol	Value	Description	Reset value
4:0	USBDIV		Selects the divide value for creating the USB clock from the selected PLL output. Only the values shown below can produce even number multiples of 48 MHz from the PLL. **Warning:** Improper setting of this value will result in incorrect operation of the USB interface. Only the main oscillator in conjunction with either PLL0 or PLL1 can provide a clock that meets USB accuracy and jitter specifications. Other values cannot produce the 48 MHz clock required for USB operation.	0
		0x0	The divider is turned off, no clock will be provided to the USB subsystem.	
		0x1	The selected output is divided by 1. The PLL output must be 48 MHz.	
		0x2	The selected output is divided by 2. The PLL output must be 96 MHz.	
		0x3	The selected output is divided by 3. The PLL output must be 144 MHz.	
7:5	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 21. USB Clock Selection register (USBCLKSEL - address 0x400F C108) bit description

Bit	Symbol	Value	Description	Reset value
9:8	USBSEL		Selects the input clock for the USB clock divider.	0
		0x0	Sysclk is used as the input to the USB clock divider. When this clock is selected, the USB can be accessed by software but cannot perform USB functions.	
		0x1	The output of the Main PLL is used as the input to the USB clock divider.	
		0x2	The output of the Alt PLL is used as the input to the USB clock divider.	
		0x3	Reserved, this setting should not be used.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.3.4 Clock Source Selection register

The CLKSRCSEL register controls selection of the clock used for sysclk and PLL0.

Table 22. Clock Source Selection register (CLKSRCSEL - address 0x400F C10C) bit description

Bit	Symbol	Value	Description	Reset value
0	CLKSRC		Selects the clock source for sysclk and PLL0 as follows:	0
		0	Selects the Internal RC oscillator as the sysclk and PLL0 clock source (default).	
		1	Selects the main oscillator as the sysclk and PLL0 clock source.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.3.5 Peripheral Clock Selection register

The PCLKSEL register controls the base clock used for all APB peripherals. A clock divider allows a range of frequencies to be used. Note that the peripheral clock rate should not be set higher than the CPU clock rate.

Table 23. Peripheral Clock Selection register (PCLKSEL - address 0x400F C1A8) bit description

Bit	Symbol	Description	Reset value
4:0	PCLKDIV	Selects the divide value for the clock used for all APB peripherals. 0 = The divider is turned off., no clock will be provided to APB peripherals. 1 = The input clock is divided by 1 to produce the APB peripheral clock. 2 = The input clock is divided by 2 to produce the APB peripheral clock. 3 = The input clock is divided by 3 to produce the APB peripheral clock. 4 = The input clock is divided by 4 to produce the APB peripheral clock. Other values = not supported.	0x04
31:5	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.3.6 SPIFI Clock Selection register

The SPIFICLKSEL register controls selection of the clock used for the SPIFI, and also controls the division of that clock before it is used by the SPIFI. If a PLL is used as the SPIFI clock source, its output must be divided in order to bring the frequency down to one that will work with the SPIFI. A 5-bit divider allows a range of frequencies to be used.

Table 24. SPIFI Clock Selection register (SPIFICLKSEL - address 0x400F C1B4) bit description

Bit	Symbol	Value	Description	Reset value
4:0	SPIFIDIV		Selects the divide value for creating the SPIFI clock from the selected clock source. 0 = The divider is turned off., no clock will be provided to the SPIFI. 1 = The input clock is divided by 1 to produce the SPIFI clock. 2 = The input clock is divided by 2 to produce the SPIFI clock. 3 = The input clock is divided by 3 to produce the SPIFI clock. ... 31 = The input clock is divided by 31 to produce the SPIFI clock.	0
7:5	-		Reserved. Read value is undefined, only zero should be written.	NA
9:8	SPIFISEL		Selects the input clock for the USB clock divider.	0
		0x0	Sysclk is used as the input to the SPIFI clock divider.	
		0x1	The output of the Main PLL is used as the input to the SPIFI clock divider.	
		0x2	The output of the Alt PLL is used as the input to the SPIFI clock divider.	
		0x3	Reserved, this setting should not be used.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.4 External interrupts

3.3.4.1 External Interrupt flag register

When a pin is selected for its external interrupt function, the level or edge on that pin (selected by its bits in the EXTPOLAR and EXTMODE registers) will set its interrupt flag in this register. This asserts the corresponding interrupt request to the NVIC, which will cause an interrupt if interrupts from the pin are enabled.

Writing ones to bits EINT0 through EINT3 in EXTINT register clears the corresponding bits. In level-sensitive mode the interrupt is cleared only when the pin is in its inactive state.

Once a bit from EINT0 to EINT3 is set and an appropriate code starts to execute (handling wake-up and/or external interrupt), this bit in EXTINT register must be cleared. Otherwise event that was just triggered by activity on the EINT pin will not be recognized in future.

Important: whenever a change of external interrupt operating mode (i.e. active level/edge) is performed (including the initialization of an external interrupt), the corresponding bit in the EXTINT register must be cleared! For details see Section 3.3.4.2 “External Interrupt Mode register” **and** Section 3.3.4.3 “External Interrupt Polarity register”.

For example, if a system wakes up from Power-down using low level on external interrupt 0 pin, its post wake-up code must reset EINT0 bit in order to allow future entry into the Power-down mode. If EINT0 bit is left set to 1, any subsequent attempt to invoke Power-down mode will fail. The same goes for external interrupt handling.

More details on Power-down mode will be discussed in the following chapters.

Table 25. External Interrupt Flag register (EXTINT - address 0x400F C140) bit description

Bit	Symbol	Description	Reset value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0

Table 25. External Interrupt Flag register (EXTINT - address 0x400F C140) bit description

Bit	Symbol	Description	Reset value
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

[1] Example: e.g. if the EINTx is selected to be low level sensitive and low level is present on corresponding pin, this bit can not be cleared; this bit can be cleared only when signal on the pin becomes high.

3.3.4.2 External Interrupt Mode register

The bits in this register select whether each EINT pin is level- or edge-sensitive. Only pins that are selected for the EINT function (see Section 7.3) and enabled in the appropriate NVIC register) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

Note: Software should only change a bit in this register when its interrupt is disabled in the NVIC (state readable in the ISERn/ICERn registers), and should write the corresponding 1 to EXTINT before enabling (initializing) or re-enabling the interrupt. An extraneous interrupt could be set by changing the mode and not having the EXTINT cleared.

Table 26. External Interrupt Mode register (EXTMODE - address 0x400F C148) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0		Level or edge sensitivity select for $\overline{\text{EINT0}}$.	0
		0	Level sensitive.	
		1	Edge sensitive.	
1	EXTMODE1		Level or edge sensitivity select for $\overline{\text{EINT1}}$.	0
		0	Level sensitive.	
		1	Edge sensitive.	
2	EXTMODE2		Level or edge sensitivity select for $\overline{\text{EINT2}}$.	0
		0	Level sensitive.	
		1	Edge sensitive.	
3	EXTMODE3		Level or edge sensitivity select for $\overline{\text{EINT3}}$.	0
		0	Level sensitive.	
		1	Edge sensitive.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.4.3 External Interrupt Polarity register

In level-sensitive mode, the bits in this register select whether the corresponding pin is high- or low-active. In edge-sensitive mode, they select whether the pin is rising- or falling-edge sensitive. Only pins that are selected for the EINT function Only pins that are selected for the EINT function (see Section 7.3) and enabled in the appropriate NVIC register) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

Note: Software should only change a bit in this register when its interrupt is disabled in the NVIC (state readable in the ISERn/ICERn registers), and should write the corresponding 1 to EXTINT before enabling (initializing) or re-enabling the interrupt. An extraneous interrupt could be set by changing the polarity and not having the EXTINT cleared.

Table 27. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTPOLAR0		External interrupt polarity for $\overline{\text{EINT0}}$.	0
		0	Low-active or falling-edge sensitive (depending on EXTMODE0).	
		1	High-active or rising-edge sensitive (depending on EXTMODE0).	
1	EXTPOLAR1		External interrupt polarity for $\overline{\text{EINT1}}$.	0
		0	Low-active or falling-edge sensitive (depending on EXTMODE1).	
		1	High-active or rising-edge sensitive (depending on EXTMODE1).	
2	EXTPOLAR2		External interrupt polarity for $\overline{\text{EINT2}}$.	0
		0	Low-active or falling-edge sensitive (depending on EXTMODE2).	
		1	High-active or rising-edge sensitive (depending on EXTMODE2).	
3	EXTPOLAR3		External interrupt polarity for $\overline{\text{EINT3}}$.	0
		0	Low-active or falling-edge sensitive (depending on EXTMODE3).	
		1	High-active or rising-edge sensitive (depending on EXTMODE3).	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

3.3.5 Device and Peripheral Reset

3.3.5.1 Reset Source Identification Register

This register contains one bit for each source of Reset. Writing a 1 to any of these bits clears the corresponding read-side bit to 0. The interactions among the four sources are described below.

Table 28. Reset Source Identification register (RSID - address 0x400F C180) bit description

Bit	Symbol	Description	Reset value
0	POR	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.	See description
1	EXTR	Assertion of the external $\overline{\text{RESET}}$ signal sets this bit. This bit is cleared only by software or POR.	See description
2	WDTR	This bit is set when the Watchdog Timer times out and the WDTRESET bit in the Watchdog Mode Register is 1. This bit is cleared only by software or POR.	See description
3	BODR	This bit is set when the $V_{DD(REG)(3V3)}$ voltage reaches a level below the BOD reset trip level (typically 1.85 V under nominal room temperature conditions). If the $V_{DD(REG)(3V3)}$ voltage dips from the normal operating range to below the BOD reset trip level and recovers, the BODR bit will be set to 1. If the $V_{DD(REG)(3V3)}$ 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 (nominally 1 V), the BODR bit is cleared. If the $V_{DD(REG)(3V3)}$ voltage rises continuously from below 1 V to a level above the BOD reset trip level, the BODR will be set to 1. This bit is cleared only by software or POR. **Note:** Only in the case where a reset occurs and the POR = 0, the BODR bit indicates if the $V_{DD(REG)(3V3)}$ voltage was below the BOD reset trip level or not.	See description
4	SYSRESET	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 only by software or POR.	See description
5	LOCKUP	This bit is set if the processor has been reset due to a lockup of the CPU (see Cortex-M4 documentation for details). The lockup state causes a chip reset. This bit is cleared only by software or POR.	See description
31:6	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.5.2 Reset control register 0

Many peripherals may be given a hardware reset using the RSTCON0 register. Some additional peripherals may be reset using the RSTCON1 register following.

Table 29. Reset control register 0 (RSTCON0 - address 0x400F C1CC) bit description

Bit	Symbol	Description	Reset value
0	RSTLCD	LCD controller reset control bit.	0
1	RSTTIM0	Timer/Counter 0 reset control bit.	0
2	RSTTIM1	Timer/Counter 1 reset control bit.	0
3	RSTUART0	UART0 reset control bit.	0
4	RSTUART1	UART1 reset control bit.	0
5	RSTPWM0	PWM0 reset control bit.	0
6	RSTPWM1	PWM1 reset control bit.	0
7	RSTI2C0	The I^2C0 interface reset control bit.	0
8	RSTUART4	UART4 reset control bit.	0
9	RSTRTC	RTC and Event Monitor/Recorder reset control bit. RTC reset is limited, see Table 630 “Register overview: Real-Time Clock (base address 0x4002 4000)” for details.	0
10	RSTSSP1	The SSP 1 interface reset control bit.	0
11	RSTEMC	External Memory Controller reset control bit.	0
12	RSTADC	A/D converter (ADC) reset control bit.	0
13	RSTCAN1	CAN Controller 1 reset control bit. Note: The CAN acceptance filter may be reset by a separate bit in the RSTCON1 register.	0
14	RSTCAN2	CAN Controller 2 reset control bit. Note: The CAN acceptance filter may be reset by a separate bit in the RSTCON1 register.	0
15	RSTGPIO	Reset control bit for GPIO, and GPIO interrupts. Note: IOCON may be reset by a separate bit in the RSTCON1 register.	0
16	RSTSPIFI	SPI Flash Interface reset control bit.	0
17	RSTMCPWM	Motor Control PWM reset control bit.	0
18	RSTQEI	Quadrature Encoder Interface reset control bit.	0
19	RSTI2C1	The I^2C1 interface reset control bit.	0
20	RSTSSP2	The SSP2 interface reset control bit.	0
21	RSTSSP0	The SSP0 interface reset control bit.	0
22	RSTTIM2	Timer 2 reset control bit.	0
23	RSTTIM3	Timer 3 reset control bit.	0
24	RSTUART2	UART 2 reset control bit.	0
25	RSTUART3	UART 3 reset control bit.	0
26	RSTI2C2	I^2C interface 2 reset control bit.	0
27	RSTI2S	I^2S interface reset control bit.	0
28	RSTSDC	SD Card interface reset control bit.	0
29	RSTGPDMA	GPDMA function reset control bit.	0
30	RSTENET	Ethernet block reset control bit.	0
31	RSTUSB	USB interface reset control bit.	0

3.3.5.3 Reset control register 1

Some additional peripherals may be given a hardware reset using the RSTCON1 register, as shown in Table 30 below.

Table 30. Reset control register 1 (RSTCON1 - address 0x400F C1D0) bit description

Bit	Symbol	Description	Reset value
0	RSTIOCON	Reset control bit for the IOCON registers.	0
1	RSTDAC	D/A converter (DAC) reset control bit.	0
2	RSTCANACC	CAN acceptance filter reset control bit.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.6 EMC delay control and calibration

3.3.6.1 EMC Delay Control register

The EMCDLYCTL register controls on-chip programmable delays that can b used to fine tune timing to external SDRAM memories. Delays can be configured in increments of approximately 250 picoseconds up to a maximum of roughly 7.75 ns. See Section 9.5.6 for an overview of the programmable delays. Figure 5 shows the detailed connections of the programmable delays. Table 31 shows the bit assignments in EMCDLYCTL.

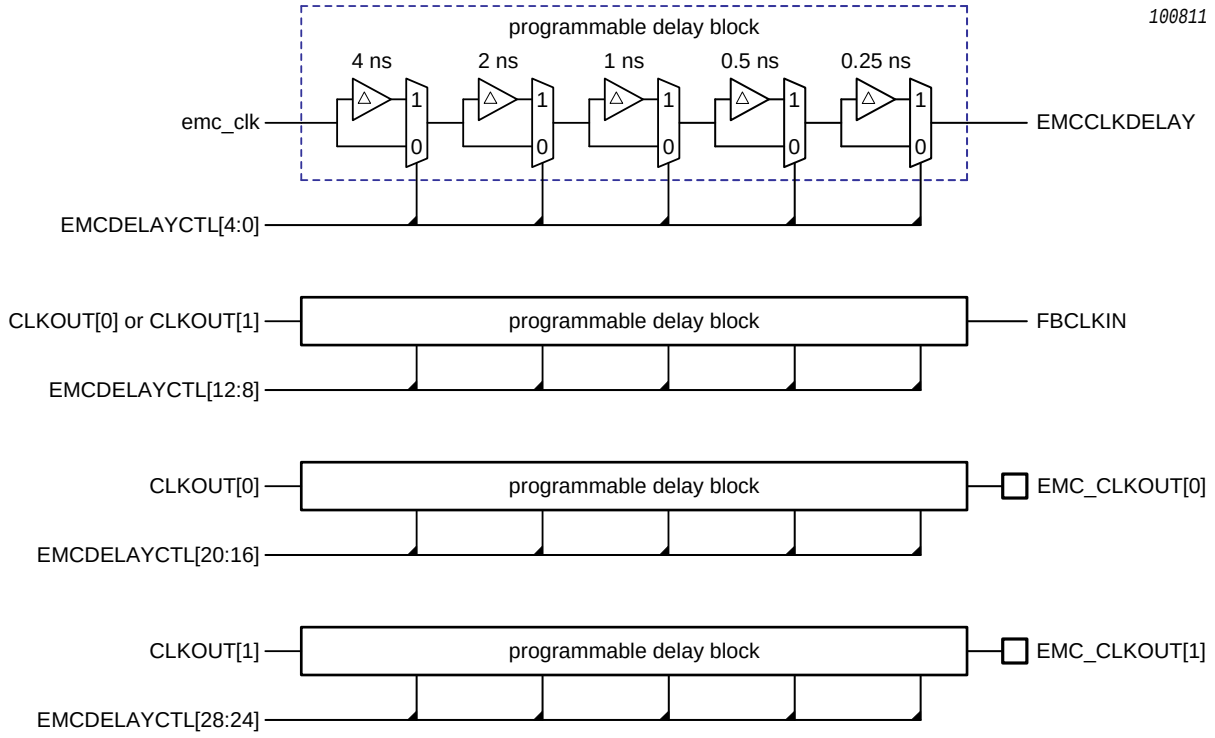


Fig 5. EMC programmable delays

Table 31. Delay Control register (EMCDLYCTL - 0x400F C1DC) bit description

Bit	Symbol	Description	Reset Value
4:0	CMDDLY	Programmable delay value for EMC outputs in command delayed mode. See Section 9.13.6. The delay amount is roughly (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.	NA
12:8	FBCLKDLY	Programmable delay value for the feedback clock that controls input data sampling. See Section 9.5.3. The delay amount is roughly (FBCLKDLY+1) * 250 picoseconds.	0x02
15:13	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 31. Delay Control register (EMCDLYCTL - 0x400F C1DC) bit description

Bit	Symbol	Description	Reset Value
20:16	CLKOUT0DLY	Programmable delay value for the CLKOUT0 output. This would typically be used in clock delayed mode. See Section 9.13.6 The delay amount is roughly (CLKOUT0DLY+1) * 250 picoseconds. Delaying the clock output changes dynamic characteristics defined in the device data sheet.	0
23:21	-	Reserved. Read value is undefined, only zero should be written.	NA
28:24	CLKOUT1DLY	Programmable delay value for the CLKOUT1 output. This would typically be used in clock delayed mode. See Section 9.13.6 The delay amount is roughly (CLKOUT1DLY+1) * 250 picoseconds.	0
31:29	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.6.2 EMC Calibration 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. Figure 6 below shows the delay calibration circuit. Table 32 shows the bit assignments in EMCCAL.

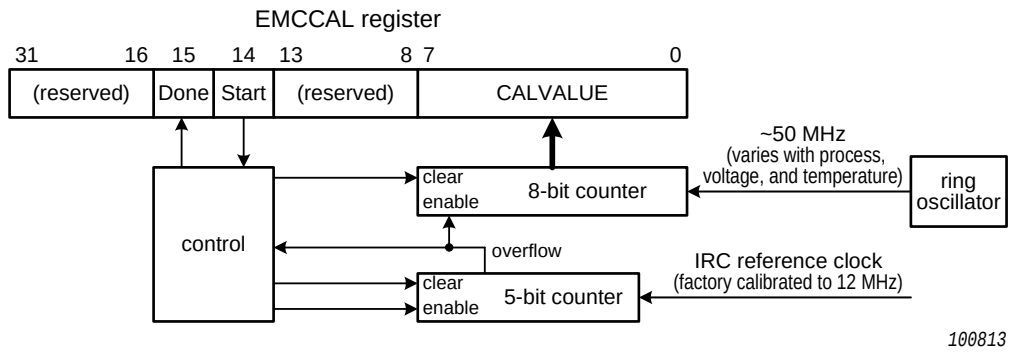


Fig 6. EMC delay calibration

Table 32. EMC Calibration register (EMCCAL - 0x400F C1E0) 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 IRC oscillator. 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.	NA
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. Read value is undefined, only zero should be written.	NA

Procedure for calibrating programmable delays

1. Write 1 to the START bit of the EMCCAL register.
2. Wait until the DONE bit of the same register becomes 1. Other operations can be done during this time, the calibration requires 32 clocks of the 12 MHz IRC clock, or about 2.7 microseconds.
3. Read the calibration value from the bottom 8 bits of the EMCCAL register. A typical value at room temperature is 0x86.
4. Adjust one or more programmable delays if needed based on the calibration result.

The calibration procedure should typically be repeated periodically, depending on how rapidly ambient conditions may change in the application environment.

3.3.7 Miscellaneous system control registers

3.3.7.1 System Controls and Status register

The SCS register contains special control and status bits related to various aspects of chip operation. These functions are described in Table 33.

Several of these bits apply to the main oscillator. Since chip operation always begins using the Internal RC Oscillator, and the main oscillator may not be used at all in some applications, it will only be started by software request. This is accomplished by setting the OSCEN bit in the SCS register, as described in Table 3-13. The main oscillator provides a status flag (the OSCSTAT bit in the SCS register) so that software can determine when the oscillator is running and stable. At that point, software can control switching to the main oscillator as a clock source. Prior to starting the main oscillator, a frequency range must be selected by configuring the OSCRANGE bit in the SCS register.

Table 33. System Controls and Status register (SCS - address 0x400F C1A0) bit description

Bit	Function	Value	Description	Access	Reset value
0	EMCSC		EMC Shift Control. Controls how addresses are output on the EMC address pins for static memories. Also see Section 9.9 in the EMC chapter.	R/W	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 2 places such that bit 2 is the LSB. In this mode, address bit 0 for the this device is connected to address bit 0 of the memory device, thus 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. This mode matches the operation of LPC23xx and LPC24xx devices.		
1	EMCRD		EMC Reset Disable[1]. External Memory Controller Reset Disable. Also see Section 9.8 in the EMC chapter.	R/W	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. Also see Section 9.10 in the EMC chapter.	R/W	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. These unrequested accesses can cause issues with some I/O devices.		

Table 33. System Controls and Status register (SCS - address 0x400F C1A0) bit description

Bit	Function	Value	Description	Access	Reset value
3	MCIPWRAL		MCIPWR Active Level[1]. Selects the active level of the SD card interface signal SD_PWR.	R/W	1
		0	SD_PWR is active low (inverted output of the SD Card interface block).		
		1	SD_PWR is active high (follows the output of the SD Card interface block).		
4	OSCRS		Main oscillator range select.	R/W	0
		0	The frequency range of the main oscillator is 1 MHz to 20 MHz.		
		1	The frequency range of the main oscillator is 15 MHz to 25 MHz.		
5	OSCEN		Main oscillator enable.	R/W	0
		0	The main oscillator is disabled.		
		1	The main oscillator is enabled, and will start up if the correct external circuitry is connected to the XTAL1 and XTAL2 pins.		
6	OSCSTAT		Main oscillator status.	RO	0
		0	The main oscillator is not ready to be used as a clock source.		
		1	The main oscillator is ready to be used as a clock source. The main oscillator must be enabled via the OSCEN bit.		
31:7	-		Reserved. Read value is undefined, only zero should be written.	-	NA

[1] 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.

3.3.7.2 IRC control register

This register is used to trim the on-chip 12 MHz oscillator. The trim value is factory-preset and written by the boot code on start-up.

Table 34. IRC control register (IRCCTRL - address 0x400F C1A4) bit description

Bit	Function	Description	Access	Reset value
7:0	TRIM	Trim value.	R/W	0x1000 0000, then flash will reprogram
31:8	-	Reserved. Read value is undefined, only zero should be written.	-	NA

3.3.7.3 LCD Configuration register

The LCD_CFG register controls the prescaling of the clock used for LCD data generation.

The contents of the LCD_CFG register are described in Table 35.

Table 35. LCD Configuration register (LCD_CFG, address - 0x400F C1B8) bit description

Bits	Symbol	Description	Reset value
4:0	CLKDIV	LCD panel clock prescaler selection. The value in the this register plus 1 is used to divide the selected input clock (see the CLKSEL bit in the LCD_POL register), to produce the panel clock.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

3.3.7.4 CAN Sleep Clear register

This register provides the current sleep state of the two CAN channels and provides a means to restore the clocks to that channel following wake-up. Refer to Section 20.8.2 “Sleep mode” for more information on the CAN sleep feature.

Table 36. CAN Sleep Clear register (CANSLEEPCLR - address 0x400F C110) bit description

Bit	Symbol	Function	Reset Value
0	-	Reserved. Read value is undefined, only zero should be written.	NA
1	CAN1SLEEP	Sleep status and control for CAN channel 1. Read: when 1, indicates that CAN channel 1 is in the sleep mode. Write: writing a 1 causes clocks to be restored to CAN channel 1.	0
2	CAN2SLEEP	Sleep status and control for CAN channel 2. Read: when 1, indicates that CAN channel 2 is in the sleep mode. Write: writing a 1 causes clocks to be restored to CAN channel 2.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.7.5 CAN Wake-up Flags register

This register provides the wake-up status for the two CAN channels and allows clearing wake-up events. Refer to Section 20.8.2 “Sleep mode” for more information on the CAN sleep feature.

Table 37. CAN Wake-up Flags register (CANWAKEFLAGS - address 0x400F C114) bit description

Bit	Symbol	Function	Reset Value
0	-	Reserved. Read value is undefined, only zero should be written.	NA
1	CAN1WAKE	Wake-up status for CAN channel 1. Read: when 1, indicates that a falling edge has occurred on the receive data line of CAN channel 1. Write: writing a 1 clears this bit.	0
2	CAN2WAKE	Wake-up status for CAN channel 2. Read: when 1, indicates that a falling edge has occurred on the receive data line of CAN channel 2. Write: writing a 1 clears this bit.	0
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

3.3.7.6 USB Interrupt Status Register

The USB OTG controller has seven interrupt lines. Only the first three interrupts (USB_INT_REQ_LP, USB_INT_REQ_HP, and USB_INT_REQ_HP) and the USB_NEED_CLK signal are used for the device controller. The interrupt lines are ORed together to a single channel of the vectored interrupt controller. This register allows software to determine their status with a single read operation.

Table 38. USB Interrupt Status register - (USBINTST - address 0x400F C1C0) bit description

Bit	Symbol	Description	Reset Value
0	USB_INT_REQ_LP	Low priority interrupt line status. This bit is read-only.	0
1	USB_INT_REQ_HP	High priority interrupt line status. This bit is read-only.	0
2	USB_INT_REQ_DMA	DMA interrupt line status. This bit is read-only.	0
3	USB_HOST_INT	USB host interrupt line status. This bit is read-only.	0
4	USB_ATX_INT	External ATX interrupt line status. This bit is read-only.	0
5	USB_OTG_INT	OTG interrupt line status. This bit is read-only.	0
6	USB_I2C_INT	I^2C module interrupt line status. This bit is read-only.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	USB_NEED_CLK	USB need clock indicator. This bit is read-only. This bit is set to 1 when USB activity or a change of state on the USB data pins is detected, and it indicates that a PLL supplied clock of 48 MHz is needed. Once USB_NEED_CLK becomes one, it resets to zero 5 ms after the last packet has been received/sent, or 2 ms after the Suspend Change (SUS_CH) interrupt has occurred. A change of this bit from 0 to 1 can wake up the microcontroller if activity on the USB bus is selected to wake up the part from the Power-down mode (see Section 3.12.8 “Wake-up from Reduced Power Modes” for details). Also see Section 3.10.3 “PLLs and Power-down mode” and Section 3.3.2.2 “Power Control for Peripherals registers” for considerations about the PLL and invoking the Power-down mode. This bit is read-only.	1
30:9	-	Reserved. Read value is undefined, only zero should be written.	NA
31	EN_USB_INTS	Enable all USB interrupts. When this bit is cleared, the NVIC does not see the ORed output of the USB interrupt lines.	1

3.3.7.7 DMA Request Select register

DMACReqSel is a read/write register that allows selecting between potential DMA requests for DMA inputs 0 through 7 and 10 through 15. Table 39 shows the bit assignments of the DMACReqSel Register.

Table 39. DMA Request Select register bit description

Bit	Name	Description	Reset value
0	DMASEL00	Selects the DMA request for GPDMA input 0: 0 - (unused) 1 - Timer 0 match 0 is selected.	0
1	DMASEL01	Selects the DMA request for GPDMA input 1: 0 - SD card interface is selected. 1 - Timer 0 match 1 is selected.	0
2	DMASEL02	Selects the DMA request for GPDMA input 2: 0 - SSP0 transmit is selected. 1 - Timer 1 match 0 is selected.	0
3	DMASEL03	Selects the DMA request for GPDMA input 3: 0 - SSP0 receive is selected. 1 - Timer 1 match 1 is selected.	0
4	DMASEL04	Selects the DMA request for GPDMA input 4: 0 - SSP1 transmit is selected. 1 - Timer 2 match 0 is selected.	0
5	DMASEL05	Selects the DMA request for GPDMA input 5: 0 - SSP1 receive is selected. 1 - Timer 2 match 1 is selected.	0
6	DMASEL06	Selects the DMA request for GPDMA input 6: 0 - SSP2 transmit is selected. 1 - I^2S channel 0 is selected.	0
7	DMASEL07	Selects the DMA request for GPDMA input 7: 0 - SSP2 receive is selected. 1 - I^2S channel 1 is selected.	0
9:8	-	Reserved. Read value is undefined, only zero should be written.	-
10	DMASEL10	Selects the DMA request for GPDMA input 10: 0 - UART0 transmit is selected. 1 - UART3 transmit is selected.	0
11	DMASEL11	Selects the DMA request for GPDMA input 11: 0 - UART0 receive is selected. 1 - UART3 receive is selected.	0
12	DMASEL12	Selects the DMA request for GPDMA input 12: 0 - UART1 transmit is selected. 1 - UART4 transmit is selected.	0
13	DMASEL13	Selects the DMA request for GPDMA input 13: 0 - UART1 receive is selected. 1 - UART4 receive is selected.	0

Table 39. DMA Request Select register bit description *…continued*

Bit	Name	Description	Reset value
14	DMASEL14	Selects the DMA request for GPDMA input 14: 0 - UART2 transmit is selected. 1 - Timer 3 match 0 is selected.	0
15	DMASEL15	Selects the DMA request for GPDMA input 15: 0 - UART2 receive is selected. 1 - Timer 3 match 1 is selected.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

3.3.7.7.1 Timer DMA requests

Timer DMA requests are generated by the timer when the timer value matches the related Match register (see Section 24.6.12). If the DMA controller is configured so that a timer DMA request is selected as an input to a DMA channel, and the DMA channel is enabled, the DMA controller will act on that request.

3.3.7.8 Clock Output Configuration register

The CLKOUTCFG register controls the selection of the internal clock that appears on the CLKOUT pin and allows dividing the clock by an integer value up to 16. The divider can be used to produce a system clock that is related to one of the on-chip clocks. For most clock sources, the division may be by 1. When the CPU clock is selected and is higher than approximately 50 MHz, the output must be divided in order to bring the frequency within the ability of the pin to switch with reasonable logic levels. If a clock is selected that is not running, there will be no signal on CLKOUT.

Note: The CLKOUT multiplexer is designed to switch cleanly, without glitches, between the possible clock sources. The divider is also designed to allow changing the divide value without glitches.

Table 40. Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8) bit description

Bit	Symbol	Description	Reset value
3:0	CLKOUTSEL	Selects the clock source for the CLKOUT function. 0x0 = Selects the CPU clock as the CLKOUT source. 0x1 = Selects the main oscillator as the CLKOUT source. 0x2 = Selects the Internal RC oscillator as the CLKOUT source. 0x3 = Selects the USB clock as the CLKOUT source. 0x4 = Selects the RTC oscillator as the CLKOUT source. 0x5 = Selects the SPIFI clock as the CLKOUT source. 0x6 = Selects the Watchdog oscillator as the CLKOUT source. Other settings are reserved. Do not use.	0
7:4	CLKOUTDIV	Integer value to divide the output clock by, minus one. 0x0 = Clock is divided by 1. 0x1 = Clock is divided by 2. 0x2 = Clock is divided by 3. ... 0xF = Clock is divided by 16.	0
8	CLKOUT_EN	CLKOUT enable control, allows switching the CLKOUT source without glitches. Clear to stop CLKOUT on the next falling edge. Set to enable CLKOUT.	0
9	CLKOUT_ACT	CLKOUT activity indication. Reads as 1 when CLKOUT is enabled. Read as 0 when CLKOUT has been disabled via the CLKOUT_EN bit and the clock has completed being stopped.	0
31:10	-	Reserved. Read value is undefined, only zero should be written.	NA

3.4 Chip reset

Reset has 6 sources: the $\overline{\text{RESET}}$ pin, Watchdog Reset, Power On Reset (POR), Brown Out Detect (BOD), system reset, and lockup.

The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin. Assertion of chip Reset by any source, once the operating voltage attains a usable level, starts the wake-up timer (see description in Section 3.13 “Wake-up timer” in this chapter), causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, a fixed number of clocks have passed, and the flash controller has completed its initialization. The reset logic is shown in the following block diagram (see Figure 7).

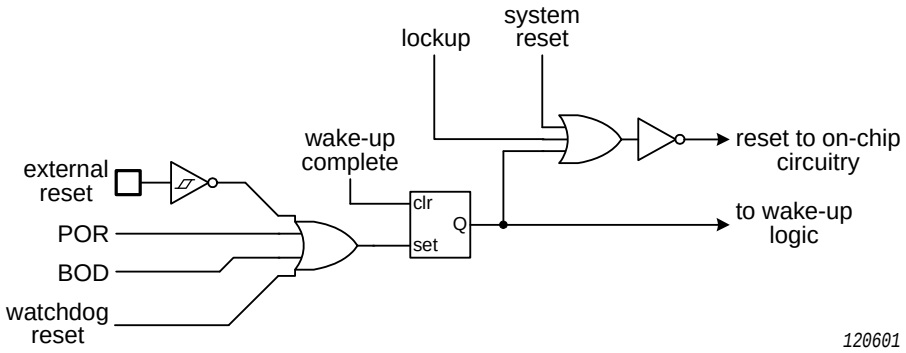


Fig 7. Reset block diagram

On the assertion of a reset source external to the CPU (POR, BOD reset, External reset, and Watchdog reset), the IRC starts up. After the IRC-start-up time (maximum of 60 μs on power-up) and after the IRC provides a stable clock output, the reset signal is latched and synchronized on the IRC clock. Then the following two sequences start simultaneously:

1. The 2-bit IRC wake-up timer starts counting when the synchronized reset is de-asserted. The boot code in the ROM starts when the 2-bit IRC wake-up timer times out. The boot code performs the boot tasks and may jump to the flash. If the flash is not ready to access, the Flash Accelerator will insert wait cycles until the flash is ready.
2. The flash wake-up timer (9-bit) starts counting when the synchronized reset is de-asserted. The flash wakeup-timer generates the 100 μs flash start-up time. Once it times out, the flash initialization sequence is started, which takes about 250 cycles. When it’s done, the Flash Accelerator will be granted access 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.

Figure 8 shows an example of the relationship between the $\overline{\text{RESET}}$, the IRC, and the processor status when the device starts up after reset. See Section 3.8.2 “Main oscillator” for start-up of the main oscillator if selected by the user code.

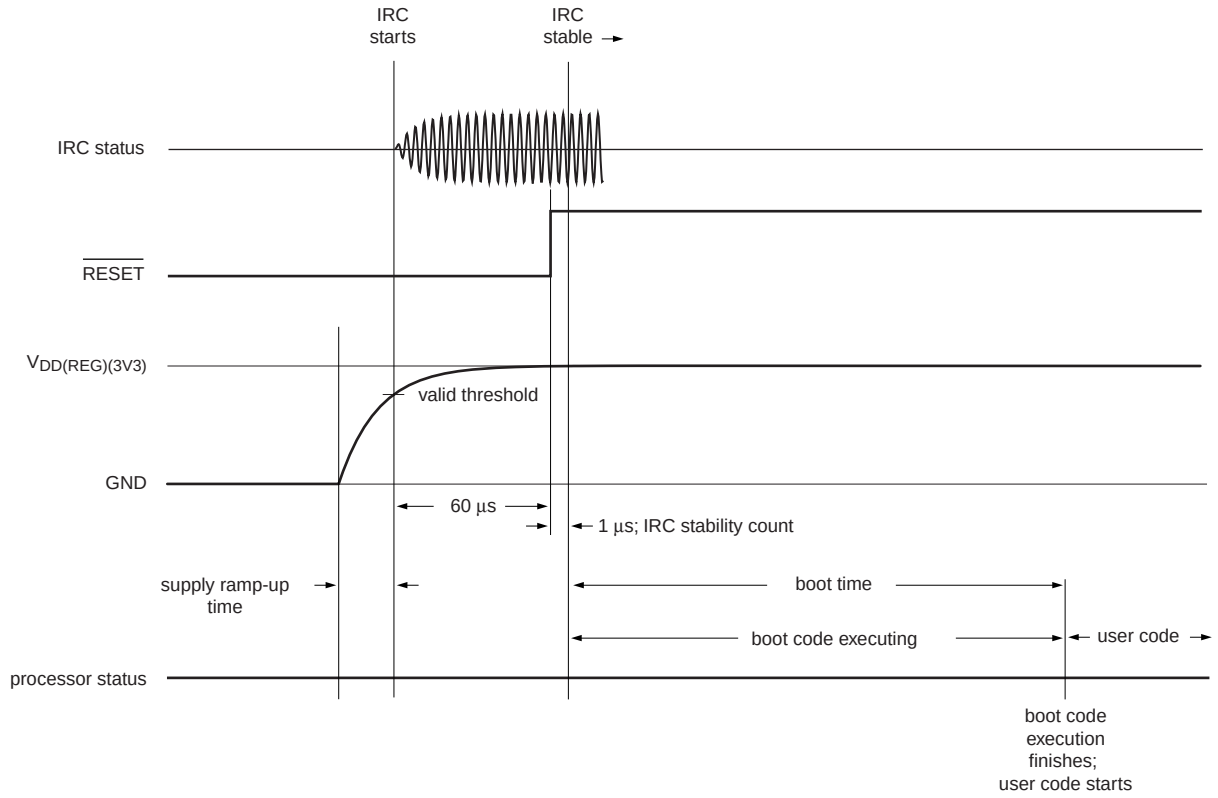


Fig 8. Example of start-up after reset

3.5 Peripheral reset control

Most peripheral functions can have a hardware reset initiated by software by setting appropriate bits in the RSTCON0 and RSTCON1 registers. Software must clear the RSTCON register after this in order to allow the peripheral to function. A peripheral remains in a hardware reset state as long as the corresponding bit in RSTCON = 1.

3.6 Brown-out detection

A Brown-Out Detector (BOD) is included that provides 2-stage monitoring of the voltage on the $V_{DD(REG)(3V3)}$ pins. If this voltage falls below the BOD interrupt trip level (typically 2.2 V under nominal room temperature conditions), the BOD asserts an interrupt signal to the NVIC. This signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC in order to cause a CPU interrupt; if not, software can monitor the signal by reading the Raw Interrupt Status Register.

The second stage of low-voltage detection asserts Reset to inactivate the device when the voltage on the $V_{DD(REG)(3V3)}$ pins falls below the BOD reset trip level (typically 1.85 V under nominal room temperature conditions). This Reset prevents alteration of the flash as operation of the various elements of the chip would otherwise become unreliable due to low voltage. The BOD circuit maintains this reset down below 1 V, at which point the Power-On Reset circuitry maintains the overall Reset.

Both the BOD reset interrupt level and the BOD reset trip level thresholds include some hysteresis. In normal operation, this hysteresis allows the BOD reset interrupt level detection to reliably interrupt, or a regularly-executed event loop to sense the condition.

But when Brown-Out Detection is enabled to bring the device out of Power-down mode (which is itself not a guaranteed operation -- see Section 3.3.2.1 “Power Mode Control register”), the supply voltage may recover from a transient before the wake-up timer has completed its delay. In this case, the net result of the transient BOD is that the part wakes up and continues operation after the instructions that set Power-down mode, without any interrupt occurring and with the BOD bit in the RSID being 0. Since all other wake-up conditions have latching flags (see Section 3.3.4.1 “External Interrupt flag register” and Section 29.6.2), a wake-up of this type, without any apparent cause, can be assumed to be a Brown-Out that was too short to be fully captured.

3.7 External interrupt inputs

Four External Interrupt Inputs are included as selectable pin functions. The logic of an individual external interrupt is represented in Figure 9. In addition, external interrupts have the ability to wake up the CPU from Power-down mode. Refer to Section 3.12.8 “Wake-up from Reduced Power Modes” for details.

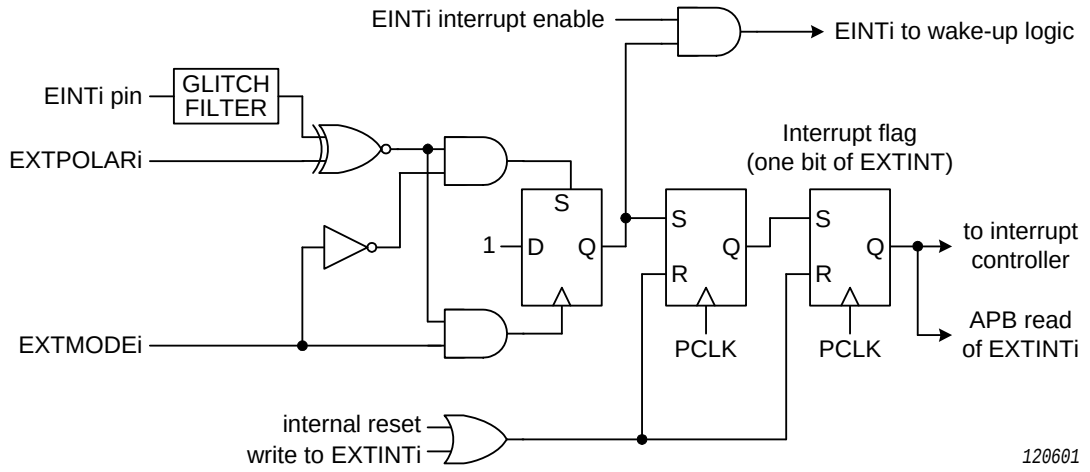


Fig 9. External interrupt logic

3.7.1 Register description

The external interrupt function has four registers associated with it. The EXTINT register contains the interrupt flags. The EXTMODE and EXTPOLAR registers specify the level and edge sensitivity parameters.

Table 41. External Interrupt registers

Name	Description	Access	Reset value[1]	Address
EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, EINT2 and EINT3. See Table 25.	R/W	0x00	0x400F C140
EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level-sensitive. See Table 26.	R/W	0x00	0x400F C148
EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt. See Table 27.	R/W	0x00	0x400F C14C

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

3.8 Oscillators

Three independent oscillators are included. These are the Main Oscillator, the Internal RC Oscillator, and the RTC oscillator. Each oscillator can be used for more than one purpose as required in a particular application. This can be seen in Figure 4.

Following Reset, the device will operate from the Internal RC Oscillator until switched by software. This allows systems to operate without any external crystal, and allows the boot loader code to operate at a known frequency.

3.8.1 Internal RC oscillator

The Internal RC Oscillator (IRC) may be used as the clock that drives PLL0 and subsequently the CPU. The precision of the IRC does not allow for use of the USB interface, which requires a much more precise time base in order to comply with the USB specification (only the main oscillator can meet that specification). Also, the IRC should not be used with the CAN1/2 block if the CAN baud rate is higher than 100 kbit/s.The IRC frequency is 12 MHz, factory trimmed to within ±1% accuracy.

Upon power-up or any chip reset, the IRC is used as the clock source. Software may later switch to one of the other available clock sources.

3.8.2 Main oscillator

The main oscillator can be used as the clock source for the CPU, with or without using PLL0. The main oscillator operates at frequencies of 1 MHz to 25 MHz. This frequency can be boosted to a higher frequency, up to the maximum CPU operating frequency, by the Main PLL (PLL0). The oscillator output is called OSC_CLK. The clock selected as the PLL0 input is PLLCLKIN and the ARM processor clock frequency is referred to as CCLK for purposes of rate equations, etc. elsewhere in this document. The frequencies of PLLCLKIN and CCLK are the same value unless the PLL0 is active and connected. Refer to Section 3.10 for details.

The on-board oscillator can operate in one of two modes: slave mode and oscillation mode.

In slave mode the input clock signal should be coupled by means of a capacitor of 100 pF (C_C in Figure 10, drawing a), with an amplitude between 200 mVrms and 1000 mVrms. This corresponds to a square wave signal with a signal swing of between 280 mV and 1.4 V. The XTAL2 pin in this configuration can be left unconnected.

External components and models used in oscillation mode are shown in Figure 10, drawings b and c, and in Table 42 and Table 43. Since the feedback resistance is integrated on chip, only a crystal and the capacitances C_{X1} and C_{X2} need to be connected externally in case of fundamental mode oscillation (the fundamental frequency is represented by L, C_L and R_S). Capacitance C_P in Figure 10, drawing c, represents the parallel package capacitance and should not be larger than 7 pF. Parameters F_C, C_L, R_S and C_P are supplied by the crystal manufacturer.

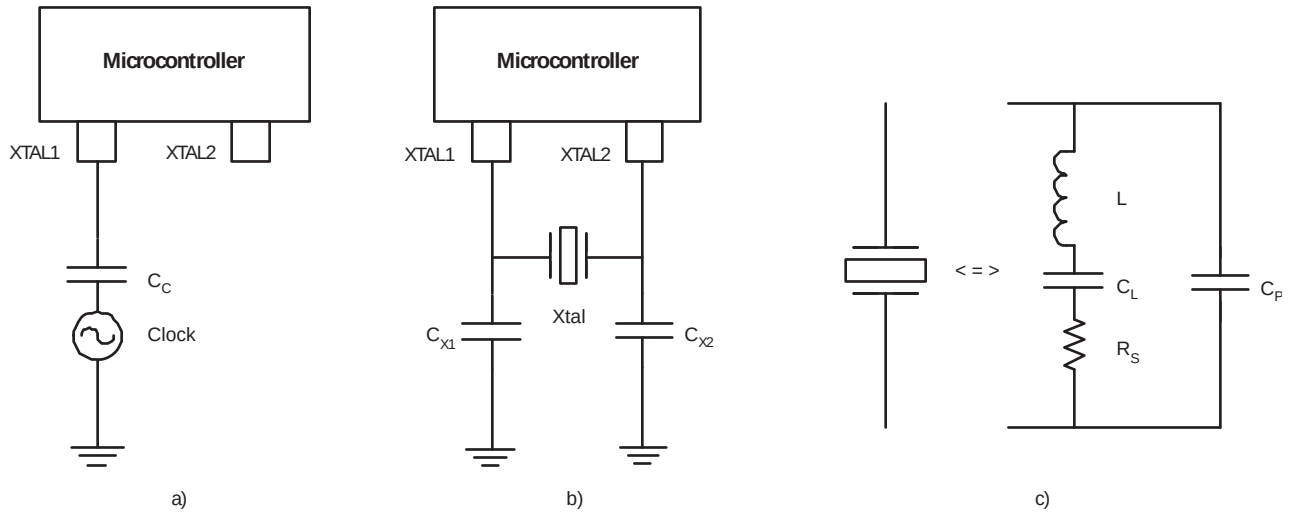


Fig 10. Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for $C_{X1}/_{X2}$ evaluation

Table 42. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) low frequency mode (OSCRANGE = 0, see Table 33)

Fundamental oscillation frequency F_{OSC}	Crystal load capacitance C_L	Maximum crystal series resistance R_S	External load capacitors C_{X1}, C_{X2}
1 MHz - 5 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 300 Ω	39 pF, 39 pF
	30 pF	< 300 Ω	57 pF, 57 pF
5 MHz - 10 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 200 Ω	39 pF, 39 pF
	30 pF	< 100 Ω	57 pF, 57 pF
10 MHz - 15 MHz	10 pF	< 160 Ω	18 pF, 18 pF
	20 pF	< 60 Ω	39 pF, 39 pF
15 MHz - 20 MHz	10 pF	< 80 Ω	18 pF, 18 pF

Table 43. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) high frequency mode (OSCRANGE = 1, see Table 33)

Fundamental oscillation frequency F_{OSC}	Crystal load capacitance C_L	Maximum crystal series resistance R_S	External load capacitors C_{X1}, C_{X2}
15 MHz - 20 MHz	10 pF	< 180 Ω	18 pF, 18 pF
	20 pF	< 100 Ω	39 pF, 39 pF
20 MHz - 25 MHz	10 pF	< 160 Ω	18 pF, 18 pF
	20 pF	< 80 Ω	39 pF, 39 pF

3.8.2.1 Main oscillator startup

Since chip operation always begins using the Internal RC Oscillator, and the main oscillator may not be used at all in some applications, it will only be started by software request. This is accomplished by setting the OSCEN bit in the SCS register, as described in Table 33. The main oscillator provides a status flag (the OSCSTAT bit in the SCS register) so that software can determine when the oscillator is running and stable. At that point, software can control switching to the main oscillator as a clock source. Prior to starting the main oscillator, a frequency range must be selected by configuring the OSCRANGE bit in the SCS register.

3.8.3 RTC oscillator

The RTC oscillator provides a 1 Hz clock to the RTC and a 32 kHz clock output that can be output on the CLKOUT pin in order to allow trimming the RTC oscillator without interference from a probe.

3.8.4 Watchdog oscillator

The Watchdog Timer has a dedicated oscillator that provides a 500 kHz clock to the Watchdog Timer that is always running if the Watchdog Timer is enabled. The Watchdog oscillator clock can be output on the CLKOUT pin in order to allow observe its frequency.

In order to allow Watchdog Timer operation with minimum power consumption, which can be important in reduced power modes, the Watchdog oscillator frequency is not tightly controlled. The Watchdog oscillator frequency will vary over temperature and power supply within a particular part, and may vary by processing across different parts. This variation should be taken into account when determining Watchdog reload values.

Within a particular part, temperature and power supply variations can produce up to a ±17% frequency variation. Frequency variation between devices under the same operating conditions can be up to ±30%.

3.9 Clock source selection multiplexer

Two clock sources may be chosen to drive the system clock (sysclk) and PLL0. These are the Internal RC oscillator and the main oscillator.

The clock source selection should only be changed safely when PLL0 is not connected. For a detailed description of how to change the clock source in a system using PLL0 see Section 3.10.6 “PLL configuration sequence”.

3.10 PLL0 and PLL1 (Phase Locked Loops)

PLL0 (also called the Main PLL) and PLL1 (also called the Alt PLL) are functionally identical, but have somewhat different input possibilities and output connections. These possibilities are shown in Figure 4. The Main PLL can receive its input from either the IRC or the Main Oscillator, and can potentially be used to provide the clocks to nearly everything on the device. The Alt PLL receives its input only from the main oscillator and is intended to be used as an alternate source of clocking to the USB and the SPIFI. This peripheral has timing needs that may not always be filled by the Main PLL.

Both PLLs are disabled and powered off on reset. If the Alt PLL is left disabled, the USB and SPIFI clocks can be supplied by PLL0 if everything is set up to provide 48 MHz to the USB clock and the desired SPIFI clock through that route. The source for each clock must be selected via the CLKSEL registers (see Section 3.11), and can be further reduced by clock dividers as needed.

PLL activation is controlled via the PLLCON registers. PLL multiplier and divider values are controlled by the PLLCFG registers. The PLLCFG registers are protected in order to prevent accidental deactivation of PLLs or accidental alteration PLL operating parameters. The protection is accomplished by a feed sequence similar to that of the Watchdog Timer. Details are provided in the descriptions of the PLLFEED registers.

PLL0 accepts an input clock frequency from either the IRC or the main oscillator. If only the Main PLL is used, then its output frequency must be an integer multiple of all other clocks needed in the system. PLL1 takes its input only from the main oscillator, requiring an external crystal in the range of 10 to 25 MHz. In each PLL, the Current Controlled Oscillator (CCO) operates in the range of 156 MHz to 320 MHz, so there are additional dividers to bring the output down to the desired frequencies. The minimum output divider value is 2, insuring that the output of the PLLs have a 50% duty cycle. Figure 11 shows a block diagram of PLL internal connections.

If the USB is used, the possibilities for the CPU clock and other clocks will be limited by the requirements that the frequency be precise and very low jitter, and that the PLL0 output must be a multiple of 48 MHz. Even multiples of 48 MHz that are within the operating range of the PLL F_{CCO} are 192 and 288 MHz. Also, only the main oscillator in conjunction with the PLL can meet the precision and jitter specifications for USB. It is due to these limitations that the Alt PLL is provided.

The Alt PLL accepts an input clock frequency from the main oscillator in the range of 10 MHz to 25 MHz only. When used as the USB clock, the input frequency is multiplied up to a multiple of 48 MHz (192 or 288 MHz as described above). The Alt PLL can also provide the clock to the SPIFI through a separate divider, if needed.

3.10.1 PLL and startup/boot code interaction

When there is no valid user code (determined by the checksum word) in the user flash or the ISP enable pin (P2[10]) is pulled low on startup, the ISP mode will be entered and the boot code will setup the Main PLL with the IRC. Therefore it can not be assumed that the Main PLL is disabled when the user opens a debug session to debug the application code. The user startup code must follow the steps described in this chapter to disconnect the Main PLL.

3.10.2 PLL register description

The PLLs are controlled by the registers shown in Table 44. More detailed descriptions follow.

Warning: Improper setting of PLL values may result in incorrect operation of the USB subsystem!

Table 44. PLL1 registers

Generic Name	Description	Access	Reset value[1]	PLLn Register Name and Address	Table
PLLCON	PLL Control Register. Holding register for updating PLL control bits. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W	0	PLL0CON - 0x400F C080 PLL1CON - 0x400F C0A0	10
PLLCFG	PLL Configuration Register. Holding register for updating PLL configuration values. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W	0	PLL0CFG - 0x400F C084 PLL1CFG - 0x400F C0A4	11
PLLSTAT	PLL Status Register. Read-back register for PLL control and configuration information. If PLLCON or PLLCFG have been written to, but a PLL feed sequence has not yet occurred, they will not reflect the current PLL state. Reading this register provides the actual values controlling PLL, as well as PLL status.	RO	0	PLL0STAT - 0x400F C088 PLL1STAT - 0x400F C0A8	12
PLLFEED	PLL Feed Register. This register enables loading of PLL control and configuration information from the PLLCON and PLLCFG registers into the shadow registers that actually affect PLL operation.	WO	NA	PLL0FEED - 0x400F C08C PLL1FEED - 0x400F C0AC	13

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

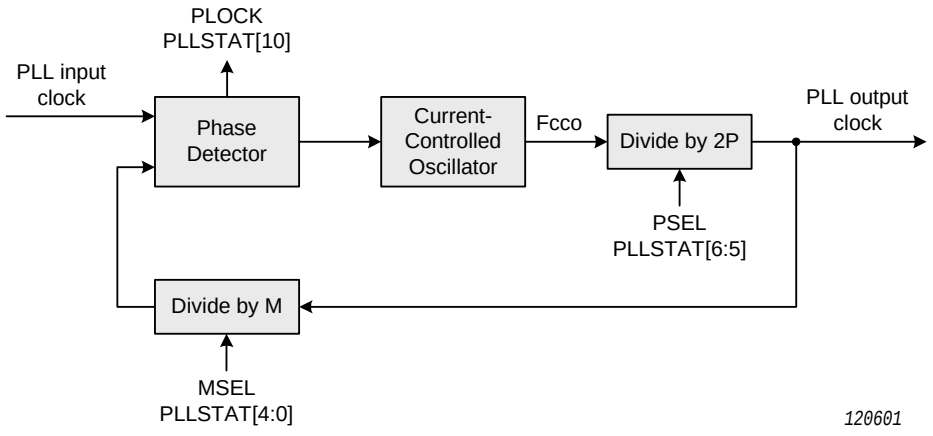


Fig 11. PLL0 and PLL1 block diagram

3.10.3 PLLs and Power-down mode

Power-down mode automatically turns off and disconnects activated PLLs, while subsequent wake-up from Power-down mode does not automatically restore PLL settings. This must be done in software. Typically, a routine to activate a PLL, wait for lock, and then select the PLL can be called at the beginning of any interrupt service routine that might be called due to the wake-up.

If activity on the USB data lines is not selected to wake the microcontroller from Power-down mode (see Section 3.12.8 for details of wake up from reduced modes), both the Main PLL (PLL0) and the Alt PLL (PLL1) will be automatically be turned off and disconnected when Power-down mode is invoked, as described above. However, if the USB activity interrupt is enabled and USB_NEED_CLK = 1 (see Table 254 for a description of USB_NEED_CLK), it is not possible to go into Power-down mode and any attempt to set the PD bit will fail, leaving the PLLs in the current state.

3.10.4 PLL frequency calculation

Equations for both the Main and Alt PLLs use the following parameters:

Table 45. Elements determining PLL frequency

Element	Description
pll_in_clk	the frequency of the input to the PLL
F_{CCO}	the frequency of the PLL current controlled oscillator
pll_out_clk	the PLL output frequency
M	PLL Multiplier value from the MSEL bits in the PLLCFG register
P	PLL Divider value from the PSEL bits in the PLLCFG register

The PLL output frequency (when the PLL is active and locked) is given by:

$pll\_out\_clk = M \times pll\_in\_clk$ - or - $pll\_out\_clk = F_{CCO} / (2 \times P)$

The CCO frequency can be computed as:

$F_{CCO} = pll\_out\_clk \times 2 \times P$ - or - $F_{CCO} = pll\_in\_clk \times M \times 2 \times P$

The PLL inputs and settings must meet the following criteria:

- M is in the range of 1 to 32.
- P is one of 1, 2, 4, 8.
- pll_in_clk is in the range of 10 MHz to 25 MHz.
- F_{CCO} is in the range of 156 MHz to 320 MHz.
- pll_out_clk is in the range of 9.75 MHz to 160 MHz.

3.10.5 Procedure for determining PLL settings

In general, PLL configuration values may be found as follows:

1. Based on the desired PLL output frequency, choose an oscillator frequency (F_{OSC}). If the USB interface is to be used, an external crystal of either 12 MHz, 16 MHz, or 24 MHz must be provided. 12 MHz is recommended for this purpose in order to save power and have more flexibility with PLL settings.
2. If the USB interface is used in the system, and if a PLL output of 96 MHz or 144 MHz can provide the desired CPU clock frequency, it is probably possible to use only PLL0.
3. Calculate the value of M to configure the MSEL1 bits to obtain the desired PLL output frequency. M = pll_out_clk / pll_in_clk. The value written to the MSEL bits in the PLLCFG register is M – 1 (or see Table 46). This is done for both PLLs if they are both used.
4. Find a value for P to configure the PSEL bits, such that F_{CCO} is within its defined operating frequency limits of 156 MHz to 320 MHz. F_{CCO} is calculated using $F_{CCO} = pll\_out\_clk \times 2 \times P$. The value written to the PSEL bits in PLLCFG can be found in

Table 47.

Table 46. PLL Multiplier values

Value of M	MSEL Bits (PLLCFG bits [4:0])	MSEL hex
1	00000	0
2	00001	0x01
3	00010	0x02
4	00011	0x03
5	00100	0x04
6	00101	0x05
7	00110	0x06
8	00111	0x07
9	01000	0x08
10	01001	0x09
11	01010	0x0A
12	01011	0x0B
13	01100	0x0C
14	01101	0x0D
15	01110	0x0E
16	01111	0x0F
17	10000	0x10
18	10001	0x11
19	10010	0x12
20	10011	0x13
21	10100	0x14
22	10101	0x15
23	10110	0x16
24	10111	0x17
25	11000	0x18
26	11001	0x19
27	11010	0x1A
28	11011	0x1B
29	11100	0x1C
30	11101	0x1D
31	11110	0x1E
32	11111	0x1F

Table 47. PLL Divider values

Value of P	PSEL Bits (PLLCFG bits [6:5])	PSEL hex
1	00	0

Table 47. PLL Divider values

Value of P	PSEL Bits (PLLCFG bits [6:5])	PSEL hex
2	01	0x1
4	10	0x2
8	11	0x3

3.10.6 PLL configuration sequence

The following discussions refer to PLLs and PLL related registers generically (e.g. PLLCFG rather than PLL0CFG or PLL1CFG). The instructions have to be adapted to the specific case being addressed in the application.

To set up a PLL and switch clocks to its output:

1. Make sure that the PLL output is not already being used. The CCLKSEL, USBCLKSEL, and SPIFICLKSEL registers must not select the PLL being set up (see "To switch clocks away from a PLL output:" below). Clock dividers included in these registers may also be set up at this time if writing to any of the noted registers.
2. If the main PLL is being set up, and the main clock source is being changed (IRC versus main oscillator), change this first by writing the correct value to the CLKSRCSEL register.
3. Write PLL new setup values to the PLLCFG register. Write a 1 to the PLLE bit in the PLLCON register. Perform a PLL feed sequence by writing first the value 0xAA, then the value 0x55 to the PLLFEED register.
4. Set up the necessary clock dividers. These may include the CCLKSEL, PCLKSEL, EMCCLKSEL, USBCLKSEL, and the SPIFICLKSEL registers.
5. Wait for the PLL to lock. This may be accomplished by polling the PLLSTAT register and testing for PLOCK = 1, or by using the PLL lock interrupt.
6. Connect the PLL by selecting it output in the appropriate places. This may include the CCLKSEL, USBCLKSEL, and SPIFICLKSEL registers.

To switch clocks away from a PLL output:

1. To switch back to the mode of not using a PLL, write values to any or all of the CCLKSEL, USBCLKSEL, and SPIFICLKSEL registers in order to select a different clock source.
2. The related PLL may now be turned off by writing to the PLLCON register and performing a PLL feed sequence, reconfigured by writing to the PLLCFG register, etc.

3.10.7 PLL configuration examples

The following examples illustrate selecting PLL values based on different system requirements.

Example 1).

Assumptions:

- The system design is planned to use the IRC to generate the CPU clock.
- A frequency as close to 80 MHz as possible is desired for the CPU clock.

Of the two PLLs, only PLL0 can supply the CPU clock, so this example is for PLL0. The nearest multiple of the 12 MHz IRC frequency to 80 MHz is 84 MHz. Since pll_out_clk = M × pll_in_clk, M = pll_out_clk / pll_in_clk = 84 / 12 = 7.

Now a value for P must be found that puts F_{CCO} within the PLL operating range of 156 MHz to 320 MHz. F_{CCO} = pll_out_clk × 2 × P. Start by finding the value of F_{CCO} with P = 1, which is 84 MHz × 2 = 168 MHz. Since that is within the PLL operating range, no further work is needed.

Set up the PLL for M = 7and P = 1. This requires putting the value 6 (M - 1, or see Table 46 “PLL Multiplier values”) in the MSEL field of the PLL0CFG register. A value of 0 (see Table 47 “PLL Divider values”) is needed in the PSEL field of PLL0CFG. A single write of both values would be PLL0CFG = 0x06. See Section 3.10.6 for a description of the PLL setup sequence.

Example 2).

Assumptions:

- The system design is planned to use a 12 MHz crystal generate both the CPU clock and the USB clock.
- A frequency close to 100 MHz is desired for the CPU clock.

Of the two PLLs, only PLL0 can supply both the CPU clock and the USB clock, so this example is for PLL0. The PLL output must be an even integer multiple of 48 MHz for the USB to operate correctly (i.e. a multiple of 96 MHz). Two multiples of 96 MHz fit within the PLL operating range: 192 MHz (2 × 96 MHz), and 288 MHz (3 × 96 MHz). Of these, only 192 MHz can produce a CPU clock near 100 MHz (96 MHz). So, a 96 MHz PLL output can be used to obtain the 2 needed frequencies. Since pll_out_clk = M × pll_in_clk, M = pll_out_clk / pll_in_clk = 96 / 12 = 8.

Now a value for P must be found that puts F_{CCO} within the PLL operating range of 156 MHz to 320 MHz. F_{CCO} = pll_out_clk × 2 × P. Start by finding the value of F_{CCO} with P = 1, which is 96 MHz × 2 = 192 MHz. Since that is within the PLL operating range, no further work is needed.

Set up the PLL for M = 8 and P = 1. This requires putting the value 7 (M - 1, or see Table 46) in the MSEL field of the PLL0CFG register. A value of 0 (see Table 47) is needed in the PSEL field of PLL0CFG. A single write of both values would be PLL0CFG = 0x07. See Section 3.10.6 for a description of the PLL setup sequence.

Example 3)

Assumptions:

- The system design will use the USB interface.
- It is desired that the CPU clock remain flexible and able to operate at frequencies unrelated to the USB clock.

In order to keep the CPU clock separate form the USB clock, the CPU will use PLL0. For USB, PLL1 may be configured with the same values used in the last example. PLL0 can be operated from either the IRC or the main oscillator to obtain whatever frequency is needed, and the PLL0 setup can be changed without compromising USB operation.

3.11 Clock selection and division

The output of each PLL that is used must be divided down to whatever frequency is needed by each subsystem. There are separate clocks for the CPU, External Memory Controller, USB interface, SPIFI, and peripherals on the APB buses. Separate clock selection multiplexers and clock dividers provide flexibility in the generation of these clocks.

3.12 Power control

A variety of power control features are supported: Sleep mode, Deep Sleep mode, Power-down mode, and Deep Power-down mode. The CPU clock rate may also be controlled as needed by changing clock sources, re-configuring PLL values, and/or altering the CPU clock divider value. This allows a trade-off of power versus processing speed based on application requirements. In addition, Peripheral Power Control allows shutting down the clocks to individual on-chip peripherals, allowing fine tuning of power consumption by eliminating all dynamic power use in any peripherals that are not required for the application. A power boost feature allows operation up to 120 MHz, or power savings when operation is at or below 100 MHz.

Entry to any reduced power mode begins with the execution of either a WFI (Wait For Interrupt) or WFE (Wait For Exception) instruction by the CPU. The CPU internally supports two reduced power modes: Sleep and Deep Sleep. These are selected by the SLEEPDEEP bit in the Cortex-M4 System Control Register. Power-down and Deep Power-down modes are selected by bits in the PCON register. See Table 14. The same register contains flags that indicate whether entry into each reduced power mode actually occurred.

A separate power domain is implemented in order to allow turning off power to the bulk of the device while maintaining operation of the Real Time Clock.

Reduced power modes have some limitation during debug, see Section 39.7 for more information.

3.12.1 Sleep mode

Note: Sleep mode on these devices corresponds to the Idle mode on older LPC2xxx series devices. The name is changed because ARM has incorporated portions of reduced power mode control into the Cortex-M4.

When Sleep mode is entered, the clock to the core is stopped, and the SMFLAG bit in PCON is set, see Table 14.Resumption from the Sleep mode does not need any special sequence but re-enabling the clock to the ARM core.

In Sleep mode, execution of instructions is suspended until either a Reset or an interrupt occurs. Peripheral functions 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 DMA controller can continue to work in Sleep mode, and has access to the peripheral SRAMs and all peripheral registers. The flash memory and the Main SRAM are not available in Sleep mode, they are disabled in order to save power.

Wake-up from Sleep mode will occur whenever any enabled interrupt occurs.

3.12.2 Deep Sleep mode

Note: Deep Sleep mode on these devices corresponds to the Sleep mode on older LPC23xx and LPC24xx series devices. The name is changed because ARM has incorporated portions of reduced power mode control into the Cortex-M4.

When the chip enters the Deep Sleep mode, the main oscillator is powered down, nearly all clocks are stopped, and the DSFLAG bit in PCON is set, see Table 14. The IRC remains running for fast startup. The 32 kHz RTC oscillator is not stopped and RTC interrupts may be used as a wake-up source. The flash is left in the standby mode allowing a quick wake-up. The PLLs are automatically turned off and the clock selection multiplexers are set to use sysclk (the reset state). The clock divider control registers are automatically reset to zero.

The processor state and registers, peripheral registers, and internal SRAM values are preserved throughout Deep Sleep mode and the logic levels of chip pins remain static. The Deep Sleep mode can be terminated and normal operation resumed by either a Reset or certain specific interrupts that are able to function without clocks. Since all dynamic operation of the chip is suspended, Deep Sleep mode reduces chip power consumption to a very low value.

On the wake-up of Deep Sleep mode, if the IRC was used before entering Deep Sleep mode, a 2-bit IRC timer starts counting and the code execution and peripherals activities will resume after the timer expires (4 cycles). If the main oscillator is used, the 12-bit main oscillator timer starts counting and the code execution will resume when the timer expires (4096 cycles). The user must remember to re-configure any required PLLs and clock dividers after the wake-up.

Wake-up from Deep Sleep mode can be brought about by NMI, External Interrupts EINT0 through EINT3, GPIO interrupts, the Ethernet Wake-on-LAN interrupt, Brownout Detect, an RTC Alarm interrupt, a USB input pin transition (USB activity interrupt), a CAN input pin transition, or a Watchdog Timer timeout, when the related interrupt is enabled. Wake-up will occur whenever any enabled interrupt occurs.

3.12.3 Power-down mode

Power-down mode does everything that Deep Sleep mode does, but also turns off the flash memory. Entry to Power-down mode causes the PDFLAG bit in PCON to be set, see Table 14. This saves more power, but requires waiting for resumption of flash operation before execution of code or data access in the flash memory can be accomplished.

When the chip enters Power-down mode, the IRC, the main oscillator, and all clocks are stopped. The RTC remains running if it has been enabled and RTC interrupts may be used to wake up the CPU. The flash is forced into Power-down mode. The PLLs are automatically turned off and the clock selection multiplexers are set to use sysclk (the reset state). The clock divider control registers are automatically reset to zero. If the Watchdog timer is running, it will continue running in Power-down mode.

Upon wake-up from Power-down mode, if the IRC was used before entering Power-down mode, after IRC-start-up time (about 60 μs), the 2-bit IRC timer starts counting and expiring in 4 cycles. Code execution can then be resumed immediately following the expiration of the IRC timer if the code was running from SRAM. In the meantime, the flash wake-up timer measures flash start-up time of about 100 μs. When it times out, access to the flash is enabled. The user must remember to re-configure any required PLLs and clock dividers after the wake-up.

Wake-up from Power-down mode can be brought about by NMI, External Interrupts EINT0 through EINT3, GPIO interrupts, the Ethernet Wake-on-LAN interrupt, Brownout Detect, an RTC Alarm interrupt, a USB input pin transition (USB activity interrupt), or a CAN input pin transition, when the related interrupt is enabled.

3.12.4 Deep Power-down mode

In Deep Power-down mode, power is shut off to the entire chip with the exception of the Real-Time Clock, the $\overline{\text{RESET}}$ pin, the WIC, and the RTC backup registers. Entry to Deep Power-down mode causes the DPDFLAG bit in PCON to be set, see Table 14.

To optimize power conservation, the user has the additional option of turning off or retaining power to the 32 kHz oscillator. It is also possible to use external circuitry to turn off power to the on-chip regulator via the $V_{DD(REG)(3V3)}$ pins and/or the I/O power via the $V_{DD(3V3)}$ pins after entering Deep Power-down mode. Power must be restored before device operation can be restarted.

Wake-up from Deep Power-down mode will occur when an external reset signal is applied, or the RTC interrupt is enabled and an RTC interrupt is generated.

3.12.5 Peripheral power control

A Power Control for Peripherals feature allows individual peripherals to be turned off if they are not needed in the application, resulting in additional power savings. This is detailed in the description of the PCONP register.

3.12.6 Power boost

A Power boost feature allows operation above 100 MHz, to the upper limit for this device of 120 MHz. This boost is on by default when user code begins after a chip reset. Power can be saved by turning of this mode when operation will be at 100 MHz or lower. See Section 3.3.2.3.

3.12.7 Register description

The Power Control function uses registers shown in Table 48. More detailed descriptions follow.

Table 48. Power Control registers

Name	Description	Access	Reset value[1]	Address	Table
PCON	Power Control Register. This register contains control bits that enable some reduced power operating modes. See Table 14.	R/W	0	0x400F C0C0	14
PCONP	Power Control for Peripherals Register. This register contains control bits that enable and disable individual peripheral functions, allowing elimination of power consumption by peripherals that are not needed. See Table 16.	R/W	0x0408 829E	0x400F C0C4	16
PBOOST	Power Boost control register. This register controls the output of the main on-chip regulator, allowing a choice between high-speed operation above 100 MHz, or power savings when operation is at 100 MHz or lower.	R/W	0x3	0x400F C1B0	18

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

3.12.8 Wake-up from Reduced Power Modes

Any enabled interrupt can wake up the CPU from Sleep mode. Certain interrupts can wake up the processor if it is in either Deep Sleep mode or Power-down mode.

Interrupts that can occur during Deep Sleep or Power-down mode will wake up the CPU if the interrupt is enabled. After wake-up, execution will continue to the appropriate interrupt service routine. These interrupts are NMI, External Interrupts EINT0 through EINT3, GPIO interrupts, Ethernet Wake-on-LAN interrupt, Brownout Detect, RTC Alarm, CAN Activity Interrupt, USB Activity Interrupt, and Watchdog timer timeout. For the wake-up process to take place, the corresponding interrupt must be enabled in the NVIC. For pin-related peripheral functions, the related functions must also be mapped to pins.

The CAN Activity Interrupt is generated by activity on the CAN bus pins, and the USB Activity Interrupt is generated by activity on the USB bus pins. These interrupts are only useful to wake up the CPU when it is on Deep Sleep or Power-down mode, when the peripheral functions are powered up, but not active. Typically, if these interrupts are used, their flags should be polled just before enabling the interrupt and entering the desired reduced power mode. This can save time and power by avoiding an immediate wake-up. Upon wake-up, the interrupt service can turn off the related activity interrupt, do any application specific setup, and exit to await a normal peripheral interrupt.

In Deep Power-down mode, internal power to most of the device is removed, which limits the possibilities for waking up from this mode. External reset can wake-up the device. Also, of the RTC is running and has been set up to cause an interrupt, that event can wake-up the device.

3.12.9 Power control usage notes

After every reset, the PCONP register contains the value that enables selected interfaces and peripherals controlled by the PCONP to be enabled. Therefore, apart from proper configuring via peripheral dedicated registers, the user's application might have to access the PCONP in order to start using some of the on-board peripherals.

Power saving oriented systems should have 1s in the PCONP register only in positions that match peripherals that are actually used in the application. All other bits, declared to be "Reserved" or dedicated to the peripherals not used in the current application, must be cleared to 0.

3.12.10 Power domains

Two independent power domains are provided that allow the bulk of the device to have power removed while maintaining operation of the Real Time Clock.

The V_{BAT} pin supplies power only to the RTC domain. The RTC requires a minimum of power to operate, which can be supplied by an external battery. Whenever the device core power is greater than V_{BAT}, that power is used to operate the RTC.

3.13 Wake-up timer

At power-up and when awakened from Power-down mode, operation begins by using the 12 MHz IRC oscillator as the clock source. This allows chip operation to begin quickly. If the main oscillator or one or both PLLs are needed by the application, software will need to enable these features and wait for them to stabilize before they are used as a clock source.

When the main oscillator is initially activated, the wake-up timer allows software to ensure that the main oscillator is fully functional before the processor uses it as a clock source and starts to execute instructions. This is important at power-on, all types of Reset, and whenever any of the aforementioned functions are turned off for any reason. Since the oscillator and other functions are turned off during Power-down mode, any wake-up of the processor from Power-down mode makes use of the Wake-up Timer.

The Wake-up Timer monitors the crystal oscillator as the means of checking whether it is safe to begin code execution. When power is applied to the chip, or some event caused the chip to exit Power-down mode, some time is required for the oscillator to produce a signal of sufficient amplitude to drive the clock logic. The amount of time depends on many factors, including the rate of $V_{DD(REG)(3V3)}$ ramp (in the case of power on), the type of crystal and its electrical characteristics (if a quartz crystal is used), as well as any other external circuitry (e.g. capacitors), and the characteristics of the oscillator itself under the existing ambient conditions.

Once a clock is detected, the Wake-up Timer counts a fixed number of clocks (4,096), then sets the flag (OSCSTAT bit in the SCS register) that indicates that the main oscillator is ready for use. Software can then switch to the main oscillator and start any required PLLs. Refer to the Main Oscillator description in this chapter for details.

3.14 External clock output pin

For system test and development purposes, any one of several internal clocks may be brought out on the CLKOUT function available on the P1[25] or P1[27] pins, as shown in Figure 12.

Clocks that may be observed via CLKOUT are the CPU clock (cclk), the main oscillator (osc_clk), the internal RC oscillator (irc_osc), the USB clock (usb_clk), the RTC clock (rtc_clk), the SPIFI clock (spifi_clk), and the Watchdog oscillator (wdt_clk).

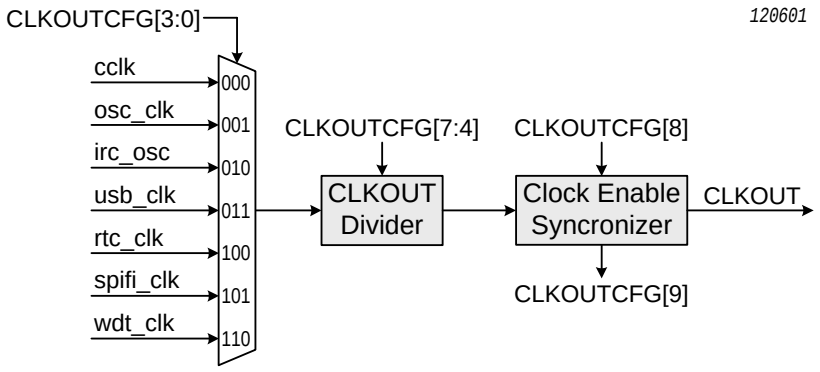


Fig 12. CLKOUT selection

4.1 Introduction

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.

4.2 Flash accelerator blocks

The flash accelerator is divided into several functional blocks:

- AHB-Lite bus interface, accessible by the I-code and D-code buses of the CPU, as well as by the General Purpose DMA Controller
- An array of eight 128-bit buffers
- Flash accelerator control logic, including address compare and flash control
- A flash memory interface

Figure 13 shows a simplified diagram of the flash accelerator blocks and data paths.

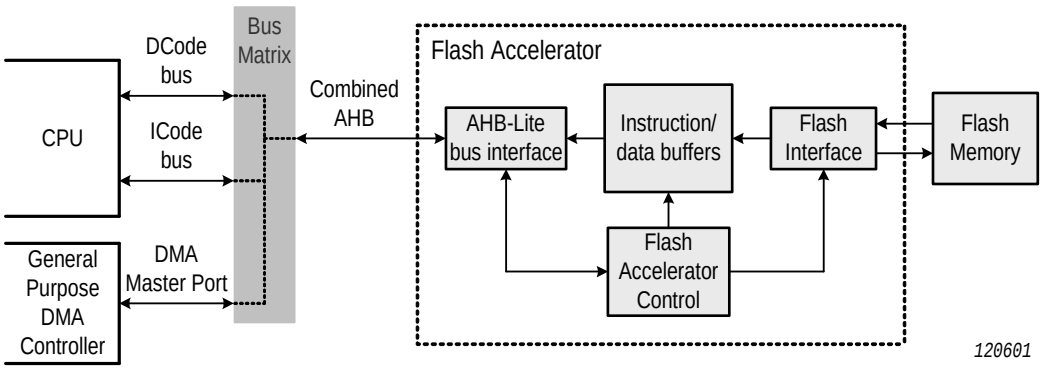


Fig 13. Simplified block diagram of the flash accelerator showing potential bus connections

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.

4.2.1 Flash memory bank

Flash programming operations are not controlled by the flash accelerator, but are handled as a separate function. A Boot ROM contains flash programming algorithms that may be called as part of the application program, and a loader that may be run to allow programming of the flash memory.

4.2.2 Flash programming Issues

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.

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.

4.3 Register description

The flash accelerator is controlled by the register shown in Table 49. More detailed descriptions follow.

Table 49. Summary of flash accelerator registers

Name	Description	Access	Reset value[1]	Address
FLASHCFG	Flash Accelerator Configuration Register. Controls flash access timing. See Table 50.	R/W	0x303A	0x400F C000

[1] Reset Value reflects the data stored in defined bits only. It does not include reserved bits content.

4.4 Flash Accelerator Configuration register

Configuration bits select the flash access time, as shown in Table 50. **The lower bits of FLASHCFG control internal flash accelerator functions and should not be altered.**

Following reset, flash accelerator functions are enabled and flash access timing is set to a default value of 4 clocks.

Changing the FLASHCFG register value causes the flash accelerator to invalidate all of the holding latches, resulting in new reads of flash information as required. This guarantees synchronization of the flash accelerator to CPU operation.

Table 50. Flash Accelerator Configuration register (FLASHCFG - address 0x400F C000) bit description

Bit	Symbol	Value	Description	Reset value
11:0	-	-	Reserved, user software should not change these bits from the reset value.	0x03A
15:12	FLASHTIM		Flash access time. The value of this field plus 1 gives the number of CPU clocks used for a flash access. **Warning:** improper setting of this value may result in incorrect operation of the device.	0x3
		0000	Flash accesses use 1 CPU clock. Use for up to 20 MHz CPU clock with power boost off (see Section 3.12.6).	
		0001	Flash accesses use 2 CPU clocks. Use for up to 40 MHz CPU clock with power boost off (see Section 3.12.6).	
		0010	Flash accesses use 3 CPU clocks. Use for up to 60 MHz CPU clock with power boost off (see Section 3.12.6).	
		0011	Flash accesses use 4 CPU clocks. Use for up to 80 MHz CPU clock with power boost off (see Section 3.12.6).	
		0100	Flash accesses use 5 CPU clocks. Use for up to 100 MHz CPU clock with power boost off (see Section 3.12.6). Use this setting for operation from 100 to 120 MHz operation with power boost on.	
		0101	Flash accesses use 6 CPU clocks. “Safe” setting for any allowed conditions.	
		Other	Intended for potential future higher speed devices.	
31:16	-		Reserved. Read value is undefined, only zero should be written.	NA

4.5 Operation

Simply put, the flash accelerator attempts to have the next instruction that will be needed in its latches in time to prevent CPU fetch stalls. The flash accelerator includes an array of eight 128-bit buffers to store both instructions and data in a configurable manner. Each 128-bit buffer in the array can include four 32-bit instructions, eight 16-bit instructions or some combination of the two. During sequential code execution, a buffer typically contains the current instruction and the entire flash line that contains that instruction, or one flash line of data containing a previously requested address. Buffers are marked according to how they are used (as instruction or data buffers), and when they have been accessed. This information is used to carry out the buffer replacement strategy.

The CPU provides a separate bus for instruction access (I-code) and data access (D-code) in the code memory space. These buses, plus the General Purpose DMA Controllers's master port, are arbitrated by the AHB multilayer matrix. Any access to the flash memory's address space is presented to the flash accelerator.

If a flash instruction fetch and a flash data access from the CPU occur at the same time, the multilayer matrix gives precedence to the data access. This is because a stalled data access always slows down execution, while a stalled instruction fetch often does not. When the flash data access is concluded, any flash fetch or prefetch that had been in progress is re-initiated.

Branches and other program flow changes cause a break in the sequential flow of instruction fetches described above. Buffer replacement strategy in the flash accelerator attempts to maximize the chances that potentially reusable information is retained until it is needed again.

If an attempt is made to write directly to the flash memory without using the normal flash programming interface (via Boot ROM function calls), the flash accelerator generates an error condition. The CPU treats this error as a data abort. The GPDMA handles error conditions as described in Section 35.4.1.6.3.

When an Instruction Fetch is not satisfied by existing contents of the buffer array, nor has a prefetch been initiated for that flash line, the CPU will be stalled while a fetch is initiated for the related 128-bit flash line. If a prefetch has been initiated but not yet completed, the CPU is stalled for a shorter time since the required flash access is already in progress.

Typically, a flash prefetch is begun whenever an access is made to a just prefetched address, or to a buffer whose immediate successor is not already in another buffer. A prefetch in progress may be aborted by a data access, in order to minimize CPU stalls.

A prefetched flash line is latched within the flash memory, but the flash accelerator does not capture the line in a buffer until the CPU presents an address that is contained within the prefetched flash line. If the core presents an instruction address that is not already buffered and is not contained in the prefetched flash line, the prefetched line will be discarded.

Some special cases include the possibility that the CPU will request a data access to an address already contained in an instruction buffer. In this case, the data will be read from the buffer as if it was a data buffer. The reverse case, if the CPU requests an instruction address that can be satisfied from an existing data buffer, causes the instruction to be

supplied from the data buffer, and the buffer to be changed into an instruction buffer. This causes the buffer to be handled differently when the flash accelerator is determining which buffer is to be overwritten next.

5.1 Features

- Nested Vectored Interrupt Controller that is an integral part of the ARM Cortex-M4
- Tightly coupled interrupt controller provides low interrupt latency
- Controls system exceptions and peripheral interrupts
- The NVIC supports 40 vectored interrupts in these devices
- 32 programmable interrupt priority levels, with hardware priority level masking
- Relocatable vector table
- Non-Maskable Interrupt
- Software interrupt generation

5.2 Description

The Nested Vectored Interrupt Controller (NVIC) is an integral part of the Cortex-M4. The tight coupling to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts. The NVIC handles interrupts in addition to system exceptions. Exceptions include Reset, NMI, Hard Fault, MemManage Fault, Bus Fault, Usage Fault, SVCall, Debug Monitor, PendSV, and Systick.

See the ARM Cortex-M4 User Guide referred to in Section 40.1 for details of NVIC operation.

5.3 Interrupt sources

Table 51 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, as noted.

Exception numbers relate to where entries are stored in the exception vector table. Interrupt numbers are used in some other contexts, such as software interrupts.

Note that system exceptions are hard-wired into the Cortex-M4 and are not shown in the table. Some other information about the Systick interrupt can be found in the System Tick Timer chapter, Section 25.1

In addition, the NVIC handles the Non-Maskable Interrupt (NMI). In order for NMI to operate from an external signal, the NMI function must be connected to the related device pin (P2[10] / EINT0n / NMI). When connected, a logic 1 on the pin will cause the NMI to be processed. For details, refer to the Cortex-M4 User Guide that is an appendix to this User Manual.

Table 51. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
0	16	0x40	WDT	Watchdog Interrupt (WDINT)
1	17	0x44	Timer 0	Match 0 - 1 (MR0, MR1) Capture 0 - 1 (CR0, CR1)
2	18	0x48	Timer 1	Match 0 - 2 (MR0, MR1, MR2) Capture 0 - 1 (CR0, CR1)
3	19	0x4C	Timer 2	Match 0-3 Capture 0-1
4	20	0x50	Timer 3	Match 0-3 Capture 0-1
5	21	0x54	UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
6	22	0x58	UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Control Change End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
7	23	0x5C	UART 2	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
8	24	0x60	UART 3	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
9	25	0x64	PWM1	Match 0 - 6 of PWM1 Capture 0-1 of PWM1
10	26	0x68	I^2C0	SI (state change)
11	27	0x6C	I^2C1	SI (state change)
12	28	0x70	I^2C2	SI (state change)
13	29	0x74	(unused)	-

Table 51. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
14	30	0x78	SSP0	Tx FIFO half empty of SSP0 Rx FIFO half full of SSP0 Rx Timeout of SSP0 Rx Overrun of SSP0
15	31	0x7C	SSP 1	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
16	32	0x80	PLL0 (Main PLL)	PLL0 Lock (PLOCK0)
17	33	0x84	RTC and Event Monitor/Recorder	Counter Increment (RTCCIF), Alarm (RTCALF) EV0, EV1, EV2
18	34	0x88	External Interrupt	External Interrupt 0 (EINT0)
19	35	0x8C	External Interrupt	External Interrupt 1 (EINT1)
20	36	0x90	External Interrupt	External Interrupt 2 (EINT2)
21	37	0x94	External Interrupt	External Interrupt 3 (EINT3)
22	38	0x98	ADC	A/D Converter end of conversion
23	39	0x9C	BOD	Brown Out detect
24	40	0xA0	USB	USB_INT_REQ_LP, USB_INT_REQ_HP, USB_INT_REQ_DMA, USB_HOST_INT, USB_ATX_INT, USB_OTG_INT, USB_I2C_INT
25	41	0xA4	CAN	CAN Common, CAN 0 Tx, CAN 0 Rx, CAN 1 Tx, CAN 1 Rx
26	42	0xA8	DMA Controller	Interrupt status of all DMA channels
27	43	0xAC	I^2S	irq, dmareq1, dmareq2
28	44	0xB0	Ethernet	WakeupInt, SoftInt, TxDoneInt, TxFinishedInt, TxErrorInt, TxUnderrunInt, RxDoneInt, RxFinishedInt, RxErrorInt, RxOverrunInt.
29	45	0xB4	SD Card Interface	RxDataAvlbl, TxDataAvlbl, RxFifoEmpty, TxFifoEmpty, RxFifoFull, TxFifoFull, RxFifoHalfFull, TxFifoHalfEmpty, RxActive, TxActive, CmdActive, DataBlockEnd, StartBitErr, DataEnd, CmdSent, CmdRespEnd, RxOverrun, TxUnderrun, DataTimeOut, CmdTimeOut, DataCrcFail, CmdCrcFail
30	46	0xB8	Motor Control PWM	IPER[2:0], IPW[2:0], ICAP[2:0], FES
31	47	0xBC	Quadrature Encoder	INX_Int, TIM_Int, VELC_Int, DIR_Int, ERR_Int, ENCLK_Int, POS0_Int, POS1_Int, POS2_Int, REV_Int, POS0REV_Int, POS1REV_Int, POS2REV_Int
32	48	0xC0	PLL1 (Alt PLL)	PLL1 Lock (PLOCK1)
33	49	0xC4	USB Activity Interrupt	USB_NEED_CLK
34	50	0xC8	CAN Activity Interrupt	CAN1WAKE, CAN2WAKE
35	51	0xCC	UART4	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)

Table 51. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
36	52	0xD0	SSP2	Tx FIFO half empty of SSP2 Rx FIFO half full of SSP2 Rx Timeout of SSP2 Rx Overrun of SSP2
37	53	0xD4	LCD controller	BER, VCompI, LNBUI, FUFI, CrsrI
38	54	0xD8	GPIO interrupts	P0xREI, P2xREI, P0xFEI, P2xFEI
39	55	0xDC	PWM0	Match 0 - 6 of PWM0 Capture 0-1 of PWM0
40	56	0xE0	EEPROM	EE_PROG_DONE, EE_RW_DONE

5.4 Vector table remapping

The Cortex-M4 incorporates a mechanism that allows remapping the interrupt vector table to alternate locations in the memory map. This is controlled via the Vector Table Offset Register (VTOR) contained in the Cortex-M4.

The vector table may be located anywhere within the bottom 1 GB of Cortex-M4 address space. The vector table should be located on a 256 word (1024 byte) boundary to insure alignment. See the ARM Cortex-M4 User Guide referred to in Section 40.1 for details of the Vector Table Offset feature.

ARM describes bit 29 of the VTOR (TBLOFF) as selecting a memory region, either code or SRAM. For simplicity, this bit can be thought as simply part of the address offset since the split between the “code” space and the “SRAM” space occurs at the location corresponding to bit 29 in a memory address.

Examples:

To place the vector table at the beginning of the Main SRAM, starting at address 0x1000 0000, place the value 0x1000 0000 in the VTOR register. This indicates address 0x1000 0000 in the code space, since bit 29 of the VTOR equals 0.

To place the vector table at the beginning of the peripheral SRAM, starting at address 0x2000 0000, place the value 0x2000 0000 in the VTOR register. This indicates address 0x2000 0000 in the SRAM space, since bit 29 of the VTOR equals 1.

5.5 Register description

The following table summarizes the registers in the NVIC as implemented in LPC408x/407x devices. See the ARM Cortex-M4 User Guide referred to in Section 40.1 for functional details of the NVIC.

Table 52. NVIC register map

Name	Description	Access	Reset value	Address	Table
ISER0 to ISER1	Interrupt Set-Enable Registers. These registers allow enabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ISER0 - 0xE000 E100	53
				ISER1 - 0xE000 E104	54
ICER0 to ICER1	Interrupt Clear-Enable Registers. These registers allow disabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ICER0 - 0xE000 E180	55
				ICER1 - 0xE000 E184	56
ISPR0 to ISPR1	Interrupt Set-Pending Registers. These registers allow changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ISPR0 - 0xE000 E200	57
				ISPR1 - 0xE000 E204	58
ICPR0 to ICPR1	Interrupt Clear-Pending Registers. These registers allow changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ICPR0 - 0xE000 E280	59
				ICPR1 - 0xE000 E284	60
IABR0 to IABR1	Interrupt Active Bit Registers. These registers allow reading the current interrupt active state for specific peripheral functions.	RO	0	IABR0 - 0xE000 E300	61
				IABR1 - 0xE000 E304	62
IPR0 to IPR10	Interrupt Priority Registers. These registers allow assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	RW	0	IPR0 - 0xE000 E400	63
				IPR1 - 0xE000 E404	64
				IPR2 - 0xE000 E408	65
				IPR3 - 0xE000 E40C	66
				IPR4 - 0xE000 E410	67
				IPR5 - 0xE000 E414	68
				IPR6 - 0xE000 E418	69
				IPR7 - 0xE000 E41C	70
				IPR8 - 0xE000 E420	71
				IPR9 - 0xE000 E424	72
				IPR10 - 0xE000 E428	73
STIR	Software Trigger Interrupt Register. This register allows software to generate an interrupt.	WO	-	STIR - 0xE000 EF00	74

5.5.1 Interrupt Set-Enable Register 0 register

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 5.5.2). Disabling interrupts is done through the ICER0 and ICER1 registers (Section 5.5.3 and Section 5.4).

Table 53. Interrupt Set-Enable Register 0 register

Bit	Name	Function
0	ISE_WDT	Watchdog Timer interrupt enable. 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.
1	ISE_TIMER0	Timer 0 interrupt enable. See functional description for bit 0.
2	ISE_TIMER1	Timer 1 interrupt enable. See functional description for bit 0.
3	ISE_TIMER2	Timer 2 interrupt enable. See functional description for bit 0.
4	ISE_TIMER3	Timer 3 interrupt enable. See functional description for bit 0.
5	ISE_UART0	UART0 interrupt enable. See functional description for bit 0.
6	ISE_UART1	UART1 interrupt enable. See functional description for bit 0.
7	ISE_UART2	UART2 interrupt enable. See functional description for bit 0.
8	ISE_UART3	UART3 interrupt enable. See functional description for bit 0.
9	ISE_PWM1	PWM1 interrupt enable. See functional description for bit 0.
10	ISE_I2C0	I^2C0 interrupt enable. See functional description for bit 0.
11	ISE_I2C1	I^2C1 interrupt enable. See functional description for bit 0.
12	ISE_I2C2	I^2C2 interrupt enable. See functional description for bit 0.
13	-	Reserved. Read value is undefined, only zero should be written.
14	ISE_SSP0	SSP0 interrupt enable. See functional description for bit 0.
15	ISE_SSP1	SSP1 interrupt enable. See functional description for bit 0.
16	ISE_PLL0	PLL0 (Main PLL) interrupt enable. See functional description for bit 0.
17	ISE_RTC	Real Time Clock (RTC) and Event Monitor/Recorder interrupt enable. See description of bit 0.
18	ISE_EINT0	External Interrupt 0 interrupt enable. See functional description for bit 0.
19	ISE_EINT1	External Interrupt 1 interrupt enable. See functional description for bit 0.
20	ISE_EINT2	External Interrupt 2 interrupt enable. See functional description for bit 0.
21	ISE_EINT3	External Interrupt 3 interrupt enable. See functional description for bit 0.
22	ISE_ADC	ADC interrupt enable. See functional description for bit 0.
23	ISE_BOD	BOD interrupt enable. See functional description for bit 0.
24	ISE_USB	USB interrupt enable. See functional description for bit 0.
25	ISE_CAN	CAN interrupt enable. See functional description for bit 0.
26	ISE_DMA	GPDMA interrupt enable. See functional description for bit 0.
27	ISE_I2S	I^2S interrupt enable. See functional description for bit 0.
28	ISE_ENET	Ethernet interrupt enable. See functional description for bit 0.
29	ISE_SD	SD card interface interrupt enable. See functional description for bit 0.
30	ISE_MCPWM	Motor Control PWM interrupt enable. See functional description for bit 0.
31	ISE_QEI	Quadrature Encoder Interface interrupt enable. See functional description for bit 0.

5.5.2 Interrupt Set-Enable Register 1 register

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 5.5.3 and Section 5.4).

Table 54. Interrupt Set-Enable Register 1 register

Bit	Name	Function
0	ISE_PLL1	PLL1 (Alt PLL) interrupt enable. 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.
1	ISE_USBACT	USB Activity interrupt enable. See functional description for bit 0.
2	ISE_CANACT	CAN Activity interrupt enable. See functional description for bit 0.
3	ISE_UART4	UART4 interrupt enable. See functional description for bit 0.
4	ISE_SSP2	SSP2 interrupt enable. See functional description for bit 0.
5	ISE_LCD	LCD interrupt enable. See functional description for bit 0.
6	ISE_GPIO	GPIO interrupt enable. See functional description for bit 0.
7	ISE_PWM0	PWM0 interrupt enable. See functional description for bit 0.
8	ISE_FLASH	Flash and EEPROM interrupt enable. See functional description for bit 0.
31:9	-	Reserved. Read value is undefined, only zero should be written.

5.5.3 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 5.4). Enabling interrupts is done through the ISER0 and ISER1 registers (Section 5.5.1 and Section 5.5.2).

Table 55. Interrupt Clear-Enable Register 0

Bit	Name	Function
0	ICE_WDT	Watchdog Timer interrupt disable. 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.
1	ICE_TIMER0	Timer 0 interrupt disable. See functional description for bit 0.
2	ICE_TIMER1	Timer 1 interrupt disable. See functional description for bit 0.
3	ICE_TIMER2	Timer 2 interrupt disable. See functional description for bit 0.
4	ICE_TIMER3	Timer 3 interrupt disable. See functional description for bit 0.
5	ICE_UART0	UART0 interrupt disable. See functional description for bit 0.
6	ICE_UART1	UART1 interrupt disable. See functional description for bit 0.
7	ICE_UART2	UART2 interrupt disable. See functional description for bit 0.
8	ICE_UART3	UART3 interrupt disable. See functional description for bit 0.
9	ICE_PWM1	PWM1 interrupt disable. See functional description for bit 0.
10	ICE_I2C0	I^2C0 interrupt disable. See functional description for bit 0.
11	ICE_I2C1	I^2C1 interrupt disable. See functional description for bit 0.
12	ICE_I2C2	I^2C2 interrupt disable. See functional description for bit 0.
13	-	Reserved. Read value is undefined, only zero should be written.
14	ICE_SSP0	SSP0 interrupt disable. See functional description for bit 0.
15	ICE_SSP1	SSP1 interrupt disable. See functional description for bit 0.
16	ICE_PLL0	PLL0 (Main PLL) interrupt disable. See functional description for bit 0.
17	ICE_RTC	Real Time Clock (RTC) and Event Monitor/Recorder interrupt disable. See description of bit 0.
18	ICE_EINT0	External Interrupt 0 interrupt disable. See functional description for bit 0.
19	ICE_EINT1	External Interrupt 1 interrupt disable. See functional description for bit 0.
20	ICE_EINT2	External Interrupt 2 interrupt disable. See functional description for bit 0.
21	ICE_EINT3	External Interrupt 3 interrupt disable. See functional description for bit 0.
22	ICE_ADC	ADC interrupt disable. See functional description for bit 0.
23	ICE_BOD	BOD interrupt disable. See functional description for bit 0.
24	ICE_USB	USB interrupt disable. See functional description for bit 0.
25	ICE_CAN	CAN interrupt disable. See functional description for bit 0.
26	ICE_DMA	GPDMA interrupt disable. See functional description for bit 0.
27	ICE_I2S	I^2S interrupt disable. See functional description for bit 0.
28	ICE_ENET	Ethernet interrupt disable. See functional description for bit 0.
29	ICE_SD	SD card interface interrupt disable. See functional description for bit 0.
30	ICE_MCPWM	Motor Control PWM interrupt disable. See functional description for bit 0.
31	ICE_QEI	Quadrature Encoder Interface interrupt disable. See functional description for bit 0.

5.4 Interrupt Clear-Enable Register 1 register

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 5.5.1 and Section 5.5.2).

Table 56. Interrupt Clear-Enable Register 1 register

Bit	Name	Function
0	ICE_PLL1	PLL1 (Alt PLL) interrupt disable. 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.
1	ICE_USBACT	USB Activity interrupt disable. See functional description for bit 0.
2	ICE_CANACT	CAN Activity interrupt disable. See functional description for bit 0.
3	ICE_UART4	UART4 interrupt disable. See functional description for bit 0.
4	ICE_SSP2	SSP2 interrupt disable. See functional description for bit 0.
5	ICE_LCD	LCD interrupt disable. See functional description for bit 0.
6	ICE_GPIO	GPIO interrupt disable. See functional description for bit 0.
7	ICE_PWM0	PWM0 interrupt disable. See functional description for bit 0.
8	ICE_EEPROM	EEPROM interrupt disable. See functional description for bit 0.
31:9	-	Reserved. Read value is undefined, only zero should be written.

5.5.5 Interrupt Set-Pending Register 0 register

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 5.5.6). Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 5.5.7 and Section 5.5.8).

Table 57. Interrupt Set-Pending Register 0 register

Bit	Name	Function
0	ISP_WDT	Watchdog Timer interrupt pending set. 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.
1	ISP_TIMER0	Timer 0 interrupt pending set. See functional description for bit 0.
2	ISP_TIMER1	Timer 1 interrupt pending set. See functional description for bit 0.
3	ISP_TIMER2	Timer 2 interrupt pending set. See functional description for bit 0.
4	ISP_TIMER3	Timer 3 interrupt pending set. See functional description for bit 0.
5	ISP_UART0	UART0 interrupt pending set. See functional description for bit 0.
6	ISP_UART1	UART1 interrupt pending set. See functional description for bit 0.
7	ISP_UART2	UART2 interrupt pending set. See functional description for bit 0.
8	ISP_UART3	UART3 interrupt pending set. See functional description for bit 0.
9	ISP_PWM1	PWM1 interrupt pending set. See functional description for bit 0.
10	ISP_I2C0	I^2C0 interrupt pending set. See functional description for bit 0.
11	ISP_I2C1	I^2C1 interrupt pending set. See functional description for bit 0.
12	ISP_I2C2	I^2C2 interrupt pending set. See functional description for bit 0.
13	-	Reserved. Read value is undefined, only zero should be written.
14	ISP_SSP0	SSP0 interrupt pending set. See functional description for bit 0.
15	ISP_SSP1	SSP1 interrupt pending set. See functional description for bit 0.
16	ISP_PLL0	PLL0 (Main PLL) interrupt pending set. See functional description for bit 0.
17	ISP_RTC	Real Time Clock (RTC) and Event Monitor/Recorder interrupt pending set. See description of bit 0.
18	ISP_EINT0	External Interrupt 0 interrupt pending set. See functional description for bit 0.
19	ISP_EINT1	External Interrupt 1 interrupt pending set. See functional description for bit 0.
20	ISP_EINT2	External Interrupt 2 interrupt pending set. See functional description for bit 0.
21	ISP_EINT3	External Interrupt 3 interrupt pending set. See functional description for bit 0.
22	ISP_ADC	ADC interrupt pending set. See functional description for bit 0.
23	ISP_BOD	BOD interrupt pending set. See functional description for bit 0.
24	ISP_USB	USB interrupt pending set. See functional description for bit 0.
25	ISP_CAN	CAN interrupt pending set. See functional description for bit 0.
26	ISP_DMA	GPDMA interrupt pending set. See functional description for bit 0.
27	ISP_I2S	I^2S interrupt pending set. See functional description for bit 0.
28	ISP_ENET	Ethernet interrupt pending set. See functional description for bit 0.
29	ISP_SD	SD card interface interrupt pending set. See functional description for bit 0.
30	ISP_MCPWM	Motor Control PWM interrupt pending set. See functional description for bit 0.
31	ISP_QEI	Quadrature Encoder Interface interrupt pending set. See functional description for bit 0.

5.5.6 Interrupt Set-Pending Register 1 register

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 5.5.7 and Section 5.5.8).

Table 58. Interrupt Set-Pending Register 1 register

Bit	Name	Function
0	ISP_PLL1	PLL1 (Alt PLL) interrupt pending set. 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.
1	ISP_USBACT	USB Activity interrupt pending set. See functional description for bit 0.
2	ISP_CANACT	CAN Activity interrupt pending set. See functional description for bit 0.
3	ISP_UART4	UART4 interrupt pending set. See functional description for bit 0.
4	ISP_SSP2	SSP2 interrupt pending set. See functional description for bit 0.
5	ISP_LCD	LCD interrupt pending set. See functional description for bit 0.
6	ISP_GPIO	GPIO interrupt pending set. See functional description for bit 0.
7	ISP_PWM0	PWM0 interrupt pending set. See functional description for bit 0.
8	ISP_EEPROM	EEPROM interrupt pending set. See functional description for bit 0.
31:9	-	Reserved. Read value is undefined, only zero should be written.

5.5.7 Interrupt Clear-Pending Register 0 register

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 5.5.8). Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 5.5.5 and Section 5.5.6).

Table 59. Interrupt Clear-Pending Register 0 register

Bit	Name	Function
0	ICP_WDT	Watchdog Timer interrupt pending clear. 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.
1	ICP_TIMER0	Timer 0 interrupt pending clear. See functional description for bit 0.
2	ICP_TIMER1	Timer 1 interrupt pending clear. See functional description for bit 0.
3	ICP_TIMER2	Timer 2 interrupt pending clear. See functional description for bit 0.
4	ICP_TIMER3	Timer 3 interrupt pending clear. See functional description for bit 0.
5	ICP_UART0	UART0 interrupt pending clear. See functional description for bit 0.
6	ICP_UART1	UART1 interrupt pending clear. See functional description for bit 0.
7	ICP_UART2	UART2 interrupt pending clear. See functional description for bit 0.
8	ICP_UART3	UART3 interrupt pending clear. See functional description for bit 0.
9	ICP_PWM1	PWM1 interrupt pending clear. See functional description for bit 0.
10	ICP_I2C0	I^2C0 interrupt pending clear. See functional description for bit 0.
11	ICP_I2C1	I^2C1 interrupt pending clear. See functional description for bit 0.
12	ICP_I2C2	I^2C2 interrupt pending clear. See functional description for bit 0.
13	-	Reserved. Read value is undefined, only zero should be written.
14	ICP_SSP0	SSP0 interrupt pending clear. See functional description for bit 0.
15	ICP_SSP1	SSP1 interrupt pending clear. See functional description for bit 0.
16	ICP_PLL0	PLL0 (Main PLL) interrupt pending clear. See functional description for bit 0.
17	ICP_RTC	Real Time Clock (RTC) and Event Monitor/Recorder interrupt pending clear. See description of bit 0.
18	ICP_EINT0	External Interrupt 0 interrupt pending clear. See functional description for bit 0.
19	ICP_EINT1	External Interrupt 1 interrupt pending clear. See functional description for bit 0.
20	ICP_EINT2	External Interrupt 2 interrupt pending clear. See functional description for bit 0.
21	ICP_EINT3	External Interrupt 3 interrupt pending clear. See functional description for bit 0.
22	ICP_ADC	ADC interrupt pending clear. See functional description for bit 0.
23	ICP_BOD	BOD interrupt pending clear. See functional description for bit 0.
24	ICP_USB	USB interrupt pending clear. See functional description for bit 0.
25	ICP_CAN	CAN interrupt pending clear. See functional description for bit 0.
26	ICP_DMA	GPDMA interrupt pending clear. See functional description for bit 0.
27	ICP_I2S	I^2S interrupt pending clear. See functional description for bit 0.
28	ICP_ENET	Ethernet interrupt pending clear. See functional description for bit 0.
29	ICP_SD	SD Card interface interrupt pending clear. See functional description for bit 0.
30	ICP_MCPWM	Motor Control PWM interrupt pending clear. See functional description for bit 0.
31	ICP_QEI	Quadrature Encoder Interface interrupt pending clear. See functional description for bit 0.

5.5.8 Interrupt Clear-Pending Register 1 register

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 5.5.5 and Section 5.5.6).

Table 60. Interrupt Clear-Pending Register 1 register

Bit	Name	Function
0	ICP_PLL1	PLL1 (Alt PLL) interrupt pending clear. 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.
1	ICP_USBACT	USB Activity interrupt pending clear. See functional description for bit 0.
2	ICP_CANACT	CAN Activity interrupt pending clear. See functional description for bit 0.
3	ICP_UART4	UART4 interrupt pending clear. See functional description for bit 0.
4	ICP_SSP2	SSP2 interrupt pending clear. See functional description for bit 0.
5	ICP_LCD	LCD interrupt pending clear. See functional description for bit 0.
6	ICP_GPIO	GPIO interrupt pending clear. See functional description for bit 0.
7	ICP_PWM0	PWM0 interrupt pending clear. See functional description for bit 0.
8	ICP_EEPROM	EEPROM interrupt pending clear. See functional description for bit 0.
31:9	-	Reserved. Read value is undefined, only zero should be written.

5.5.9 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 5.5.10).

Table 61. Interrupt Active Bit Register 0

Bit	Name	Function
0	IAB_WDT	Watchdog Timer interrupt active. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.
1	IAB_TIMER0	Timer 0 interrupt active. See functional description for bit 0.
2	IAB_TIMER1	Timer 1 interrupt active. See functional description for bit 0.
3	IAB_TIMER2	Timer 2 interrupt active. See functional description for bit 0.
4	IAB_TIMER3	Timer 3 interrupt active. See functional description for bit 0.
5	IAB_UART0	UART0 interrupt active. See functional description for bit 0.
6	IAB_UART1	UART1 interrupt active. See functional description for bit 0.
7	IAB_UART2	UART2 interrupt active. See functional description for bit 0.
8	IAB_UART3	UART3 interrupt active. See functional description for bit 0.
9	IAB_PWM1	PWM1 interrupt active. See functional description for bit 0.
10	IAB_I2C0	I^2C0 interrupt active. See functional description for bit 0.
11	IAB_I2C1	I^2C1 interrupt active. See functional description for bit 0.
12	IAB_I2C2	I^2C2 interrupt active. See functional description for bit 0.
13	-	Reserved. Read value is undefined, only zero should be written.
14	IAB_SSP0	SSP0 interrupt active. See functional description for bit 0.
15	IAB_SSP1	SSP1 interrupt active. See functional description for bit 0.
16	IAB_PLL0	PLL0 (Main PLL) interrupt active. See functional description for bit 0.
17	IAB_RTC	Real Time Clock (RTC) and Event Monitor/Recorder interrupt active. See description of bit 0.
18	IAB_EINT0	External Interrupt 0 interrupt active. See functional description for bit 0.
19	IAB_EINT1	External Interrupt 1 interrupt active. See functional description for bit 0.
20	IAB_EINT2	External Interrupt 2 interrupt active. See functional description for bit 0.
21	IAB_EINT3	External Interrupt 3 interrupt active. See functional description for bit 0.
22	IAB_ADC	ADC interrupt active. See functional description for bit 0.
23	IAB_BOD	BOD interrupt active. See functional description for bit 0.
24	IAB_USB	USB interrupt active. See functional description for bit 0.
25	IAB_CAN	CAN interrupt active. See functional description for bit 0.
26	IAB_DMA	GPDMA interrupt active. See functional description for bit 0.
27	IAB_I2S	I^2S interrupt active. See functional description for bit 0.
28	IAB_ENET	Ethernet interrupt active. See functional description for bit 0.
29	IAB_SD	Repetitive Interrupt Timer interrupt active. See functional description for bit 0.
30	IAB_MCPWM	Motor Control PWM interrupt active. See functional description for bit 0.
31	IAB_QEI	Quadrature Encoder Interface interrupt active. See functional description for bit 0.

5.5.10 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 62. Interrupt Active Bit Register 1

Bit	Name	Function
0	IAB_PLL1	PLL1 (Alt PLL) interrupt active. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.
1	IAB_USBACT	USB Activity interrupt active. See functional description for bit 0.
2	IAB_CANACT	CAN Activity interrupt active. See functional description for bit 0.
3	IAB_UART4	UART4 interrupt active. See functional description for bit 0.
4	IAB_SSP2	SSP2 interrupt active. See functional description for bit 0.
5	IAB_LCD	LCD interrupt active. See functional description for bit 0.
6	IAB_GPIO	GPIO interrupt active. See functional description for bit 0.
7	IAB_PWM0	PWM0 interrupt active. See functional description for bit 0.
8	IAB_EEPROM	EEPROM interrupt active. See functional description for bit 0.
31:9	-	Reserved. The value read from a reserved bit is not defined.

5.5.11 Interrupt Priority Register 0

The IPR0 register controls the priority of the first 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 63. Interrupt Priority Register 0

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_WDT	Watchdog Timer interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_TIMER0	Timer 0 interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_TIMER1	Timer 1 interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_TIMER2	Timer 2 interrupt priority. See functional description for bits 7-3.

5.5.12 Interrupt Priority Register 1

The IPR1 register controls the priority of the second group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 64. Interrupt Priority Register 1

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_TIMER3	Timer 3 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_UART0	UART0 interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_UART1	UART1 interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_UART2	UART2 interrupt priority. See functional description for bits 7-3.

5.5.13 Interrupt Priority Register 2

The IPR2 register controls the priority of the third group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 65. Interrupt Priority Register 2

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_UART3	UART3 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_PWM1	PWM1 interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_I2C0	I^2C0 interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_I2C1	I^2C1 interrupt priority. See functional description for bits 7-3.

5.5.14 Interrupt Priority Register 3

The IPR3 register controls the priority of the fourth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 66. Interrupt Priority Register 3

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_I2C2	I^2C2 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	-	Reserved. Read value is undefined, only zero should be written.
18:8	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_SSP0	SSP0 interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_SSP1	SSP1 interrupt priority. See functional description for bits 7-3.

5.5.15 Interrupt Priority Register 4

The IPR4 register controls the priority of the fifth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 67. Interrupt Priority Register 4

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_PLL0	PLL0 (Main PLL) interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_RTC	Real Time Clock (RTC) interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_EINT0	External Interrupt 0 interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_EINT1	External Interrupt 1 interrupt priority. See functional description for bits 7-3.

5.5.16 Interrupt Priority Register 5

The IPR5 register controls the priority of the sixth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 68. Interrupt Priority Register 5

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_EINT2	External Interrupt 2 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_EINT3	External Interrupt 3 interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_ADC	ADC interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_BOD	BOD interrupt priority. See functional description for bits 7-3.

5.5.17 Interrupt Priority Register 6

The IPR6 register controls the priority of the seventh group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 69. Interrupt Priority Register 6

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_USB	USB interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_CAN	CAN interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_DMA	GPDMA interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_I2S	I^2S interrupt priority. See functional description for bits 7-3.

5.5.18 Interrupt Priority Register 7

The IPR7 register controls the priority of the eighth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 70. Interrupt Priority Register 7

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_ENET	Ethernet interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_SD	SD Card interface interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_MCPWM	Motor Control PWM interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_QEI	Quadrature Encoder Interface interrupt priority. See functional description for bits 7-3.

5.5.19 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 32 priorities, where 0 is the highest priority.

Table 71. Interrupt Priority Register 8

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_PLL1	PLL1 (Alt PLL) interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_USBACT	USB Activity interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_CANACT	CAN Activity interrupt priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_UART4	UART4 interrupt priority. See functional description for bits 7-3.

5.5.20 Interrupt Priority Register 9

The IPR9 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 72. Interrupt Priority Register 9

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_SSP2	SSP2 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_LCD	LCD controller interrupt priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_GPIO	Priority of GPIO interrupts. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_PWM0	PWM0 interrupt priority. See functional description for bits 7-3.

5.5.21 Interrupt Priority Register 10

The IPR10 register controls the priority of the eleventh group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 73. Interrupt Priority Register 10

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_EEPROM	EEPROM programming interrupt. 0 = highest priority. 31 (0x1F) = lowest priority.
31:8	Unimplemented	These bits ignore writes, and read as 0.

5.5.22 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 (see the ARM Cortex-M4 User Guide referred to in Section 40.1 for details).

Table 74. Software Trigger Interrupt Register

Bit	Name	Function
8:0	INTID	Writing a value to this field generates an interrupt for the specified Interrupt ID (see Table 51).
31:9	-	Reserved. Read value is undefined, only zero should be written.

6.1 Pin configuration

For information about the individual LPC408x/407x devices, refer to specific data sheets. Table 75 lists pins in order by pin name, and includes description of each potential pin function.

See the IOCON registers (Section 7.4.1) to configure pins for the desired function.

I/O pins are 5V tolerant and have input hysteresis unless otherwise indicated in the table below. Crystal pins, power pins, and reference voltage pins are not 5V tolerant. In addition, when pins are selected to be A to D converter inputs, they are no longer 5V tolerant and must be limited to the voltage at the ADC positive reference pin (VREFP).

Table 75. Pin description

Symbol	Type	IOCON select[1]	Description
P0[0] to P0[31]	I/O		**Port 0:** Port 0 provides up to 32 I/O pins, depending on the package. Each pin has individual direction control, pin mode configuration, and function selection.
P0[0]/ CAN_RD1/ U3_TXD/ I2C1_SDA/ U0_TXD	I/O	0	**P0[0] —** General purpose digital input/output pin.
	I	1	**CAN_RD1 —** CAN1 receiver input.
	O	2	**U3_TXD —** Transmitter output for UART 3.
	I/O	3	**I2C1_SDA —** I^2C1 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	I/O	4	**U0_TXD —** Transmitter output for UART 0.
P0[1]/ CAN1_TD/ U3_RXD/ I2C1_SCL/ U0_RXD	I/O	0	**P0[1] —** General purpose digital input/output pin.
	O	1	**CAN1_TD —** CAN1 transmitter output.
	I	2	**U3_RXD —** Receiver input for UART 3.
	I/O	3	**I2C1_SCL —** I^2C1 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	I	4	**U0_RXD —** Receiver input for UART 0.
P0[2]/ U0_TXD/ U3_TXD	I/O	0	**P0[2] —** General purpose digital input/output pin.
	O	1	**U0_TXD —** Transmitter output for UART 0. Used for ISP communication, see Section 38.1.
	O	2	**U3_TXD —** Transmitter output for UART 3.
P0[3]/ U0_RXD/ U3_RXD	I/O	0	**P0[3] —** General purpose digital input/output pin.
	I	1	**U0_RXD —** Receiver input for UART 0. Used for ISP communication, see Section 38.1.
	I	2	**U3_RXD —** Receiver input for UART 3.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P0[4]/ I2S_RX_SCK/ CAN_RD2/ T2_CAP0/ CMP_ROSC/ LCD_VD[0]	I/O	0	**P0[4] —** General purpose digital input/output pin.
	I/O	1	**I2S_RX_SCK —** I^2S Receive clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
	I	2	**CAN_RD2 —** CAN2 receiver input.
	I	3	**T2_CAP0 —** Capture input for Timer 2, channel 0.
	O	5	**CMP_ROSC —** Comparator relaxation oscillator output.
	O	7	**LCD_VD[0] —** LCD data.
P0[5]/ I2S_RX_WS/ CAN_TD2/ T2_CAP1/ CMP_RESET/ LCD_VD[1]	I/O	0	**P0[5] —** General purpose digital input/output pin.
	I/O	1	**I2S_RX_WS —** I^2S Receive word select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
	O	2	**CAN_TD2 —** CAN2 transmitter output.
	I	3	**T2_CAP1 —** Capture input for Timer 2, channel 1.
	O	5	**CMP_RESET —** Comparator reset input.
	O	7	**LCD_VD[1] —** LCD data.
P0[6]/ I2S_RX_SDA/ SSP1_SSEL/ T2_MAT0/ U1_RTS/ CMP_ROSC/ LCD_VD[8]	I/O	0	**P0[6] —** General purpose digital input/output pin.
	I/O	1	**I2S_RX_SDA —** I^2S Receive data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
	I/O	2	**SSP1_SSEL1 —** Slave Select for SSP1.
	O	3	**T2_MAT0 —** Match output for Timer 2, channel 0.
	O	4	**U1_RTS —** Request to Send output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
	O	5	**CMP_ROSC —** Comparator relaxation oscillator output.
	O	7	**LCD_VD[8] —** LCD data.
P0[7]/ I2S_TX_SCK/ SSP1_SCK/ T2_MAT1/ RTC_EV0/ CMP_VREF/ LCD_VD[9]	I/O	0	**P0[7] —** General purpose digital input/output pin.
	I/O	1	**I2S_TX_SCK —** I^2S transmit clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
	I/O	2	**SSP1_SCK —** Serial Clock for SSP1.
	O	3	**T2_MAT1 —** Match output for Timer 2, channel 1.
	I	4	**RTC_EV0 —** Event input 0 to Event Monitor/Recorder.
	O	5	**CMP_VREF —** Comparator voltage reference input.
	O	7	**LCD_VD[9] —** LCD data.
P0[8]/ I2S_TX_WS/ SSP1_MISO/ T2_MAT2/ RTC_EV1/ CMP1_IN[4]/ LCD_VD[16]	I/O	0	**P0[8] —** General purpose digital input/output pin.
	I/O	1	**I2S_TX_WS —** I^2S Transmit word select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
	I/O	2	**SSP1_MISO —** Master In Slave Out for SSP1.
	O	3	**T2_MAT2 —** Match output for Timer 2, channel 2.
	I	4	**RTC_EV1 —** Event input 1 to Event Monitor/Recorder.
	O	5	**CMP1_IN[4] —** Comparator input.
	O	7	**LCD_VD[16] —** LCD data.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P0[9]/ I2S_TX_SDA/ SSP1_MOSI/ T2_MAT3/ RTC_EV2/ CMP1_IN[3]/ LCD_VD[17]	I/O	0	**P0[9] —** General purpose digital input/output pin.
	I/O	1	**I2S_TX_SDA —** I^2S transmit data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
	I/O	2	**SSP1_MOSI —** Master Out Slave In for SSP1.
	O	3	**T2_MAT3 —** Match output for Timer 2, channel 3.
	I	4	**RTC_EV2 —** Event input 2 to Event Monitor/Recorder.
	O	5	**CMP1_IN[3] —** Comparator input.
	O	7	**LCD_VD[17] —** LCD data.
P0[10]/ U2_TXD/ I2C2_SDA/ T3_MAT0/ LCD_VD[5]	I/O	0	**P0[10] —** General purpose digital input/output pin.
	O	1	**U2_TXD —** Transmitter output for UART 2.
	I/O	2	**I2C2_SDA —** I^2C2 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	3	**T3_MAT0 —** Match output for Timer 3, channel 0.
	O	7	**LCD_VD[5] —** LCD data.
P0[11]/ U2_RXD/ I2C2_SCL/ T3_MAT1/ LCD_VD[10]	I/O	0	**P0[11] —** General purpose digital input/output pin.
	I	1	**U2_RXD —** Receiver input for UART 2.
	I/O	2	**I2C2_SCL —** I^2C2 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	3	**T3_MAT1 —** Match output for Timer 3, channel 1.
	O	7	**LCD_VD[10] —** LCD data.
P0[12]/ USB_PPWR2/ SSP1_MISO/ AD0[6]	I/O	0	**P0[12] —** General purpose digital input/output pin.
	O	1	**USB_PPWR2 —** Port Power enable signal for USB port 2.
	I/O	2	**SSP1_MISO —** Master In Slave Out for SSP1.
	I	3	**AD0[6] —** A/D converter 0, input 6. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
P0[13]/ USB_UP_LED2/ SSP1_MOSI/ AD0[7]	I/O	0	**P0[13] —** General purpose digital input/output pin.
	O	1	**USB_UP_LED2 —** USB port 2 GoodLink LED indicator. It is LOW when device is configured (non-control endpoints enabled) or when host is enabled and has detected a device on the bus. It is HIGH when the device is not configured, when host is enabled and has not detected a device on the bus, or during global suspend. It toggles between low and high when host is enabled and detects activity on the bus.
	I/O	2	**SSP1_MOSI —** Master Out Slave In for SSP1.
	I	3	**AD0[7] —** A/D converter 0, input 7. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
P0[14]/ USB_HSTEN2/ SSP1_SSEL/ USB_CONNECT2	I/O	0	**P0[14] —** General purpose digital input/output pin.
	O	1	**USB_HSTEN2 —** Host Enabled status for USB port 2.
	I/O	2	**SSP1_SSEL —** Slave Select for SSP1.
	O	3	**USB_CONNECT2 —** SoftConnect control for USB port 2. The USB_CONNECT pin indicates when the pull-up resistor must be enabled when running in USB device mode. If it is used in USB device mode, this function can be implemented by using another GPIO pin. If the chip is only used in USB host mode, there is no need to use this pin.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P0[15]/ U1_TXD/ SSP0_SCK/ SPIFI_IO[2]	I/O	0	**P0[15] —** General purpose digital input/output pin.
	O	1	**U1_TXD —** Transmitter output for UART 1.
	I/O	2	**SSP0_SCK —** Serial clock for SSP0.
	I/O	5	**SPIFI_IO[2] —** Data bit 2 for SPIFI.
P0[16]/ U1_RXD/ SSP0_ SSEL/ SPIFI_IO[3]	I/O	0	**P0 [16] —** General purpose digital input/output pin.
	I	1	**U1_RXD —** Receiver input for UART 1.
	I/O	2	**SSP0_SSEL —** Slave Select for SSP0.
	I/O	5	**SPIFI_IO[3] —** Data bit 3 for SPIFI.
P0[17]/ U1_CTS/ SSP0_MISO/ SPIFI_IO[1]	I/O	0	**P0[17] —** General purpose digital input/output pin.
	I	1	**U1_CTS —** Clear to Send input for UART 1.
	I/O	2	**SSP0_MISO —** Master In Slave Out for SSP0.
	I/O	5	**SPIFI_IO[1] —** Data bit 1 for SPIFI.
P0[18]/ U1_DCD/ SSP0_MOSI/ SPIFI_IO[0]	I/O	0	**P0[18] —** General purpose digital input/output pin.
	I	1	**U1_DCD —** Data Carrier Detect input for UART 1.
	I/O	2	**SSP0_MOSI —** Master Out Slave In for SSP0.
	I/O	5	**SPIFI_IO[0] —** Data bit 0 for SPIFI.
P0[19]/ U1_DSR/ SD_CLK/ I2C1_SDA/ LCD_VD[13]	I/O	0	**P0[19] —** General purpose digital input/output pin.
	I	1	**U1_DSR —** Data Set Ready input for UART 1.
	O	2	**SD_CLK —** Clock output line for SD card interface.
	I/O	3	**I2C1_SDA —** I^2C1 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	7	**LCD_VD[13] —** LCD data.
P0[20]/ U1_DTR/ SD_CMD/ I2C1_SCL/ LCD_VD[14]	I/O	0	**P0[20] —** General purpose digital input/output pin.
	O	1	**U1_DTR —** Data Terminal Ready output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
	I/O	2	**SD_CMD —** Command line for SD card interface.
	I/O	3	**I2C1_SCL —** I^2C1 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	7	**LCD_VD[14] —** LCD data.
P0[21]/ U1_RI/ SD_PWR/ U4_OE/ CAN_RD1/ U4_SCLK	I/O	0	**P0[21] —** General purpose digital input/output pin.
	I	1	**U1_RI —** Ring Indicator input for UART 1.
	O	2	**SD_PWR —** Power Supply Enable for external SD card power supply.
	O	3	**U4_OE —** RS-485/EIA-485 output enable signal for UART 4.
	I	4	**CAN_RD1 —** CAN1 receiver input.
	I/O	5	**U4_SCLK —** UART 4 clock input or output in synchronous mode.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P0[22]/ U1_RTS/ SD_DAT[0]/ U4_TXD/ CAN_TD1/ SPIFI_CLK	I/O	0	**P0[22] —** General purpose digital input/output pin.
	O	1	**U1_RTS —** Request to Send output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
	I/O	2	**SD_DAT[0] —** Data line 0 for SD card interface.
	O	3	**U4_TXD —** Transmitter output for UART 4 (input/output in smart card mode).
	O	4	**CAN_TD1 —** CAN1 transmitter output.
	O	5	**SPIFI_CLK —** Clock output for SPIFI.
P0[23]/ AD0[0]/ I2S_RX_SCK/ T3_CAP0	I/O	0	**P0[23] —** General purpose digital input/output pin.
	I	1	**AD0[0] —** A/D converter 0, input 0. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
	I/O	2	**I2S_RX_SCK —** Receive Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
	I	3	**T3_CAP0 —** Capture input for Timer 3, channel 0.
P0[24]/ AD0[1]/ I2S_RX_WS/ T3_CAP1	I/O	0	**P0[24] —** General purpose digital input/output pin.
	I	1	**AD0[1] —** A/D converter 0, input 1. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
	I/O	2	**I2S_RX_WS —** Receive Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
	I	3	**T3_CAP1 —** Capture input for Timer 3, channel 1.
P0[25]/ AD0[2]/ I2S_RX_SDA/ U3_TXD	I/O	0	**P0[25] —** General purpose digital input/output pin.
	I	1	**AD0[2] —** A/D converter 0, input 2. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
	I/O	2	**I2S_RX_SDA —** Receive data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
	O	3	**U3_TXD —** Transmitter output for UART 3.
P0[26]/ AD0[3]/ DAC_OUT/ U3_RXD	I/O	0	**P0[26] —** General purpose digital input/output pin.
	I	1	**AD0[3] —** A/D converter 0, input 3. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
	O	2	**DAC_OUT —** D/A converter output. When configured as the DAC output, the digital function of the pin must be disabled (see Section 7.4.1).
	I	3	**U3_RXD —** Receiver input for UART 3.
P0[27]/ I2C0_SDA/ USB_SDA	I/O	0	**P0[27] —** General purpose digital input/output pin.
	I/O	1	**I2C0_SDA —** I^2C0 data input/output. (this pin uses a specialized I^2C pad, see Section 22.1 for details).
	I/O	2	**USB_SDA —** I^2C serial data for communication with an external USB transceiver.
P0[28]/ I2C0_SCL/ USB_SCL	I/O	0	**P0[28] —** General purpose digital input/output pin.
	I/O	1	**I2C0_SCL0 —** I^2C0 clock input/output (this pin uses a specialized I^2C pad, see Section 22.1 for details).
	I/O	2	**USB_SCL —** I^2C serial clock for communication with an external USB transceiver.
P0[29]/ USB_D+1/ EINT0	I/O	0	**P0[29] —** General purpose digital input/output pin. When used as GPIO, P0[29] shares a direction control with P0[30].
	I/O	1	**USB_D+1 —** USB port 1 bidirectional D+ line.
	I	2	**EINT0 —** External interrupt 0 input.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P0[30]/ USB_D−1/ EINT1	I/O	0	**P0[30] —** General purpose digital input/output pin. When used as GPIO, P0[30] shares a direction control with P0[29].
	I/O	1	**USB_D−1 —** USB port 1 bidirectional D− line.
	I	2	**EINT1 —** External interrupt 1 input.
P0[31]/ USB_D+2	I/O	0	**P0[31] —** General purpose digital input/output pin.
	I/O	1	**USB_D+2 —** USB port 2 bidirectional D+ line.
P1[0] to P1[31]	I/O		**Port 1:** Port 1 provides up to 32 I/O pins, depending on the package. Each pin has individual direction control, pin mode configuration, and function selection.
P1[0]/ ENET_TXD0/ T3_CAP1/ SSP2_SCK	I/O	0	**P1[0] —** General purpose digital input/output pin.
	O	1	**ENET_TXD0 —** Ethernet transmit data 0 (RMII/MII interface).
	I	3	**T3_CAP1 —** Capture input for Timer 3, channel 1.
	I/O	4	**SSP2_SCK —** Serial clock for SSP2.
P1[1]/ ENET_TXD1/ T3_MAT3/ SSP2_MOSI	I/O	0	**P1[1] —** General purpose digital input/output pin.
	O	1	**ENET_TXD1 —** Ethernet transmit data 1 (RMII/MII interface).
	O	3	**T3_MAT3 —** Match output for Timer 3, channel 3.
	I/O	4	**SSP2_MOSI —** Master Out Slave In for SSP2.
P1[2]/ ENET_TXD2/ SD_CLK/ PWM0[1]	I/O	0	**P1[2] —** General purpose digital input/output pin.
	O	1	**ENET_TXD2 —** Ethernet transmit data 2 (MII interface).
	O	2	**SD_CLK —** Clock output line for SD card interface.
	O	3	**PWM0[1] —** Pulse Width Modulator 0, output 1.
P1[3]/ ENET_TXD3/ SD_CMD/ PWM0[2]	I/O	0	**P1[3] —** General purpose digital input/output pin.
	O	1	**ENET_TXD3 —** Ethernet transmit data 3 (MII interface).
	I/O	2	**SD_CMD —** Command line for SD card interface.
	O	3	**PWM0[2] —** Pulse Width Modulator 0, output 2.
P1[4]/ ENET_TX_EN/ T3_MAT2/ SSP2_MISO	I/O	0	**P1[4] —** General purpose digital input/output pin.
	O	1	**ENET_TX_EN —** Ethernet transmit data enable (RMII/MII interface).
	O	3	**T3_MAT2 —** Match output for Timer 3, channel 2.
	I/O	4	**SSP2_MISO —** Master In Slave Out for SSP2.
P1[5]/ ENET_TX_ER/ SD_PWR/ PWM0[3]/ CMP1_IN[2]	I/O	0	**P1[5] —** General purpose digital input/output pin.
	O	1	**ENET_TX_ER —** Ethernet Transmit Error (MII interface).
	O	2	**SD_PWR —** Power Supply Enable for external SD card power supply.
	O	3	**PWM0[3] —** Pulse Width Modulator 0, output 3.
	O	5	**CMP1_IN[2] —** Comparator input.
P1[6]/ ENET_TX_CLK/ SD_DAT[0]/ PWM0[4]/ CMP0_IN[4]	I/O	0	**P1[6] —** General purpose digital input/output pin.
	I	1	**ENET_TX_CLK —** Ethernet Transmit Clock (MII interface).
	I/O	2	**SD_DAT[0] —** Data line 0 for SD card interface.
	O	3	**PWM0[4] —** Pulse Width Modulator 0, output 4.
	O	5	**CMP0_IN[4] —** Comparator input.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P1[7]/ ENET_COL/ SD_DAT[1]/ PWM0[5] / CMP1_IN[1]	I/O	0	**P1[7] —** General purpose digital input/output pin.
	I	1	**ENET_COL —** Ethernet Collision detect (MII interface).
	I/O	2	**SD_DAT[1] —** Data line 1 for SD card interface.
	O	3	**PWM0[5] —** Pulse Width Modulator 0, output 5.
	O	5	**CMP1_IN[1] —** Comparator input.
P1[8]/ ENET_CRS (ENET_CRS_DV)/ T3_MAT1/ SSP2_SSEL	I/O	0	**P1[8] —** General purpose digital input/output pin.
	I	1	**ENET_CRS (ENET_CRS_DV) —** Ethernet Carrier Sense (MII interface) or Ethernet Carrier Sense/Data Valid (RMII interface).
	O	3	**T3_MAT1 —** Match output for Timer 3, channel 1.
	I/O	4	**SSP2_SSEL —** Slave Select for SSP2.
P1[9]/ ENET_RXD0/ T3_MAT0	I/O	0	**P1[9] —** General purpose digital input/output pin.
	I	1	**ENET_RXD0 —** Ethernet receive data 0 (RMII/MII interface).
	O	3	**T3_MAT0 —** Match output for Timer 3, channel 0.
P1[10]/ ENET_RXD1/ T3_CAP0	I/O	0	**P1[10] —** General purpose digital input/output pin.
	I	1	**ENET_RXD1 —** Ethernet receive data 1 (RMII/MII interface).
	I	3	**T3_CAP0 —** Capture input for Timer 3, channel 0.
P1[11]/ ENET_RXD2/ SD_DAT[2]/ PWM0[6]	I/O	0	**P1[11] —** General purpose digital input/output pin.
	I	1	**ENET_RXD2 —** Ethernet Receive Data 2 (MII interface).
	I/O	2	**SD_DAT[2] —** Data line 2 for SD card interface.
	O	3	**PWM0[6] —** Pulse Width Modulator 0, output 6.
P1[12]/ ENET_RXD3/ SD_DAT[3]/ PWM0_CAP0/ CMP1_OUT	I/O	0	**P1[12] —** General purpose digital input/output pin.
	I	1	**ENET_RXD3 —** Ethernet Receive Data (MII interface).
	I/O	2	**SD_DAT[3] —** Data line 3 for SD card interface.
	I	3	**PWM0_CAP0 —** Capture input for PWM0, channel 0.
	O	5	**CMP1_OUT —** Comparator 1 output.
P1[13]/ ENET_RX_DV	I/O	0	**P1[13] —** General purpose digital input/output pin.
	I	1	**ENET_RX_DV —** Ethernet Receive Data Valid (MII interface).
P1[14]/ ENET_RX_ER/ T2_CAP0/ CMP0_IN[1]	I/O	0	**P1[14] —** General purpose digital input/output pin.
	I	1	**ENET_RX_ER —** Ethernet receive error (RMII/MII interface).
	I	3	**T2_CAP0 —** Capture input for Timer 2, channel 0.
	O	5	**CMP0_IN[1] —** Comparator input.
P1[15]/ ENET_RX_CLK (ENET_REF_CLK)/ I2C2_SDA	I/O	0	**P1[15] —** General purpose digital input/output pin.
	I	1	**ENET_RX_CLK (ENET_REF_CLK) —** Ethernet Receive Clock (MII interface) or Ethernet Reference Clock (RMII interface).
	I/O	3	**I2C2_SDA —** I^2C2 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
P1[16]/ ENET_MDC/ I2S_TX_MCLK/ CMP0_IN[2]	I/O	0	**P1[16] —** General purpose digital input/output pin.
	O	1	**ENET_MDC —** Ethernet MIIM clock.
	O	2	**I2S_TX_MCLK —** I^2S transmitter master clock output.
	O	5	**CMP0_IN[2] —** Comparator input.

Table 75. Pin description *…continued*

Symbol	Type	IOCON select[1]	Description
P1[17]/ ENET_MDIO/ I2S_RX_MCLK/ CMP0_IN[3]	I/O	0	**P1[17] —** General purpose digital input/output pin.
	I/O	1	**ENET_MDIO —** Ethernet MIIM data input and output.
	O	2	**I2S_RX_MCLK —** I^2S receiver master clock output.
	O	5	**CMP0_IN[3] —** Comparator input.
P1[18]/ USB_UP_LED1/ PWM1[1]/ T1_CAP0/ SSP1_MISO	I/O	0	**P1[18] —** General purpose digital input/output pin.
	O	1	**USB_UP_LED1 —** USB port 1 GoodLink LED indicator. It is LOW when device is configured (non-control endpoints enabled) or when host is enabled and has detected a device on the bus. It is HIGH when the device is not configured, when host is enabled and has not detected a device on the bus, or during global suspend. It toggles between low and high when host is enabled and detects activity on the bus.
	O	2	**PWM1[1] —** Pulse Width Modulator 1, channel 1 output.
	I	3	**T1_CAP0 —** Capture input for Timer 1, channel 0.
	I/O	5	**SSP1_MISO —** Master In Slave Out for SSP1.
P1[19]/ $\overline{\text{USB\_TX\_E1}}$/ USB_PPWR1/ T1_CAP1/ MC_0A/ SSP1_SCK/ U2_OE	I/O	0	**P1[19] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{USB\_TX\_E1}}$ —** Transmit Enable signal for USB port 1 (OTG transceiver).
	O	2	**$\overline{\text{USB\_PPWR1}}$ —** Port Power enable signal for USB port 1.
	I	3	**T1_CAP1 —** Capture input for Timer 1, channel 1.
	O	4	**MC_0A —** Motor control PWM channel 0, output A.
	I/O	5	**SSP1_SCK —** Serial clock for SSP1.
	O	6	**U2_OE —** RS-485/EIA-485 output enable signal for UART 2.
P1[20]/ USB_TX_DP1/ PWM1[2]/ QEI_PHA/ MC_FB0/ SSP0_SCK/ LCD_VD[6]/ LCD_VD[10]	I/O	0	**P1[20] —** General purpose digital input/output pin.
	O	1	**USB_TX_DP1 —** D+ transmit data for USB port 1 (OTG transceiver).
	O	2	**PWM1[2] —** Pulse Width Modulator 1, channel 2 output.
	I	3	**QEI_PHA —** Quadrature Encoder Interface PHA input.
	I	4	**MC_FB0 —** Motor control PWM channel 0 feedback input.
	I/O	5	**SSP0_SCK0 —** Serial clock for SSP0.
	O	6	**LCD_VD[6] —** LCD data.
	O	7	**LCD_VD[10] —** LCD data.
P1[21]/ USB_TX_DM1/ PWM1[3]/ $\overline{\text{SSP0\_SSEL}}$/ $\overline{\text{MC\_ABORT}}$/ LCD_VD[7]/ LCD_VD[11]	I/O	0	**P1[21] —** General purpose digital input/output pin.
	O	1	**USB_TX_DM1 —** D− transmit data for USB port 1 (OTG transceiver).
	O	2	**PWM1[3] —** Pulse Width Modulator 1, channel 3 output.
	I/O	3	**$\overline{\text{SSP0\_SSEL}}$ —** Slave Select for SSP0.
	I	4	**$\overline{\text{MC\_ABORT}}$ —** Motor control PWM, active low fast abort.
	O	6	**LCD_VD[7] —** LCD data.
	O	7	**LCD_VD[11] —** LCD data.

Table 75. Pin description *…continued*

Symbol	Type	IOCON select[1]	Description
P1[22]/ USB_RCV1/ USB_PWRD1/ T1_MAT0/ MC_0B/ SSP1_MOSI/ LCD_VD[8]/ LCD_VD[12]	I/O	0	**P1[22] —** General purpose digital input/output pin.
	I	1	**USB_RCV1 —** Differential receive data for USB port 1 (OTG transceiver).
	I	2	**USB_PWRD1 —** Power Status for USB port 1 (host power switch). When using the chip in USB host mode, the USB_PWRD input must be enabled. The USB host controller will only detect a device connect event when the port power bit is set in the OHCI and the USB_PWRD bit is asserted for the corresponding port.
	O	3	**T1_MAT0 —** Match output for Timer 1, channel 0.
	O	4	**MC_0B —** Motor control PWM channel 0, output B.
	I/O	5	**SSP1_MOSI —** Master Out Slave In for SSP1.
	O	6	**LCD_VD[8] —** LCD data.
	O	7	**LCD_VD[12] —** LCD data.
P1[23]/ USB_RX_DP1/ PWM1[4]/ QEI_PHB/ MC_FB1/ SSP0_MISO/ LCD_VD[9]/ LCD_VD[13]	I/O	0	**P1[23] —** General purpose digital input/output pin.
	I	1	**USB_RX_DP1 —** D+ receive data for USB port 1 (OTG transceiver).
	O	2	**PWM1[4] —** Pulse Width Modulator 1, channel 4 output.
	I	3	**QEI_PHB —** Quadrature Encoder Interface PHB input.
	I	4	**MC_FB1 —** Motor control PWM channel 1 feedback input.
	I/O	5	**SSP0_MISO —** Master In Slave Out for SSP0.
	O	6	**LCD_VD[9] —** LCD data.
	O	7	**LCD_VD[13] —** LCD data.
P1[24]/ USB_RX_DM1/ PWM1[5]/ QEI_IDX/ MC_FB2/ SSP0_MOSI/ LCD_VD[10]/ LCD_VD[14]	I/O	0	**P1[24] —** General purpose digital input/output pin.
	I	1	**USB_RX_DM1 —** D− receive data for USB port 1 (OTG transceiver).
	O	2	**PWM1[5] —** Pulse Width Modulator 1, channel 5 output.
	I	3	**QEI_IDX —** Quadrature Encoder Interface INDEX input.
	I	4	**MC_FB2 —** Motor control PWM channel 2 feedback input.
	I/O	5	**SSP0_MOSI —** Master Out Slave in for SSP0.
	O	6	**LCD_VD[10]/LCD_VD[14] —** LCD data.
	O	7	**LCD_VD[10]/LCD_VD[14] —** LCD data.
P1[25]/ $\overline{\text{USB\_LS1}}$/ $\overline{\text{USB\_HSTEN1}}$/ T1_MAT1/ MC_1A/ CLKOUT/ LCD_VD[11]/ LCD_VD[15]	I/O	0	**P1[25] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{USB\_LS1}}$ —** Low Speed status for USB port 1 (OTG transceiver).
	O	2	**$\overline{\text{USB\_HSTEN1}}$ —** Host Enabled status for USB port 1.
	O	3	**T1_MAT1 —** Match output for Timer 1, channel 1.
	O	4	**MC_1A —** Motor control PWM channel 1, output A.
	O	5	**CLKOUT —** Selectable clock output.
	O	6	**LCD_VD[11] —** LCD data.
	O	7	**LCD_VD[15] —** LCD data.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P1[26]/ USB_SSPND1/ PWM1[6]/ T0_CAP0/ MC_1B/ SSP1_SSEL/ LCD_VD[12]/ LCD_VD[20]	I/O	0	**P1[26] —** General purpose digital input/output pin.
	O	1	**USB_SSPND1 —** USB port 1 Bus Suspend status (OTG transceiver).
	O	2	**PWM1[6] —** Pulse Width Modulator 1, channel 6 output.
	I	3	**T0_CAP0 —** Capture input for Timer 0, channel 0.
	O	4	**MC_1B —** Motor control PWM channel 1, output B.
	I/O	5	**SSP1_SSEL —** Slave Select for SSP1.
	O	6	**LCD_VD[12] —** LCD data.
	O	7	**LCD_VD[20] —** LCD data.
P1[27]/ USB_INT1/ USB_OVRCR1/ T0_CAP1/ CLKOUT/ LCD_VD[13]/ LCD_VD[21]	I/O	0	**P1[27] —** General purpose digital input/output pin.
	I	1	**USB_INT1 —** USB port 1 OTG transceiver interrupt (OTG transceiver).
	I	2	**USB_OVRCR1 —** USB port 1 Over-Current status. The USB_OVRCR pin is used to set status in the OHCI controller to inform the host firmware that there is an overcurrent condition. It is possible to use instead a GPIO pin and observe that pin for overcurrent situations.
	I	3	**T0_CAP1 —** Capture input for Timer 0, channel 1.
	O	4	**CLKOUT —** Selectable clock output.
	O	6	**LCD_VD[13] —** LCD data.
	O	7	**LCD_VD[21] —** LCD data.
P1[28]/ USB_SCL1/ PWM1_CAP0/ T0_MAT0/ MC_2A/ SSP0_SSEL/ LCD_VD[14]/ LCD_VD[22]	I/O	0	**P1[28] —** General purpose digital input/output pin.
	I/O	1	**USB_SCL1 —** USB port 1 I^2C serial clock (OTG transceiver).
	I	2	**PWM1_CAP0 —** Capture input for PWM1, channel 0.
	O	3	**T0_MAT0 —** Match output for Timer 0, channel 0.
	O	4	**MC_2A —** Motor control PWM channel 2, output A.
	I/O	5	**SSP0_SSEL —** Slave Select for SSP0.
	O	6	**LCD_VD[14] —** LCD data.
	O	7	**LCD_VD[22] —** LCD data.
P1[29]/ USB_SDA1/ PWM1_CAP1/ T0_MAT1/ MC_2B/ U4_TXD/ LCD_VD[15]/ LCD_VD[23]	I/O	0	**P1[29] —** General purpose digital input/output pin.
	I/O	1	**USB_SDA1 —** USB port 1 I^2C serial data (OTG transceiver).
	I	2	**PWM1_CAP1 —** Capture input for PWM1, channel 1.
	O	3	**T0_MAT1 —** Match output for Timer 0, channel 1.
	O	4	**MC_2B —** Motor control PWM channel 2, output B.
	O	5	**U4_TXD —** Transmitter output for UART 4 (input/output in smart card mode).
	O	6	**LCD_VD[15] —** LCD data.
	O	7	**LCD_VD[23] —** LCD data.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P1[30]/ USB_PWRD2/ USB_VBUS/ AD0[4]/ I2C0_SDA/ U3_OE	I/O	0	**P1[30] —** General purpose digital input/output pin.
	I	1	**USB_PWRD2 —** Power Status for USB port 2. When using the chip in USB host mode, the USB_PWRD input must be enabled. The USB host controller will only detect a device connect event when the port power bit is set in the OHCI and the USB_PWRD bit is asserted for the corresponding port.
	I	2	**USB_VBUS —** Monitors the presence of USB bus power. **Note:** This signal must be HIGH for USB reset to occur.
	I	3	**AD0[4] —** A/D converter 0, input 4. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
	I/O	4	**I2C0_SDA —** I^2C0 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	5	**U3_OE —** RS-485/EIA-485 output enable signal for UART 3.
P1[31]/ USB_OVRCR2/ SSP1_SCK/ AD0[5]/ I2C0_SCL	I/O	0	**P1[31] —** General purpose digital input/output pin.
	I	1	**USB_OVRCR2 —** Over-Current status for USB port 2. The USB_OVRCR pin is used to set status in the OHCI controller to inform the host firmware that there is an overcurrent condition. It is possible to use instead a GPIO pin and observe that pin for overcurrent situations.
	I/O	2	**SSP1_SCK —** Serial Clock for SSP1.
	I	3	**AD0[5] —** A/D converter 0, input 5. When configured as an ADC input, the digital function of the pin must be disabled (see Section 7.4.1).
	I/O	4	**I2C0_SCL —** I^2C0 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
P2[0] to P2[31]	I/O		**Port 2:** Port 2 provides up to 32 I/O pins, depending on the package. Each pin has individual direction control, pin mode configuration, and function selection.
P2[0]/ PWM1[1]/ U1_TXD/ LCD_PWR	I/O	0	**P2[0] —** General purpose digital input/output pin.
	O	1	**PWM1[1] —** Pulse Width Modulator 1, channel 1 output.
	O	2	**U1_TXD —** Transmitter output for UART 1.
	O	7	**LCD_PWR —** LCD panel power enable.
P2[1]/ PWM1[2]/ U1_RXD/ LCD_LE	I/O	0	**P2[1] —** General purpose digital input/output pin.
	O	1	**PWM1[2] —** Pulse Width Modulator 1, channel 2 output.
	I	2	**U1_RXD —** Receiver input for UART 1.
	O	7	**LCD_LE —** Line end signal.
P2[2]/ PWM1[3]/ U1_CTS/ T2_MAT3/ TRACEDATA[3]/ LCD_DCLK	I/O	0	**P2[2] —** General purpose digital input/output pin.
	O	1	**PWM1[3] —** Pulse Width Modulator 1, channel 3 output.
	I	2	**U1_CTS —** Clear to Send input for UART 1.
	O	3	**T2_MAT3 —** Match output for Timer 2, channel 3.
	O	5	**TRACEDATA[3] —** Trace data, bit 3.
	O	7	**LCD_DCLK —** LCD panel clock.

Table 75. Pin description *…continued*

Symbol	Type	IOCON select[1]	Description
P2[3]/ PWM1[4]/ U1_DCD/ T2_MAT2/ TRACEDATA[2]/ LCD_FP	I/O	0	**P2[3] —** General purpose digital input/output pin.
	O	1	**PWM1[4] —** Pulse Width Modulator 1, channel 4 output.
	I	2	**U1_DCD —** Data Carrier Detect input for UART 1.
	O	3	**T2_MAT2 —** Match output for Timer 2, channel 2.
	O	5	**TRACEDATA[2] —** Trace data, bit 2.
	O	7	**LCD_FP —** Frame pulse (STN). Vertical synchronization pulse (TFT).
P2[4]/ PWM1[5]/ U1_DSR/ T2_MAT1/ TRACEDATA[1]/ LCD_ENAB_M	I/O	0	**P2[4] —** General purpose digital input/output pin.
	O	1	**PWM1[5] —** Pulse Width Modulator 1, channel 5 output.
	I	2	**U1_DSR —** Data Set Ready input for UART 1.
	O	3	**T2_MAT1 —** Match output for Timer 2, channel 1.
	O	5	**TRACEDATA[1] —** Trace data, bit 1.
	O	7	**LCD_ENAB_M —** STN AC bias drive or TFT data enable output.
P2[5]/ PWM1[6]/ U1_DTR/ T2_MAT0/ TRACEDATA[0]/ LCD_LP	I/O	0	**P2[5] —** General purpose digital input/output pin.
	O	1	**PWM1[6] —** Pulse Width Modulator 1, channel 6 output.
	O	2	**U1_DTR —** Data Terminal Ready output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
	O	3	**T2_MAT0 —** Match output for Timer 2, channel 0.
	O	5	**TRACEDATA[0] —** Trace data, bit 0.
	O	7	**LCD_LP —** Line synchronization pulse (STN). Horizontal synchronization pulse (TFT).
P2[6]/ PWM1_CAP0/ U1_RI/ T2_CAP0/ U2_OE/ TRACECLK/ LCD_VD[0]/ LCD_VD[4]	I/O	0	**P2[6] —** General purpose digital input/output pin.
	I	1	**PWM1_CAP0 —** Capture input for PWM1, channel 0.
	I	2	**U1_RI —** Ring Indicator input for UART 1.
	I	3	**T2_CAP0 —** Capture input for Timer 2, channel 0.
	O	4	**U2_OE —** RS-485/EIA-485 output enable signal for UART 2.
	O	5	**TRACECLK —** Trace clock.
	O	6	**LCD_VD[0] —** LCD data.
	O	7	**LCD_VD[4] —** LCD data.
P2[7]/ CAN_RD2/ U1_RTS/ SPIFI_CS/ LCD_VD[1]/ LCD_VD[5]	I/O	0	**P2[7] —** General purpose digital input/output pin.
	I	1	**CAN_RD2 —** CAN2 receiver input.
	O	2	**U1_RTS —** Request to Send output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
	O	5	**SPIFI_CS —** Chip select output for SPIFI.
	O	6	**LCD_VD[1] —** LCD data.
	O	7	**LCD_VD[5] —** LCD data.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P2[8]/ CAN_TD2/ U2_TXD/ U1_CTS/ ENET_MDC/ LCD_VD[2]/ LCD_VD[6]	I/O	0	**P2[8] —** General purpose digital input/output pin.
	O	1	**CAN_TD2 —** CAN2 transmitter output.
	O	2	**U2_TXD —** Transmitter output for UART 2.
	I	3	**U1_CTS —** Clear to Send input for UART 1.
	O	4	**ENET_MDC —** Ethernet MIIM clock.
	O	6	**LCD_VD[2] —** LCD data.
	O	7	**LCD_VD[6] —** LCD data.
P2[9]/ USB_CONNECT1/ U2_RXD/ U4_RXD/ ENET_MDIO/ LCD_VD[3]/ LCD_VD[7]	I/O	0	**P2[9] —** General purpose digital input/output pin.
	O	1	**USB_CONNECT1 —** USB1 SoftConnect control. The USB_CONNECT pin indicates when the pull-up resistor must be enabled when running in USB device mode. If it is used in USB device mode, this function can be implemented by using another GPIO pin. If the chip is only used in USB host mode, there is no need to use this pin.
	I	2	**U2_RXD —** Receiver input for UART 2.
	I	3	**U4_RXD —** Receiver input for UART 4.
	I/O	4	**ENET_MDIO —** Ethernet MIIM data input and output.
	I	6	**LCD_VD[3] —** LCD data.
	I	7	**LCD_VD[7] —** LCD data.
P2[10]/ EINT0/ NMI	I/O	0	**P2[10] —** General purpose digital input/output pin. This pin includes a 5 ns input glitch filter. **Note:** A LOW on this pin while $\overline{\text{RESET}}$ is LOW forces the on-chip boot loader to take over control of the part after a reset and go into ISP mode. See Section 38.3.
	I	1	**EINT0 —** External interrupt 0 input.
	I	2	**NMI —** Non-maskable interrupt input.
P2[11]/ EINT1/ SD_DAT[1]/ I2S_TX_SCK/ LCD_CLKIN	I/O	0	**P2[11] —** General purpose digital input/output pin. This pin includes a 5 ns input glitch filter.
	I	1	**EINT1 —** External interrupt 1 input.
	I/O	2	**SD_DAT[1] —** Data line 1 for SD card interface.
	I/O	3	**I2S_TX_SCK —** Transmit Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
	I	7	**LCD_CLKIN —** LCD clock.
P2[12]/ EINT2/ SD_DAT[2]/ I2S_TX_WS/ LCD_VD[4]/ LCD_VD[3]/ LCD_VD[8]/ LCD_VD[18]	I/O	0	**P2[12] —** General purpose digital input/output pin. This pin includes a 5 ns input glitch filter.
	I	1	**EINT2 —** External interrupt 2 input.
	I/O	2	**SD_DAT[2] —** Data line 2 for SD card interface.
	I/O	3	**I2S_TX_WS —** Transmit Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
	O	4	**LCD_VD[4] —** LCD data.
	O	5	**LCD_VD[3] —** LCD data.
	O	6	**LCD_VD[8] —** LCD data.
	O	7	**LCD_VD[18] —** LCD data.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P2[13]/ EINT3/ SD_DAT[3]/ I2S_TX_SDA/ LCD_VD[5]/ LCD_VD[9]/ LCD_VD[19]	I/O	0	**P2[13] —** General purpose digital input/output pin. This pin includes a 5 ns input glitch filter.
	I	1	**EINT3 —** External interrupt 3 input.
	I/O	2	**SD_DAT[3] —** Data line 3 for SD card interface.
	I/O	3	**I2S_TX_SDA —** Transmit data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
	O	5	**LCD_VD[5] —** LCD data.
	O	6	**LCD_VD[9] —** LCD data.
	O	7	**LCD_VD[19] —** LCD data.
P2[14]/ $\overline{\text{EMC\_CS2}}$/ I2C1_SDA/ T2_CAP0	I/O	0	**P2[14] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_CS2}}$ —** LOW active Chip Select 2 signal.
	I/O	2	**I2C1_SDA —** I^2C1 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	I	3	**T2_CAP0 —** Capture input for Timer 2, channel 0.
P2[15]/ $\overline{\text{EMC\_CS3}}$/ I2C1_SCL/ T2_CAP1	I/O	0	**P2[15] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_CS3}}$ —** LOW active Chip Select 3 signal.
	I/O	2	**I2C1_SCL —** I^2C1 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	I	3	**T2_CAP1 —** Capture input for Timer 2, channel 1.
P2[16]/ $\overline{\text{EMC\_CAS}}$	I/O	0	**P2[16] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_CAS}}$ —** LOW active SDRAM Column Address Strobe.
P2[17]/ $\overline{\text{EMC\_RAS}}$	I/O	0	**P2[17] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_RAS}}$ —** LOW active SDRAM Row Address Strobe.
P2[18]/ EMC_CLK0	I/O	0	**P2[18] —** General purpose digital input/output pin.
	O	1	**EMC_CLK0 —** SDRAM clock 0.
P2[19]/ EMC_CLK1	I/O	0	**P2[19] —** General purpose digital input/output pin.
	O	1	**EMC_CLK1 —** SDRAM clock 1.
P2[20]/ $\overline{\text{EMC\_DYCS0}}$	I/O	0	**P2[20] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_DYCS0}}$ —** SDRAM chip select 0.
P2[21]/ $\overline{\text{EMC\_DYCS1}}$	I/O	0	**P2[21] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_DYCS1}}$ —** SDRAM chip select 1.
P2[22]/ $\overline{\text{EMC\_DYCS2}}$/ SSP0_SCK/ T3_CAP0	I/O	0	**P2[22] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_DYCS2}}$ —** SDRAM chip select 2.
	I/O	2	**SSP0_SCK —** Serial clock for SSP0.
	I	3	**T3_CAP0 —** Capture input for Timer 3, channel 0.
P2[23]/ $\overline{\text{EMC\_DYCS3}}$/ SSP0_SSEL/ T3_CAP1	I/O	0	**P2[23] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_DYCS3}}$ —** SDRAM chip select 3.
	I/O	2	**SSP0_SSEL —** Slave Select for SSP0.
	I	3	**T3_CAP1 —** Capture input for Timer 3, channel 1.
P2[24]/ EMC_CKE0	I/O	0	**P2[24] —** General purpose digital input/output pin.
	O	1	**EMC_CKE0 —** SDRAM clock enable 0.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P2[25]/ EMC_CKE1	I/O	0	**P2[25] —** General purpose digital input/output pin.
	O	1	**EMC_CKE1 —** SDRAM clock enable 1.
P2[26]/ EMC_CKE2/ SSP0_MISO/ T3_MAT0	I/O	0	**P2[26] —** General purpose digital input/output pin.
	O	1	**EMC_CKE2 —** SDRAM clock enable 2.
	I/O	2	**SSP0_MISO —** Master In Slave Out for SSP0.
	O	3	**T3_MAT0 —** Match output for Timer 3, channel 0.
P2[27]/ EMC_CKE3/ SSP0_MOSI/ T3_MAT1	I/O	0	**P2[27] —** General purpose digital input/output pin.
	O	1	**EMC_CKE3 —** SDRAM clock enable 3.
	I/O	2	**SSP0_MOSI —** Master Out Slave In for SSP0.
	O	3	**T3_MAT1 —** Match output for Timer 3, channel 1.
P2[28]/ EMC_DQM0	I/O	0	**P2[28] —** General purpose digital input/output pin.
	O	1	**EMC_DQM0 —** Data mask 0 used with SDRAM and static devices.
P2[29]/ EMC_DQM1	I/O	0	**P2[29] —** General purpose digital input/output pin.
	O	1	**EMC_DQM1 —** Data mask 1 used with SDRAM and static devices.
P2[30]/ EMC_DQM2/ I2C2_SDA/ T3_MAT2	I/O	0	**P2[30] —** General purpose digital input/output pin.
	O	1	**EMC_DQM2 —** Data mask 2 used with SDRAM and static devices.
	I/O	2	**I2C2_SDA —** I^2C2 data input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	3	**T3_MAT2 —** Match output for Timer 3, channel 2.
P2[31]/ EMC_DQM3/ I2C2_SCL/ T3_MAT3	I/O	0	**P2[31] —** General purpose digital input/output pin.
	O	1	**EMC_DQM3 —** Data mask 3 used with SDRAM and static devices.
	I/O	2	**I2C2_SCL —** I^2C2 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	3	**T3_MAT3 —** Match output for Timer 3, channel 3.
P3[0] to P3[31]	I/O		**Port 3:** Port 3 provides up to 32 I/O pins, depending on the package. Each pin has individual direction control, pin mode configuration, and function selection.
P3[0]/ EMC_D[0]	I/O	0	**P3[0] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[0] —** External memory data line 0.
P3[1]/ EMC_D[1]	I/O	0	**P3[1] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[1] —** External memory data line 1.
P3[2]/ EMC_D[2]	I/O	0	**P3[2] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[2] —** External memory data line 2.
P3[3]/ EMC_D[3]	I/O	0	**P3[3] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[3] —** External memory data line 3.
P3[4]/ EMC_D[4]	I/O	0	**P3[4] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[4] —** External memory data line 4.
P3[5]/ EMC_D[5]	I/O	0	**P3[5] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[5] —** External memory data line 5.
P3[6]/ EMC_D[6]	I/O	0	**P3[6] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[6] —** External memory data line 6.

Table 75. Pin description *…continued*

Symbol	Type	IOCON select[1]	Description
P3[7]/ EMC_D[7]	I/O	0	**P3[7] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[7] —** External memory data line 7.
P3[8]/ EMC_D[8]	I/O	0	**P3[8] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[8] —** External memory data line 8.
P3[9]/ EMC_D[9]	I/O	0	**P3[9] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[9] —** External memory data line 9.
P3[10]/ EMC_D[10]	I/O	0	**P3[10] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[10] —** External memory data line 10.
P3[11]/ EMC_D[11]	I/O	0	**P3[11] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[11] —** External memory data line 11.
P3[12]/ EMC_D[12]	I/O	0	**P3[12] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[12] —** External memory data line 12.
P3[13]/ EMC_D[13]	I/O	0	**P3[13] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[13] —** External memory data line 13.
P3[14]/ EMC_D[14]	I/O	0	**P3[14] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[14] —** External memory data line 14. On POR, this pin serves as the BOOT0 pin (see P3[15] description below.
P3[15]/ EMC_D[15]	I/O	0	**P3[15] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[15] —** External memory data line 15. BOOT[1:0] = 00 selects 8-bit external memory on $\overline{\text{EMC\_CS1}}$. BOOT[1:0] = 01 is reserved. Do not use. BOOT[1:0] = 10 selects 32-bit external memory on $\overline{\text{EMC\_CS1}}$. BOOT[1:0] = 11 selects 16-bit external memory on $\overline{\text{EMC\_CS1}}$.
P3[16]/ EMC_D[16]/ PWM0[1]/ U1_TXD	I/O	0	**P3[16] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[16] —** External memory data line 16.
	O	2	**PWM0[1] —** Pulse Width Modulator 0, output 1.
	O	3	**U1_TXD —** Transmitter output for UART 1.
P3[17]/ EMC_D[17]/ PWM0[2]/ U1_RXD	I/O	0	**P3[17] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[17] —** External memory data line 17.
	O	2	**PWM0[2] —** Pulse Width Modulator 0, output 2.
	I	3	**U1_RXD —** Receiver input for UART 1.
P3[18]/ EMC_D[18]/ PWM0[3]/ U1_CTS	I/O	0	**P3[18] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[18] —** External memory data line 18.
	O	2	**PWM0[3] —** Pulse Width Modulator 0, output 3.
	I	3	**U1_CTS —** Clear to Send input for UART 1.
P3[19]/ EMC_D[19]/ PWM0[4]/ U1_DCD	I/O	0	**P3[19] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[19] —** External memory data line 19.
	O	2	**PWM0[4] —** Pulse Width Modulator 0, output 4.
	I	3	**U1_DCD —** Data Carrier Detect input for UART 1.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P3[20]/ EMC_D[20]/ PWM0[5]/ U1_DSR	I/O	0	**P3[20] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[20] —** External memory data line 20.
	O	2	**PWM0[5] —** Pulse Width Modulator 0, output 5.
	I	3	**U1_DSR —** Data Set Ready input for UART 1.
P3[21]/ EMC_D[21]/ PWM0[6]/ U1_DTR	I/O	0	**P3[21] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[21] —** External memory data line 21.
	O	2	**PWM0[6] —** Pulse Width Modulator 0, output 6.
	O	3	**U1_DTR —** Data Terminal Ready output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
P3[22]/ EMC_D[22]/ PWM0_CAP0/ U1_RI	I/O	0	**P3[22] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[22] —** External memory data line 22.
	I	2	**PWM0_CAP0 —** Capture input for PWM0, channel 0.
	I	3	**U1_RI —** Ring Indicator input for UART 1.
P3[23]/ EMC_D[23]/ PWM1_CAP0/ T0_CAP0	I/O	0	**P3[23] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[23] —** External memory data line 23.
	I	2	**PWM1_CAP0 —** Capture input for PWM1, channel 0.
	I	3	**T0_CAP0 —** Capture input for Timer 0, channel 0.
P3[24]/ EMC_D[24]/ PWM1[1]/ T0_CAP1	I/O	0	**P3[24] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[24] —** External memory data line 24.
	O	2	**PWM1[1] —** Pulse Width Modulator 1, output 1.
	I	3	**T0_CAP1 —** Capture input for Timer 0, channel 1.
P3[25]/ EMC_D[25]/ PWM1[2]/ T0_MAT0	I/O	0	**P3[25] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[25] —** External memory data line 25.
	O	2	**PWM1[2] —** Pulse Width Modulator 1, output 2.
	O	3	**T0_MAT0 —** Match output for Timer 0, channel 0.
P3[26]/ EMC_D[26]/ PWM1[3]/ T0_MAT1/ STCLK	I/O	0	**P3[26] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[26] —** External memory data line 26.
	O	2	**PWM1[3] —** Pulse Width Modulator 1, output 3.
	O	3	**T0_MAT1 —** Match output for Timer 0, channel 1.
	I	4	**STCLK —** System tick timer clock input.
P3[27]/ EMC_D[27]/ PWM1[4]/ T1_CAP0	I/O	0	**P3[27] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[27] —** External memory data line 27.
	O	2	**PWM1[4] —** Pulse Width Modulator 1, output 4.
	I	3	**T1_CAP0 —** Capture input for Timer 1, channel 0.
P3[28]/ EMC_D[28]/ PWM1[5]/ T1_CAP1	I/O	0	**P3[28] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[28] —** External memory data line 28.
	O	2	**PWM1[5] —** Pulse Width Modulator 1, output 5.
	I	3	**T1_CAP1 —** Capture input for Timer 1, channel 1.

Table 75. Pin description *…continued*

Symbol	Type	IOCON select[1]	Description
P3[29]/ EMC_D[29]/ PWM1[6]/ T1_MAT0	I/O	0	**P3[29] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[29] —** External memory data line 29.
	O	2	**PWM1[6] —** Pulse Width Modulator 1, output 6.
	O	3	**T1_MAT0 —** Match output for Timer 1, channel 0.
P3[30]/ EMC_D[30]/ U1_RTS/ T1_MAT1	I/O	0	**P3[30] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[30] —** External memory data line 30.
	O	2	**U1_RTS —** Request to Send output for UART 1. Can also be configured to be an RS-485/EIA-485 output enable signal for UART 1.
	O	3	**T1_MAT1 —** Match output for Timer 1, channel 1.
P3[31]/ EMC_D[31]/ T1_MAT2	I/O	0	**P3[31] —** General purpose digital input/output pin.
	I/O	1	**EMC_D[31] —** External memory data line 31.
	O	3	**T1_MAT2 —** Match output for Timer 1, channel 2.
P4[0] to P4[31]	I/O		**Port 4:** Port 4 provides up to 32 I/O pins, depending on the package. Each pin has individual direction control, pin mode configuration, and function selection.
P4[0]/ EMC_A[0]	I/O	0	**P4[0] —**]General purpose digital input/output pin.
	I/O	1	**EMC_A[0] —** External memory address line 0.
P4[1]/ EMC_A[1]	I/O	0	**P4[1] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[1] —** External memory address line 1.
P4[2]/ EMC_A[2]	I/O	0	**P4[2] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[2] —** External memory address line 2.
P4[3]/ EMC_A[3]	I/O	0	**P4[3] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[3] —** External memory address line 3.
P4[4]/ EMC_A[4]	I/O	0	**P4[4] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[4] —** External memory address line 4.
P4[5]/ EMC_A[5]	I/O	0	**P4[5] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[5] —** External memory address line 5.
P4[6]/ EMC_A[6]	I/O	0	**P4[6] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[6] —** External memory address line 6.
P4[7]/ EMC_A[7]	I/O	0	**P4[7] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[7] —** External memory address line 7.
P4[8]/ EMC_A[8]	I/O	0	**P4[8] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[8] —** External memory address line 8.
P4[9]/ EMC_A[9]	I/O	0	**P4[9] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[9] —** External memory address line 9.
P4[10]/ EMC_A[10]	I/O	0	**P4[10] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[10] —** External memory address line 10.
P4[11]/ EMC_A[11]	I/O	0	**P4[11] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[11] —** External memory address line 11.
P4[12]/ EMC_A[12]	I/O	0	**P4[12] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[12] —** External memory address line 12.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P4[13]/ EMC_A[13]	I/O	0	**P4[13] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[13] —** External memory address line 13.
P4[14]/ EMC_A[14]	I/O	0	**P4[14] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[14] —** External memory address line 14.
P4[15]/ EMC_A[15]	I/O	0	**P4[15] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[15] —** External memory address line 15.
P4[16]/ EMC_A[16]	I/O	0	**P4[16] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[16] —** External memory address line 16.
P4[17]/ EMC_A[17]	I/O	0	**P4[17] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[17] —** External memory address line 17.
P4[18]/ EMC_A[18]	I/O	0	**P4[18] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[18] —** External memory address line 18.
P4[19]/ EMC_A[19]	I/O	0	**P4[19] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[19] —** External memory address line 19.
P4[20]/ EMC_A[20]/ I2C2_SDA/ SSP1_SCK	I/O	0	**P4[20] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[20] —** External memory address line 20.
	I/O	2	**I2C2_SDA —** I^2C2 data input/output ((this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	I/O	3	**SSP1_SCK —** Serial Clock for SSP1.
P4[21]/ EMC_A[21]/ I2C2_SCL/ SSP1_SSEL	I/O	0	**P4[21] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[21] —** External memory address line 21.
	I/O	2	**I2C2_SCL —** I^2C2 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	I/O	3	**SSP1_SSEL —** Slave Select for SSP1.
P4[22]/ EMC_A[22]/ U2_TXD/ SSP1_MISO	I/O	0	**P4[22] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[22] —** External memory address line 22.
	O	2	**U2_TXD —** Transmitter output for UART 2.
	I/O	3	**SSP1_MISO —** Master In Slave Out for SSP1.
P4[23]/ EMC_A[23]/ U2_RXD/ SSP1_MOSI	I/O	0	**P4[23] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[23] —** External memory address line 23.
	I	2	**U2_RXD —** Receiver input for UART 2.
	I/O	3	**SSP1_MOSI —** Master Out Slave In for SSP1.
P4[24]/ $\overline{\text{EMC\_OE}}$	I/O	0	**P4[24] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_OE}}$ —** LOW active Output Enable signal.
P4[25]/ $\overline{\text{EMC\_WE}}$	I/O	0	**P4[25] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_WE}}$ —** LOW active Write Enable signal.
P4[26]/ $\overline{\text{EMC\_BLS0}}$	I/O	0	**P4[26] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_BLS0}}$ —** LOW active Byte Lane select signal 0.
P4[27]/ $\overline{\text{EMC\_BLS1}}$	I/O	0	**P4[27] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_BLS1}}$ —** LOW active Byte Lane select signal 1.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P4[28]/ $\overline{\text{EMC\_BLS2}}$/ U3_TXD/ T2_MAT0/ LCD_VD[6]/ LCD_VD[10]/ LCD_VD[2]	I/O	0	**P4 [28] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_BLS2}}$ —** LOW active Byte Lane select signal 2.
	O	2	**TXD3 —** Transmitter output for UART 3.
	O	3	**T2_MAT0 —** Match output for Timer 2, channel 0.
	O	5	**LCD_VD[6] —** LCD data.
	O	6	**LCD_VD[10] —** LCD data.
	O	7	**LCD_VD[2] —** LCD data.
P4[29]/ $\overline{\text{EMC\_BLS3}}$/ U3_RXD/ T2_MAT1/ I2C2_SCL/ LCD_VD[7]/ LCD_VD[11]/ LCD_VD[3]	I/O	0	**P4[29] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_BLS3}}$ —** LOW active Byte Lane select signal 3.
	I	2	**U3_RXD —** Receiver input for UART 3.
	O	3	**T2_MAT1 —** Match output for Timer 2, channel 1.
	I/O	4	**I2C2_SCL —** I^2C2 clock input/output (this pin does not use a specialized I^2C pad, see Section 22.1 for details).
	O	5	**LCD_VD[7] —** LCD data.
	O	6	**LCD_VD[11] —** LCD data.
	O	7	**LCD_VD[3] —** LCD data.
P4[30]/ $\overline{\text{EMC\_CS0}}$/ CMP0_OUT	I/O	0	**P4[30] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_CS0}}$ —** LOW active Chip Select 0 signal.
	O	5	**CMP0_OUT —** Comparator 0 output.
P4[31]/ $\overline{\text{EMC\_CS1}}$	I/O	0	**P4[31] —** General purpose digital input/output pin.
	O	1	**$\overline{\text{EMC\_CS1}}$ —** LOW active Chip Select 1 signal.
P5[0] to P5[4]	I/O		**Port 5:** Port 5 provides up to 5 I/O pins, depending on the package. Each pin has individual direction control, pin mode configuration, and function selection.
P5[0]/ EMC_A[24]/ SSP2_MOSI/ T2_MAT2	I/O	0	**P5[0] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[24] —** External memory address line 24.
	I/O	2	**SSP2_MOSI —** Master Out Slave In for SSP2.
	O	3	**T2_MAT2 —** Match output for Timer 2, channel 2.
P5[1]/ EMC_A[25]/ SSP2_MISO/ T2_MAT3	I/O	0	**P5[1] —** General purpose digital input/output pin.
	I/O	1	**EMC_A[25] —** External memory address line 25.
	I/O	2	**SSP2_MISO —** Master In Slave Out for SSP2.
	O	3	**T2_MAT3 —** Match output for Timer 2, channel 3.
P5[2]/ T3_MAT2/ I2C0_SDA	I/O	0	**P5[2] —** General purpose digital input/output pin.
	I/O	2	**SSP2_SCK —** Serial clock for SSP2. When using this pin, the SSP2 bit rate is limited to 1 MHz.
	O	3	**T3_MAT2 —** Match output for Timer 3, channel 2.
	I/O	5	**I2C0_SDA —** I^2C0 data input/output (this pin uses a specialized I^2C pad that supports I^2C Fast Mode Plus).

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
P5[3]/ U4_RXD/ I2C0_SCL	I/O	0	**P5[3] —** General purpose digital input/output pin.
	I/O	2	**SSP2_SSEL —** Slave select for SSP2. When using this pin, the SSP2 bit rate is limited to 1 MHz.
	I	4	**U4_RXD —** Receiver input for UART 4.
	I/O	5	**I2C0_SCL0 —** I^2C0 clock input/output (this pin uses a specialized I^2C pad that supports I^2C Fast Mode Plus.
P5[4]/ U0_OE/ T3_MAT3/ U4_TXD	I/O	0	**P5[4] —** General purpose digital input/output pin.
	O	1	**U0_OE —** RS-485/EIA-485 output enable signal for UART 0.
	O	3	**T3_MAT3 —** Match output for Timer 3, channel 3.
	O	4	**U4_TXD —** Transmitter output for UART 4 (input/output in smart card mode).
RTC_ALARM	O		**RTC_ALARM —** RTC controlled output. This pin has a low drive strength and is powered by V_{BAT} (see data sheet for details). It is driven high when an RTC alarm is generated.
USB_D−2	I/O		**USB_D−2 —** USB port 2 bidirectional D− line.
JTAG_TDO (SWO)	O		**JTAG_TDO —** Test Data Out for JTAG interface. **SWO —** Serial wire trace output.
JTAG_TDI	I		**TDI —** Test Data In for JTAG interface. This pin includes an internal pull-up, see Section 39.1.
JTAG_TMS (SWDIO)	I		**TMS —** Test Mode Select for JTAG interface. This pin includes an internal pull-up, see Section 39.1. **SWDIO —** Serial wire debug data input/output.
$\overline{\text{JTAG\_TRST}}$	I		$\overline{\textbf{TRST}}$ **—** Test Reset for JTAG interface. This pin includes an internal pull-up, see Section 39.1.
JTAG_TCK (SWDCLK)	I		**TCK —** Test Clock for JTAG interface. This clock must be slower than 1/6 of the CPU clock (CCLK) for the JTAG interface to operate. **SWDCLK —** Serial wire clock.
$\overline{\text{RSTOUT}}$	O		Reset status output. A LOW output on this pin indicates that the device is in the reset state, for any reason. This reflects the $\overline{\text{RESET}}$ input pin and all internal reset sources.
$\overline{\text{RESET}}$	I		External reset input. A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. This pin includes a 20 ns input glitch filter.
XTAL1[2]	I		Input to the oscillator circuit and internal clock generator circuits.
XTAL2[2]	O		Output from the oscillator amplifier.
RTCX1[2]	I		Input to the RTC 32 kHz ultra-low power oscillator circuit.
RTCX2[2]	O		Output from the RTC 32 kHz ultra-low power oscillator circuit.
V_{SS}[2]	I		**ground:** 0 V reference for digital IO pins.
V_{SSREG}[2]	I		**ground:** 0 V reference for internal logic.
V_{SSA}[2]	I		**analog ground:** 0 V power supply and reference for the ADC and DAC. This should be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{DD(3V3)}$[2]	I		**3.3 V supply voltage:** This is the power supply voltage for I/O other than pins in the Vbat domain.
$V_{DD(REG)(3V3)}$[2]	I		**3.3 V regulator supply voltage:** This is the power supply for the on-chip voltage regulator that supplies internal logic.

Table 75. Pin description *...continued*

Symbol	Type	IOCON select[1]	Description
V_{DDA}[2]	I		**analog 3.3 V pad supply voltage:** This can be connected to the same supply as $V_{DD(3V3)}$ but should be isolated to minimize noise and error. This voltage is used to power the ADC and DAC. **Note: this pin should be tied to 3.3V if the ADC and DAC are not used.**
VREFP[2]	I		**ADC positive reference voltage:** This should be the same voltage as V_{DDA}, but should be isolated to minimize noise and error. The voltage level on this pin is used as a reference for ADC and DAC. **Note: this pin should be tied to 3.3V if the ADC and DAC are not used.**
VBAT[2]	I		**RTC power supply:** 3.3 V on this pin supplies power to the RTC.

[1] These values are used in the FUNC field of the IOCON registers, described in Section 7.4.1.

[2] These pins provide special analog functionality.

7.1 Introduction

A separate register is provided to configure each GPIO pin. This configuration includes which internal function is connected to the pin, the output mode (plain, pull-up, pull-down, or repeater), open drain mode control, hysteresis enable, slew rate control, and buffer setup for analog functions. Some pins include additional special controls, such as for I^2C buffer modes. These registers are summarized in Table 76.

Table 76. Summary of I/O pin configuration registers

Port	Registers	Detail Table
Port 0 pins	IOCON_P0_nn, where nn is the port pin number, from 0 to 31 [1]	Table 77
Port 1 pins	IOCON_P1_nn, where nn is the port pin number, from 0 to 31 [1]	Table 78
Port 2 pins	IOCON_P2_nn, where nn is the port pin number, from 0 to 31 [1]	Table 79
Port 3 pins	IOCON_P3_nn, where nn is the port pin number, from 0 to 31 [1]	Table 80
Port 4 pins	IOCON_P4_nn, where nn is the port pin number, from 0 to 31 [1]	Table 81
Port 5 pins	IOCON_P5_nn, where nn is the port pin number, from 0 to 4 [1]	Table 82

[1] Which pins are available depends on the part number and package combination.

7.2 Description

The pin connect block allows most pins of the microcontroller to have more than one potential function. Configuration registers control the multiplexers to allow connection between the pin and the on-chip peripherals.

Peripherals should be connected to the appropriate pins prior to being activated and prior to any related interrupt(s) being enabled. Activity of any enabled peripheral function that is not mapped to a related pin should be considered undefined.

Selection of a single function on a port pin excludes other peripheral functions available on the same pin. However, the GPIO input stays connected and may be read by software or used to contribute to the GPIO interrupt feature.

7.3 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 7.4.1. 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.

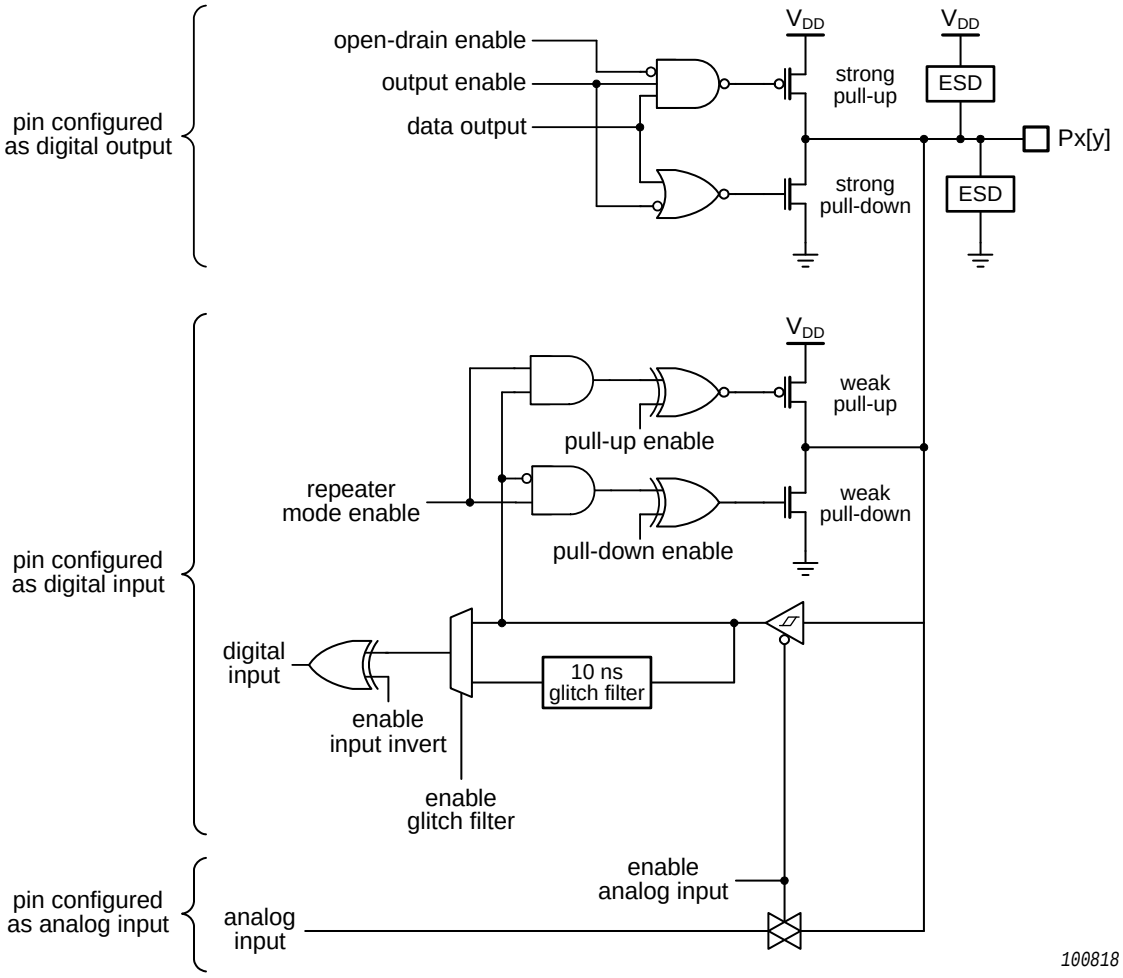


Fig 14. I/O configurations

7.3.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (typically value 000) or to a special function. For pins set to GPIO, the FIOnDIR registers determine whether the pin is configured as an input or output (see Section 8.5.1.1). For any special function, the pin direction is controlled automatically depending on the function. The FIOnDIR registers have no effect for special functions.

7.3.2 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.

7.3.3 Hysteresis

The input buffer for digital functions can be configured with or without hysteresis. See the appropriate specific device data sheet for quantitative details.

7.3.4 Input Inversion

This option is included to save users from having to include an external inverter on an input that is only available in the opposite polarity from an external source. Do not set this option on a GPIO output. Doing so can result in inadvertent toggling of an output with input inversion selected, as a result of operations on other pins in the same port. For example, if software reads a GPIO Port register, modifies other bits/outputs in the value, and writes the result back to the Port register, any output in the port that has input inversion selected will change state.

7.3.5 Analog/digital mode

In Analog mode, the analog input connection is enabled. In digital mode, the analog input connection is disabled. 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.

If Analog mode is selected, the MODE field should be “Inactive” (00); the HYS, INV, FILTR, SLEW, and OD settings have no effect. For an unconnected pin that has an analog function, keep the ADMODE bit set to 1 (digital mode), and pull-up or pull-down mode selected in the MODE field.

7.3.6 Input filter

Type A and W pins include a filter that can be selectively enabled. The filter suppresses input pulses smaller than about 10 ns.

7.3.7 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.

7.3.8 I²C modes

Pins that support I²C with specialized pad electronics (P0[27], P0[28], P5[2], and P5[3]) have additional configuration bits. These are not hardwired so that the pins can be more easily used for non-I²C functions.

The HS bit applies to standard, Fast-mode, and Fast-mode Plus I²C, and is available for all the pins noted above.

The HIDRIVE bit applies only to pins P5[2] and P5[3], and is used to select between Standard mode and Fast-mode I²C or Fast-mode Plus I²C.

- For any I²C mode, clear the HS bit so that the input glitch filter is enabled. Clear the HIDRIVE bit if it exists for that pin to select the correct drive strength for Standard mode or Fast-mode I²C
- For Fast-mode Plus I²C operation, set the HIDRIVE bit to select the correct drive strength for Fast-mode Plus I²C.
- For non-I²C operation, these pins remain open-drain and can only drive low, regardless of how HS and HIDRIVE 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 HS = 1 and HIDRIVE = 0 (if applicable) to maximize compatibility with other GPIO pins.

7.3.9 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 its FIOnDIR 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.

7.3.10 DAC enable

The pin that includes the DAC output as a potential function includes an enable for the function that must be set if the DAC output is used.

7.4 Register description

The pin connect block contains an I/O Control register (IOCON) for each pin that has programmable attributes, and selects peripheral functions associated with that pin. These registers are shown by GPIO port number in Tables 7–77 through 7–82.

Table 77. I/O Control registers for port 0

Port pin	Register	Access	Reset Value[1]	Address	IOCON type[2]	208-pin	180-pin	144-pin	80-pin
P0[0]	IOCON_P0_0	R/W	0x030	0x4002 C000	D (tables 83, 84)	x	x	x	x
P0[1]	IOCON_P0_1	R/W	0x030	0x4002 C004	D (tables 83, 84)	x	x	x	x
P0[2]	IOCON_P0_2	R/W	0x030	0x4002 C008	D (tables 83, 84)	x	x	x	x
P0[3]	IOCON_P0_3	R/W	0x030	0x4002 C00C	D (tables 83, 84)	x	x	x	x
P0[4]	IOCON_P0_4	R/W	0x030	0x4002 C010	D (tables 83, 84)	x	x	x	-
P0[5]	IOCON_P0_5	R/W	0x030	0x4002 C014	D (tables 83, 84)	x	x	x	-
P0[6]	IOCON_P0_6	R/W	0x030	0x4002 C018	D (tables 83, 84)	x	x	x	x
P0[7]	IOCON_P0_7	R/W	0x0A0	0x4002 C01C	W (tables 91, 92)	x	x	x	x
P0[8]	IOCON_P0_8	R/W	0x0A0	0x4002 C020	W (tables 91, 92)	x	x	x	x
P0[9]	IOCON_P0_9	R/W	0x0A0	0x4002 C024	W (tables 91, 92)	x	x	x	x
P0[10]	IOCON_P0_10	R/W	0x030	0x4002 C028	D (tables 83, 84)	x	x	x	x
P0[11]	IOCON_P0_11	R/W	0x030	0x4002 C02C	D (tables 83, 84)	x	x	x	x
P0[12]	IOCON_P0_12	R/W	0x1B0	0x4002 C030	A (tables 85, 86)	x	x	x	-
P0[13]	IOCON_P0_13	R/W	0x1B0	0x4002 C034	A (tables 85, 86)	x	x	x	-
P0[14]	IOCON_P0_14	R/W	0x030	0x4002 C038	D (tables 83, 84)	x	x	x	-
P0[15]	IOCON_P0_15	R/W	0x030	0x4002 C03C	D (tables 83, 84)	x	x	x	x
P0[16]	IOCON_P0_16	R/W	0x030	0x4002 C040	D (tables 83, 84)	x	x	x	x
P0[17]	IOCON_P0_17	R/W	0x030	0x4002 C044	D (tables 83, 84)	x	x	x	x
P0[18]	IOCON_P0_18	R/W	0x030	0x4002 C048	D (tables 83, 84)	x	x	x	x
P0[19]	IOCON_P0_19	R/W	0x030	0x4002 C04C	D (tables 83, 84)	x	x	x	-
P0[20]	IOCON_P0_20	R/W	0x030	0x4002 C050	D (tables 83, 84)	x	x	x	-
P0[21]	IOCON_P0_21	R/W	0x030	0x4002 C054	D (tables 83, 84)	x	x	x	-
P0[22]	IOCON_P0_22	R/W	0x030	0x4002 C058	D (tables 83, 84)	x	x	x	x
P0[23]	IOCON_P0_23	R/W	0x1B0	0x4002 C05C	A (tables 85, 86)	x	x	x	-
P0[24]	IOCON_P0_24	R/W	0x1B0	0x4002 C060	A (tables 85, 86)	x	x	x	-
P0[25]	IOCON_P0_25	R/W	0x1B0	0x4002 C064	A (tables 85, 86)	x	x	x	x
P0[26]	IOCON_P0_26	R/W	0x1B0	0x4002 C068	A (tables 85, 86)	x	x	x	x
P0[27]	IOCON_P0_27	R/W	0	0x4002 C06C	I (tables 89, 90)	x	x	x	-
P0[28]	IOCON_P0_28	R/W	0	0x4002 C070	I (tables 89, 90)	x	x	x	-
P0[29]	IOCON_P0_29	R/W	0	0x4002 C074	U (tables 87, 88)	x	x	x	x
P0[30]	IOCON_P0_30	R/W	0	0x4002 C078	U (tables 87, 88)	x	x	x	x
P0[31]	IOCON_P0_31	R/W	0	0x4002 C07C	U (tables 87, 88)	x	x	x	-

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

[2] IOCON types are D (standard digital pin), and other pins with a specialized function: A (analog), U (USB), I (I2C), and W.

Table 78. I/O Control registers for port 1

Port pin	Register	Access	Reset Value[1]	Address	IOCON type	208-pin	180-pin	144-pin	80-pin
P1[0]	IOCON_P1_0	R/W	0x030	0x4002 C080	D (tables 83, 84)	x	x	x	x
P1[1]	IOCON_P1_1	R/W	0x030	0x4002 C084	D (tables 83, 84)	x	x	x	x
P1[2]	IOCON_P1_2	R/W	0x030	0x4002 C088	D (tables 83, 84)	x	x	-	-
P1[3]	IOCON_P1_3	R/W	0x030	0x4002 C08C	D (tables 83, 84)	x	x	-	-
P1[4]	IOCON_P1_4	R/W	0x030	0x4002 C090	D (tables 83, 84)	x	x	x	x
P1[5]	IOCON_P1_5	R/W	0x0B0	0x4002 C094	W (tables 91, 92)	x	x	-	-
P1[6]	IOCON_P1_6	R/W	0x0B0	0x4002 C098	W (tables 91, 92)	x	x	-	-
P1[7]	IOCON_P1_7	R/W	0x0B0	0x4002 C09C	W (tables 91, 92)	x	x	-	-
P1[8]	IOCON_P1_8	R/W	0x030	0x4002 C0A0	D (tables 83, 84)	x	x	x	x
P1[9]	IOCON_P1_9	R/W	0x030	0x4002 C0A4	D (tables 83, 84)	x	x	x	x
P1[10]	IOCON_P1_10	R/W	0x030	0x4002 C0A8	D (tables 83, 84)	x	x	x	x
P1[11]	IOCON_P1_11	R/W	0x030	0x4002 C0AC	D (tables 83, 84)	x	x	-	-
P1[12]	IOCON_P1_12	R/W	0x030	0x4002 C0B0	D (tables 83, 84)	x	x	-	-
P1[13]	IOCON_P1_13	R/W	0x030	0x4002 C0B4	D (tables 83, 84)	x	x	-	-
P1[14]	IOCON_P1_14	R/W	0x0B0	0x4002 C0B8	W (tables 91, 92)	x	x	x	x
P1[15]	IOCON_P1_15	R/W	0x030	0x4002 C0BC	D (tables 83, 84)	x	x	x	x
P1[16]	IOCON_P1_16	R/W	0x0B0	0x4002 C0C0	W (tables 91, 92)	x	x	x	-
P1[17]	IOCON_P1_17	R/W	0x0B0	0x4002 C0C4	W (tables 91, 92)	x	x	x	-
P1[18]	IOCON_P1_18	R/W	0x030	0x4002 C0C8	D (tables 83, 84)	x	x	x	x
P1[19]	IOCON_P1_19	R/W	0x030	0x4002 C0CC	D (tables 83, 84)	x	x	x	x
P1[20]	IOCON_P1_20	R/W	0x030	0x4002 C0D0	D (tables 83, 84)	x	x	x	x
P1[21]	IOCON_P1_21	R/W	0x030	0x4002 C0D4	D (tables 83, 84)	x	x	x	-
P1[22]	IOCON_P1_22	R/W	0x030	0x4002 C0D8	D (tables 83, 84)	x	x	x	x
P1[23]	IOCON_P1_23	R/W	0x030	0x4002 C0DC	D (tables 83, 84)	x	x	x	x
P1[24]	IOCON_P1_24	R/W	0x030	0x4002 C0E0	D (tables 83, 84)	x	x	x	x
P1[25]	IOCON_P1_25	R/W	0x030	0x4002 C0E4	D (tables 83, 84)	x	x	x	x
P1[26]	IOCON_P1_26	R/W	0x030	0x4002 C0E8	D (tables 83, 84)	x	x	x	x
P1[27]	IOCON_P1_27	R/W	0x030	0x4002 C0EC	D (tables 83, 84)	x	x	x	-
P1[28]	IOCON_P1_28	R/W	0x030	0x4002 C0F0	D (tables 83, 84)	x	x	x	x
P1[29]	IOCON_P1_29	R/W	0x030	0x4002 C0F4	D (tables 83, 84)	x	x	x	x
P1[30]	IOCON_P1_30	R/W	0x1B0	0x4002 C0F8	A (tables 85, 86)	x	x	x	x
P1[31]	IOCON_P1_31	R/W	0x1B0	0x4002 C0FC	A (tables 85, 86)	x	x	x	x

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 79. I/O Control registers for port 2

Port pin	Register	Access	Reset Value[1]	Address	IOCON type	208-pin	180-pin	144-pin	80-pin
P2[0]	IOCON_P2_0	R/W	0x030	0x4002 C100	D (tables 83, 84)	x	x	x	x
P2[1]	IOCON_P2_1	R/W	0x030	0x4002 C104	D (tables 83, 84)	x	x	x	x
P2[2]	IOCON_P2_2	R/W	0x030	0x4002 C108	D (tables 83, 84)	x	x	x	x
P2[3]	IOCON_P2_3	R/W	0x030	0x4002 C10C	D (tables 83, 84)	x	x	x	x
P2[4]	IOCON_P2_4	R/W	0x030	0x4002 C110	D (tables 83, 84)	x	x	x	x
P2[5]	IOCON_P2_5	R/W	0x030	0x4002 C114	D (tables 83, 84)	x	x	x	x
P2[6]	IOCON_P2_6	R/W	0x030	0x4002 C118	D (tables 83, 84)	x	x	x	x
P2[7]	IOCON_P2_7	R/W	0x030	0x4002 C11C	D (tables 83, 84)	x	x	x	x
P2[8]	IOCON_P2_8	R/W	0x030	0x4002 C120	D (tables 83, 84)	x	x	x	x
P2[9]	IOCON_P2_9	R/W	0x030	0x4002 C124	D (tables 83, 84)	x	x	x	x
P2[10]	IOCON_P2_10	R/W	0x030	0x4002 C128	D (tables 83, 84)	x	x	x	x
P2[11]	IOCON_P2_11	R/W	0x030	0x4002 C12C	D (tables 83, 84)	x	x	x	-
P2[12]	IOCON_P2_12	R/W	0x030	0x4002 C130	D (tables 83, 84)	x	x	x	-
P2[13]	IOCON_P2_13	R/W	0x030	0x4002 C134	D (tables 83, 84)	x	x	x	-
P2[14]	IOCON_P2_14	R/W	0x030	0x4002 C138	D (tables 83, 84)	x	-	-	-
P2[15]	IOCON_P2_15	R/W	0x030	0x4002 C13C	D (tables 83, 84)	x	-	-	-
P2[16]	IOCON_P2_16	R/W	0x030	0x4002 C140	D (tables 83, 84)	x	x	-	-
P2[17]	IOCON_P2_17	R/W	0x030	0x4002 C144	D (tables 83, 84)	x	x	-	-
P2[18]	IOCON_P2_18	R/W	0x030	0x4002 C148	D (tables 83, 84)	x	x	-	-
P2[19]	IOCON_P2_19	R/W	0x030	0x4002 C14C	D (tables 83, 84)	x	x	-	-
P2[20]	IOCON_P2_20	R/W	0x030	0x4002 C150	D (tables 83, 84)	x	x	-	-
P2[21]	IOCON_P2_21	R/W	0x030	0x4002 C154	D (tables 83, 84)	x	x	-	-
P2[22]	IOCON_P2_22	R/W	0x030	0x4002 C158	D (tables 83, 84)	x	-	-	-
P2[23]	IOCON_P2_23	R/W	0x030	0x4002 C15C	D (tables 83, 84)	x	-	-	-
P2[24]	IOCON_P2_24	R/W	0x030	0x4002 C160	D (tables 83, 84)	x	x	-	-
P2[25]	IOCON_P2_25	R/W	0x030	0x4002 C164	D (tables 83, 84)	x	x	-	-
P2[26]	IOCON_P2_26	R/W	0x030	0x4002 C168	D (tables 83, 84)	x	-	-	-
P2[27]	IOCON_P2_27	R/W	0x030	0x4002 C16C	D (tables 83, 84)	x	-	-	-
P2[28]	IOCON_P2_28	R/W	0x030	0x4002 C170	D (tables 83, 84)	x	x	-	-
P2[29]	IOCON_P2_29	R/W	0x030	0x4002 C174	D (tables 83, 84)	x	x	-	-
P2[30]	IOCON_P2_30	R/W	0x030	0x4002 C178	D (tables 83, 84)	x	-	-	-
P2[31]	IOCON_P2_31	R/W	0x030	0x4002 C17C	D (tables 83, 84)	x	-	-	-

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 80. I/O Control registers for port 3

Port pin	Register	Access	Reset Value[1]	Address	IOCON type	208-pin	180-pin	144-pin	80-pin
P3[0]	IOCON_P3_0	R/W	0x030	0x4002 C180	D (tables 83, 84)	x	x	x	-
P3[1]	IOCON_P3_1	R/W	0x030	0x4002 C184	D (tables 83, 84)	x	x	x	-
P3[2]	IOCON_P3_2	R/W	0x030	0x4002 C188	D (tables 83, 84)	x	x	x	-
P3[3]	IOCON_P3_3	R/W	0x030	0x4002 C18C	D (tables 83, 84)	x	x	x	-
P3[4]	IOCON_P3_4	R/W	0x030	0x4002 C190	D (tables 83, 84)	x	x	x	-
P3[5]	IOCON_P3_5	R/W	0x030	0x4002 C194	D (tables 83, 84)	x	x	x	-
P3[6]	IOCON_P3_6	R/W	0x030	0x4002 C198	D (tables 83, 84)	x	x	x	-
P3[7]	IOCON_P3_7	R/W	0x030	0x4002 C19C	D (tables 83, 84)	x	x	x	-
P3[8]	IOCON_P3_8	R/W	0x030	0x4002 C1A0	D (tables 83, 84)	x	x	-	-
P3[9]	IOCON_P3_9	R/W	0x030	0x4002 C1A4	D (tables 83, 84)	x	x	-	-
P3[10]	IOCON_P3_10	R/W	0x030	0x4002 C1A8	D (tables 83, 84)	x	x	-	-
P3[11]	IOCON_P3_11	R/W	0x030	0x4002 C1AC	D (tables 83, 84)	x	x	-	-
P3[12]	IOCON_P3_12	R/W	0x030	0x4002 C1B0	D (tables 83, 84)	x	x	-	-
P3[13]	IOCON_P3_13	R/W	0x030	0x4002 C1B4	D (tables 83, 84)	x	x	-	-
P3[14]	IOCON_P3_14	R/W	0x030	0x4002 C1B8	D (tables 83, 84)	x	x	-	-
P3[15]	IOCON_P3_15	R/W	0x030	0x4002 C1BC	D (tables 83, 84)	x	x	-	-
P3[16]	IOCON_P3_16	R/W	0x030	0x4002 C1C0	D (tables 83, 84)	x	-	-	-
P3[17]	IOCON_P3_17	R/W	0x030	0x4002 C1C4	D (tables 83, 84)	x	-	-	-
P3[18]	IOCON_P3_18	R/W	0x030	0x4002 C1C8	D (tables 83, 84)	x	-	-	-
P3[19]	IOCON_P3_19	R/W	0x030	0x4002 C1CC	D (tables 83, 84)	x	-	-	-
P3[20]	IOCON_P3_20	R/W	0x030	0x4002 C1D0	D (tables 83, 84)	x	-	-	-
P3[21]	IOCON_P3_21	R/W	0x030	0x4002 C1D4	D (tables 83, 84)	x	-	-	-
P3[22]	IOCON_P3_22	R/W	0x030	0x4002 C1D8	D (tables 83, 84)	x	-	-	-
P3[23]	IOCON_P3_23	R/W	0x030	0x4002 C1DC	D (tables 83, 84)	x	x	x	-
P3[24]	IOCON_P3_24	R/W	0x030	0x4002 C1E0	D (tables 83, 84)	x	x	x	-
P3[25]	IOCON_P3_25	R/W	0x030	0x4002 C1E4	D (tables 83, 84)	x	x	x	-
P3[26]	IOCON_P3_26	R/W	0x030	0x4002 C1E8	D (tables 83, 84)	x	x	x	-
P3[27]	IOCON_P3_27	R/W	0x030	0x4002 C1EC	D (tables 83, 84)	x	-	-	-
P3[28]	IOCON_P3_28	R/W	0x030	0x4002 C1F0	D (tables 83, 84)	x	-	-	-
P3[29]	IOCON_P3_29	R/W	0x030	0x4002 C1F4	D (tables 83, 84)	x	-	-	-
P3[30]	IOCON_P3_30	R/W	0x030	0x4002 C1F8	D (tables 83, 84)	x	-	-	-
P3[31]	IOCON_P3_31	R/W	0x030	0x4002 C1FC	D (tables 83, 84)	x	-	-	-

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 81. I/O Control registers for port 4

Port pin	Register	Access	Reset Value[1]	Address	IOCON type	208-pin	180-pin	144-pin	80-pin
P4[0]	IOCON_P4_0	R/W	0x030	0x4002 C200	D (tables 83, 84)	x	x	x	-
P4[1]	IOCON_P4_1	R/W	0x030	0x4002 C204	D (tables 83, 84)	x	x	x	-
P4[2]	IOCON_P4_2	R/W	0x030	0x4002 C208	D (tables 83, 84)	x	x	x	-
P4[3]	IOCON_P4_3	R/W	0x030	0x4002 C20C	D (tables 83, 84)	x	x	x	-
P4[4]	IOCON_P4_4	R/W	0x030	0x4002 C210	D (tables 83, 84)	x	x	x	-
P4[5]	IOCON_P4_5	R/W	0x030	0x4002 C214	D (tables 83, 84)	x	x	x	-
P4[6]	IOCON_P4_6	R/W	0x030	0x4002 C218	D (tables 83, 84)	x	x	x	-
P4[7]	IOCON_P4_7	R/W	0x030	0x4002 C21C	D (tables 83, 84)	x	x	x	-
P4[8]	IOCON_P4_8	R/W	0x030	0x4002 C220	D (tables 83, 84)	x	x	x	-
P4[9]	IOCON_P4_9	R/W	0x030	0x4002 C224	D (tables 83, 84)	x	x	x	-
P4[10]	IOCON_P4_10	R/W	0x030	0x4002 C228	D (tables 83, 84)	x	x	x	-
P4[11]	IOCON_P4_11	R/W	0x030	0x4002 C22C	D (tables 83, 84)	x	x	x	-
P4[12]	IOCON_P4_12	R/W	0x030	0x4002 C230	D (tables 83, 84)	x	x	x	-
P4[13]	IOCON_P4_13	R/W	0x030	0x4002 C234	D (tables 83, 84)	x	x	x	-
P4[14]	IOCON_P4_14	R/W	0x030	0x4002 C238	D (tables 83, 84)	x	x	x	-
P4[15]	IOCON_P4_15	R/W	0x030	0x4002 C23C	D (tables 83, 84)	x	x	x	-
P4[16]	IOCON_P4_16	R/W	0x030	0x4002 C240	D (tables 83, 84)	x	x	-	-
P4[17]	IOCON_P4_17	R/W	0x030	0x4002 C244	D (tables 83, 84)	x	x	-	-
P4[18]	IOCON_P4_18	R/W	0x030	0x4002 C248	D (tables 83, 84)	x	x	-	-
P4[19]	IOCON_P4_19	R/W	0x030	0x4002 C24C	D (tables 83, 84)	x	x	-	-
P4[20]	IOCON_P4_20	R/W	0x030	0x4002 C250	D (tables 83, 84)	x	-	-	-
P4[21]	IOCON_P4_21	R/W	0x030	0x4002 C254	D (tables 83, 84)	x	-	-	-
P4[22]	IOCON_P4_22	R/W	0x030	0x4002 C258	D (tables 83, 84)	x	-	-	-
P4[23]	IOCON_P4_23	R/W	0x030	0x4002 C25C	D (tables 83, 84)	x	-	-	-
P4[24]	IOCON_P4_24	R/W	0x030	0x4002 C260	D (tables 83, 84)	x	x	x	-
P4[25]	IOCON_P4_25	R/W	0x030	0x4002 C264	D (tables 83, 84)	x	x	x	-
P4[26]	IOCON_P4_26	R/W	0x030	0x4002 C268	D (tables 83, 84)	x	x	-	-
P4[27]	IOCON_P4_27	R/W	0x030	0x4002 C26C	D (tables 83, 84)	x	x	-	-
P4[28]	IOCON_P4_28	R/W	0x030	0x4002 C270	D (tables 83, 84)	x	x	x	x
P4[29]	IOCON_P4_29	R/W	0x030	0x4002 C274	D (tables 83, 84)	x	x	x	x
P4[30]	IOCON_P4_30	R/W	0x030	0x4002 C278	D (tables 83, 84)	x	x	x	-
P4[31]	IOCON_P4_31	R/W	0x030	0x4002 C27C	D (tables 83, 84)	x	x	x	-

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 82. I/O Control registers for port 5

Port pin	Register	Access	Reset Value[1]	Address	IOCON type	208-pin	180-pin	144-pin	80-pin
P5[0]	IOCON_P5_0	R/W	0x030	0x4002 C280	D (tables 83, 84)	x	x	x	-
P5[1]	IOCON_P5_1	R/W	0x030	0x4002 C284	D (tables 83, 84)	x	x	x	-
P5[2]	IOCON_P5_2	R/W	0	0x4002 C288	I (tables 89, 90)	x	x	x	-
P5[3]	IOCON_P5_3	R/W	0	0x4002 C28C	I (tables 89, 90)	x	x	x	-
P5[4]	IOCON_P5_4	R/W	0x030	0x4002 C290	D (tables 83, 84)	x	x	x	-

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

7.4.1 I/O configuration register contents (IOCON)

The functions of bits in the IOCON register for each GPIO port pin is described in the following sections. There are some differences in IOCON for special port pins compared to most other port pins. These include pins that support analog functions (such as ADC inputs and the DAC output), the USB D+/D- pins, and specialized I^2C pins:

- "Type D IOCON registers (applies to most GPIO port pins)"
- "Type A IOCON registers (applies to pins that include an analog function)"
- "Type U IOCON registers (applies to pins that include a USB D+ or D- function)"
- "Type I IOCON registers (applies to pins that include a specialized I^2C function)"
- "Type W IOCON registers (these pins are otherwise the same as Type D, but include a selectable input glitch filter"

7.4.1.1 Type D IOCON registers (applies to most GPIO port pins)

This IOCON table applies to all port pins except P0[7 to 9], P0[12 to 13], P0[23 to 31], P1[5 to 7], P1[14], P1[16 to 17], P1[30 to 31], and P5[2 to 3]. Those pins include DAC, ADC, USB, I^2C, or input glitch filter functions that alter the contents of the related IOCON registers.

Table 83. Type D IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 84 for specific values.	000
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control). See Section 7.3.2 “Pin mode”.	10
		00	Inactive (no pull-down/pull-up resistor enabled).	
		01	Pull-down resistor enabled.	
		10	Pull-up resistor enabled.	
		11	Repeater mode.	
5	HYS		Hysteresis. See Section 7.3.3 “Hysteresis”.	1
		0	Disable.	
		1	Enable.	
6	INV		Input polarity. See Section 7.3.4 “Input Inversion”.	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
8:7	-		Reserved. Read value is undefined, only zero should be written.	NA
9	SLEW		Driver slew rate. See Section 7.3.7 “Output slew rate”.	0
		0	Standard mode. Output slew rate control is enabled. More outputs can be switched simultaneously.	
		1	Fast mode. Slew rate control is disabled. This mode reduces the output delay by 1 ns compared to the standard mode. Fast mode is recommended for pins used with the EMC, LCD, and SPIFI interfaces.	
10	OD		Controls open-drain mode. See Section 7.3.9 “Open-Drain Mode”.	0
		0	Normal push-pull output	
		1	Simulated open-drain output (high drive disabled)	
31:11	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_0	P0[0]	CAN_RD1	U3_TXD	I2C1_SDA	U0_TXD			
IOCON_P0_1	P0[1]	CAN_TD1	U3_RXD	I2C1_SCL	U0_RXD			
IOCON_P0_2	P0[2]	U0_TXD	U3_TXD					
IOCON_P0_3	P0[3]	U0_RXD	U3_RXD					
IOCON_P0_4	P0[4]	I2S_RX_SCK	CAN_RD2	T2_CAP0		CMP_ROSC		LCD_VD[0]
IOCON_P0_5	P0[5]	I2S_RX_WS	CAN_TD2	T2_CAP1		CMP_RESET		LCD_VD[1]
IOCON_P0_6	P0[6]	I2S_RX_SDA	SSP1_SSEL	T2_MAT0	U1_RTS	CMP_ROSC		LCD_VD[8]
IOCON_P0_10	P0[10]	U2_TXD	I2C2_SDA	T3_MAT0				LCD_VD[5]
IOCON_P0_11	P0[11]	U2_RXD	I2C2_SCL	T3_MAT1				LCD_VD[10]
IOCON_P0_14	P0[14]	$\overline{\text{USB\_HSTEN2}}$	SSP1_SSEL	USB_CONNECT2				
IOCON_P0_15	P0[15]	U1_TXD	SSP0_SCK			SPIFI_IO[2]		
IOCON_P0_16	P0[16]	U1_RXD	SSP0_SSEL			SPIFI_IO[3]		
IOCON_P0_17	P0[17]	U1_CTS	SSP0_MISO			SPIFI_IO[1]		
IOCON_P0_18	P0[18]	U1_DCD	SSP0_MOSI			SPIFI_IO[0]		
IOCON_P0_19	P0[19]	U1_DSR	SD_CLK	I2C1_SDA				LCD_VD[13]
IOCON_P0_20	P0[20]	U1_DTR	SD_CMD	I2C1_SCL				LCD_VD[14]
IOCON_P0_21	P0[21]	U1_RI	SD_PWR	U4_OE	CAN_RD1	U4_SCLK		
IOCON_P0_22	P0[22]	U1_RTS	SD_DAT[0]	U4_TXD	CAN_TD1	SPIFI_CLK		
IOCON_P1_0	P1[0]	ENET_TXD0		T3_CAP1	SSP2_SCK			
IOCON_P1_1	P1[1]	ENET_TXD1		T3_MAT3	SSP2_MOSI			
IOCON_P1_2	P1[2]	ENET_TXD2	SD_CLK	PWM0[1]				
IOCON_P1_3	P1[3]	ENET_TXD3	SD_CMD	PWM0[2]				
IOCON_P1_4	P1[4]	ENET_TX_EN		T3_MAT2	SSP2_MISO			
IOCON_P1_8	P1[8]	ENET_CRS		T3_MAT1	SSP2_SSEL			
IOCON_P1_9	P1[9]	ENET_RXD0		T3_MAT0				
IOCON_P1_10	P1[10]	ENET_RXD1		T3_CAP0				
IOCON_P1_11	P1[11]	ENET_RXD2	SD_DAT[2]	PWM0[6]				
IOCON_P1_12	P1[12]	ENET_RXD3	SD_DAT[3]	PWM0_CAP0		CMP1_OUT		
IOCON_P1_13	P1[13]	ENET_RX_DV						

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P1_15	P1[15]	ENET_RX_CLK		I2C2_SDA				
IOCON_P1_18	P1[18]	USB_UP_LED1	PWM1[1]	T1_CAP0		SSP1_MISO		
IOCON_P1_19	P1[19]	USB_TX_E1	USB_PPWR1	T1_CAP1	MC_0A	SSP1_SCK	U2_OE	
IOCON_P1_20	P1[20]	USB_TX_DP1	PWM1[2]	QEI_PHA	MC_FB0	SSP0_SCK	LCD_VD[6]	LCD_VD[10]
IOCON_P1_21	P1[21]	USB_TX_DM1	PWM1[3]	SSP0_SSEL	MC_ABORT		LCD_VD[7]	LCD_VD[11]
IOCON_P1_22	P1[22]	USB_RCV1	USB_PWRD1	T1_MAT0	MC_0B	SSP1_MOSI	LCD_VD[8]	LCD_VD[12]
IOCON_P1_23	P1[23]	USB_RX_DP1	PWM1[4]	QEI_PHB	MC_FB1	SSP0_MISO	LCD_VD[9]	LCD_VD[13]
IOCON_P1_24	P1[24]	USB_RX_DM1	PWM1[5]	QEI_IDX	MC_FB2	SSP0_MOSI	LCD_VD[10]	LCD_VD[14]
IOCON_P1_25	P1[25]	USB_LS1	USB_HSTEN1	T1_MAT1	MC_1A	CLKOUT	LCD_VD[11]	LCD_VD[15]
IOCON_P1_26	P1[26]	USB_SSPND1	PWM1[6]	T0_CAP0	MC_1B	SSP1_SSEL	LCD_VD[12]	LCD_VD[20]
IOCON_P1_27	P1[27]	USB_INT1	USB_OVRCR1	T0_CAP1	CLKOUT		LCD_VD[13]	LCD_VD[21]
IOCON_P1_28	P1[28]	USB_SCL1	PWM1_CAP0	T0_MAT0	MC_2A	SSP0_SSEL	LCD_VD[14]	LCD_VD[22]
IOCON_P1_29	P1[29]	USB_SDA1	PWM1_CAP1	T0_MAT1	MC_2B	U4_TXD	LCD_VD[15]	LCD_VD[23]
IOCON_P2_0	P2[0]	PWM1[1]	U1_TXD					LCD_PWR
IOCON_P2_1	P2[1]	PWM1[2]	U1_RXD					LCD_LE
IOCON_P2_2	P2[2]	PWM1[3]	U1_CTS	T2_MAT3		TRACEDATA[3]		LCD_DCLK
IOCON_P2_3	P2[3]	PWM1[4]	U1_DCD	T2_MAT2		TRACEDATA[2]		LCD_FP
IOCON_P2_4	P2[4]	PWM1[5]	U1_DSR	T2_MAT1		TRACEDATA[1]		LCD_ENAB_M
IOCON_P2_5	P2[5]	PWM1[6]	U1_DTR	T2_MAT0		TRACEDATA[0]		LCD_LP
IOCON_P2_6	P2[6]	PWM1_CAP0	U1_RI	T2_CAP0	U2_OE	TRACECLK	LCD_VD[0]	LCD_VD[4]
IOCON_P2_7	P2[7]	CAN_RD2	U1_RTS			SPIFI_CS	LCD_VD[1]	LCD_VD[5]
IOCON_P2_8	P2[8]	CAN_TD2	U2_TXD	U1_CTS	ENET_MDC		LCD_VD[2]	LCD_VD[6]
IOCON_P2_9	P2[9]	USB_CONNECT1	U2_RXD	U4_RXD	ENET_MDIO		LCD_VD[3]	LCD_VD[7]
IOCON_P2_10	P2[10]	EINT0	NMI					
IOCON_P2_11	P2[11]	EINT1	SD_DAT[1]	I2S_TX_SCK				LCD_CLKIN
IOCON_P2_12	P2[12]	EINT2	SD_DAT[2]	I2S_TX_WS	LCD_VD[4]	LCD_VD[3]	LCD_VD[8]	LCD_VD[18]
IOCON_P2_13	P2[13]	EINT3	SD_DAT[3]	I2S_TX_SDA		LCD_VD[5]	LCD_VD[9]	LCD_VD[19]
IOCON_P2_14	P2[14]	EMC_CS2	I2C1_SDA	T2_CAP0				
IOCON_P2_15	P2[15]	EMC_CS3	I2C1_SCL	T2_CAP1				
IOCON_P2_16	P2[16]	EMC_CAS						

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P2_17	P2[17]	$\overline{\text{EMC\_RAS}}$						
IOCON_P2_18	P2[18]	EMC_CLK0						
IOCON_P2_19	P2[19]	EMC_CLK1						
IOCON_P2_20	P2[20]	$\overline{\text{EMC\_DYCS0}}$						
IOCON_P2_21	P2[21]	$\overline{\text{EMC\_DYCS1}}$						
IOCON_P2_22	P2[22]	$\overline{\text{EMC\_DYCS2}}$	SSP0_SCK	T3_CAP0				
IOCON_P2_23	P2[23]	$\overline{\text{EMC\_DYCS3}}$	SSP0_SSEL	T3_CAP1				
IOCON_P2_24	P2[24]	EMC_CKE0						
IOCON_P2_25	P2[25]	EMC_CKE1						
IOCON_P2_26	P2[26]	EMC_CKE2	SSP0_MISO	T3_MAT0				
IOCON_P2_27	P2[27]	EMC_CKE3	SSP0_MOSI	T3_MAT1				
IOCON_P2_28	P2[28]	EMC_DQM0						
IOCON_P2_29	P2[29]	EMC_DQM1						
IOCON_P2_30	P2[30]	EMC_DQM2	I2C2_SDA	T3_MAT2				
IOCON_P2_31	P2[31]	EMC_DQM3	I2C2_SCL	T3_MAT3				
IOCON_P3_0	P3[0]	EMC_D[0]						
IOCON_P3_1	P3[1]	EMC_D[1]						
IOCON_P3_2	P3[2]	EMC_D[2]						
IOCON_P3_3	P3[3]	EMC_D[3]						
IOCON_P3_4	P3[4]	EMC_D[4]						
IOCON_P3_5	P3[5]	EMC_D[5]						
IOCON_P3_6	P3[6]	EMC_D[6]						
IOCON_P3_7	P3[7]	EMC_D[7]						
IOCON_P3_8	P3[8]	EMC_D[8]						
IOCON_P3_9	P3[9]	EMC_D[9]						
IOCON_P3_10	P3[10]	EMC_D[10]						
IOCON_P3_11	P3[11]	EMC_D[11]						
IOCON_P3_12	P3[12]	EMC_D[12]						
IOCON_P3_13	P3[13]	EMC_D[13]						
IOCON_P3_14	P3[14]	EMC_D[14]						

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P3_15	P3[15]	EMC_D[15]						
IOCON_P3_16	P3[16]	EMC_D[16]	PWM0[1]	U1_TXD				
IOCON_P3_17	P3[17]	EMC_D[17]	PWM0[2]	U1_RXD				
IOCON_P3_18	P3[18]	EMC_D[18]	PWM0[3]	U1_CTS				
IOCON_P3_19	P3[19]	EMC_D[19]	PWM0[4]	U1_DCD				
IOCON_P3_20	P3[20]	EMC_D[20]	PWM0[5]	U1_DSR				
IOCON_P3_21	P3[21]	EMC_D[21]	PWM0[6]	U1_DTR				
IOCON_P3_22	P3[22]	EMC_D[22]	PWM0_CAP0	U1_RI				
IOCON_P3_23	P3[23]	EMC_D[23]	PWM1_CAP0	T0_CAP0				
IOCON_P3_24	P3[24]	EMC_D[24]	PWM1[1]	T0_CAP1				
IOCON_P3_25	P3[25]	EMC_D[25]	PWM1[2]	T0_MAT0				
IOCON_P3_26	P3[26]	EMC_D[26]	PWM1[3]	T0_MAT1	STCLK			
IOCON_P3_27	P3[27]	EMC_D[27]	PWM1[4]	T1_CAP0				
IOCON_P3_28	P3[28]	EMC_D[28]	PWM1[5]	T1_CAP1				
IOCON_P3_29	P3[29]	EMC_D[29]	PWM1[6]	T1_MAT0				
IOCON_P3_30	P3[30]	EMC_D[30]	U1_RTS	T1_MAT1				
IOCON_P3_31	P3[31]	EMC_D[31]		T1_MAT2				
IOCON_P4_0	P4[0]	EMC_A[0]						
IOCON_P4_1	P4[1]	EMC_A[1]						
IOCON_P4_2	P4[2]	EMC_A[2]						
IOCON_P4_3	P4[3]	EMC_A[3]						
IOCON_P4_4	P4[4]	EMC_A[4]						
IOCON_P4_5	P4[5]	EMC_A[5]						
IOCON_P4_6	P4[6]	EMC_A[6]						
IOCON_P4_7	P4[7]	EMC_A[7]						
IOCON_P4_8	P4[8]	EMC_A[8]						
IOCON_P4_9	P4[9]	EMC_A[9]						
IOCON_P4_10	P4[10]	EMC_A[10]						
IOCON_P4_11	P4[11]	EMC_A[11]						
IOCON_P4_12	P4[12]	EMC_A[12]						

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P4_13	P4[13]	EMC_A[13]						
IOCON_P4_14	P4[14]	EMC_A[14]						
IOCON_P4_15	P4[15]	EMC_A[15]						
IOCON_P4_16	P4[16]	EMC_A[16]						
IOCON_P4_17	P4[17]	EMC_A[17]						
IOCON_P4_18	P4[18]	EMC_A[18]						
IOCON_P4_19	P4[19]	EMC_A[19]						
IOCON_P4_20	P4[20]	EMC_A[20]	I2C2_SDA	SSP1_SCK				
IOCON_P4_21	P4[21]	EMC_A[21]	I2C2_SCL	SSP1_SSEL				
IOCON_P4_22	P4[22]	EMC_A[22]	U2_TXD	SSP1_MISO				
IOCON_P4_23	P4[23]	EMC_A[23]	U2_RXD	SSP1_MOSI				
IOCON_P4_24	P4[24]	$\overline{\text{EMC\_OE}}$						
IOCON_P4_25	P4[25]	$\overline{\text{EMC\_WE}}$						
IOCON_P4_26	P4[26]	$\overline{\text{EMC\_BLS0}}$						
IOCON_P4_27	P4[27]	$\overline{\text{EMC\_BLS1}}$						
IOCON_P4_28	P4[28]	$\overline{\text{EMC\_BLS2}}$	U3_TXD	T2_MAT0		LCD_VD[6]	LCD_VD[10]	LCD_VD[2]
IOCON_P4_29	P4[29]	$\overline{\text{EMC\_BLS3}}$	U3_RXD	T2_MAT1	I2C2_SCL	LCD_VD[7]	LCD_VD[11]	LCD_VD[3]
IOCON_P4_30	P4[30]	$\overline{\text{EMC\_CS0}}$				CMP0_OUT		
IOCON_P4_31	P4[31]	$\overline{\text{EMC\_CS1}}$						
IOCON_P5_0	P5[0]	EMC_A[24]	SSP2_MOSI	T2_MAT2				
IOCON_P5_1	P5[1]	EMC_A[25]	SSP2_MISO	T2_MAT3				
IOCON_P5_4	P5[4]	U0_OE		T3_MAT3	U4_TXD			

7.4.1.2 Type A IOCON registers (applies to pins that include an analog function)

This IOCON table applies to pins P0[12 to 13], P0[23 to 26], and P1[30 to 31]. The presence of the DAC output on P0[26] makes that pin slightly different, see the description of bit 16 below.

Table 85. Type A IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 86 for specific values.	0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control). See Section 7.3.2 “Pin mode”.	10
		00	Inactive (no pull-down/pull-up resistor enabled).	
		01	Pull-down resistor enabled.	
		10	Pull-up resistor enabled.	
		11	Repeater mode.	
5	-		Reserved. Read value is undefined, only zero should be written.	NA
6	INV		Input polarity. See Section 7.3.4 “Input Inversion”.	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
7	ADMODE		Select Analog/Digital mode. See Section 7.3.5 “Analog/digital mode”.	1
		0	Analog mode.	
		1	Digital mode.	
8	FILTER		Controls glitch filter. See Section 7.3.6 “Input filter”.	1
		0	Noise pulses below approximately 10 ns are filtered out	
		1	No input filtering is done	
9	-		Reserved. Read value is undefined, only zero should be written.	NA
10	OD		Controls open-drain mode. See Section 7.3.9 “Open-Drain Mode”.	0
		0	Normal push-pull output	
		1	Simulated open-drain output (high drive disabled)	
14:11	-		Reserved. Read value is undefined, only zero should be written.	NA
16	DACEN		DAC enable control. This bit applies only to P0[26], which includes the DAC output function DAC_OUT. See Section 7.3.10 “DAC enable”.	0
		0	DAC is disabled	
		1	DAC is enabled	
31:17	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 86. Type A I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_12	P0[12]	$\overline{\text{USB\_PPWR2}}$	SSP1_MISO	ADC0[6]				
IOCON_P0_13	P0[13]	USB_UP_LED2	SSP1_MOSI	ADC0[7]				
IOCON_P0_23	P0[23]	ADC0[0]	I2S_RX_SCK	T3_CAP0				
IOCON_P0_24	P0[24]	ADC0[1]	I2S_RX_WS	T3_CAP1				
IOCON_P0_25	P0[25]	ADC0[2]	I2S_RX_SDA	U3_TXD				
IOCON_P0_26	P0[26]	ADC0[3]	DAC_OUT	U3_RXD				
IOCON_P1_30	P1[30]	USB_PWRD2	USB_VBUS	ADC[4]	I2C0_SDA	U3_OE		
IOCON_P1_31	P1[31]	$\overline{\text{USB\_OVRCR2}}$	SSP1_SCK	ADC[5]	I2C0_SCL			

7.4.1.3 Type U IOCON registers (applies to pins that include a USB D+ or D- function)

This IOCON table applies to pins P0[29], P0[30], and P0[31]. These special function pins do not include the selectable modes and options of other pins.

Table 87. Type U IOCON registers bit description

Bit	Symbol	Description	Reset value
2:0	FUNC	Selects pin function. See Table 88 for specific values.	000
31:3	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 88. Type U I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_29	P0[29]	USB_D+1	EINT0					
IOCON_P0_30	P0[30]	USB_D-1	EINT1					
IOCON_P0_31	P0[31]	USB_D+2						

7.4.1.4 Type I IOCON registers (applies to pins that include a specialized I²C function)

This IOCON table applies to pins P0[27 to 28] and P5[2 to 3].

Table 89. Type I IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 90 for specific values.	0
5:3	-		Reserved. Read value is undefined, only zero should be written.	NA
6	INV		Input polarity. See Section 7.3.4 “Input Inversion”.	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
7	-		Reserved. Read value is undefined, only zero should be written.	NA
8	HS		Configures I²C features for standard mode, fast mode, and Fast Mode Plus operation. See Section 7.3.8 “I²C modes”.	0
		0	I²C 50ns glitch filter and slew rate control enabled.	
		1	I²C 50ns glitch filter and slew rate control disabled.	
9	HIDRIVE		Controls sink current capability of the pin, only for P5[2] and P5[3]. See Section 7.3.8 “I²C modes”.	0
		0	Output drive sink is 4 mA. This is sufficient for standard and fast mode I²C.	
		1	Output drive sink is 20 mA. This is needed for Fast Mode Plus I²C. Refer to the appropriate specific device data sheet for details.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 90. Type I I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_27	P0[27]	I2C0_SDA	USB_SDA1					
IOCON_P0_28	P0[28]	I2C0_SCL	USB_SCL1					
IOCON_P5_2	P5[2]		SSP2_SCK	T3_MAT2		I2C0_SDA		
IOCON_P5_3	P5[3]		SSP2_SSEL		U4_RXD	I2C0_SCL		

7.4.1.5 Type W IOCON registers (these pins are otherwise the same as Type D, but include a selectable input glitch filter

This IOCON table applies to pins P0[7 to 9], P1[5 to 7], P1[14], and P1[16 to 17].

P0[7 to 9] reset with the pull-down/pull-up disabled. P1[5 to 7], P1[14], and P1[16 to 17] reset with the pull-down/pull-up enabled.

Table 91. Type W IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. See Table 92 for specific values.	000
4:3	MODE		Selects the output functional mode for the pin (on-chip pull-up/pull-down resistor control). See Section 7.3.2 “Pin mode”.	[1]
		00	Inactive (no pull-down/pull-up resistor enabled).	
		01	Pull-down resistor enabled.	
		10	Pull-up resistor enabled.	
		11	Repeater mode.	
5	HYS		Hysteresis. See Section 7.3.3 “Hysteresis”.	1
		0	Disable.	
		1	Enable.	
6	INV		Input polarity. See Section 7.3.4 “Input Inversion”.	0
		0	Input is not inverted (a HIGH on the pin reads as 1)	
		1	Input is inverted (a HIGH on the pin reads as 0)	
7	ADMODE		Select Analog/Digital mode. See Section 7.3.5 “Analog/digital mode”.	1
		0	Analog mode.	
		1	Digital mode.	
8	FILTER		Controls glitch filter. See Section 7.3.6 “Input filter”.	0
		0	Noise pulses below approximately 10 ns are filtered out	
		1	No input filtering is done	
9	SLEW		Driver slew rate. See Section 7.3.7 “Output slew rate”.	0
		0	Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously.	
		1	Fast mode, slew rate control is disabled. Refer to the appropriate specific device data sheet for details.	
10	OD		Controls open-drain mode. See Section 7.3.9 “Open-Drain Mode”.	0
		0	Normal push-pull output	
		1	Simulated open-drain output (high drive disabled)	
31:11	-		Reserved. Read value is undefined, only zero should be written.	NA

[1] The mode field reset value is 00 for P0[7 to 9], and 10 for P1[5 to 7, P1[14], and P1[16 to 17].

Table 92. Type W I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_7	P0[7]	I2S_TX_SCK	SSP1_SCK	T2_MAT1	RTC_EV0	CMP_VREF		LCD_VD[9]
IOCON_P0_8	P0[8]	I2S_TX_WS	SSP1_MISO	T2_MAT2	RTC_EV1	CMP1_IN[4]		LCD_VD[16]
IOCON_P0_9	P0[9]	I2S_TX_SDA	SSP1_MOSI	T2_MAT3	RTC_EV2	CMP1_IN[3]		LCD_VD[17]
IOCON_P1_5	P1[5]	ENET_TX_ER	SD_PWR	PWM0[3]		CMP1_IN[2]		
IOCON_P1_6	P1[6]	ENET_TX_CLK	SD_DAT[0]	PWM0[4]		CMP0_IN[4]		
IOCON_P1_7	P1[7]	ENET_COL	SD_DAT[1]	PWM0[5]		CMP1_IN[1]		
IOCON_P1_14	P1[14]	ENET_RX_ER		T2_CAP0		CMP0_IN[1]		
IOCON_P1_16	P1[16]	ENET_MDC	I2S_TX_MCLK			CMP0_IN[2]		
IOCON_P1_17	P1[17]	ENET_MDIO	I2S_RX_MCLK			CMP0_IN[3]		

8.1 Basic configuration

GPIOs are configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCGPIO. This enables the GPIOs themselves, GPIO interrupts, and the IOCON block.
2. Pins: See Section 7.4.1 for GPIO pins and their modes.
3. Wake-up: GPIO ports 0 and 2 can be used for wake-up if needed, see (Section 3.12.8).
4. Interrupts: Enable GPIO interrupts in EnR (Table 105 or Table 110) and EnF (Table 106 or Table 111). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

8.2 Features

8.2.1 Digital I/O ports

- Accelerated GPIO functions:
  - GPIO registers are located on a peripheral AHB bus for fast I/O timing.
  - Mask registers allow treating sets of port bits as a group, leaving other bits unchanged.
  - All GPIO registers are byte, half-word, and word addressable.
  - Entire port value can be written in one instruction.
  - GPIO registers are accessible by the GPDMA.
- Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.
- All GPIO registers support bit-banding operations by the CPU.
- GPIO registers are accessible by the GPDMA controller to allow DMA of data to or from GPIOs, synchronized to any DMA request.
- Direction control of individual port bits.
- All I/Os default to input with pull-up after reset.

8.2.2 Interrupt generating digital ports

- Port 0 and Port 2 can provide a single interrupt for any combination of port pins.
- Each port pin can be programmed to generate an interrupt on a rising edge, a falling edge, or both.
- Edge detection is asynchronous, so it may operate when clocks are not present, such as during Power-down mode. With this feature, level triggered interrupts are not needed.
- Each enabled interrupt contributes to a wake-up signal that can be used to bring the part out of Power-down mode.

- Registers provide a software view of pending rising edge interrupts, pending falling edge interrupts, and overall pending GPIO interrupts.
- The GPIO interrupt function does not require that the pin be configured for GPIO. This allows interrupting on a change to a pin that is part of an operating peripheral interface.

8.3 Applications

- General purpose I/O
- Driving LEDs or other indicators
- Controlling off-chip devices
- Sensing digital inputs, detecting edges
- Bringing the part out of Power-down mode

8.4 Pin description

Table 93. GPIO pin description

Pin Name	Type	Description
P0[31:0]; P1[31:0]; P2[31:0]; P3[31:0]; P4[31:0]; P5[4:0]	Input/ Output	General purpose input/output. These are typically shared with other peripherals functions and will therefore not all be available in an application. Packaging options may affect the number of GPIOs available in a particular device. Some pins may be limited by requirements of the alternate functions of the pin. For example, some pins that can be used for I^2C are special pins, and some of that behavior is inherited by any other function selected on that pin. Details may be found in Section 6.1.

8.5 Register description

The registers represent the enhanced GPIO features available on all of the GPIO ports. These registers are located on an AHB bus for fast read and write timing. They can all be accessed in byte, half-word, and word sizes. A mask register allows access to a group of bits in a single GPIO port independently from other bits in the same port.

Table 94. Register overview: GPIO (base address 0x2009 8000)

Name	Access	Address offset	Description	Reset value	Table
DIR0	R/W	0x000	GPIO Port0 Direction control register.	0	96
MASK0	R/W	0x010	Mask register for Port0.	0	97
PIN0	R/W	0x014	Port0 Pin value register using FIOMASK.	0	98
SET0	R/W	0x018	Port0 Output Set register using FIOMASK.	0	99
CLR0	WO	0x01C	Port0 Output Clear register using FIOMASK.	-	100
DIR1	R/W	0x020	GPIO Port1 Direction control register.	0	96
MASK1	R/W	0x030	Mask register for Port1.	0	97
PIN1	R/W	0x034	Port1 Pin value register using FIOMASK.	0	98
SET1	R/W	0x038	Port1 Output Set register using FIOMASK.	0	99
CLR1	WO	0x03C	Port1 Output Clear register using FIOMASK.	-	100
DIR2	R/W	0x040	GPIO Port2 Direction control register.	0	96
MASK2	R/W	0x050	Mask register for Port2.	0	97
PIN2	R/W	0x054	Port2 Pin value register using FIOMASK.	0	98
SET2	R/W	0x058	Port2 Output Set register using FIOMASK.	0	99
CLR2	WO	0x05C	Port2 Output Clear register using FIOMASK.	-	100
DIR3	R/W	0x060	GPIO Port3 Direction control register.	0	96
MASK3	R/W	0x070	Mask register for Port3.	0	97
PIN3	R/W	0x074	Port3 Pin value register using FIOMASK.	0	98
SET3	R/W	0x078	Port3 Output Set register using FIOMASK.	0	99
CLR3	WO	0x07C	Port3 Output Clear register using FIOMASK.	-	100
DIR4	R/W	0x080	GPIO Port4 Direction control register.	0	96
MASK4	R/W	0x090	Mask register for Port4.	0	97
PIN4	R/W	0x094	Port4 Pin value register using FIOMASK.	0	98
SET4	R/W	0x098	Port4 Output Set register using FIOMASK.	0	99
CLR4	WO	0x09C	Port4 Output Clear register using FIOMASK.	-	100
DIR5	R/W	0x0A0	GPIO Port5 Direction control register.	0	96
MASK5	R/W	0x0B0	Mask register for Port5.	0	97
PIN5	R/W	0x0B4	Port5 Pin value register using FIOMASK.	0	98
SET5	R/W	0x0B8	Port5 Output Set register using FIOMASK.	0	99
CLR5	WO	0x0BC	Port5 Output Clear register using FIOMASK.	-	100

Table 95. Register overview: GPIO interrupt (base address 0x4002 8000)

Name	Access	Address offset	Description	Reset value[1]	Table
STATUS	RO	0x080	GPIO overall Interrupt Status.	0	101
STATR0	RO	0x084	GPIO Interrupt Status for Rising edge for Port 0.	0	102
STATF0	RO	0x088	GPIO Interrupt Status for Falling edge for Port 0.	0	103
CLR0	WO	0x08C	GPIO Interrupt Clear.	-	104
ENR0	R/W	0x090	GPIO Interrupt Enable for Rising edge for Port 0.	0	105
ENF0	R/W	0x094	GPIO Interrupt Enable for Falling edge for Port 0.	0	106
STATR2	RO	0x0A4	GPIO Interrupt Status for Rising edge for Port 0.	0	107
STATF2	RO	0x0A8	GPIO Interrupt Status for Falling edge for Port 0.	0	108
CLR2	WO	0x0AC	GPIO Interrupt Clear.	-	109
ENR2	R/W	0x0B0	GPIO Interrupt Enable for Rising edge for Port 0.	0	110
ENF2	R/W	0x0B4	GPIO Interrupt Enable for Falling edge for Port 0.	0	111

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

8.5.1 GPIO port registers

8.5.1.1 GPIO port Direction register

This word accessible register is used to control the direction of the pins when they are configured as GPIO port pins. Direction bit for any pin must be set according to the pin functionality.

Note that GPIO pins P0[29] and P0[30] are shared with the USB_D+ and USB_D- pins and must have the same direction. If either FIO0DIR bit 29 or 30 are configured as zero, both P0[29] and P0[30] will be inputs. If both FIO0DIR bits 29 and 30 are ones, both P0[29] and P0[30] will be outputs.

Aside from the 32-bit long and word only accessible DIRx register, every fast GPIO port can also be controlled via byte and half-word access.

Table 96. GPIO port Direction register (DIR[0:5] - addresses 0x2009 8000 (DIR0) to 0x200980A0 (DIR5)) bit description

Bit	Symbol	Description	Reset value
31:0	PINDIR	Fast GPIO Direction PORTx control bits. Bit 0 in DIRx controls pin Px[0], bit 31 in DIRx controls pin Px[31]. 0 = Controlled pin is input. 1 = Controlled pin is output.	0x0

8.5.1.2 Fast GPIO port Mask register

This register is used to select port pins that will and will not be affected by write accesses to the PINx, SETx or CLRx register. Mask register also filters out port's content when the PINx register is read.

A zero in this register's bit enables an access to the corresponding physical pin via a read or write access. If a bit in this register is one, corresponding pin will not be changed with write access and if read, will not be reflected in the updated PINx register. For software examples, see Section 8.6.

Table 97. Fast GPIO port Mask register (MASK[0:5] - addresses 0x2009 8010 (MASK0) to 0x2009 80B0 (MASK5)) bit description

Bit	Symbol	Description	Reset value
31:0	PINMASK	Fast GPIO physical pin access control. 0 = Controlled pin is affected by writes to the port's SETx, CLRx, and PINx registers. Current state of the pin can be read from the PINx register. 1 = Controlled pin is not affected by writes into the port's SETx, CLRx and PINx registers. When the PINx register is read, this bit will not be updated with the state of the physical pin.	0x0

Aside from the 32-bit long and word only accessible MASKx register, every fast GPIO port can also be controlled via byte and half-word access.

8.5.1.3 GPIO port Pin value register

This register provides the value of port pins that are configured to perform only digital functions. The register will give the logic value of the pin regardless of whether the pin is configured for input or output, or as GPIO or an alternate digital function. As an example,

a particular port pin may have GPIO input, GPIO output, UART receive, and PWM output as selectable functions. Any configuration of that pin will allow its current logic state to be read from the corresponding PINx register.

If a pin has an analog function as one of its options, the pin state cannot be read if the analog configuration is selected. Selecting the pin as an A/D input disconnects the digital features of the pin. In that case, the pin value read in the PINx register is not valid.

Writing to the PINx register stores the value in the port output register, bypassing the need to use both the SETx and CLRx registers to obtain the entire written value. This feature should be used carefully in an application since it affects the entire port.

Access to a port pin via the PINx register is conditioned by the corresponding bit of the MASKx register (see Section 8.5.1.2).

Only pins masked with zeros in the Mask register (see Section 8.5.1.2) will be correlated to the current content of the Fast GPIO port pin value register.

Table 98. Fast GPIO port Pin value register (PIN[0:5] - addresses 0x2009 8014 (PIN0) to 0x2009 80B4 (PIN5)) bit description

Bit	Symbol	Description	Reset value
31:0	PINVAL	Fast GPIO output value Set bits. Bit 0 in CLRx corresponds to pin Px[0], bit 31 in CLRx corresponds to pin Px[31]. 0 = Controlled pin output is set to LOW. 1 = Controlled pin output is set to HIGH.	0x0

Aside from the 32-bit long and word only accessible PINx register, every fast GPIO port can also be controlled via byte and half-word access.

8.5.1.4 GPIO port output Set register

This register is used to produce a HIGH level output at the port pins configured as GPIO in an OUTPUT mode. Writing 1 produces a HIGH level at the corresponding port pins. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing 1 to the corresponding bit in the SETx has no effect.

Reading the SETx register returns the value of this register, as determined by previous writes to SETx and CLRx (or PINx as noted above). This value does not reflect the effect of any outside world influence on the I/O pins.

Access to a port pin via the SETx register is conditioned by the corresponding bit of the MASKx register (see Section 8.5.1.2).

Table 99. Fast GPIO port output Set register (SET[0:5] - addresses 0x2009 8018 (SET0) to 0x2009 80B8 (SET5)) bit description

Bit	Symbol	Description	Reset value
31:0	PINSET	Fast GPIO output value Set bits. Bit 0 in SETx controls pin Px[0], bit 31 in SETx controls pin Px[31]. 0 = Controlled pin output is unchanged. 1 = Controlled pin output is set to HIGH.	0x0

Aside from the 32-bit long and word only accessible SETx register, every fast GPIO port can also be controlled via byte and half-word access.

8.5.1.5 GPIO port output Clear register

This register is used to produce a LOW level output at port pins configured as GPIO in an OUTPUT mode. Writing 1 produces a LOW level at the corresponding port pin and clears the corresponding bit in the SETx register. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to CLRx has no effect.

Access to a port pin via the CLRx register is conditioned by the corresponding bit of the MASKx register (see Section 8.5.1.2).

Table 100. Fast GPIO port output Clear register (CLR[0:5] - addresses 0x2009 801C (CLR0) to 0x2009 80BC (CLR5)) bit description

Bit	Symbol	Description
31:0	PINCLR	Fast GPIO output value Clear bits. Bit 0 in CLRx controls pin Px[0], bit 31 controls pin Px[31]. 0 = Controlled pin output is unchanged. 1 = Controlled pin output is set to LOW.

Aside from the 32-bit long and word only accessible CLRx register, every fast GPIO port can also be controlled via byte and half-word access.

8.5.2 GPIO interrupt registers

The following registers configure the pins of Port 0 and Port 2 to generate interrupts.

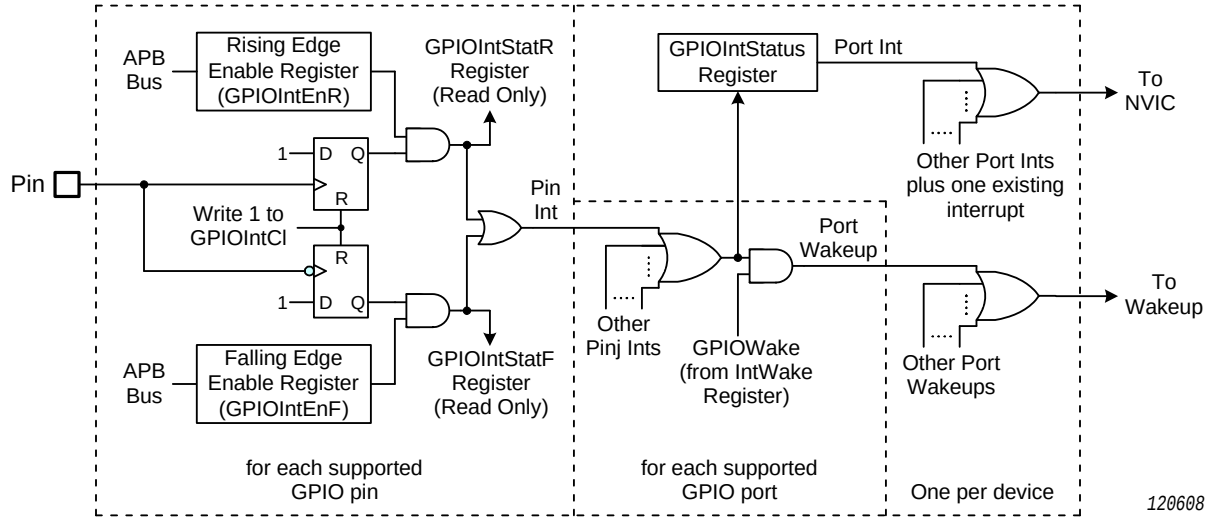


Fig 15. GPIO interrupt block diagram

8.5.2.1 GPIO overall Interrupt Status register

This read-only register indicates the presence of interrupt pending on all of the GPIO ports that support GPIO interrupts. Only status one bit per port is required.

Table 101. GPIO overall Interrupt Status register (STATUS - address 0x4002 8080) bit description

Bit	Symbol	Value	Description	Reset value
0	P0INT		Port 0 GPIO interrupt pending.	0
		0	No pending interrupts on Port 0.	
		1	At least one pending interrupt on Port 0.	
1	-		Reserved. The value read from a reserved bit is not defined.	NA
2	P2INT		Port 2 GPIO interrupt pending.	0
		0	No pending interrupts on Port 2.	
		1	At least one pending interrupt on Port 2.	
31:2	-		Reserved. The value read from a reserved bit is not defined.	NA

8.5.2.2 GPIO Interrupt Status for port 0 Rising Edge Interrupt

Each bit in these read-only registers indicates the rising edge interrupt status for port 0.

Table 102. GPIO Interrupt Status for port 0 Rising Edge Interrupt (STATR0 - 0x4002 8084) bit description

Bit	Symbol	Description	Reset value
0	P0_0REI	Rising Edge Interrupt status for P0[0]. 0 = No rising edge detected. 1 = Rising edge interrupt generated.	0
1	P0_1REI	Rising Edge Interrupt status for P0[1]. See bit 0 description.	0
2	P0_2REI	Rising Edge Interrupt status for P0[2]. See bit 0 description.	0
3	P0_3REI	Rising Edge Interrupt status for P0[3]. See bit 0 description.	0
4	P0_4REI	Rising Edge Interrupt status for P0[4]. See bit 0 description.	0
5	P0_5REI	Rising Edge Interrupt status for P0[5]. See bit 0 description.	0
6	P0_6REI	Rising Edge Interrupt status for P0[6]. See bit 0 description.	0
7	P0_7REI	Rising Edge Interrupt status for P0[7]. See bit 0 description.	0
8	P0_8REI	Rising Edge Interrupt status for P0[8]. See bit 0 description.	0
9	P0_9REI	Rising Edge Interrupt status for P0[9]. See bit 0 description.	0
10	P0_10REI	Rising Edge Interrupt status for P0[10]. See bit 0 description.	0
11	P0_11REI	Rising Edge Interrupt status for P0[11]. See bit 0 description.	0
12	P0_12REI	Rising Edge Interrupt status for P0[12]. See bit 0 description.	0
13	P0_13REI	Rising Edge Interrupt status for P0[13]. See bit 0 description.	0
14	P0_14REI	Rising Edge Interrupt status for P0[14]. See bit 0 description.	0
15	P0_15REI	Rising Edge Interrupt status for P0[15]. See bit 0 description.	0
16	P0_16REI	Rising Edge Interrupt status for P0[16]. See bit 0 description.	0
17	P0_17REI	Rising Edge Interrupt status for P0[17]. See bit 0 description.	0
18	P0_18REI	Rising Edge Interrupt status for P0[18]. See bit 0 description.	0
19	P0_19REI	Rising Edge Interrupt status for P0[19]. See bit 0 description.	0
20	P0_20REI	Rising Edge Interrupt status for P0[20]. See bit 0 description.	0
21	P0_21REI	Rising Edge Interrupt status for P0[21]. See bit 0 description.	0
22	P0_22REI	Rising Edge Interrupt status for P0[22]. See bit 0 description.	0
23	P0_23REI	Rising Edge Interrupt status for P0[23]. See bit 0 description.	0
24	P0_24REI	Rising Edge Interrupt status for P0[24]. See bit 0 description.	0
25	P0_25REI	Rising Edge Interrupt status for P0[25]. See bit 0 description.	0
26	P0_26REI	Rising Edge Interrupt status for P0[26]. See bit 0 description.	0
27	P0_27REI	Rising Edge Interrupt status for P0[27]. See bit 0 description.	0
28	P0_28REI	Rising Edge Interrupt status for P0[28]. See bit 0 description.	0
29	P0_29REI	Rising Edge Interrupt status for P0[29]. See bit 0 description.	0
30	P0_30REI	Rising Edge Interrupt status for P0[30]. See bit 0 description.	0
31	P0_31REI	Rising Edge Interrupt status for P0[31]. See bit 0 description.	0

8.5.2.3 GPIO Interrupt Status for port 0 Falling Edge Interrupt

Each bit in these read-only registers indicates the falling edge interrupt status for port 0.

Table 103. GPIO Interrupt Status for port 0 Falling Edge Interrupt (STATF0 - 0x4002 8088) bit description

Bit	Symbol	Description	Reset value
0	P0_0FEI	Falling Edge Interrupt status for P0[0]. 0 = No falling edge detected. 1 = Falling edge interrupt generated.	0
1	P0_1FEI	Falling Edge Interrupt status for P0[1]. See bit 0 description.	0
2	P0_2FEI	Falling Edge Interrupt status for P0[2]. See bit 0 description.	0
3	P0_3FEI	Falling Edge Interrupt status for P0[3]. See bit 0 description.	0
4	P0_4FEI	Falling Edge Interrupt status for P0[4]. See bit 0 description.	0
5	P0_5FEI	Falling Edge Interrupt status for P0[5]. See bit 0 description.	0
6	P0_6FEI	Falling Edge Interrupt status for P0[6]. See bit 0 description.	0
7	P0_7FEI	Falling Edge Interrupt status for P0[7]. See bit 0 description.	0
8	P0_8FEI	Falling Edge Interrupt status for P0[8]. See bit 0 description.	0
9	P0_9FEI	Falling Edge Interrupt status for P0[9]. See bit 0 description.	0
10	P0_10FEI	Falling Edge Interrupt status for P0[10]. See bit 0 description.	0
11	P0_11FEI	Falling Edge Interrupt status for P0[11]. See bit 0 description.	0
12	P0_12FEI	Falling Edge Interrupt status for P0[12]. See bit 0description.	0
13	P0_13FEI	Falling Edge Interrupt status for P0[13]. See bit 0 description.	0
14	P0_14FEI	Falling Edge Interrupt status for P0[14]. See bit 0 description.	0
15	P0_15FEI	Falling Edge Interrupt status for P0[15]. See bit 0 description.	0
16	P0_16FEI	Falling Edge Interrupt status for P0[16]. See bit 0 description.	0
17	P0_17FEI	Falling Edge Interrupt status for P0[17]. See bit 0 description.	0
18	P0_18FEI	Falling Edge Interrupt status for P0[18]. See bit 0 description.	0
19	P0_19FEI	Falling Edge Interrupt status for P0[19]. See bit 0 description.	0
20	P0_20FEI	Falling Edge Interrupt status for P0[20]. See bit 0 description.	0
21	P0_21FEI	Falling Edge Interrupt status for P0[21]. See bit 0 description.	0
22	P0_22FEI	Falling Edge Interrupt status for P0[22]. See bit 0 description.	0
23	P0_23FEI	Falling Edge Interrupt status for P0[23]. See bit 0 description.	0
24	P0_24FEI	Falling Edge Interrupt status for P0[24]. See bit 0 description.	0
25	P0_25FEI	Falling Edge Interrupt status for P0[25]. See bit 0 description.	0
26	P0_26FEI	Falling Edge Interrupt status for P0[26]. See bit 0 description.	0
27	P0_27FEI	Falling Edge Interrupt status for P0[27]. See bit 0 description.	0
28	P0_28FEI	Falling Edge Interrupt status for P0[28]. See bit 0 description.	0
29	P0_29FEI	Falling Edge Interrupt status for P0[29]. See bit 0 description.	0
30	P0_30FEI	Falling Edge Interrupt status for P0[30]. See bit 0 description.	0
31	P0_31FEI	Falling Edge Interrupt status for P0[31]. See bit 0 description.	0

8.5.2.4 GPIO Interrupt Clear register for port 0

Writing a 1 into a bit in this write-only register clears any interrupts for the corresponding port 0 pin.

Table 104. GPIO Interrupt Clear register for port 0 (CLR0 - 0x4002 808C) bit description

Bit	Symbol	Description
0	P0_0CI	Clear GPIO port Interrupts for P0[0]. 0 = No effect. 1 = Clear corresponding bits in IOnINTSTATR and IOnSTATF.
1	P0_1CI	Clear GPIO port Interrupts for P0[1]. See bit 0 description.
2	P0_2CI	Clear GPIO port Interrupts for P0[2]. See bit 0 description.
3	P0_3CI	Clear GPIO port Interrupts for P0[3]. See bit 0 description.
4	P0_4CI	Clear GPIO port Interrupts for P0[4]. See bit 0 description.
5	P0_5CI	Clear GPIO port Interrupts for P0[5]. See bit 0 description.
6	P0_6CI	Clear GPIO port Interrupts for P0[6]. See bit 0 description.
7	P0_7CI	Clear GPIO port Interrupts for P0[7]. See bit 0 description.
8	P0_8CI	Clear GPIO port Interrupts for P0[8]. See bit 0 description.
9	P0_9CI	Clear GPIO port Interrupts for P0[9]. See bit 0 description.
10	P0_10CI	Clear GPIO port Interrupts for P0[10]. See bit 0 description.
11	P0_11CI	Clear GPIO port Interrupts for P0[11]. See bit 0 description.
12	P0_12CI	Clear GPIO port Interrupts for P0[12]. See bit 0 description.
13	P0_13CI	Clear GPIO port Interrupts for P0[13]. See bit 0 description.
14	P0_14CI	Clear GPIO port Interrupts for P0[14]. See bit 0 description.
15	P0_15CI	Clear GPIO port Interrupts for P0[15]. See bit 0 description.
16	P0_16CI	Clear GPIO port Interrupts for P0[16]. See bit 0 description.
17	P0_17CI	Clear GPIO port Interrupts for P0[17]. See bit 0 description.
18	P0_18CI	Clear GPIO port Interrupts for P0[18]. See bit 0 description.
19	P0_19CI	Clear GPIO port Interrupts for P0[19]. See bit 0 description.
20	P0_20CI	Clear GPIO port Interrupts for P0[20]. See bit 0 description.
21	P0_21CI	Clear GPIO port Interrupts for P0[21]. See bit 0 description.
22	P0_22CI	Clear GPIO port Interrupts for P0[22]. See bit 0 description.
23	P0_23CI	Clear GPIO port Interrupts for P0[23]. See bit 0 description.
24	P0_24CI	Clear GPIO port Interrupts for P0[24]. See bit 0 description.
25	P0_25CI	Clear GPIO port Interrupts for P0[25]. See bit 0 description.
26	P0_26CI	Clear GPIO port Interrupts for P0[26]. See bit 0 description.
27	P0_27CI	Clear GPIO port Interrupts for P0[27]. See bit 0 description.
28	P0_28CI	Clear GPIO port Interrupts for P0[28]. See bit 0 description.
29	P0_29CI	Clear GPIO port Interrupts for P0[29]. See bit 0 description.
30	P0_30CI	Clear GPIO port Interrupts for P0[30]. See bit 0 description.
31	P0_31CI	Clear GPIO port Interrupts for P0[31]. See bit 0 description.

8.5.2.5 GPIO Interrupt Enable for port 0 Rising Edge

Each bit in these read-write registers enables the rising edge interrupt for the corresponding port 0 pin.

Which pins are available depends on the part number and package combination. See the specific device data sheet for details.

Table 105. GPIO Interrupt Enable for port 0 Rising Edge (ENR0 - 0x4002 8090) bit description

Bit	Symbol	Description	Reset value
0	P0_0ER	Enable rising edge interrupt for P0[0]. 0 = Disable rising edge interrupt. 1 = Enable rising edge interrupt.	0
1	P0_1ER	Enable rising edge interrupt for P0[1]. See bit 0 description.	0
2	P0_2ER	Enable rising edge interrupt for P0[2]. See bit 0 description.	0
3	P0_3ER	Enable rising edge interrupt for P0[3]. See bit 0 description.	0
4	P0_4ER	Enable rising edge interrupt for P0[4]. See bit 0 description.	0
5	P0_5ER	Enable rising edge interrupt for P0[5]. See bit 0 description.	0
6	P0_6ER	Enable rising edge interrupt for P0[6]. See bit 0 description.	0
7	P0_7ER	Enable rising edge interrupt for P0[7]. See bit 0 description.	0
8	P0_8ER	Enable rising edge interrupt for P0[8]. See bit 0 description.	0
9	P0_9ER	Enable rising edge interrupt for P0[9]. See bit 0 description.	0
10	P0_10ER	Enable rising edge interrupt for P0[10]. See bit 0 description.	0
11	P0_11ER	Enable rising edge interrupt for P0[11]. See bit 0 description.	0
12	P0_12ER	Enable rising edge interrupt for P0[12]. See bit 0 description.	0
13	P0_13ER	Enable rising edge interrupt for P0[13]. See bit 0 description.	0
14	P0_14ER	Enable rising edge interrupt for P0[14]. See bit 0 description.	0
15	P0_15ER	Enable rising edge interrupt for P0[15]. See bit 0 description.	0
16	P0_16ER	Enable rising edge interrupt for P0[16]. See bit 0 description.	0
17	P0_17ER	Enable rising edge interrupt for P0[17]. See bit 0 description.	0
18	P0_18ER	Enable rising edge interrupt for P0[18]. See bit 0 description.	0
19	P0_19ER	Enable rising edge interrupt for P0[19]. See bit 0 description.	0
20	P0_20ER	Enable rising edge interrupt for P0[20]. See bit 0 description.	0
21	P0_21ER	Enable rising edge interrupt for P0[21]. See bit 0 description.	0
22	P0_22ER	Enable rising edge interrupt for P0[22]. See bit 0 description.	0
23	P0_23ER	Enable rising edge interrupt for P0[23]. See bit 0 description.	0
24	P0_24ER	Enable rising edge interrupt for P0[24]. See bit 0 description.	0
25	P0_25ER	Enable rising edge interrupt for P0[25]. See bit 0 description.	0
26	P0_26ER	Enable rising edge interrupt for P0[26]. See bit 0 description.	0
27	P0_27ER	Enable rising edge interrupt for P0[27]. See bit 0 description.	0
28	P0_28ER	Enable rising edge interrupt for P0[28]. See bit 0 description.	0
29	P0_29ER	Enable rising edge interrupt for P0[29]. See bit 0 description.	0
30	P0_30ER	Enable rising edge interrupt for P0[30]. See bit 0 description.	0
31	P0_31ER	Enable rising edge interrupt for P0[31]. See bit 0 description.	0

8.5.2.6 GPIO Interrupt Enable for port 0 Falling Edge

Each bit in these read-write registers enables the falling edge interrupt for the corresponding GPIO port 0 pin.

Table 106. GPIO Interrupt Enable for port 0 Falling Edge (ENF0 - address 0x4002 8094) bit description

Bit	Symbol	Description	Reset value
0	P0_0EF	Enable falling edge interrupt for P0[0]. 0 = Disable falling edge interrupt. 1 = Enable falling edge interrupt.	0
1	P0_1EF	Enable falling edge interrupt for P0[1]. See bit 0 description.	0
2	P0_2EF	Enable falling edge interrupt for P0[2]. See bit 0 description.	0
3	P0_3EF	Enable falling edge interrupt for P0[3]. See bit 0 description.	0
4	P0_4EF	Enable falling edge interrupt for P0[4]. See bit 0 description.	0
5	P0_5EF	Enable falling edge interrupt for P0[5]. See bit 0 description.	0
6	P0_6EF	Enable falling edge interrupt for P0[6]. See bit 0 description.	0
7	P0_7EF	Enable falling edge interrupt for P0[7]. See bit 0 description.	0
8	P0_8EF	Enable falling edge interrupt for P0[8]. See bit 0 description.	0
9	P0_9EF	Enable falling edge interrupt for P0[9]. See bit 0 description.	0
10	P0_10EF	Enable falling edge interrupt for P0[10]. See bit 0 description.	0
11	P0_11EF	Enable falling edge interrupt for P0[11]. See bit 0 description.	0
12	P0_12EF	Enable falling edge interrupt for P0[12]. See bit 0 description.	0
13	P0_13EF	Enable falling edge interrupt for P0[13]. See bit 0 description.	0
14	P0_14EF	Enable falling edge interrupt for P0[14]. See bit 0 description.	0
15	P0_15EF	Enable falling edge interrupt for P0[15]. See bit 0 description.	0
16	P0_16EF	Enable falling edge interrupt for P0[16]. See bit 0 description.	0
17	P0_17EF	Enable falling edge interrupt for P0[17]. See bit 0 description.	0
18	P0_18EF	Enable falling edge interrupt for P0[18]. See bit 0 description.	0
19	P0_19EF	Enable falling edge interrupt for P0[19]. See bit 0 description.	0
20	P0_20EF	Enable falling edge interrupt for P0[20]. See bit 0 description.	0
21	P0_21EF	Enable falling edge interrupt for P0[21]. See bit 0 description.	0
22	P0_22EF	Enable falling edge interrupt for P0[22]. See bit 0 description.	0
23	P0_23EF	Enable falling edge interrupt for P0[23]. See bit 0 description.	0
24	P0_24EF	Enable falling edge interrupt for P0[24]. See bit 0 description.	0
25	P0_25EF	Enable falling edge interrupt for P0[25]. See bit 0 description.	0
26	P0_26EF	Enable falling edge interrupt for P0[26]. See bit 0 description.	0
27	P0_27EF	Enable falling edge interrupt for P0[27]. See bit 0 description.	0
28	P0_28EF	Enable falling edge interrupt for P0[28]. See bit 0 description.	0
29	P0_29EF	Enable falling edge interrupt for P0[29]. See bit 0 description.	0
30	P0_30EF	Enable falling edge interrupt for P0[30]. See bit 0 description.	0
31	P0_31EF	Enable falling edge interrupt for P0[31]. See bit 0 description.	0

8.5.2.7 GPIO Interrupt Status for port 2 Rising Edge Interrupt

Each bit in these read-only registers indicates the rising edge interrupt status for port 2.

Table 107. GPIO Interrupt Status for port 2 Rising Edge Interrupt (STATR2 - 0x4002 80A4) bit description

Bit	Symbol	Description	Reset value
0	P2_0REI	Status of Rising Edge Interrupt for P2[0]. 0 = No rising edge detected. 1 = Rising edge interrupt generated.	0
1	P2_1REI	Status of Rising Edge Interrupt for P2[1]. See bit 0 description.	0
2	P2_2REI	Status of Rising Edge Interrupt for P2[2]. See bit 0 description.	0
3	P2_3REI	Status of Rising Edge Interrupt for P2[3]. See bit 0 description.	0
4	P2_4REI	Status of Rising Edge Interrupt for P2[4]. See bit 0 description.	0
5	P2_5REI	Status of Rising Edge Interrupt for P2[5]. See bit 0 description.	0
6	P2_6REI	Status of Rising Edge Interrupt for P2[6]. See bit 0 description.	0
7	P2_7REI	Status of Rising Edge Interrupt for P2[7]. See bit 0 description.	0
8	P2_8REI	Status of Rising Edge Interrupt for P2[8]. See bit 0 description.	0
9	P2_9REI	Status of Rising Edge Interrupt for P2[9]. See bit 0 description.	0
10	P2_10REI	Status of Rising Edge Interrupt for P2[10]. See bit 0 description.	0
11	P2_11REI	Status of Rising Edge Interrupt for P2[11]. See bit 0 description.	0
12	P2_12REI	Status of Rising Edge Interrupt for P2[12]. See bit 0 description.	0
13	P2_13REI	Status of Rising Edge Interrupt for P2[13]. See bit 0 description.	0
14	P2_14REI	Status of Rising Edge Interrupt for P2[14]. See bit 0 description.	0
15	P2_15REI	Status of Rising Edge Interrupt for P2[15]. See bit 0 description.	0
16	P2_16REI	Status of Rising Edge Interrupt for P2[16]. See bit 0 description.	0
17	P2_17REI	Status of Rising Edge Interrupt for P2[17]. See bit 0 description.	0
18	P2_18REI	Status of Rising Edge Interrupt for P2[18]. See bit 0 description.	0
19	P2_19REI	Status of Rising Edge Interrupt for P2[19]. See bit 0 description.	0
20	P2_20REI	Status of Rising Edge Interrupt for P2[20]. See bit 0 description.	0
21	P2_21REI	Status of Rising Edge Interrupt for P2[21]. See bit 0 description.	0
22	P2_22REI	Status of Rising Edge Interrupt for P2[22]. See bit 0 description.	0
23	P2_23REI	Status of Rising Edge Interrupt for P2[23]. See bit 0 description.	0
24	P2_24REI	Status of Rising Edge Interrupt for P2[24]. See bit 0 description.	0
25	P2_25REI	Status of Rising Edge Interrupt for P2[25]. See bit 0 description.	0
26	P2_26REI	Status of Rising Edge Interrupt for P2[26]. See bit 0 description.	0
27	P2_27REI	Status of Rising Edge Interrupt for P2[27]. See bit 0 description.	0
28	P2_28REI	Status of Rising Edge Interrupt for P2[28]. See bit 0 description.	0
29	P2_29REI	Status of Rising Edge Interrupt for P2[29]. See bit 0 description.	0
30	P2_30REI	Status of Rising Edge Interrupt for P2[30]. See bit 0 description.	0
31	P2_31REI	Status of Rising Edge Interrupt for P2[31]. See bit 0 description.	0

8.5.2.8 GPIO Interrupt Status for port 2 Falling Edge Interrupt

Each bit in these read-only registers indicates the falling edge interrupt status for port 2.

Table 108. GPIO Interrupt Status for port 2 Falling Edge Interrupt (STATF2 - 0x4002 80A8) bit description

Bit	Symbol	Description	Reset value
0	P2_0FEI	Status of Falling Edge Interrupt for P2[0]. 0 = No falling edge detected. 1 = Falling edge interrupt generated.	0
1	P2_1FEI	Status of Falling Edge Interrupt for P2[1]. See bit 0 description.	0
2	P2_2FEI	Status of Falling Edge Interrupt for P2[2]. See bit 0 description.	0
3	P2_3FEI	Status of Falling Edge Interrupt for P2[3]. See bit 0 description.	0
4	P2_4FEI	Status of Falling Edge Interrupt for P2[4]. See bit 0 description.	0
5	P2_5FEI	Status of Falling Edge Interrupt for P2[5]. See bit 0 description.	0
6	P2_6FEI	Status of Falling Edge Interrupt for P2[6]. See bit 0 description.	0
7	P2_7FEI	Status of Falling Edge Interrupt for P2[7]. See bit 0 description.	0
8	P2_8FEI	Status of Falling Edge Interrupt for P2[8]. See bit 0 description.	0
9	P2_9FEI	Status of Falling Edge Interrupt for P2[9]. See bit 0 description.	0
10	P2_10FEI	Status of Falling Edge Interrupt for P2[10]. See bit 0 description.	0
11	P2_11FEI	Status of Falling Edge Interrupt for P2[11]. See bit 0 description.	0
12	P2_12FEI	Status of Falling Edge Interrupt for P2[12]. See bit 0 description.	0
13	P2_13FEI	Status of Falling Edge Interrupt for P2[13]. See bit 0 description.	0
14	P2_14FEI	Status of Falling Edge Interrupt for P2[14]. See bit 0 description.	0
15	P2_15FEI	Status of Falling Edge Interrupt for P2[15]. See bit 0 description.	0
16	P2_16FEI	Status of Falling Edge Interrupt for P2[16]. See bit 0 description.	0
17	P2_17FEI	Status of Falling Edge Interrupt for P2[17]. See bit 0 description.	0
18	P2_18FEI	Status of Falling Edge Interrupt for P2[18]. See bit 0 description.	0
19	P2_19FEI	Status of Falling Edge Interrupt for P2[19]. See bit 0 description.	0
20	P2_20FEI	Status of Falling Edge Interrupt for P2[20]. See bit 0 description.	0
21	P2_21FEI	Status of Falling Edge Interrupt for P2[21]. See bit 0 description.	0
22	P2_22FEI	Status of Falling Edge Interrupt for P2[22]. See bit 0 description.	0
23	P2_23FEI	Status of Falling Edge Interrupt for P2[23]. See bit 0 description.	0
24	P2_24FEI	Status of Falling Edge Interrupt for P2[24]. See bit 0 description.	0
25	P2_25FEI	Status of Falling Edge Interrupt for P2[25]. See bit 0 description.	0
26	P2_26FEI	Status of Falling Edge Interrupt for P2[26]. See bit 0 description.	0
27	P2_27FEI	Status of Falling Edge Interrupt for P2[27]. See bit 0 description.	0
28	P2_28FEI	Status of Falling Edge Interrupt for P2[28]. See bit 0 description.	0
29	P2_29FEI	Status of Falling Edge Interrupt for P2[29]. See bit 0 description.	0
30	P2_30FEI	Status of Falling Edge Interrupt for P2[30]. See bit 0 description.	0
31	P2_31FEI	Status of Falling Edge Interrupt for P2[31]. See bit 0 description.	0

8.5.2.9 GPIO Interrupt Clear register for port 2

Writing a 1 into a bit in this write-only register clears any interrupts for the corresponding port 2 pin.

Table 109. GPIO Interrupt Clear register for port 0 (CLR2 - 0x4002 80AC) bit description

Bit	Symbol	Description
0	P2_0CI	Clear GPIO port Interrupts for P2[0]. 0 = No effect. 1 = Clear corresponding bits in IOnINTSTATR and IOnSTATF.
1	P2_1CI	Clear GPIO port Interrupts for P2[1]. See bit 0 description.
2	P2_2CI	Clear GPIO port Interrupts for P2[2]. See bit 0 description.
3	P2_3CI	Clear GPIO port Interrupts for P2[3]. See bit 0 description.
4	P2_4CI	Clear GPIO port Interrupts for P2[4]. See bit 0 description.
5	P2_5CI	Clear GPIO port Interrupts for P2[5]. See bit 0 description.
6	P2_6CI	Clear GPIO port Interrupts for P2[6]. See bit 0 description.
7	P2_7CI	Clear GPIO port Interrupts for P2[7]. See bit 0 description.
8	P2_8CI	Clear GPIO port Interrupts for P2[8]. See bit 0 description.
9	P2_9CI	Clear GPIO port Interrupts for P2[9]. See bit 0 description.
10	P2_10CI	Clear GPIO port Interrupts for P2[10]. See bit 0 description.
11	P2_11CI	Clear GPIO port Interrupts for P2[11]. See bit 0 description.
12	P2_12CI	Clear GPIO port Interrupts for P2[12]. See bit 0 description.
13	P2_13CI	Clear GPIO port Interrupts for P2[13]. See bit 0 description.
14	P2_14CI	Clear GPIO port Interrupts for P2[14]. See bit 0 description.
15	P2_15CI	Clear GPIO port Interrupts for P2[15]. See bit 0 description.
16	P2_16CI	Clear GPIO port Interrupts for P2[16]. See bit 0 description.
17	P2_17CI	Clear GPIO port Interrupts for P2[17]. See bit 0 description.
18	P2_18CI	Clear GPIO port Interrupts for P2[18]. See bit 0 description.
19	P2_19CI	Clear GPIO port Interrupts for P2[19]. See bit 0 description.
20	P2_20CI	Clear GPIO port Interrupts for P2[20]. See bit 0 description.
21	P2_21CI	Clear GPIO port Interrupts for P2[21]. See bit 0 description.
22	P2_22CI	Clear GPIO port Interrupts for P2[22]. See bit 0 description.
23	P2_23CI	Clear GPIO port Interrupts for P2[23]. See bit 0 description.
24	P2_24CI	Clear GPIO port Interrupts for P2[24]. See bit 0 description.
25	P2_25CI	Clear GPIO port Interrupts for P2[25]. See bit 0 description.
26	P2_26CI	Clear GPIO port Interrupts for P2[26]. See bit 0 description.
27	P2_27CI	Clear GPIO port Interrupts for P2[27]. See bit 0 description.
28	P2_28CI	Clear GPIO port Interrupts for P2[28]. See bit 0 description.
29	P2_29CI	Clear GPIO port Interrupts for P2[29]. See bit 0 description.
30	P2_30CI	Clear GPIO port Interrupts for P2[30]. See bit 0 description.
31	P2_31CI	Clear GPIO port Interrupts for P2[31]. See bit 0 description.

8.5.2.10 GPIO Interrupt Enable for port 2 Rising Edge

Each bit in these read-write registers enables the rising edge interrupt for the corresponding port 2 pin.

Which pins are available depends on the part number and package combination. See the specific device data sheet for details.

Table 110. GPIO Interrupt Enable for port 2 Rising Edge (ENR2 - 0x4002 80B0) bit description

Bit	Symbol	Description	Reset value
0	P2_0ER	Enable rising edge interrupt for P2[0]. 0 = Disable rising edge interrupt. 1 = Enable rising edge interrupt.	0
1	P2_1ER	Enable rising edge interrupt for P2[1]. See bit 0 description.	0
2	P2_2ER	Enable rising edge interrupt for P2[2]. See bit 0 description.	0
3	P2_3ER	Enable rising edge interrupt for P2[3]. See bit 0 description.	0
4	P2_4ER	Enable rising edge interrupt for P2[4]. See bit 0 description.	0
5	P2_5ER	Enable rising edge interrupt for P2[5]. See bit 0 description.	0
6	P2_6ER	Enable rising edge interrupt for P2[6]. See bit 0 description.	0
7	P2_7ER	Enable rising edge interrupt for P2[7]. See bit 0 description.	0
8	P2_8ER	Enable rising edge interrupt for P2[8]. See bit 0 description.	0
9	P2_9ER	Enable rising edge interrupt for P2[9]. See bit 0 description.	0
10	P2_10ER	Enable rising edge interrupt for P2[10]. See bit 0 description.	0
11	P2_11ER	Enable rising edge interrupt for P2[11]. See bit 0 description.	0
12	P2_12ER	Enable rising edge interrupt for P2[12]. See bit 0 description.	0
13	P2_13ER	Enable rising edge interrupt for P2[13]. See bit 0 description.	0
14	P2_14ER	Enable rising edge interrupt for P2[14]. See bit 0 description.	0
15	P2_15ER	Enable rising edge interrupt for P2[15]. See bit 0 description.	0
16	P2_16ER	Enable rising edge interrupt for P2[16]. See bit 0 description.	0
17	P2_17ER	Enable rising edge interrupt for P2[17]. See bit 0 description.	0
18	P2_18ER	Enable rising edge interrupt for P2[18]. See bit 0 description.	0
19	P2_19ER	Enable rising edge interrupt for P2[19]. See bit 0 description.	0
20	P2_20ER	Enable rising edge interrupt for P2[20]. See bit 0 description.	0
21	P2_21ER	Enable rising edge interrupt for P2[21]. See bit 0 description.	0
22	P2_22ER	Enable rising edge interrupt for P2[22]. See bit 0 description.	0
23	P2_23ER	Enable rising edge interrupt for P2[23]. See bit 0 description.	0
24	P2_24ER	Enable rising edge interrupt for P2[24]. See bit 0 description.	0
25	P2_25ER	Enable rising edge interrupt for P2[25]. See bit 0 description.	0
26	P2_26ER	Enable rising edge interrupt for P2[26]. See bit 0 description.	0
27	P2_27ER	Enable rising edge interrupt for P2[27]. See bit 0 description.	0
28	P2_28ER	Enable rising edge interrupt for P2[28]. See bit 0 description.	0
29	P2_29ER	Enable rising edge interrupt for P2[29]. See bit 0 description.	0
30	P2_30ER	Enable rising edge interrupt for P2[30]. See bit 0 description.	0
31	P2_31ER	Enable rising edge interrupt for P2[31]. See bit 0 description.	0

8.5.2.11 GPIO Interrupt Enable for port 2 Falling Edge

Each bit in these read-write registers enables the falling edge interrupt for the corresponding GPIO port 2 pin.

Table 111. GPIO Interrupt Enable for port 2 Falling Edge (ENF2 - 0x4002 80B4) bit description

Bit	Symbol	Description	Reset value
0	P2_0EF	Enable falling edge interrupt for P2[0]. 0 = Disable falling edge interrupt. 1 = Enable falling edge interrupt.	0
1	P2_1EF	Enable falling edge interrupt for P2[1]. See bit 0 description.	0
2	P2_2EF	Enable falling edge interrupt for P2[2]. See bit 0 description.	0
3	P2_3EF	Enable falling edge interrupt for P2[3]. See bit 0 description.	0
4	P2_4EF	Enable falling edge interrupt for P2[4]. See bit 0 description.	0
5	P2_5EF	Enable falling edge interrupt for P2[5]. See bit 0 description.	0
6	P2_6EF	Enable falling edge interrupt for P2[6]. See bit 0 description.	0
7	P2_7EF	Enable falling edge interrupt for P2[7]. See bit 0 description.	0
8	P2_8EF	Enable falling edge interrupt for P2[8]. See bit 0 description.	0
9	P2_9EF	Enable falling edge interrupt for P2[9]. See bit 0 description.	0
10	P2_10EF	Enable falling edge interrupt for P2[10]. See bit 0 description.	0
11	P2_11EF	Enable falling edge interrupt for P2[11]. See bit 0 description.	0
12	P2_12EF	Enable falling edge interrupt for P2[12]. See bit 0 description.	0
13	P2_13EF	Enable falling edge interrupt for P2[13]. See bit 0 description.	0
14	P2_14EF	Enable falling edge interrupt for P2[14]. See bit 0 description.	0
15	P2_15EF	Enable falling edge interrupt for P2[15]. See bit 0 description.	0
16	P2_16EF	Enable falling edge interrupt for P2[16]. See bit 0 description.	0
17	P2_17EF	Enable falling edge interrupt for P2[17]. See bit 0 description.	0
18	P2_18EF	Enable falling edge interrupt for P2[18]. See bit 0 description.	0
19	P2_19EF	Enable falling edge interrupt for P2[19]. See bit 0 description.	0
20	P2_20EF	Enable falling edge interrupt for P2[20]. See bit 0 description.	0
21	P2_21EF	Enable falling edge interrupt for P2[21]. See bit 0 description.	0
22	P2_22EF	Enable falling edge interrupt for P2[22]. See bit 0 description.	0
23	P2_23EF	Enable falling edge interrupt for P2[23]. See bit 0 description.	0
24	P2_24EF	Enable falling edge interrupt for P2[24]. See bit 0 description.	0
25	P2_25EF	Enable falling edge interrupt for P2[25]. See bit 0 description.	0
26	P2_26EF	Enable falling edge interrupt for P2[26]. See bit 0 description.	0
27	P2_27EF	Enable falling edge interrupt for P2[27]. See bit 0 description.	0
28	P2_28EF	Enable falling edge interrupt for P2[28]. See bit 0 description.	0
29	P2_29EF	Enable falling edge interrupt for P2[29]. See bit 0 description.	0
30	P2_30EF	Enable falling edge interrupt for P2[30]. See bit 0 description.	0
31	P2_31EF	Enable falling edge interrupt for P2[31]. See bit 0 description.	0

8.6 GPIO usage notes

8.6.1 Example: An instantaneous output of 0s and 1s on a GPIO port

Solution 1: using 32-bit (word) accessible fast GPIO registers

```
FIO0MASK = 0xFFFF00FF ;
FIO0PIN  = 0x0000A500;
```

Solution 2: using 16-bit (half-word) accessible fast GPIO registers

```
FIO0MASKL = 0x00FF;
FIO0PINL  = 0xA500;
```

Solution 3: using 8-bit (byte) accessible fast GPIO registers

```
FIO0PIN1  = 0xA5;
```

8.6.2 Writing to FIOSET/FIOCLR vs. FIOPIN

Writing to the FIOSET/FIOCLR registers allow a program to easily change a port's output pin(s) to both high and low levels at the same time. When FIOSET or FIOCLR are used, only pin/bit(s) written with 1 will be changed, while those written as 0 will remain unaffected.

Writing to the FIOPIN register enables instantaneous output of a desired value on the parallel GPIO. Data written to the FIOPIN register will affect all pins configured as outputs on that port: zeroes in the value will produce low level pin outputs and ones in the value will produce high level pin outputs.

A subset of a port's pins may be changed by using the FIOMASK register to define which pins are affected. FIOMASK is set up to contain zeroes in bits corresponding to pins that will be changed, and ones for all others. Solution 2 from Section 8.6.1 above illustrates output of 0xA5 on PORT0 pins 15 to 8 while preserving all other PORT0 output pins as they were before.

9.1 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 112.

Table 112. EMC configuration

Device package	Data bus widths supported	Pins available	Dynamic memory configuration registers[1][2]	Static memory configuration registers[1][3]	External memory connections
144-pin	8-bit	EMC_A[15:0] EMC_D[7:0] $\overline{\text{EMC\_OE}}$ $\overline{\text{EMC\_WE}}$ $\overline{\text{EMC\_CS1:0}}$		EMCStaticConfig1/0 EMCStaticWaitWen1/0 EMCStaticWaitOen1/0 EMCStaticWaitRd1/0 EMCStaticWaitPage1/0 EMCStaticWaitWr1/0 EMCStaticWaitTurn1/0	Section 9.14.3
180-pin	16-bit, 8-bit	EMC_A[19:0] EMC_D[15:0] $\overline{\text{EMC\_OE}}$ $\overline{\text{EMC\_WE}}$ $\overline{\text{EMC\_BLS1:0}}$ $\overline{\text{EMC\_CS1:0}}$ $\overline{\text{EMC\_DYCS1:0}}$ $\overline{\text{EMC\_CAS}}$ $\overline{\text{EMC\_RAS}}$ EMC_CLK1:0 EMC_CKE1:0 EMC_DQM1:0	EMCDynamicConfig1/0 EMCDynamicRasCas1/0	EMCStaticConfig1/0 EMCStaticWaitWen1/0 EMCStaticWaitOen1/0 EMCStaticWaitRd1/0 EMCStaticWaitPage1/0 EMCStaticWaitWr1/0 EMCStaticWaitTurn1/0	Section 9.14.2 Section 9.14.3
208-pin	32-bit, 16-bit, 8-bit	EMC_A[25:0] EMC_D[31:0] $\overline{\text{EMC\_OE}}$ $\overline{\text{EMC\_WE}}$ $\overline{\text{EMC\_BLS3:0}}$ $\overline{\text{EMC\_CS3:0}}$ $\overline{\text{EMC\_DYCS3:0}}$ $\overline{\text{EMC\_CAS}}$ $\overline{\text{EMC\_RAS}}$ EMC_CLK1:0 EMC_CKE3:0 EMC_DQM3:0	EMCDynamicConfig3/2/1/0 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 9.14.1 Section 9.14.2 Section 9.14.3

[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.

9.2 Basic configuration

The EMC is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCEMC.

   Remark: The EMC is enabled on reset (PCEMC = 1). On POR and warm reset, the EMC is enabled as well, see Table 116 and Table 119.

2. Clock: The EMC clock (EMCCLK) can be the same as the CPU clock (the default), or half of the CPU clock CCLK. 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 3.3.3.1.

3. Pins: Select EMC pins and pin modes through the relevant IOCON registers (Section 7.4.1).

4. Configuration: See Table 116 to Table 119. Also see additional EMC configurations in Section 3.3.7.1 “System Controls and Status 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 40.1 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 3.3.6.1).

7. To calibrate the EMC clock, see Section 3.3.6.2.

9.3 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, as well as dynamic memories such as Single Data Rate SDRAM. The EMC is an Advanced Microcontroller Bus Architecture (AMBA) compliant peripheral.

9.4 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 SCS register bit 0 (Section 3.3.7.1)
- 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.

9.5 EMC functional description

Figure 16 shows a block diagram of the EMC.

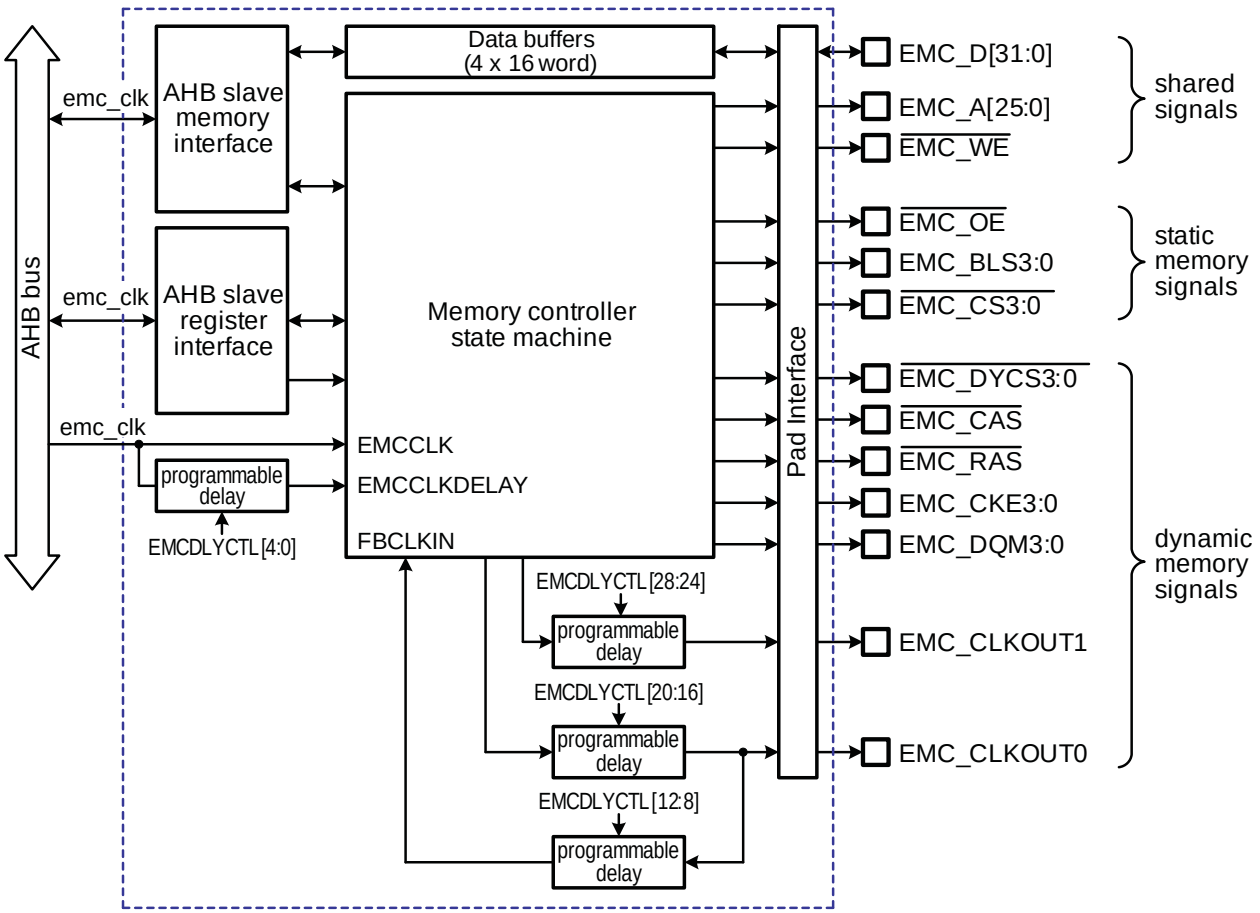


Fig 16. EMC block diagram

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.

9.5.1 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.

9.5.2 AHB slave memory interface

The AHB slave memory interface allows access to external memories.

9.5.2.1 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.

9.5.2.2 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.

9.5.2.3 Write protected memory areas

Write transactions to write-protected memory areas generate an ERROR response to the AHB bus and the transfer is terminated.

9.5.3 Pad interface

The pad interface block provides the interface to the pads. 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.

9.5.4 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.

9.5.4.1 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.

9.5.4.2 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.

9.5.5 Memory controller state machine

The memory controller state machine comprises a static memory controller and a dynamic memory controller.

9.5.6 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 clock delayed operating mode, separate programmable delays are provided for each potential clock output, CLKOUT0 and CLKOUT1.
- For the command delayed operating mode, a programmable delay is provided to control delay of all command outputs.
- For both operating modes, a programmable delay is provided to control the time at which input data from SDRAM memory is sampled.

The locations of the programmable delays are shown in the EMC overall block diagram (Figure 16). See descriptions of the EMCDLYCTL and EMCCAL registers for more information.

9.6 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.

9.6.1 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.

9.6.2 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.

9.7 Memory bank select

Eight independently-configurable memory chip selects are supported:

- Pins $\overline{\text{EMC\_CS3}}$ to $\overline{\text{EMC\_CS0}}$ are used to select static memory devices.
- Pins $\overline{\text{EMC\_DYCS3}}$ to $\overline{\text{EMC\_DYCS0}}$ 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 113 shows the address ranges of the chip selects.

Table 113. Memory bank selection

Chip select pin	Address range	Memory type	Size of range
$\overline{\text{EMC\_CS0}}$	0x8000 0000 - 0x83FF FFFF	Static	64 MB
$\overline{\text{EMC\_CS1}}$	0x9000 0000 - 0x93FF FFFF	Static	64 MB
$\overline{\text{EMC\_CS2}}$	0x9800 0000 - 0x9BFF FFFF	Static	64 MB
$\overline{\text{EMC\_CS3}}$	0x9C00 0000 - 0x9FFF FFFF	Static	64 MB
$\overline{\text{EMC\_DYCS0}}$	0xA000 0000 - 0xAFFF FFFF	Dynamic	256 MB
$\overline{\text{EMC\_DYCS1}}$	0xB000 0000 - 0xBFFF FFFF	Dynamic	256 MB
$\overline{\text{EMC\_DYCS2}}$	0xC000 0000 - 0xCFFF FFFF	Dynamic	256 MB
$\overline{\text{EMC\_DYCS3}}$	0xD000 0000 - 0xDFFF FFFF	Dynamic	256 MB

9.8 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 EMC_Reset_Disable, allows control of how the EMC is reset (see Section 3.3.7.1 “System Controls and Status register”). The default configuration (EMC_Reset_Disable = 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 EMC_Reset_Disable 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.

9.9 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 3.3.7.1 “System Controls and Status register”).

When the address shift mode is not activated (the EMC Shift Control bit in the SCS 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 SCS 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.

9.10 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 SCS register (see Section 3.3.7.1 “System Controls and Status register”).

9.11 Using the EMC with SDRAM

9.11.1 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 17 shows the layout for a JEDEC standard SDRAM mode register.

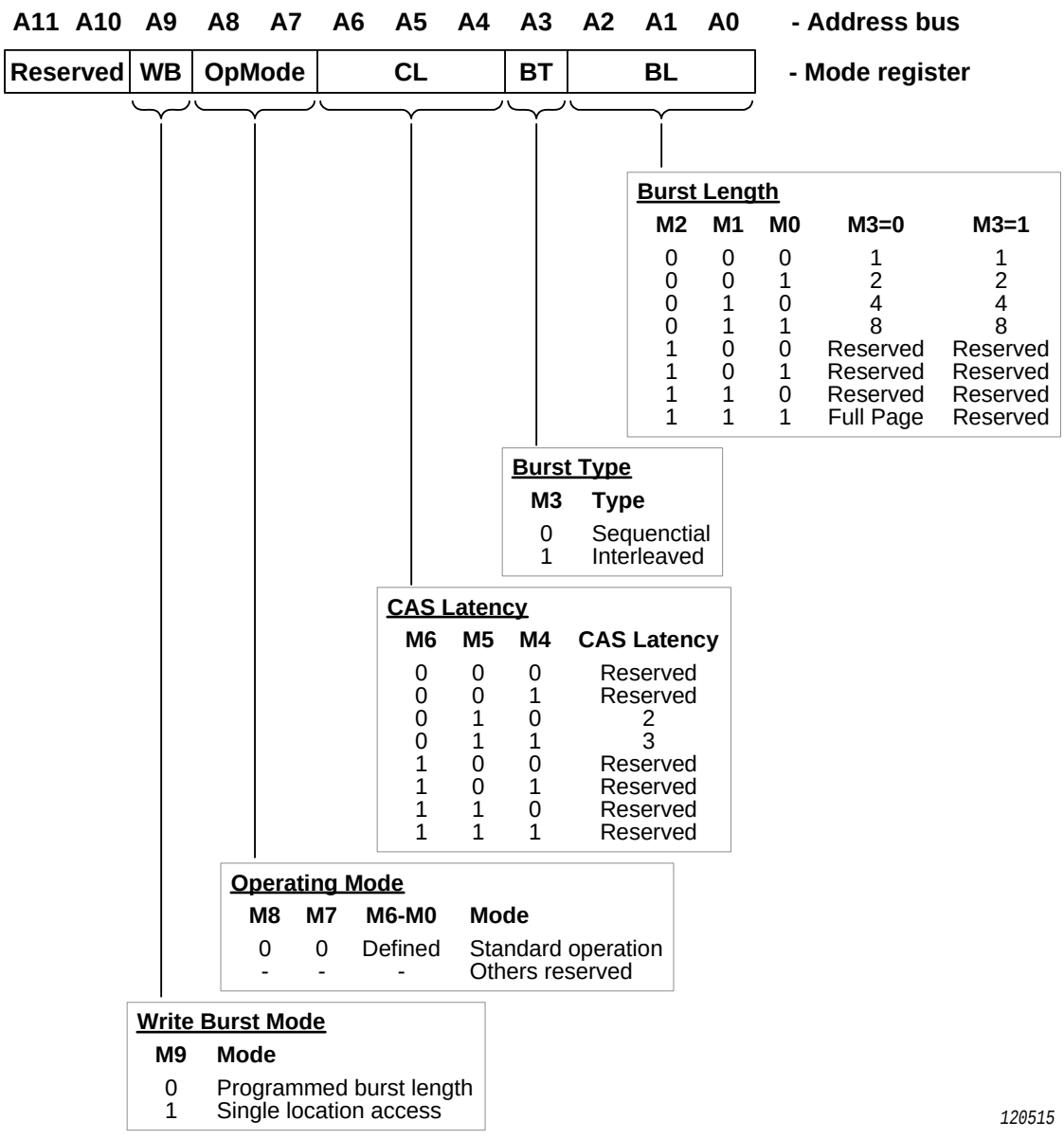


Fig 17. SDRAM mode register

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:

- a 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 17, 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 9.5.
- Bank bits and column bits, look up in Table 135. 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 + total bus width + bank select bits (RBC mode)

  OFFSET = number of columns + total bus width (BRC mode)
- 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

9.12 Pin description

Table 114 shows the interface and control signal pins for the EMC.

Table 114. 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.

9.13 Register description

This chapter describes the EMC registers and provides details required when programming the microcontroller. .

The EMC clock configuration and clock calibration registers are located in the system control block. See Section 3.3.6.1 and Section 3.3.6.2.

Table 115. Register overview: EMC (base address 0x2009 C000)

Register Name	Access	Address offset	Description	Warm Reset Value[1]	POR Reset Value[1]	Table
CONTROL	R/W	0x000	Controls operation of the memory controller.	0x1	0x3	116
STATUS	RO	0x004	Provides EMC status information.	-	0x5	117
CONFIG	R/W	0x008	Configures operation of the memory controller	-	0x0	118
DYNAMICCONTROL	R/W	0x020	Controls dynamic memory operation.	-	0x006	119
DYNAMICREFRESH	R/W	0x024	Configures dynamic memory refresh.	-	0x0	120
DYNAMICREADCONFIG	R/W	0x028	Configures dynamic memory read strategy.	-	0x0	121
DYNAMICRP	R/W	0x030	Precharge command period.	-	0x0F	122
DYNAMICRAS	R/W	0x034	Active to precharge command period.	-	0xF	123
DYNAMICSREX	R/W	0x038	Self-refresh exit time.	-	0xF	124
DYNAMICAPR	R/W	0x03C	Last-data-out to active command time.	-	0xF	125
DYNAMICDAL	R/W	0x040	Data-in to active command time.	-	0xF	126
DYNAMICWR	R/W	0x044	Write recovery time.	-	0xF	127
DYNAMICRC	R/W	0x048	Selects the active to active command period.	-	0x1F	128
DYNAMICRFC	R/W	0x04C	Selects the auto-refresh period.	-	0x1F	129
DYNAMICXSR	R/W	0x050	Time for exit self-refresh to active command.	-	0x1F	130
DYNAMICRRD	R/W	0x054	Latency for active bank A to active bank B.	-	0xF	131
DYNAMICMRD	R/W	0x058	Time for load mode register to active command.	-	0xF	132
STATICEXTENDEDWAIT	R/W	0x080	Time for long static memory read and write transfers.	-	0x0	133
DYNAMICCONFIG0	R/W	0x100	Configuration information for $\overline{\text{EMC\_DYCS0}}$.	-	0x0	134
DYNAMICRASCAS0	R/W	0x104	RAS and CAS latencies for $\overline{\text{EMC\_DYCS0}}$.	-	0x303	136
DYNAMICCONFIG1	R/W	0x120	Configuration information for $\overline{\text{EMC\_DYCS1}}$.	-	0x0	134
DYNAMICRASCAS1	R/W	0x124	RAS and CAS latencies for $\overline{\text{EMC\_DYCS1}}$.	-	0x303	136
DYNAMICCONFIG2	R/W	0x140	Configuration information for $\overline{\text{EMC\_DYCS2}}$.	-	0x0	134
DYNAMICRASCAS2	R/W	0x144	RAS and CAS latencies for $\overline{\text{EMC\_DYCS2}}$.	-	0x303	136
DYNAMICCONFIG3	R/W	0x160	Configuration information for $\overline{\text{EMC\_DYCS3}}$.	-	0x0	134
DYNAMICRASCAS3	R/W	0x164	RAS and CAS latencies for $\overline{\text{EMC\_DYCS3}}$.	-	0x303	136
STATICCONFIG0	R/W	0x200	Configuration for $\overline{\text{EMC\_CS0}}$.	-	0x0	137
STATICWAITWEN0	R/W	0x204	Delay from $\overline{\text{EMC\_CS0}}$ to write enable.	-	0x0	138
STATICWAITOEN0	R/W	0x208	Delay from $\overline{\text{EMC\_CS0}}$ or address change, whichever is later, to output enable.	-	0x0	139
STATICWAITRD0	R/W	0x20C	Delay from $\overline{\text{EMC\_CS0}}$ to a read access.	-	0x1F	140

Table 115. Register overview: EMC (base address 0x2009 C000) *…continued*

Register Name	Access	Address offset	Description	Warm Reset Value[1]	POR Reset Value[1]	Table
STATICWAITPAGE0	R/W	0x210	Delay for asynchronous page mode sequential accesses for EMC_CS0.	-	0x1F	141
STATICWAITWR0	R/W	0x214	Delay from EMC_CS0 to a write access.	-	0x1F	142
STATICWAITTURN0	R/W	0x218	Number of bus turnaround cycles EMC_CS0.	-	0xF	143
STATICCONFIG1	R/W	0x220	Memory configuration for EMC_CS1.	-	0x0	137
STATICWAITWEN1	R/W	0x224	Delay from EMC_CS1 to write enable.	-	0x0	138
STATICWAITOEN1	R/W	0x228	Delay from EMC_CS1 or address change, whichever is later, to output enable.	-	0x0	139
STATICWAITRD1	R/W	0x22C	Delay from EMC_CS1 to a read access.	-	0x1F	140
STATICWAITPAGE1	R/W	0x230	Delay for asynchronous page mode sequential accesses for EMC_CS1.	-	0x1F	141
STATICWAITWR1	R/W	0x234	Delay from EMC_CS1 to a write access.	-	0x1F	142
STATICWAITTURN1	R/W	0x238	Bus turnaround cycles for EMC_CS1.	-	0xF	143
STATICCONFIG2	R/W	0x240	Memory configuration for EMC_CS2.	-	0x0	137
STATICWAITWEN2	R/W	0x244	Delay from EMC_CS2 to write enable.	-	0x0	138
STATICWAITOEN2	R/W	0x248	Delay from EMC_CS2 or address change, whichever is later, to output enable.	-	0x0	139
STATICWAITRD2	R/W	0x24C	Delay from EMC_CS2 to a read access.	-	0x1F	140
STATICWAITPAGE2	R/W	0x250	Delay for asynchronous page mode sequential accesses for EMC_CS2.	-	0x1F	141
STATICWAITWR2	R/W	0x254	Delay from EMC_CS2 to a write access.	-	0x1F	142
EMCStaticWaitTurn2	R/W	0x258	Bus turnaround cycles for EMC_CS2.	-	0xF	143
STATICCONFIG3	R/W	0x260	Memory configuration for EMC_CS3.	-	0x0	137
STATICWAITWEN3	R/W	0x264	Delay from EMC_CS3 to write enable.	-	0x0	138
STATICWAITOEN3	R/W	0x268	Delay from EMC_CS3 or address change, whichever is later, to output enable.	-	0x0	139
STATICWAITRD3	R/W	0x26C	Delay from EMC_CS3 to a read access.	-	0x1F	140
STATICWAITPAGE3	R/W	0x270	Delay for asynchronous page mode sequential accesses for EMC_CS3.	-	0x1F	141
STATICWAITWR3	R/W	0x274	Delay from EMC_CS3 to a write access.	-	0x1F	142
STATICWAITTURN3	R/W	0x278	Bus turnaround cycles for EMC_CS3.	-	0xF	143

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

9.13.1 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 116 shows the bit assignments for the EMCControl register.

Table 116. EMC Control register (Control - address 0x2009 C000) 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.	NA

[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.

9.13.2 EMC Status register

The read-only EMCStatus register provides EMC status information.

Table 117. EMC Status register (STATUS - address 0x2009 C008) 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

9.13.3 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 118. EMC Configuration register (CONFIG - address 0x2009 C008) 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

9.13.4 Dynamic Memory Control register

The EMCDynamicControl register controls dynamic memory operation. The control bits can be altered during normal operation.

Table 119. Dynamic Control register (DYNAMICCONTROL - address 0x2009 C020) 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.

9.13.5 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 120. Dynamic Memory Refresh Timer register (DYNAMICREFRESH - address 0x2009 C024) 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.

9.13.6 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 121 shows the bit assignments for the EMCDynamicReadConfig register.

Table 121. Dynamic Memory Read Configuration register (DYNAMICREADCONFIG - address 0x2009 C028) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	RD		Read data strategy	0x0
		0x0	Clock out delayed strategy, using EMC_CLKOUTx (command not delayed, clock out delayed). POR reset value.	
		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 9.5.6 and Section 3.3.6.1.

9.13.7 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 122. Dynamic Memory Precharge Command Period register (DYNAMICRP - address 0x2009 C030) 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.	-

9.13.8 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 123. Dynamic Memory Active to Precharge Command Period register (DYNAMICRAS - address 0x2009 C034) 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.	-

9.13.9 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 124. Dynamic Memory Self Refresh Exit Time register (DYNAMICSREX - address 0x2009 C038) 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.	-

9.13.10 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 125. Dynamic Memory Last Data Out to Active Time register (DYNAMICAPR - address 0x2009 C03C) 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.	-

9.13.11 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 126. Dynamic Memory Data In to Active Command Time register (DYNAMICDAL - address 0x2009 C040) 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.	-

9.13.12 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 127. Dynamic Memory Write Recovery Time register (DYNAMICWR - address 0x2009 C044) 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.	-

9.13.13 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 128. Dynamic Memory Active to Active Command Period register (DYNAMICRC - address 0x2009 C048) 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.	-

9.13.14 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 129. Dynamic Memory Auto Refresh Period register (DYNAMICRFC - address 0x2009 C04C) 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.	-

9.13.15 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 130. Dynamic Memory Exit Self Refresh register (DYNAMICXSR - address 0x2009 C050) 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.	-

9.13.16 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 131. Dynamic Memory Active Bank A to Active Bank B Time register (DYNAMICRRD - address 0x2009 C054) 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.	-

9.13.17 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 132. Dynamic Memory Load Mode register to Active Command Time (DYNAMICMRD - address 0x2009 C058) 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.	-

9.13.18 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 133. Static Memory Extended Wait register (STATICEXTENDEDWAIT - address 0x2009 C080) 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.	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 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 - 1 = 49

9.13.19 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 134 shows the bit assignments for the EMCDynamicConfig0-3 registers.

Table 134. Dynamic Memory Configuration registers (DYNAMICCONFIG[0:3], address 0x2009 C100 (DYNAMICCONFIG0), 0x2009 C120 (DYNAMICCONFIG1), 0x2009 C140 (DYNAMICCONFIG2), 0x2009 C160 (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 135. 000000 = reset value.[1]	0
13	-		Reserved. Read value is undefined, only zero should be written.	NA
14	AM1		See Table 135. 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 135 are reserved.

Table 135. Address mapping

14	12	11:9	8:7	Description	Banks	Row length	Column length
16 bit bus width (Row, Bank, Column)							
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)							
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 bus width (Row, Bank, Column)							
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
32 bit bus width (Bank, Row, Column)							
1	1	000	00	16 Mbits (2M x 8)	2	11	9
1	1	000	01	16 Mbits (1M x 16)	2	11	8

Table 135. Address mapping

14	12	11:9	8:7	Description	Banks	Row length	Column length
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.

9.13.20 Dynamic Memory RAS & 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 136. Dynamic Memory RASCAS Delay registers (DYNAMICRASCAS[0:3], address 0x2009 C104 (DYNAMICRASCAS0), 0x2009 C124 (DYNAMICRASCAS1), 0x2009 C144 (DYNAMICRASCAS2), 0x2009 C164 (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.	-

9.13.21 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 137 shows the bit assignments for the EMCStaticConfig0-3 registers. Note that synchronous burst mode memory devices are not supported.

Table 137. Static Memory Configuration registers (STATICCONFIG[0:3], address 0x2009 C200 (STATICCONFIG0), 0x2009 C220 (STATICCONFIG1), 0x2009 C240 (STATICCONFIG2), 0x2009 C260 (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 $\overline{BLS3:0}$ signal from the EMC is usually connected to $\overline{WE}$ (write enable). In this case for reads all the $\overline{BLS3:0}$ 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 $\overline{BLS3:0}$ 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 $\overline{BLS3:0}$ are HIGH. For writes the respective active bits in $\overline{BLS3:0}$ are LOW (POR reset value).	
		1	For reads the respective active bits in $\overline{BLS3:0}$ are LOW. For writes the respective active bits in $\overline{BLS3:0}$ are LOW.	

Table 137. Static Memory Configuration registers (STATICCONFIG[0:3], address 0x2009 C200 (STATICCONFIG0), 0x2009 C220 (STATICCONFIG1), 0x2009 C240 (STATICCONFIG2), 0x2009 C260 (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.

9.13.22 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 138. Static Memory Write Enable Delay registers (STATICWAITWEN[0:3], address 0x2009 C204 (STATICWAITWEN0), 0x2009 C224 (STATICWAITWEN1),0x2009 C244 (STATICWAITWEN2), 0x2009 C264 (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.	-

9.13.23 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 139. Static Memory Output Enable delay registers (STATICWAITOEN[0:3], address 0x2009 C208 (STATICWAITOEN0), 0x0x2009 C228 (STATICWAITOEN1), 0x0x2009 C248 (STATICWAITOEN2), 0x0x2009 C268 (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.	-

9.13.24 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 140. Static Memory Read Delay registers (STATICWAITRD[0:3], address 0x2009 C20C (STATICWAITRD0), 0x2009 C22C (STATICWAITRD1), 0x2009 C24C (STATICWAITRD2), 0x2009 C26C (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.

9.13.25 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 141. Static Memory Page Mode Read Delay registers (STATICWAITPAGE[0:3], address 0x2009 C210 (STATICWAITPAGE0), 2009 C230 (STATICWAITPAGE1), 0x2009 C250 (STATICWAITPAGE2), 0x2009 C270 (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.	-

9.13.26 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 142. Static Memory Write Delay registers (STATICWAITWR[0:3], address 0x2009 C214 (STATICWAITWR0), 0x2009 C234 (STATICWAITWR1), 0x2009 C254 (STATICWAITWR2), 0x2009 C274 (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.	-

9.13.27 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 143. Static Memory Turn-around Delay registers (STATICWAITTURN[0:3], address 0x2009 C218 (STATICWAITTURN0),0x2009 C238 (STATICWAITTURN1), 0x2009 C258 (STATICWAITTURN2), 0x2009 C278 (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.

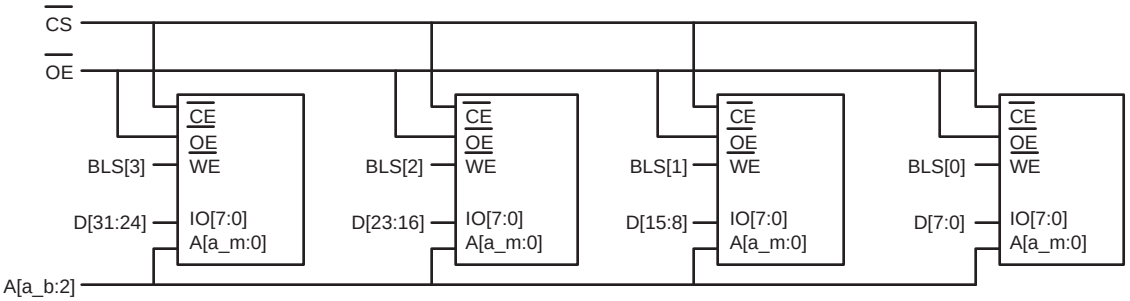
9.14 External static memory interface

External memory interfacing depends on the bank width (32, 16 or 8 bit selected via MW bits in corresponding EMCStaticConfig register).

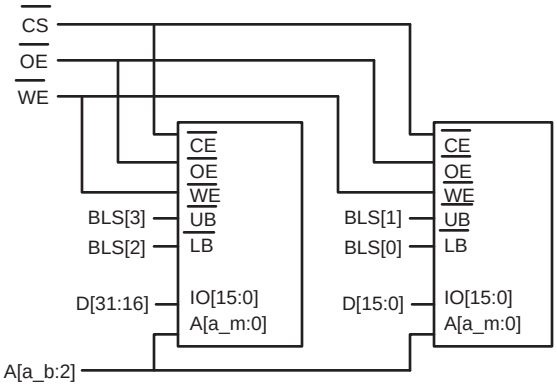
If a memory bank is configured to be 32 bits wide, address lines A0 and A1 can be used as non-address lines. If a memory bank is configured to 16 bits wide, A0 is not required. However, 8 bit wide memory banks do require all address lines down to A0. Configuring the A1 and/or A0 lines to provide address or non-address function is accomplished using the IOCON registers (see Section 7.4.1).

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.

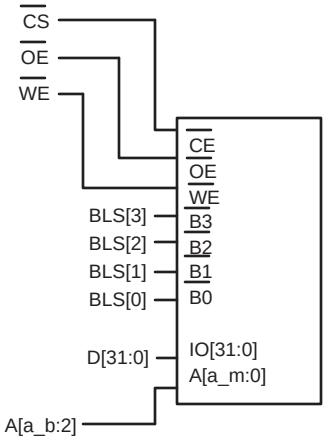
9.14.1 32-bit wide memory bank connection



a. 32 bit wide memory bank interfaced to four 8 bit memory chips



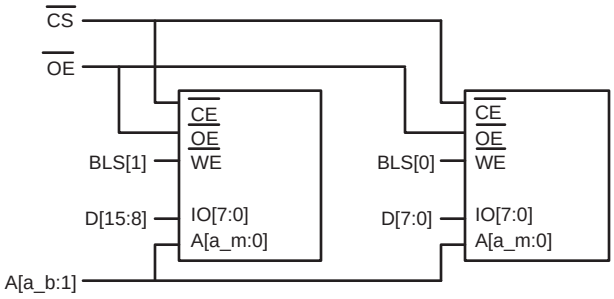
b. 32 bit wide memory bank interfaced to two 16 bit memory chips



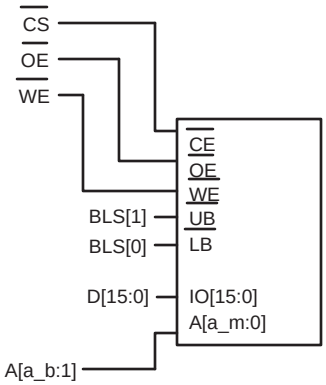
c. 32 bit wide memory bank interfaced to one 8 bit memory chip

Fig 18. 32 bit bank external memory interfaces (bits MW = 10)

9.14.2 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 19. 16 bit bank external memory interfaces (bits MW = 01)

9.14.3 8-bit wide memory bank connection

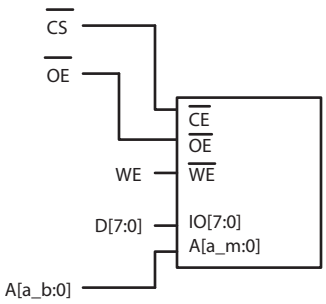


Fig 20. 8 bit bank external memory interface (bits MW = 00)

9.14.4 Memory configuration example

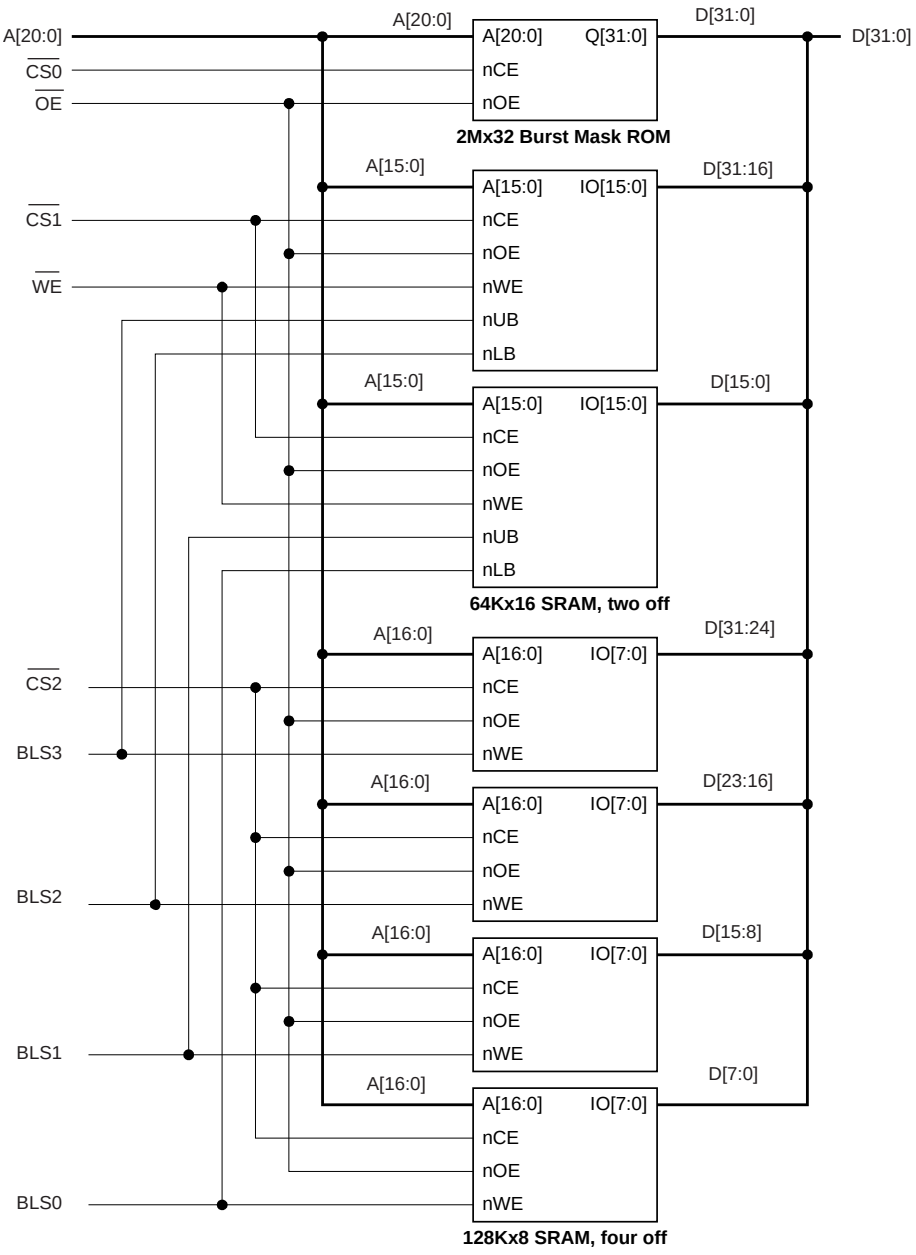


Fig 21. Typical memory configuration diagram

10.1 Basic configuration

The Ethernet controller is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCENET.

   Remark: On reset, the Ethernet block is disabled (PCENET = 0).
2. Clock: See Section 3.3.3.2.
3. Pins: Enable Ethernet pins and select their modes through the IOCON registers, see Section 7.4.1.
4. Wake-up: Activity on the Ethernet port can wake up the microcontroller from Power-down mode, see Section 3.12.8.
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. Initialization: See Section 10.13.2.

10.2 Introduction

The Ethernet block contains a full featured 10 Mbps or 100 Mbps Ethernet MAC (Media Access Controller) designed to provide optimized performance through the use of DMA hardware acceleration. Features include a generous suite of control registers, half or full duplex operation, flow control, control frames, hardware acceleration for transmit retry, receive packet filtering and wake-up on LAN activity. Automatic frame transmission and reception with Scatter-Gather DMA off-loads many operations from the CPU.

The Ethernet block is an AHB master that drives the AHB bus matrix. Through the matrix, it has access to all on-chip RAM memories. A recommended use of RAM by the Ethernet is to use one of the RAM blocks exclusively for Ethernet traffic. That RAM would then be accessed only by the Ethernet and the CPU, and possibly the GPDMA, giving maximum bandwidth to the Ethernet function.

The Ethernet block interfaces between an off-chip Ethernet PHY using the MII (Media Independent Interface) or RMII (reduced MII) protocol and the on-chip MIIM (Media Independent Interface Management) serial bus, also referred to as MDIO (Management Data Input/Output).

Table 144. Ethernet acronyms, abbreviations, and definitions

Acronym or Abbreviation	Definition
AHB	Advanced High-performance bus
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
Double-word	64-bit entity
FCS	Frame Check Sequence (CRC)
Fragment	A (part of an) Ethernet frame; one or multiple fragments can add up to a single Ethernet frame.

Table 144. Ethernet acronyms, abbreviations, and definitions

Acronym or Abbreviation	Definition
Frame	An Ethernet frame consists of destination address, source address, length type field, payload and frame check sequence.
Half-word	16-bit entity
LAN	Local Area Network
MAC	Media Access Control sublayer
MII	Media Independent Interface
MIIM	MII management
Octet	An 8-bit data entity, used in lieu of "byte" by IEEE 802.3
Packet	A frame that is transported across Ethernet; a packet consists of a preamble, a start of frame delimiter and an Ethernet frame.
PHY	Ethernet Physical Layer
RMII	Reduced MII
Rx	Receive
TCP/IP	Transmission Control Protocol / Internet Protocol. The most common high-level protocol used with Ethernet.
Tx	Transmit
VLAN	Virtual LAN
WoL	Wake-up on LAN
Word	32-bit entity

10.3 Features

- Ethernet standards support:
  - Supports 10 or 100 Mbps PHY devices including 10 Base-T, 100 Base-TX, 100 Base-FX, and 100 Base-T4.
  - Fully compliant with IEEE standard 802.3.
  - Fully compliant with 802.3x Full Duplex Flow Control and Half Duplex back pressure.
  - Flexible transmit and receive frame options.
  - VLAN frame support.
- Memory management:
  - Independent transmit and receive buffers memory mapped to shared SRAM.
  - DMA managers with scatter/gather DMA and arrays of frame descriptors.
  - Memory traffic optimized by buffering and prefetching.
- Enhanced Ethernet features:
  - Receive filtering.
  - Multicast and broadcast frame support for both transmit and receive.
  - Optional automatic FCS insertion (CRC) for transmit.
  - Selectable automatic transmit frame padding.

- Over-length frame support for both transmit and receive allows any length frames.
- Promiscuous receive mode.
- Automatic collision backoff and frame retransmission.
- Includes power management by clock switching.
- Wake-on-LAN power management support allows system wake-up: using the receive filters or a magic frame detection filter.

- Physical interface:
  - Attachment of external PHY chip through standard Media Independent Interface (MII) or standard Reduced MII (RMII) interface, software selectable.
  - PHY register access is available via the Media Independent Interface Management (MIIM) interface.

10.4 Architecture and operation

Figure 22 shows the internal architecture of the Ethernet block.

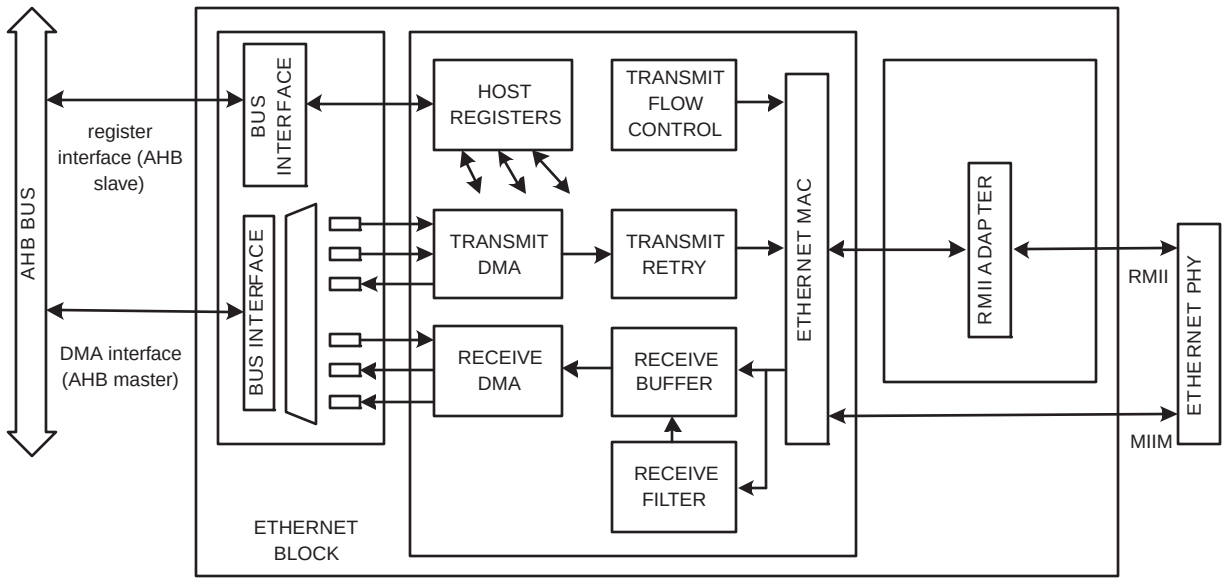


Fig 22. Ethernet block diagram

The block diagram for the Ethernet block consists of:

- The host registers module containing the registers in the software view and handling AHB accesses to the Ethernet block. The host registers connect to the transmit and receive data path as well as the MAC.
- The DMA to AHB interface. This provides an AHB master connection that allows the Ethernet block to access on-chip SRAM for reading of descriptors, writing of status, and reading and writing data buffers.
- The Ethernet MAC, which interfaces to the off-chip PHY via an MII or RMII interface.
- The transmit data path, including:

  - The transmit DMA manager which reads descriptors and data from memory and writes status to memory.
  - The transmit retry module handling Ethernet retry and abort situations.
  - The transmit flow control module which can insert Ethernet pause frames.
- The receive data path, including:
  - The receive DMA manager which reads descriptors from memory and writes data and status to memory.
  - The Ethernet MAC which detects frame types by parsing part of the frame header.
  - The receive filter which can filter out certain Ethernet frames by applying different filtering schemes.
  - The receive buffer implementing a delay for receive frames to allow the filter to filter out certain frames before storing them to memory.

10.5 DMA engine functions

The Ethernet block is designed to provide optimized performance via DMA hardware acceleration. Independent scatter/gather DMA engines connected to the AHB bus off-load many data transfers from the CPU.

Descriptors, which are stored in memory, contain information about fragments of incoming or outgoing Ethernet frames. A fragment may be an entire frame or a much smaller amount of data. Each descriptor contains a pointer to a memory buffer that holds data associated with a fragment, the size of the fragment buffer, and details of how the fragment will be transmitted or received.

Descriptors are stored in arrays in memory, which are located by pointer registers in the Ethernet block. Other registers determine the size of the arrays, point to the next descriptor in each array that will be used by the DMA engine, and point to the next descriptor in each array that will be used by the Ethernet device driver.

10.6 Overview of DMA operation

The DMA engine makes use of a Receive descriptor array and a Transmit descriptor array in memory. All or part of an Ethernet frame may be contained in a memory buffer associated with a descriptor. When transmitting, the transmit DMA engine uses as many descriptors as needed (one or more) to obtain (gather) all of the parts of a frame, and sends them out in sequence. When receiving, the receive DMA engine also uses as many descriptors as needed (one or more) to find places to store (scatter) all of the data in the received frame.

The base address registers for the descriptor array, registers indicating the number of descriptor array entries, and descriptor array input/output pointers are contained in the Ethernet block. The descriptor entries and all transmit and receive packet data are stored in memory which is not a part of the Ethernet block. The descriptor entries tell where related frame data is stored in memory, certain aspects of how the data is handled, and the result status of each Ethernet transaction.

Hardware in the DMA engine controls how data incoming from the Ethernet MAC is saved to memory, causes fragment related status to be saved, and advances the hardware receive pointer for incoming data. Driver software must handle the disposition of received data, changing of descriptor data addresses (to avoid unnecessary data movement), and advancing the software receive pointer. The two pointers create a circular queue in the descriptor array and allow both the DMA hardware and the driver software to know which descriptors (if any) are available for their use, including whether the descriptor array is empty or full.

Similarly, driver software must set up pointers to data that will be transmitted by the Ethernet MAC, giving instructions for each fragment of data, and advancing the software transmit pointer for outgoing data. Hardware in the DMA engine reads this information and sends the data to the Ethernet MAC interface when possible, updating the status and advancing the hardware transmit pointer.

10.7 Ethernet packet

Figure 23 illustrates the different fields in an Ethernet packet.

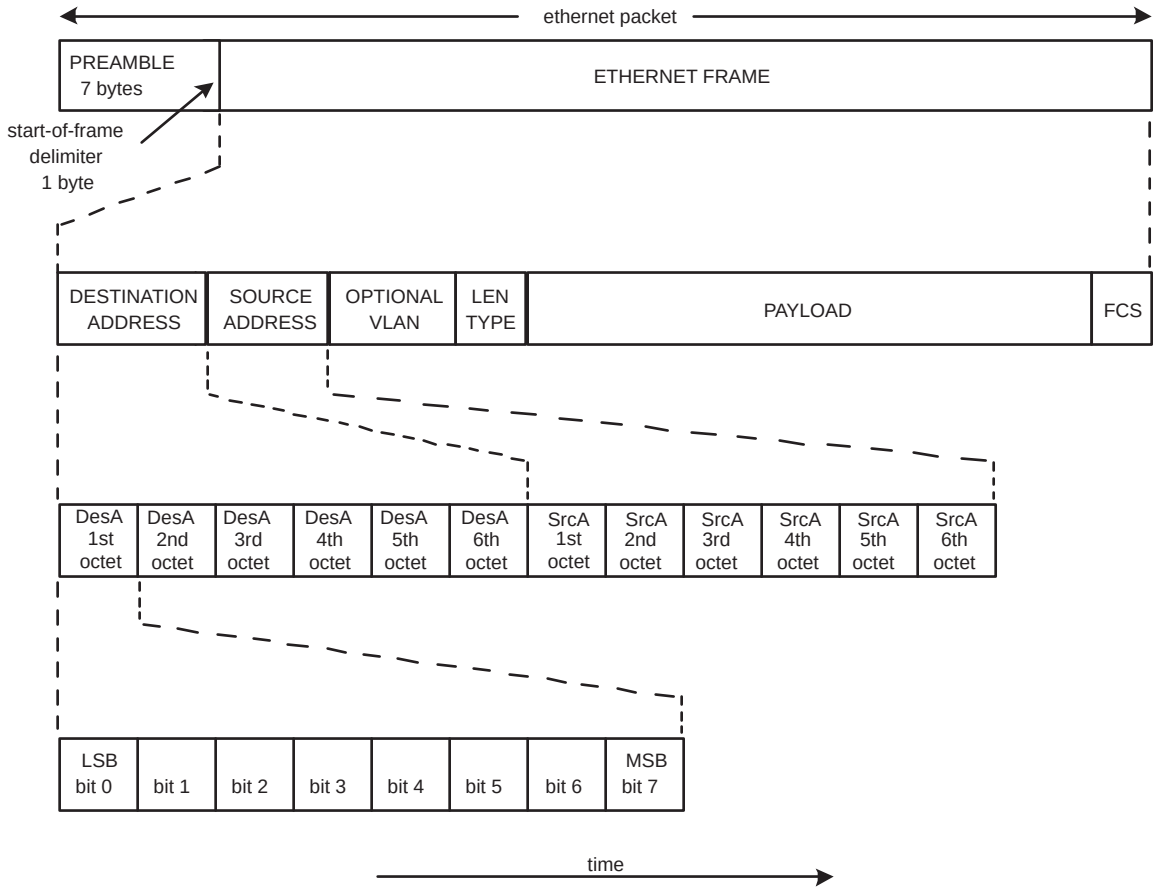


Fig 23. Ethernet packet fields

A packet consists of a preamble, a start-of-frame delimiter and an Ethernet frame.

The Ethernet frame consists of the destination address, the source address, an optional VLAN field, the length/type field, the payload and the frame check sequence.

Each address consists of 6 bytes where each byte consists of 8 bits. Bits are transferred starting with the least significant bit.

10.8 Overview

10.8.1 Partitioning

The Ethernet block and associated device driver software offer the functionality of the Media Access Control (MAC) sublayer of the data link layer in the OSI reference model (see IEEE std 802.3). The MAC sublayer offers the service of transmitting and receiving frames to the next higher protocol level, the MAC client layer, typically the Logical Link Control sublayer. The device driver software implements the interface to the MAC client layer. It sets up registers in the Ethernet block, maintains descriptor arrays pointing to frames in memory and receives results back from the Ethernet block through interrupts. When a frame is transmitted, the software partially sets up the Ethernet frames by providing pointers to the destination address field, source address field, the length/type field, the MAC client data field and optionally the CRC in the frame check sequence field. Preferably concatenation of frame fields should be done by using the scatter/gather functionality of the Ethernet core to avoid unnecessary copying of data. The hardware adds the preamble and start frame delimiter fields and can optionally add the CRC, if requested by software. When a packet is received the hardware strips the preamble and start frame delimiter and passes the rest of the packet - the Ethernet frame - to the device driver, including destination address, source address, length/type field, MAC client data and frame check sequence (FCS).

Apart from the MAC, the Ethernet block contains receive and transmit DMA managers that control receive and transmit data streams between the MAC and the AHB interface. Frames are passed via descriptor arrays located in host memory, so that the hardware can process many frames without software/CPU support. Frames can consist of multiple fragments that are accessed with scatter/gather DMA. The DMA managers optimize memory bandwidth using prefetching and buffering.

A receive filter block is used to identify received frames that are not addressed to this Ethernet station, so that they can be discarded. The Rx filters include a perfect address filter and a hash filter.

Wake-on-LAN power management support makes it possible to wake the system up from a power-down state -a state in which some of the clocks are switched off -when wake-up frames are received over the LAN. Wake-up frames are recognized by the receive filtering modules or by a Magic Frame detection technology. System wake-up occurs by triggering an interrupt.

An interrupt logic block raises and masks interrupts and keeps track of the cause of interrupts. The interrupt block sends an interrupt request signal to the host system. Interrupts can be enabled, cleared and set by software.

Support for IEEE 802.3/clause 31 flow control is implemented in the flow control block. Receive flow control frames are automatically handled by the MAC. Transmit flow control frames can be initiated by software. In half duplex mode, the flow control module will generate back pressure by sending out continuous preamble only, interrupted by pauses to prevent the jabber limit from being exceeded.

The Ethernet block has both a standard Media Independent Interface (MII) bus and a Reduced Media Independent Interface (RMII) to connect to an external Ethernet PHY chip. MII or RMII mode can be selected by the RMII bit in the Command register. The standard nibble-wide MII interface allows a low speed data connection to the PHY chip: 2.5 MHz at 10 Mbps or 25 MHz at 100 Mbps. The RMII interface allows a low pin count double clock data connection to the PHY. Registers in the PHY chip are accessed via the AHB interface through the serial management connection of the MIIM bus, typically operating at 2.5 MHz.

10.8.2 Example PHY Devices

Some examples of compatible PHY devices are shown in Table 145.

Table 145. Example PHY Devices

Manufacturer	Part Numbers
Broadcom	BCM5221
ICS	ICS1893
Intel	LXT971A
LSI Logic	L80223, L80225, L80227
Micrel	KS8721
National	DP83847, DP83846, DP83843
SMSC	LAN83C185

10.9 Pin description

Table 146 shows the signals used for connecting the Media Independent Interface (MII), and Table 147 shows the signals used for connecting the Reduced Media Independent Interface (RMII) to the external PHY.

Table 146. Ethernet MII pin descriptions

Pin Name	Type	Pin Description
ENET_TX_EN	Output	Transmit data enable, active low.
ENET_TXD3:0	Output	Transmit data, 4 bits.
ENET_TX_ER	Output	Transmit error.
ENET_TX_CLK	Input	Transmitter clock.
ENET_RX_DV	Input	Receive data valid.
ENET_RXD3:0	Input	Receive data, 4 bits.
ENET_RX_ER	Input	Receive error.
ENET_RX_CLK	Input	Receive clock.
ENET_COL	Input	Collision detect.
ENET_CRS	Input	Carrier sense.

Table 147. Ethernet RMII pin descriptions

Pin Name	Type	Pin Description
ENET_TX_EN	Output	Transmit data enable, active low.
ENET_TXD1:0	Output	Transmit data, 2 bits
ENET_RXD1:0	Input	Receive data, 2 bits.
ENET_RX_ER	Input	Receive error.
ENET_CRS	Input	ENET_CRS_DV. Carrier sense/data valid.
ENET_RX_CLK	Input	ENET_REF_CLK. Reference clock.

Table 148 shows the signals used for Media Independent Interface Management (MIIM) to the external PHY.

Table 148. Ethernet MIIM pin descriptions

Pin Name	Type	Pin Description
ENET_MDC	Output	MIIM clock.
ENET_MDIO	Input/Output	MI data input and output

10.10 Register description

The software interface of the Ethernet block consists of a register view and the format definitions for the transmit and receive descriptors. These two aspects are addressed in the next two subsections.

The total AHB address space required for the ethernet is 4 kilobytes.

After a hard reset or a soft reset via the RegReset bit of the Command register all bits in all registers are reset to 0 unless stated otherwise in the following register descriptions.

Some registers will have unused bits which will return a 0 on a read via the AHB interface. Writing to unused register bits of an otherwise writable register will not have side effects.

The register map consists of registers in the Ethernet MAC and registers around the core for controlling DMA transfers, flow control and filtering.

Reading from reserved addresses or reserved bits leads to unpredictable data. Writing to reserved addresses or reserved bits has no effect.

Reading of write-only registers will return a read error on the AHB interface. Writing of read-only registers will return a write error on the AHB interface.

Table 149. Register overview: Ethernet (base address 0x2008 4000)

Name	Access	Address offset	Description	Reset Value	Table
MAC registers					
MAC1	R/W	0x000	MAC configuration register 1.	0x8000	150
MAC2	R/W	0x004	MAC configuration register 2.	0	151
IPGT	R/W	0x008	Back-to-Back Inter-Packet-Gap register.	0	153
IPGR	R/W	0x00C	Non Back-to-Back Inter-Packet-Gap register.	0	154
CLRT	R/W	0x010	Collision window / Retry register.	0x370F	155
MAXF	R/W	0x014	Maximum Frame register.	0x0600	156
SUPP	R/W	0x018	PHY Support register.	0	157
TEST	R/W	0x01C	Test register.	0	158
MCFG	R/W	0x020	MII Mgmt Configuration register.	0	159
MCMD	R/W	0x024	MII Mgmt Command register.	0	161
MADR	R/W	0x028	MII Mgmt Address register.	0	162
MWTD	WO	0x02C	MII Mgmt Write Data register.	-	163
MRDD	RO	0x030	MII Mgmt Read Data register.	0	164
MIND	RO	0x034	MII Mgmt Indicators register.	0	165
SA0	R/W	0x040	Station Address 0 register.	0	166
SA1	R/W	0x044	Station Address 1 register.	0	167
SA2	R/W	0x048	Station Address 2 register.	0	168
Control registers					
COMMAND	R/W	0x100	Command register.	0	169
STATUS	RO	0x104	Status register.	0	170
RXDESCRIPTOR	R/W	0x108	Receive descriptor base address register.	0	171

Table 149. Register overview: Ethernet (base address 0x2008 4000)

Name	Access	Address offset	Description	Reset Value	Table
RXSTATUS	R/W	0x10C	Receive status base address register.	0	172
RXDESCRIPTORNUMBER	R/W	0x110	Receive number of descriptors register.	0	173
RXPRODUCEINDEX	RO	0x114	Receive produce index register.	0	174
RXCONSUMEINDEX	R/W	0x118	Receive consume index register.	0	175
TXDESCRIPTOR	R/W	0x11C	Transmit descriptor base address register.	0	176
TXSTATUS	R/W	0x120	Transmit status base address register.	0	177
TXDESCRIPTORNUMBER	R/W	0x124	Transmit number of descriptors register.	0	178
TXPRODUCEINDEX	R/W	0x128	Transmit produce index register.	0	179
TXCONSUMEINDEX	RO	0x12C	Transmit consume index register.	0	180
TSV0	RO	0x158	Transmit status vector 0 register.	0	181
TSV1	RO	0x15C	Transmit status vector 1 register.	0	182
RSV	RO	0x160	Receive status vector register.	0	183
FLOWCONTROLCOUNTER	R/W	0x170	Flow control counter register.	0	184
FLOWCONTROLSTATUS	RO	0x174	Flow control status register.	0	185
Rx filter registers					
RXFILTERCTRL	R/W	0x200	Receive filter control register.	0	186
RXFILTERWOLSTATUS	RO	0x204	Receive filter WoL status register.	0	187
RXFILTERWOLCLEAR	WO	0x208	Receive filter WoL clear register.	-	188
HASHFILTERL	R/W	0x210	Hash filter table LSBs register.	0	189
HASHFILTERH	R/W	0x214	Hash filter table MSBs register.	0	190
Module control registers					
INTSTATUS	RO	0xFE0	Interrupt status register.	0	191
INTENABLE	R/W	0xFE4	Interrupt enable register.	0	192
INTCLEAR	WO	0xFE8	Interrupt clear register.	-	193
INTSET	WO	0xFEC	Interrupt set register.	-	194
POWERDOWN	R/W	0xFF4	Power-down register.	0	195

The third column in the table lists the accessibility of the register: read-only, write-only, read/write.

All AHB register write transactions except for accesses to the interrupt registers are posted i.e. the AHB transaction will complete before write data is actually committed to the register. Accesses to the interrupt registers will only be completed by accepting the write data when the data has been committed to the register.

10.10.1 Ethernet MAC register definitions

This section defines the bits in the individual registers of the Ethernet block register map.

10.10.1.1 MAC Configuration Register 1

The MAC configuration register 1 (MAC1) has an address of 0x2008 4000. Its bit definition is shown in Table 150.

Table 150. MAC Configuration register 1 (MAC1 - address 0x2008 4000) bit description

Bit	Symbol	Function	Reset value
0	RXENABLE	RECEIVE ENABLE. Set this to allow receive frames to be received. Internally the MAC synchronizes this control bit to the incoming receive stream.	0
1	PARF	PASS ALL RECEIVE FRAMES. When enabled (set to 1), the MAC will pass all frames regardless of type (normal vs. Control). When disabled, the MAC does not pass valid Control frames.	0
2	RXFLOWCTRL	RX FLOW CONTROL. When enabled (set to 1), the MAC acts upon received PAUSE Flow Control frames. When disabled, received PAUSE Flow Control frames are ignored.	0
3	TXFLOWCTRL	TX FLOW CONTROL. When enabled (set to 1), PAUSE Flow Control frames are allowed to be transmitted. When disabled, Flow Control frames are blocked.	0
4	LOOPBACK	Setting this bit will cause the MAC Transmit interface to be looped back to the MAC Receive interface. Clearing this bit results in normal operation.	0
7:5	-	Unused	0
8	RESETTX	Setting this bit will put the Transmit Function logic in reset.	0
9	RESETMCSTX	Setting this bit resets the MAC Control Sublayer / Transmit logic. The MCS logic implements flow control.	0
10	RESETRX	Setting this bit will put the Ethernet receive logic in reset.	0
11	RESETMCSRX	Setting this bit resets the MAC Control Sublayer / Receive logic. The MCS logic implements flow control.	0
13:12	-	Reserved. Read value is undefined, only zero should be written.	0
14	SIMRESET	SIMULATION RESET. Setting this bit will cause a reset to the random number generator within the Transmit Function.	0
15	SOFTRESET	SOFT RESET. Setting this bit will put all modules within the MAC in reset except the Host Interface.	1
31:16	-	Reserved. Read value is undefined, only zero should be written.	0

10.10.1.2 MAC Configuration Register 2

Table 151. MAC Configuration register 2 (MAC2 - address 0x2008 4004) bit description

Bit	Symbol	Function	Reset value
0	FULLDUPLEX	When enabled (set to 1), the MAC operates in Full-Duplex mode. When disabled, the MAC operates in Half-Duplex mode.	0
1	FLC	FRAMELENGTH CHECKING. When enabled (set to 1), both transmit and receive frame lengths are compared to the Length/Type field. If the Length/Type field represents a length then the check is performed. Mismatches are reported in the StatusInfo word for each received frame.	0
2	HFEN	HUGE FRAME ENABLE. When enabled (set to 1), frames of any length are transmitted and received.	0
3	DELAYEDCRC	DELAYED CRC. This bit determines the number of bytes, if any, of proprietary header information that exist on the front of IEEE 802.3 frames. When 1, four bytes of header (ignored by the CRC function) are added. When 0, there is no proprietary header.	0
4	CRCEN	CRC ENABLE. Set this bit to append a CRC to every frame whether padding was required or not. Must be set if PAD/CRC ENABLE is set. Clear this bit if frames presented to the MAC contain a CRC.	0
5	PADCRCEN	PAD CRC ENABLE. Set this bit to have the MAC pad all short frames. Clear this bit if frames presented to the MAC have a valid length. This bit is used in conjunction with AUTO PAD ENABLE and VLAN PAD ENABLE. See Table 153 - Pad Operation for details on the pad function.	0
6	VLANPADEN	VLAN PAD ENABLE. Set this bit to cause the MAC to pad all short frames to 64 bytes and append a valid CRC. Consult Table 153 - Pad Operation for more information on the various padding features. **Note:** This bit is ignored if PAD / CRC ENABLE is cleared.	0
7	AUTODETPADEN	AUTODETECTPAD ENABLE. Set this bit to cause the MAC to automatically detect the type of frame, either tagged or un-tagged, by comparing the two octets following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly. Table 153 - Pad Operation provides a description of the pad function based on the configuration of this register. **Note:** This bit is ignored if PAD / CRC ENABLE is cleared.	0
8	PPENF	PURE PREAMBLE ENFORCEMENT. When enabled (set to 1), the MAC will verify the content of the preamble to ensure it contains 0x55 and is error-free. A packet with an incorrect preamble is discarded. When disabled, no preamble checking is performed.	0
9	LPENF	LONG PREAMBLE ENFORCEMENT. When enabled (set to 1), the MAC only allows receive packets which contain preamble fields less than 12 bytes in length. When disabled, the MAC allows any length preamble as per the Standard.	0
11:10	-	Reserved. Read value is undefined, only zero should be written.	0
12	NOBACKOFF	When enabled (set to 1), the MAC will immediately retransmit following a collision rather than using the Binary Exponential Backoff algorithm as specified in the Standard.	0
13	BP_NOBACKOFF	BACK PRESSURE / NO BACKOFF. When enabled (set to 1), after the MAC incidentally causes a collision during back pressure, it will immediately retransmit without backoff, reducing the chance of further collisions and ensuring transmit packets get sent.	0
14	EXCESSDEFER	When enabled (set to 1) the MAC will defer to carrier indefinitely as per the Standard. When disabled, the MAC will abort when the excessive deferral limit is reached.	0
31:15	-	Reserved. Read value is undefined, only zero should be written.	0

Table 152. Pad operation

Type	Auto detect pad enable MAC2 [7]	VLAN pad enable MAC2 [6]	Pad/CRC enable MAC2 [5]	Action
Any	x	x	0	No pad or CRC check
Any	0	0	1	Pad to 60 bytes, append CRC
Any	x	1	1	Pad to 64 bytes, append CRC
Any	1	0	1	If untagged, pad to 60 bytes and append CRC. If VLAN tagged: pad to 64 bytes and append CRC.

10.10.1.3 Back-to-Back Inter-Packet-Gap Register

Table 153. Back-to-back Inter-packet-gap register (IPGT - address 0x2008 4008) bit description

Bit	Symbol	Function	Reset value
6:0	BTOBINTEGAP	BACK-TO-BACK INTER-PACKET-GAP.This is a programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet to the beginning of the next. In Full-Duplex mode, the register value should be the desired period in nibble times minus 3. In Half-Duplex mode, the register value should be the desired period in nibble times minus 6. In Full-Duplex the recommended setting is 0x15 (21d), which represents the minimum IPG of 960 ns (in 100 Mbps mode) or 9.6 µs (in 10 Mbps mode). In Half-Duplex the recommended setting is 0x12 (18d), which also represents the minimum IPG of 960 ns (in 100 Mbps mode) or 9.6 µs (in 10 Mbps mode).	0
31:7	-	Reserved. Read value is undefined, only zero should be written.	0

10.10.1.4 Non Back-to-Back Inter-Packet-Gap Register

Table 154. Non Back-to-back Inter-packet-gap register (IPGR - address 0x2008 400C) bit description

Bit	Symbol	Function	Reset value
6:0	NBTOBINTEGAP2	NON-BACK-TO-BACK INTER-PACKET-GAP PART2. This is a programmable field representing the Non-Back-to-Back Inter-Packet-Gap. The recommended value is 0x12 (18d), which represents the minimum IPG of 960 ns (in 100 Mbps mode) or 9.6 µs (in 10 Mbps mode).	0
7	-	Reserved. Read value is undefined, only zero should be written.	0
14:8	NBTOBINTEGAP1	NON-BACK-TO-BACK INTER-PACKET-GAP PART1. This is a programmable field representing the optional carrierSense window referenced in IEEE 802.3/4.2.3.2.1 'Carrier Deference'. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x0 to IPGR2. The recommended value is 0xC (12d)	0
31:15	-	Reserved. Read value is undefined, only zero should be written.	0

10.10.1.5 Collision Window / Retry Register

Table 155. Collision Window / Retry register (CLRT - address 0x2008 4010) bit description

Bit	Symbol	Function	Reset value
3:0	RETRANSMAX	RETRANSMISSION MAXIMUM.This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The Standard specifies the attemptLimit to be 0xF (15d). See IEEE 802.3/4.2.3.2.5.	0xF
7:4	-	Reserved. Read value is undefined, only zero should be written.	0
13:8	COLLWIN	COLLISION WINDOW. This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. The default value of 0x37 (55d) represents a 56 byte window following the preamble and SFD.	0x37
31:14	-	Reserved. Read value is undefined, only zero should be written.	NA

10.10.1.6 Maximum Frame Register

Table 156. Maximum Frame register (MAXF - address 0x2008 4014) bit description

Bit	Symbol	Function	Reset value
15:0	MAXFLEN	MAXIMUM FRAME LENGTH. This field resets to the value 0x0600, which represents a maximum receive frame of 1536 octets. An untagged maximum size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. If a shorter maximum length restriction is desired, program this 16-bit field.	0x0600
31:16	-	Unused	0

10.10.1.7 PHY Support Register

The SUPP register provides additional control over the RMII interface.

Table 157. PHY Support register (SUPP - address 0x2008 4018) bit description

Bit	Symbol	Function	Reset value
7:0	-	Unused	0
8	SPEED	This bit configures the Reduced MII logic for the current operating speed. When set, 100 Mbps mode is selected. When cleared, 10 Mbps mode is selected.	0
31:9	-	Unused	0

Unused bits in the PHY support register should be left as zeroes.

10.10.1.8 Test Register

Table 158. Test register (TEST - address 0x2008 401C) bit description

Bit	Symbol	Function	Reset value
0	SCPQ	SHORTCUT PAUSE QUANTA. This bit reduces the effective PAUSE quanta from 64 byte-times to 1 byte-time.	0
1	TESTPAUSE	This bit causes the MAC Control sublayer to inhibit transmissions, just as if a PAUSE Receive Control frame with a nonzero pause time parameter was received.	0
2	TESTBP	TEST BACKPRESSURE. Setting this bit will cause the MAC to assert backpressure on the link. Backpressure causes preamble to be transmitted, raising carrier sense. A transmit packet from the system will be sent during backpressure.	0
31:3	-	Unused	0

10.10.1.9 MII Mgmt Configuration Register

Table 159. MII Mgmt Configuration register (MCFG - address 0x2008 4020) bit description

Bit	Symbol	Function	Reset value
0	SCANINC	SCAN INCREMENT. Set this bit to cause the MII Management hardware to perform read cycles across a range of PHYs. When set, the MII Management hardware will perform read cycles from address 1 through the value set in PHY ADDRESS[4:0]. Clear this bit to allow continuous reads of the same PHY.	0
1	SUPPPREAMBLE	SUPPRESS PREAMBLE. Set this bit to cause the MII Management hardware to perform read/write cycles without the 32-bit preamble field. Clear this bit to cause normal cycles to be performed. Some PHYs support suppressed preamble.	0
5:2	CLOCKSEL	CLOCK SELECT. This field is used by the clock divide logic in creating the MII Management Clock (MDC) which IEEE 802.3u defines to be no faster than 2.5 MHz. Some PHYs support clock rates up to 12.5 MHz, however. The AHB bus clock (HCLK) is divided by the specified amount. Refer to Table 160 below for the definition of values for this field.	0
14:6	-	Unused	0
15	RESETMIIMGMT	RESET MII MGMT. This bit resets the MII Management hardware.	0
31:16	-	Unused	0

Table 160. Clock select encoding

Clock Select	Bit 5	Bit 4	Bit 3	Bit 2	Maximum AHB clock supported
Host Clock divided by 4	0	0	0	x	10
Host Clock divided by 6	0	0	1	0	15
Host Clock divided by 8	0	0	1	1	20
Host Clock divided by 10	0	1	0	0	25
Host Clock divided by 14	0	1	0	1	35
Host Clock divided by 20	0	1	1	0	50
Host Clock divided by 28	0	1	1	1	70
Host Clock divided by 36	1	0	0	0	80[1]
Host Clock divided by 40	1	0	0	1	90[1]
Host Clock divided by 44	1	0	1	0	100[1]
Host Clock divided by 48	1	0	1	1	120[1]
Host Clock divided by 52	1	1	0	0	130[1]
Host Clock divided by 56	1	1	0	1	140[1]
Host Clock divided by 60	1	1	1	0	150[1]
Host Clock divided by 64	1	1	1	1	160[1]

[1] The maximum AHB clock rate allowed is limited to the maximum CPU clock rate for the device.

10.10.1.10 MII Mgmt Command Register

Table 161. MII Mgmt Command register (MCMD - address 0x2008 4024) bit description

Bit	Symbol	Function	Reset value
0	READ	This bit causes the MII Management hardware to perform a single Read cycle. The Read data is returned in Register MRDD (MII Mgmt Read Data).	0
1	SCAN	This bit causes the MII Management hardware to perform Read cycles continuously. This is useful for monitoring Link Fail for example.	0
31:2	-	Unused	0

10.10.1.11 MII Mgmt Address Register

Table 162. MII Mgmt Address register (MADR - address 0x2008 4028) bit description

Bit	Symbol	Function	Reset value
4:0	REGADDR	REGISTER ADDRESS. This field represents the 5-bit Register Address field of Mgmt cycles. Up to 32 registers can be accessed.	0
7:5	-	Unused	0
12:8	PHYADDR	PHY ADDRESS. This field represents the 5-bit PHY Address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved).	0
31:13	-	Unused	0

10.10.1.12 MII Mgmt Write Data Register

Table 163. MII Mgmt Write Data register (MWTD - address 0x2008 402C) bit description

Bit	Symbol	Function
15:0	WRITEDATA	WRITE DATA. When written, an MII Mgmt write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from the MII Mgmt Address register (MADR).
31:16	-	Unused

10.10.1.13 MII Mgmt Read Data Register

Table 164. MII Mgmt Read Data register (MRDD - address 0x2008 4030) bit description

Bit	Symbol	Function	Reset value
15:0	READDATA	READ DATA. Following an MII Mgmt Read Cycle, the 16-bit data can be read from this location.	0
31:16	-	Unused	0

10.10.1.14 MII Mgmt Indicators Register

Table 165. MII Mgmt Indicators register (MIND - address 0x2008 4034) bit description

Bit	Symbol	Function	Reset value
0	BUSY	When 1 is returned - indicates MII Mgmt is currently performing an MII Mgmt Read or Write cycle.	0
1	SCANNING	When 1 is returned - indicates a scan operation (continuous MII Mgmt Read cycles) is in progress.	0
2	NOTVALID	When 1 is returned - indicates MII Mgmt Read cycle has not completed and the Read Data is not yet valid.	0
3	MIILINKFAIL	When 1 is returned - indicates that an MII Mgmt link fail has occurred.	0
31:4	-	Unused	0

Here are two examples to access PHY via the MII Management Controller.

For PHY Write if scan is not used:

1. Write 0 to MCMD
2. Write PHY address and register address to MADR
3. Write data to MWTD
4. Wait for busy bit to be cleared in MIND

For PHY Read if scan is not used:

1. Write 1 to MCMD
2. Write PHY address and register address to MADR
3. Wait for busy bit to be cleared in MIND
4. Write 0 to MCMD
5. Read data from MRDD

10.10.1.15 Station Address 0 Register

Table 166. Station Address register (SA0 - address 0x2008 4040) bit description

Bit	Symbol	Function	Reset value
7:0	SADDR2	STATION ADDRESS, 2nd octet. This field holds the second octet of the station address.	0
15:8	SADDR1	STATION ADDRESS, 1st octet. This field holds the first octet of the station address.	0
31:16	-	Unused	0

The station address is used for perfect address filtering and for sending pause control frames. For the ordering of the octets in the packet please refer to Figure 23.

10.10.1.16 Station Address 1 Register

Table 167. Station Address register (SA1 - address 0x2008 4044) bit description

Bit	Symbol	Function	Reset value
7:0	SADDR4	STATION ADDRESS, 4th octet. This field holds the fourth octet of the station address.	0
15:8	SADDR3	STATION ADDRESS, 3rd octet. This field holds the third octet of the station address.	0
31:16	-	Unused	0

The station address is used for perfect address filtering and for sending pause control frames. For the ordering of the octets in the packet please refer to Figure 23.

10.10.1.17 Station Address 2 Register

Table 168. Station Address register (SA2 - address 0x2008 4048) bit description

Bit	Symbol	Function	Reset value
7:0	SADDR6	STATION ADDRESS, 6th octet. This field holds the sixth octet of the station address.	0
15:8	SADDR5	STATION ADDRESS, 5th octet. This field holds the fifth octet of the station address.	0
31:16	-	Unused	0

The station address is used for perfect address filtering and for sending pause control frames. For the ordering of the octets in the packet please refer to Figure 23.

10.10.2 Control register definitions

10.10.2.1 Command Register

Table 169. Command register (COMMAND - address 0x2008 4100) bit description

Bit	Symbol	Function	Reset value
0	RXENABLE	Enable receive.	0
1	TXENABLE	Enable transmit.	0
2	-	Unused	0
3	REGRESET	When a 1 is written, all datapaths and the host registers are reset. The MAC needs to be reset separately.	0
4	TXRESET	When a 1 is written, the transmit datapath is reset.	-
5	RXRESET	When a 1 is written, the receive datapath is reset.	-
6	PASSRUNTFRAME	When set to 1 , passes runt frames s1maller than 64 bytes to memory unless they have a CRC error. If 0 runt frames are filtered out.	0
7	PASSRXFILTER	When set to 1 , disables receive filtering i.e. all frames received are written to memory.	0
8	TXFLOWCONTROL	Enable IEEE 802.3 / clause 31 flow control sending pause frames in full duplex and continuous preamble in half duplex.	0
9	RMII	When set to 1 , RMII mode is selected; if 0, MII mode is selected.	0
10	FULLDUPLEX	When set to 1 , indicates full duplex operation.	0
31:11	-	Unused	0

All bits can be written and read. The Tx/RxReset bits are write-only, reading will return a 0.

10.10.2.2 Status Register

The Status register (Status) is a read-only register.

Table 170. Status register (STATUS - address 0x2008 4104) bit description

Bit	Symbol	Function	Reset value
0	RXSTATUS	If 1, the receive channel is active. If 0, the receive channel is inactive.	0
1	TXSTATUS	If 1, the transmit channel is active. If 0, the transmit channel is inactive.	0
31:2	-	Unused	0

The values represent the status of the two channels/data paths. When the status is 1, the channel is active, meaning:

- It is enabled and the Rx/TxEnable bit is set in the Command register or it just got disabled while still transmitting or receiving a frame.
- Also, for the transmit channel, the transmit queue is not empty i.e. ProduceIndex != ConsumeIndex.
- Also, for the receive channel, the receive queue is not full i.e. ProduceIndex != ConsumeIndex - 1.

The status transitions from active to inactive if the channel is disabled by a software reset of the Rx/TxEnable bit in the Command register and the channel has committed the status and data of the current frame to memory. The status also transitions to inactive if the transmit queue is empty or if the receive queue is full and status and data have been committed to memory.

10.10.2.3 Receive Descriptor Base Address Register

Table 171. Receive Descriptor Base Address register (RXDESCRIPTOR - address 0x2008 4108) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to 00	-
31:2	RXDESCRIPTOR	MSBs of receive descriptor base address.	0

The receive descriptor base address is a byte address aligned to a word boundary i.e. LSB 1:0 are fixed to “00”. The register contains the lowest address in the array of descriptors.

10.10.2.4 Receive Status Base Address Register

The receive descriptor base address is a byte address aligned to a word boundary i.e. LSB 1:0 are fixed to “00”. The register contains the lowest address in the array of descriptors.

Table 172. Receive Status Base Address register (RXSTATUS - address 0x2008 410C) bit description

Bit	Symbol	Function	Reset value
2:0	-	Fixed to 000	-
31:3	RXSTATUS	MSBs of receive status base address.	0

The receive status base address is a byte address aligned to a double word boundary i.e. LSB 2:0 are fixed to “000”.

10.10.2.5 Receive Number of Descriptors Register

Table 173. Receive Number of Descriptors register (RXDESCRIPTORNUMBER - address 0x2008 4110) bit description

Bit	Symbol	Function	Reset value
15:0	RXDESCRIPTORN	RxDescriptorNumber. Number of descriptors in the descriptor array for which RxDescriptor is the base address. The number of descriptors is minus one encoded.	0
31:16	-	Unused	0

The receive number of descriptors register defines the number of descriptors in the descriptor array for which RxDescriptor is the base address. The number of descriptors should match the number of statuses. The register uses minus one encoding i.e. if the array has 8 elements, the value in the register should be 7.

10.10.2.6 Receive Produce Index Register

Table 174. Receive Produce Index register (RXPRODUCEINDEX - address 0x2008 4114) bit description

Bit	Symbol	Function	Reset value
15:0	RXPRODUCEIX	Index of the descriptor that is going to be filled next by the receive datapath.	0
31:16	-	Unused	0

The receive produce index register defines the descriptor that is going to be filled next by the hardware receive process. After a frame has been received, hardware increments the index. The value is wrapped to 0 once the value of RxDescriptorNumber has been reached. If the RxProduceIndex equals RxConsumeIndex - 1, the array is full and any further frames being received will cause a buffer overrun error.

10.10.2.7 Receive Consume Index Register

Table 175. Receive Consume Index register (RXCONSUMEINDEX - address 0x2008 4118) bit description

Bit	Symbol	Function	Reset value
15:0	RXCONSUMEIX	Index of the descriptor that is going to be processed next by the receive	
31:16	-	Unused	0

The receive consume register defines the descriptor that is going to be processed next by the software receive driver. The receive array is empty as long as RxProduceIndex equals RxConsumeIndex. As soon as the array is not empty, software can process the frame pointed to by RxConsumeIndex. After a frame has been processed by software, software should increment the RxConsumeIndex. The value must be wrapped to 0 once the value of RxDescriptorNumber has been reached. If the RxProduceIndex equals RxConsumeIndex - 1, the array is full and any further frames being received will cause a buffer overrun error.

10.10.2.8 Transmit Descriptor Base Address Register

Table 176. Transmit Descriptor Base Address register (TXDESCRIPTOR - address 0x2008 411C) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to “00”	-
31:2	TXD	TxDescriptor. MSBs of transmit descriptor base address.	0

The transmit descriptor base address is a byte address aligned to a word boundary i.e. LSB 1:0 are fixed to “00”. The register contains the lowest address in the array of descriptors.

10.10.2.9 Transmit Status Base Address Register

Table 177. Transmit Status Base Address register (TXSTATUS - address 0x2008 4120) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to “00”	-
31:2	TXSTAT	TxStatus. MSBs of transmit status base address.	0

The transmit status base address is a byte address aligned to a word boundary i.e. LSB 1:0 are fixed to “00”. The register contains the lowest address in the array of statuses.

10.10.2.10 Transmit Number of Descriptors Register

Table 178. Transmit Number of Descriptors register (TXDESCRIPTORNUMBER - address 0x2008 4124) bit description

Bit	Symbol	Function	Reset value
15:0	TXDN	TxDescriptorNumber. Number of descriptors in the descriptor array for which TxDescriptor is the base address. The register is minus one encoded.	0
31:16	-	Unused	0

The transmit number of descriptors register defines the number of descriptors in the descriptor array for which TxDescriptor is the base address. The number of descriptors should match the number of statuses. The register uses minus one encoding i.e. if the array has 8 elements, the value in the register should be 7.

10.10.2.11 Transmit Produce Index Register

Table 179. Transmit Produce Index register (TXPRODUCEINDEX - address 0x2008 4128) bit description

Bit	Symbol	Function	Reset value
15:0	TXPI	TxProduceIndex. Index of the descriptor that is going to be filled next by the transmit software driver.	0
31:16	-	Unused	0

The transmit produce index register defines the descriptor that is going to be filled next by the software transmit driver. The transmit descriptor array is empty as long as TxProduceIndex equals TxConsumeIndex. If the transmit hardware is enabled, it will start transmitting frames as soon as the descriptor array is not empty. After a frame has been processed by software, it should increment the TxProduceIndex. The value must be wrapped to 0 once the value of TxDescriptorNumber has been reached. If the TxProduceIndex equals TxConsumeIndex - 1 the descriptor array is full and software should stop producing new descriptors until hardware has transmitted some frames and updated the TxConsumeIndex.

10.10.2.12 Transmit Consume Index Register

Table 180. Transmit Consume Index register (TXCONSUMEINDEX - address 0x2008 412C) bit description

Bit	Symbol	Function	Reset value
15:0	TXCI	TxConsumeIndex. Index of the descriptor that is going to be transmitted next by the transmit datapath.	0
31:16	-	Unused	0

The transmit consume index register defines the descriptor that is going to be transmitted next by the hardware transmit process. After a frame has been transmitted hardware increments the index, wrapping the value to 0 once the value of TxDescriptorNumber has been reached. If the TxConsumeIndex equals TxProduceIndex the descriptor array is empty and the transmit channel will stop transmitting until software produces new descriptors.

10.10.2.13 Transmit Status Vector 0 Register

The transmit status vector registers store the most recent transmit status returned by the MAC. Since the status vector consists of more than 4 bytes, status is distributed over two registers TSV0 and TSV1. These registers are provided for debug purposes, because the communication between driver software and the Ethernet block takes place primarily through the frame descriptors. The status register contents are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

Table 181. Transmit Status Vector 0 register (TSV0 - address 0x2008 4158) bit description

Bit	Symbol	Function	Reset value
0	CRCERR	CRC error. The attached CRC in the packet did not match the internally generated CRC.	0
1	LCE	Length check error. Indicates the frame length field does not match the actual number of data items and is not a type field.	0
2	LOR	Length out of range. Indicates that frame type/length field was larger than 1500 bytes. The EMAC doesn't distinguish the frame type and frame length, so, e.g. when the IP(0x8000) or ARP(0x0806) packets are received, it compares the frame type with the max length and gives the "Length out of range" error. In fact, this bit is not an error indication, but simply a statement by the chip regarding the status of the received frame.	0
3	DONE	Transmission of packet was completed.	0
4	MULTICAST	Packet’s destination was a multicast address.	0
5	BROADCAST	Packet’s destination was a broadcast address.	0
6	PACKETDEFER	Packet was deferred for at least one attempt, but less than an excessive defer.	0
7	EXDF	Excessive Defer. Packet was deferred in excess of 6071 nibble times in 100 Mbps or 24287 bit times in 10 Mbps mode.	0
8	EXCOL	Excessive Collision. Packet was aborted due to exceeding of maximum allowed number of collisions.	0
9	LCOL	Late Collision. Collision occurred beyond collision window, 512 bit times.	0
10	GIANT	Byte count in frame was greater than can be represented in the transmit byte count field in TSV1.	0
11	UNDERRUN	Host side caused buffer underrun.	0
27:12	TOTALBYTES	The total number of bytes transferred including collided attempts.	0
28	CONTROLFRAME	The frame was a control frame.	0
29	PAUSE	The frame was a control frame with a valid PAUSE opcode.	0
30	BACKPRESSURE	Carrier-sense method backpressure was previously applied.	0
31	VLAN	Frame’s length/type field contained 0x8100 which is the VLAN protocol identifier.	0

[1] The EMAC doesn't distinguish the frame type and frame length, so, e.g. when the IP(0x8000) or ARP(0x0806) packets are received, it compares the frame type with the max length and gives the "Length out of range" error. In fact, this bit is not an error indication, but simply a statement by the chip regarding the status of the received frame.

10.10.2.14 Transmit Status Vector 1 Register

The Transmit Status Vector 1 register (TSV1) is a read-only register. The transmit status vector registers store the most recent transmit status returned by the MAC. Since the status vector consists of more than 4 bytes, status is distributed over two registers TSV0 and TSV1. These registers are provided for debug purposes, because the communication between driver software and the Ethernet block takes place primarily through the frame descriptors. The status register contents are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

Table 182. Transmit Status Vector 1 register (TSV1 - address 0x2008 415C) bit description

Bit	Symbol	Function	Reset value
15:0	TBC	Transmit byte count. The total number of bytes in the frame, not counting the collided bytes.	0
19:16	TCC	Transmit collision count. Number of collisions the current packet incurred during transmission attempts. The maximum number of collisions (16) cannot be represented.	0
31:20	-	Unused	0

10.10.2.15 Receive Status Vector Register

The Receive status vector register (RSV) is a read-only register. The receive status vector register stores the most recent receive status returned by the MAC. This register is provided for debug purposes, because the communication between driver software and the Ethernet block takes place primarily through the frame descriptors. The status register contents are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

Table 183. Receive Status Vector register (RSV - address 0x2008 4160) bit description

Bit	Symbol	Function	Reset value
15:0	RBC	Received byte count. Indicates length of received frame.	0
16	PPI	Packet previously ignored. Indicates that a packet was dropped.	0
17	RXDVSEEN	RXDV event previously seen. Indicates that the last receive event seen was not long enough to be a valid packet.	0
18	CESEEN	Carrier event previously seen. Indicates that at some time since the last receive statistics, a carrier event was detected.	0
19	RCV	Receive code violation. Indicates that received PHY data does not represent a valid receive code.	0
20	CRCERR	CRC error. The attached CRC in the packet did not match the internally generated CRC.	0
21	LCERR	Length check error. Indicates the frame length field does not match the actual number of data items and is not a type field.	0
22	LOR	Length out of range. Indicates that frame type/length field was larger than 1518 bytes. The EMAC doesn't distinguish the frame type and frame length, so, e.g. when the IP(0x8000) or ARP(0x0806) packets are received, it compares the frame type with the max length and gives the "Length out of range" error. In fact, this bit is not an error indication, but simply a statement by the chip regarding the status of the received frame.	0
23	ROK	Receive OK. The packet had valid CRC and no symbol errors.	0
24	MULTICAST	The packet destination was a multicast address.	0
25	BROADCAST	The packet destination was a broadcast address.	0
26	DRIBBLENIBBLE	Indicates that after the end of packet another 1-7 bits were received. A single nibble, called dribble nibble, is formed but not sent out.	0
27	CONTROLFRAME	The frame was a control frame.	0
28	PAUSE	The frame was a control frame with a valid PAUSE opcode.	0
29	UO	Unsupported Opcode. The current frame was recognized as a Control Frame but contains an unknown opcode.	0
30	VLAN	Frame’s length/type field contained 0x8100 which is the VLAN protocol identifier.	0
31	-	Unused	0

10.10.2.16 Flow Control Counter Register

Table 184. Flow Control Counter register (FLOWCONTROLCOUNTER - address 0x2008 4170) bit description

Bit	Symbol	Function	Reset value
15:0	MC	MirrorCounter. In full duplex mode the MirrorCounter specifies the number of cycles before re-issuing the Pause control frame.	0
31:16	PT	PauseTimer. In full-duplex mode the PauseTimer specifies the value that is inserted into the pause timer field of a pause flow control frame. In half duplex mode the PauseTimer specifies the number of backpressure cycles.	0

10.10.2.17 Flow Control Status Register

Table 185. Flow Control Status register (FLOWCONTROLSTATUS - address 0x2008 4174) bit description

Bit	Symbol	Function	Reset value
15:0	MCC	MirrorCounterCurrent. In full duplex mode this register represents the current value of the datapath's mirror counter which counts up to the value specified by the MirrorCounter field in the FlowControlCounter register. In half duplex mode the register counts until it reaches the value of the PauseTimer bits in the FlowControlCounter register.	0
31:16	-	Unused	0

10.10.3 Receive filter register definitions

10.10.3.1 Receive Filter Control Register

Table 186. Receive Filter Control register (RXFILTERCTRL - address 0x2008 4200) bit description

Bit	Symbol	Function	Reset value
0	AUE	AcceptUnicastEn. When set to 1, all unicast frames are accepted.	0
1	ABE	AcceptBroadcastEn. When set to 1, all broadcast frames are accepted.	0
2	AME	AcceptMulticastEn. When set to 1, all multicast frames are accepted.	0
3	AUHE	AcceptUnicastHashEn. When set to 1, unicast frames that pass the imperfect hash filter are accepted.	0
4	AMHE	AcceptMulticastHashEn. When set to 1, multicast frames that pass the imperfect hash filter are accepted.	0
5	APE	AcceptPerfectEn. When set to 1, the frames with a destination address identical to the station address are accepted.	0
11:6	-	Reserved. Read value is undefined, only zero should be written.	NA
12	MPEW	MagicPacketEnWoL. When set to 1, the result of the magic packet filter will generate a WoL interrupt when there is a match.	0
13	RFEW	RxFilterEnWoL. When set to 1, the result of the perfect address matching filter and the imperfect hash filter will generate a WoL interrupt when there is a match.	0
31:14	-	Unused	0

10.10.3.2 Receive Filter WoL Status Register

The Receive Filter Wake-up on LAN Status register (RxFilterWoLStatus) is a read-only register.

Table 187. Receive Filter WoL Status register (RXFILTERWOLSTATUS - address 0x2008 4204) bit description

Bit	Symbol	Function	Reset value
0	AUW	AcceptUnicastWoL. When the value is 1, a unicast frames caused WoL.	0
1	ABW	AcceptBroadcastWoL. When the value is 1, a broadcast frame caused WoL.	0
2	AMW	AcceptMulticastWoL. When the value is 1, a multicast frame caused WoL.	0
3	AUHW	AcceptUnicastHashWoL. When the value is 1, a unicast frame that passes the imperfect hash filter caused WoL.	0
4	AMHW	AcceptMulticastHashWoL. When the value is 1, a multicast frame that passes the imperfect hash filter caused WoL.	0
5	APW	AcceptPerfectWoL. When the value is 1, the perfect address matching filter caused WoL.	0
6	-	Unused	0
7	RFW	RxFilterWoL. When the value is 1, the receive filter caused WoL.	0
8	MPW	MagicPacketWoL. When the value is 1, the magic packet filter caused WoL.	0
31:9	-	Unused	0

The bits in this register record the cause for a WoL. Bits in RxFilterWoLStatus can be cleared by writing the RxFilterWoLClear register.

10.10.3.3 Receive Filter WoL Clear Register

The Receive Filter Wake-up on LAN Clear register (RxFilterWoLClear) is a write-only register.

Table 188. Receive Filter WoL Clear register (RxFilterWoLClear - address 0x2008 4208) bit description

Bit	Symbol	Function
0	AUWCLR	AcceptUnicastWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
1	ABWCLR	AcceptBroadcastWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
2	AMWCLR	AcceptMulticastWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
3	AUHWCLR	AcceptUnicastHashWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
4	AMHWCLR	AcceptMulticastHashWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
5	APWCLR	AcceptPerfectWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
6	-	Unused
7	RFWCLR	RxFilterWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
8	MPWCLR	MagicPacketWoLClr. When a 1 is written, the corresponding status bit in the RxFilterWoLStatus register is cleared.
31:9	-	Unused

The bits in this register are write-only; writing resets the corresponding bits in the RxFilterWoLStatus register.

10.10.3.4 Hash Filter Table LSBs Register

Details of Hash filter table use can be found in Section 10.13.10 “Receive filtering” on page 261.

Table 189. Hash Filter Table LSBs register (HASHFILTERL - address 0x2008 4210) bit description

Bit	Symbol	Function	Reset value
31:0	HFL	HashFilterL. Bits 31:0 of the imperfect filter hash table for receive filtering.	0

10.10.3.5 Hash Filter Table MSBs Register

Details of Hash filter table use can be found in Section 10.13.10 “Receive filtering” on page 261.

Table 190. Hash Filter MSBs register (HASHFILTERH - address 0x2008 4214) bit description

Bit	Symbol	Function	Reset value
31:0	HFH	Bits 63:32 of the imperfect filter hash table for receive filtering.	0

10.10.4 Module control register definitions

10.10.4.1 Interrupt Status Register

The Interrupt Status register (IntStatus) is a read-only register.Note that all bits are flip-flops with an asynchronous set in order to be able to generate interrupts if there are wake-up events while clocks are disabled.

Table 191. Interrupt Status register (INTSTATUS - address 0x2008 4FE0) bit description

Bit	Symbol	Function	Reset value
0	RXOVERRUNINT	Interrupt set on a fatal overrun error in the receive queue. The fatal interrupt should be resolved by a Rx soft-reset. The bit is not set when there is a nonfatal overrun error.	0
1	RXERRORINT	Interrupt trigger on receive errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError or NoDescriptor or Overrun.	0
2	RXFINISHEDINT	Interrupt triggered when all receive descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
3	RXDONEINT	Interrupt triggered when a receive descriptor has been processed while the Interrupt bit in the Control field of the descriptor was set.	0
4	TXUNDERRUNINT	Interrupt set on a fatal underrun error in the transmit queue. The fatal interrupt should be resolved by a Tx soft-reset. The bit is not set when there is a nonfatal underrun error.	0
5	TXERRORINT	Interrupt trigger on transmit errors: LateCollision, ExcessiveCollision and ExcessiveDefer, NoDescriptor or Underrun.	0
6	TXFINISHEDINT	Interrupt triggered when all transmit descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
7	TXDONEINT	Interrupt triggered when a descriptor has been transmitted while the Interrupt bit in the Control field of the descriptor was set.	0
11:8	-	Unused	0
12	SOFTINT	Interrupt triggered by software writing a 1 to the SoftIntSet bit in the IntSet register.	0
13	WAKEUPINT	Interrupt triggered by a Wake-up event detected by the receive filter.	0
31:14	-	Unused	0

The interrupt status register is read-only. Setting can be done via the IntSet register. Reset can be accomplished via the IntClear register.

10.10.4.2 Interrupt Enable Register

Table 192. Interrupt Enable register (INTENABLE - address 0x2008 4FE4) bit description

Bit	Symbol	Function	Reset value
0	RXOVERRUNINTEN	Enable for interrupt trigger on receive buffer overrun or descriptor underrun situations.	0
1	RXERRORINTEN	Enable for interrupt trigger on receive errors.	0
2	RXFINISHEDINTEN	Enable for interrupt triggered when all receive descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
3	RXDONEINTEN	Enable for interrupt triggered when a receive descriptor has been processed while the Interrupt bit in the Control field of the descriptor was set.	0
4	TXUNDERRUNINTEN	Enable for interrupt trigger on transmit buffer or descriptor underrun situations.	0
5	TXERRORINTEN	Enable for interrupt trigger on transmit errors.	0
6	TXFINISHEDINTEN	Enable for interrupt triggered when all transmit descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
7	TXDONEINTEN	Enable for interrupt triggered when a descriptor has been transmitted while the Interrupt bit in the Control field of the descriptor was set.	0
11:8	-	Unused	0
12	SOFTINTEN	Enable for interrupt triggered by the SoftInt bit in the IntStatus register, caused by software writing a 1 to the SoftIntSet bit in the IntSet register.	0
13	WAKEUPINTEN	Enable for interrupt triggered by a Wake-up event detected by the receive filter.	0
31:14	-	Unused	0

10.10.4.3 Interrupt Clear Register

Table 193. Interrupt Clear register (INTCLEAR - address 0x2008 4FE8) bit description

Bit	Symbol	Function
0	RXOVERRUNINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
1	RXERRORINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
2	RXFINISHEDINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
3	RXDONEINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
4	TXUNDERRUNINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
5	TXERRORINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
6	TXFINISHEDINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
7	TXDONEINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
11:8	-	Unused
12	SOFTINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
13	WAKEUPINTCLR	Writing a 1 clears the corresponding status bit in interrupt status register IntStatus.
31:14	-	Unused

The interrupt clear register is write-only. Writing a 1 to a bit of the IntClear register clears the corresponding bit in the status register. Writing a 0 will not affect the interrupt status.

10.10.4.4 Interrupt Set Register

Table 194. Interrupt Set register (INTSET - address 0x2008 4FEC) bit description

Bit	Symbol	Function
0	RXOVERRUNINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
1	RXERRORINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
2	RXFINISHEDINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
3	RXDONEINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
4	TXUNDERRUNINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
5	TXERRORINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
6	TXFINISHEDINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
7	TXDONEINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
11:8	-	Unused
12	SOFTINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
13	WAKEUPINTSET	Writing a 1 to one sets the corresponding status bit in interrupt status register IntStatus.
31:14	-	Unused

The interrupt set register is write-only. Writing a 1 to a bit of the IntSet register sets the corresponding bit in the status register. Writing a 0 will not affect the interrupt status.

10.10.4.5 Power-Down Register

The Power-Down register (PowerDown) is used to block all AHB accesses except accesses to the Power-Down register.

Table 195. Power-Down register (POWERDOWN - address 0x2008 4FF4) bit description

Bit	Symbol	Function	Reset value
30:0	-	Unused	0
31	PD	PowerDownMACAHB. If true, all AHB accesses will return a read/write error, except accesses to the Power-Down register.	0

Setting the bit will return an error on all read and write accesses on the MACAHB interface except for accesses to the Power-Down register.

10.11 Descriptor and status formats

This section defines the descriptor format for the transmit and receive scatter/gather DMA engines. Each Ethernet frame can consist of one or more fragments. Each fragment corresponds to a single descriptor. The DMA managers in the Ethernet block scatter (for receive) and gather (for transmit) multiple fragments for a single Ethernet frame.

10.11.1 Receive descriptors and statuses

Figure 24 depicts the layout of the receive descriptors in memory.

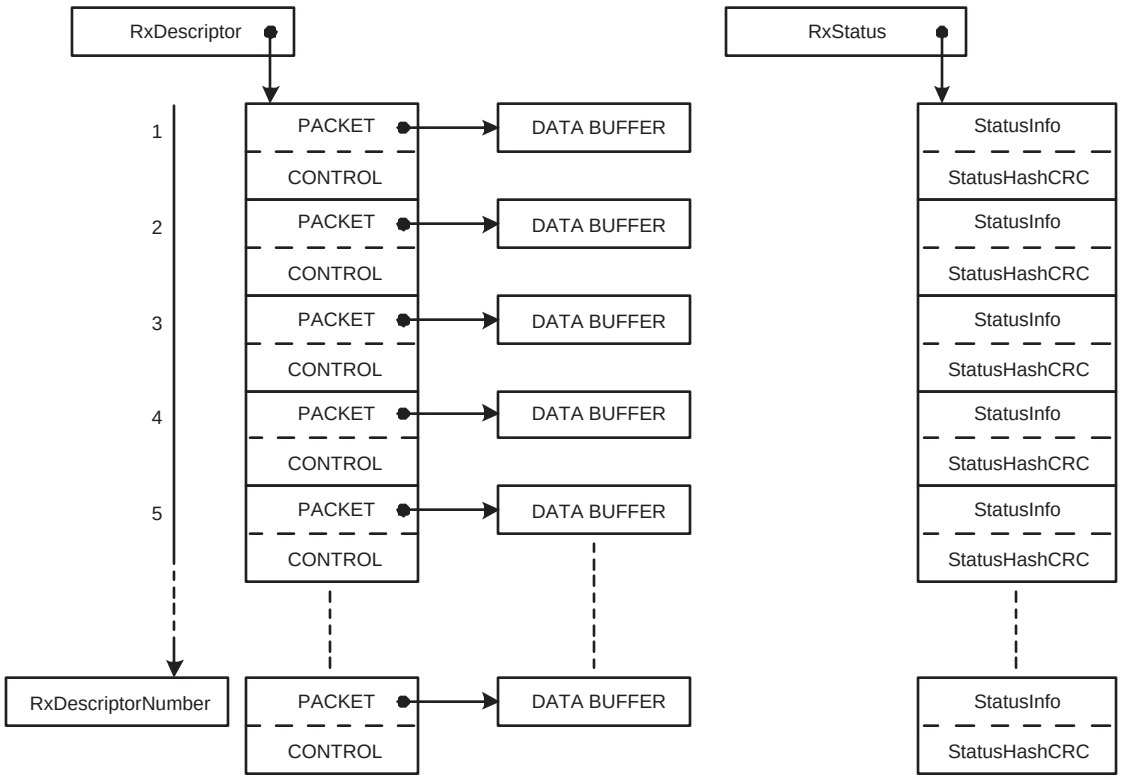


Fig 24. Receive descriptor memory layout

Receive descriptors are stored in an array in memory. The base address of the array is stored in the RxDescriptor register, and should be aligned on a 4 byte address boundary. The number of descriptors in the array is stored in the RxDescriptorNumber register using a minus one encoding style e.g. if the array has 8 elements the register value should be 7. Parallel to the descriptors there is an array of statuses. For each element of the descriptor array there is an associated status field in the status array. The base address of the status array is stored in the RxStatus register, and must be aligned on an 8 byte address boundary. During operation (when the receive data path is enabled) the RxDescriptor, RxStatus and RxDescriptorNumber registers should not be modified.

Two registers, RxConsumeIndex and RxProduceIndex, define the descriptor locations that will be used next by hardware and software. Both registers act as counters starting at 0 and wrapping when they reach the value of RxDescriptorNumber. The RxProduceIndex contains the index of the descriptor that is going to be filled with the next frame being

received. The RxConsumeIndex is programmed by software and is the index of the next descriptor that the software receive driver is going to process. When RxProduceIndex == RxConsumeIndex, the receive buffer is empty. When RxProduceIndex == RxConsumeIndex -1 (taking wraparound into account), the receive buffer is full and newly received data would generate an overflow unless the software driver frees up one or more descriptors.

Each receive descriptor takes two word locations (8 bytes) in memory. Likewise each status field takes two words (8 bytes) in memory. Each receive descriptor consists of a pointer to the data buffer for storing receive data (Packet) and a control word (Control). The Packet field has a zero address offset, the control field has a 4 byte address offset with respect to the descriptor address as defined in Table 196.

Table 196. Receive Descriptor Fields

Symbol	Address offset	Bytes	Description
Packet	0x0	4	Base address of the data buffer for storing receive data.
Control	0x4	4	Control information, see Table 197.

The data buffer pointer (Packet) is a 32-bit, byte aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in Table 197.

Table 197. Receive Descriptor Control Word

Bit	Symbol	Description
10:0	Size	Size in bytes of the data buffer. This is the size of the buffer reserved by the device driver for a frame or frame fragment i.e. the byte size of the buffer pointed to by the Packet field. The size is -1 encoded e.g. if the buffer is 8 bytes the size field should be equal to 7.
30:11	-	Unused
31	Interrupt	If true generate an RxDone interrupt when the data in this frame or frame fragment and the associated status information has been committed to memory.

Table 198 lists the fields in the receive status elements from the status array.

Table 198. Receive Status Fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Receive status return flags, see Table 200.
StatusHashCRC	0x4	4	The concatenation of the destination address hash CRC and the source address hash CRC.

Each receive status consists of two words. The StatusHashCRC word contains a concatenation of the two 9-bit hash CRCs calculated from the destination and source addresses contained in the received frame. After detecting the destination and source addresses, StatusHashCRC is calculated once, then held for every fragment of the same frame.

The concatenation of the two CRCs is shown in Table 199:

Table 199. Receive Status HashCRC Word

Bit	Symbol	Description
8:0	SAHashCRC	Hash CRC calculated from the source address.
15:9	-	Unused
24:16	DAHashCRC	Hash CRC calculated from the destination address.
31:25	-	Unused

The StatusInfo word contains flags returned by the MAC and flags generated by the receive data path reflecting the status of the reception. Table 200 lists the bit definitions in the StatusInfo word.

Table 200. Receive status information word

Bit	Symbol	Description
10:0	RxSize	The size in bytes of the actual data transferred into one fragment buffer. In other words, this is the size of the frame or fragment as actually written by the DMA manager for one descriptor. This may be different from the Size bits of the Control field in the descriptor that indicate the size of the buffer allocated by the device driver. Size is -1 encoded e.g. if the buffer has 8 bytes the RxSize value will be 7.
17:11	-	Unused
18	ControlFrame	Indicates this is a control frame for flow control, either a pause frame or a frame with an unsupported opcode.
19	VLAN	Indicates a VLAN frame.
20	FailFilter	Indicates this frame has failed the Rx filter. These frames will not normally pass to memory. But due to the limitation of the size of the buffer, part of this frame may already be passed to memory. Once the frame is found to have failed the Rx filter, the remainder of the frame will be discarded without being passed to the memory. However, if the PassRxFilter bit in the Command register is set, the whole frame will be passed to memory.
21	Multicast	Set when a multicast frame is received.
22	Broadcast	Set when a broadcast frame is received.
23	CRCError	The received frame had a CRC error.
24	SymbolError	The PHY reports a bit error over the PHY interface during reception.
25	LengthError	The frame length field value in the frame specifies a valid length, but does not match the actual data length.
26	RangeError[1]	The received packet exceeds the maximum packet size.
27	AlignmentError	An alignment error is flagged when dribble bits are detected and also a CRC error is detected. This is in accordance with IEEE std. 802.3/clause 4.3.2.
28	Overrun	Receive overrun. The adapter can not accept the data stream.
29	NoDescriptor	No new Rx descriptor is available and the frame is too long for the buffer size in the current receive descriptor.
30	LastFlag	When set to 1, indicates this descriptor is for the last fragment of a frame. If the frame consists of a single fragment, this bit is also set to 1.
31	Error	An error occurred during reception of this frame. This is a logical OR of AlignmentError, RangeError, LengthError, SymbolError, CRCError, and Overrun.

[1] The EMAC doesn't distinguish the frame type and frame length, so, e.g. when the IP(0x8000) or ARP(0x0806) packets are received, it compares the frame type with the max length and gives the "Range" error. In fact, this bit is not an error indication, but simply a statement by the chip regarding the status of the received frame.

For multi-fragment frames, the value of the AlignmentError, RangeError, LengthError, SymbolError and CRCError bits in all but the last fragment in the frame will be 0; likewise the value of the FailFilter, Multicast, Broadcast, VLAN and ControlFrame bits is undefined. The status of the last fragment in the frame will copy the value for these bits from the MAC. All fragment statuses will have valid LastFrag, RxSize, Error, Overrun and NoDescriptor bits.

10.11.2 Transmit descriptors and statuses

Figure 25 depicts the layout of the transmit descriptors in memory.

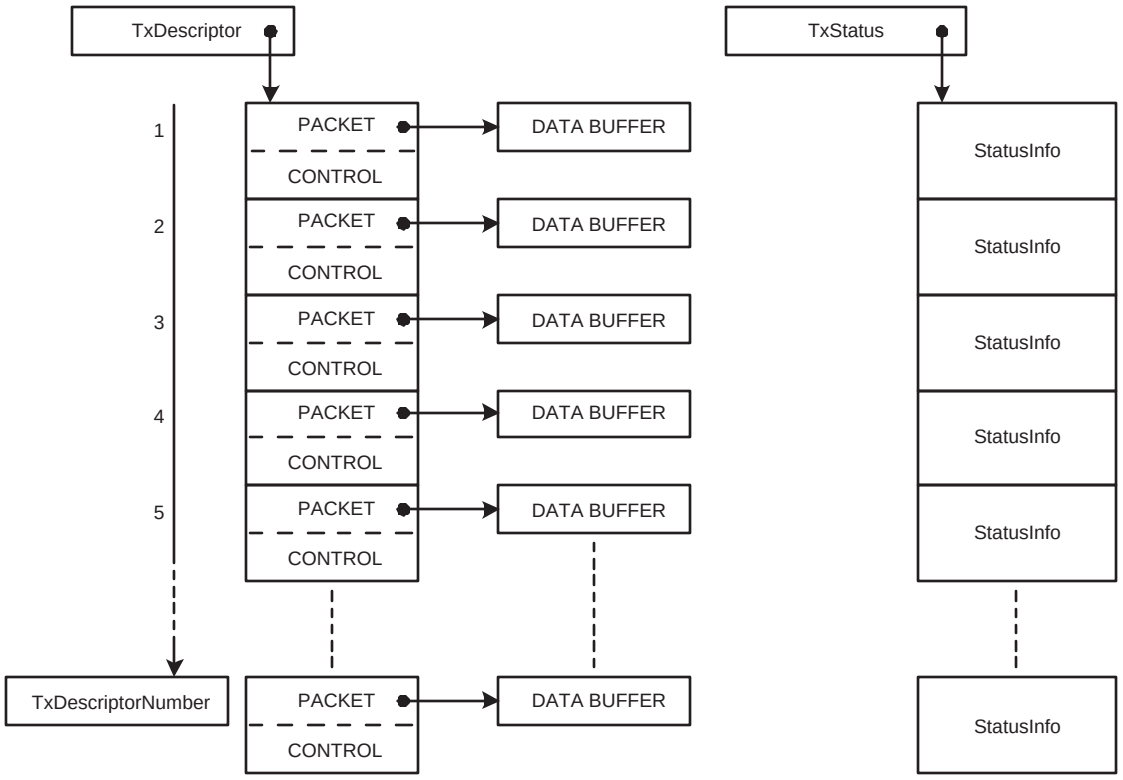


Fig 25. Transmit descriptor memory layout

Transmit descriptors are stored in an array in memory. The lowest address of the transmit descriptor array is stored in the TxDescriptor register, and must be aligned on a 4 byte address boundary. The number of descriptors in the array is stored in the TxDescriptorNumber register using a minus one encoding style i.e. if the array has 8 elements the register value should be 7. Parallel to the descriptors there is an array of statuses. For each element of the descriptor array there is an associated status field in the status array. The base address of the status array is stored in the TxStatus register, and must be aligned on a 4 byte address boundary. During operation (when the transmit data path is enabled) the TxDescriptor, TxStatus, and TxDescriptorNumber registers should not be modified.

Two registers, TxConsumeIndex and TxProduceIndex, define the descriptor locations that will be used next by hardware and software. Both register act as counters starting at 0 and wrapping when they reach the value of TxDescriptorNumber. The TxProduceIndex

contains the index of the next descriptor that is going to be filled by the software driver. The TxConsumeIndex contains the index of the next descriptor going to be transmitted by the hardware. When TxProduceIndex == TxConsumeIndex, the transmit buffer is empty. When TxProduceIndex == TxConsumeIndex -1 (taking wraparound into account), the transmit buffer is full and the software driver cannot add new descriptors until the hardware has transmitted one or more frames to free up descriptors.

Each transmit descriptor takes two word locations (8 bytes) in memory. Likewise each status field takes one word (4 bytes) in memory. Each transmit descriptor consists of a pointer to the data buffer containing transmit data (Packet) and a control word (Control). The Packet field has a zero address offset, whereas the control field has a 4 byte address offset, see Table 201.

Table 201. Transmit descriptor fields

Symbol	Address offset	Bytes	Description
Packet	0x0	4	Base address of the data buffer containing transmit data.
Control	0x4	4	Control information, see Table 202.

The data buffer pointer (Packet) is a 32-bit, byte aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in Table 202.

Table 202. Transmit descriptor control word

Bit	Symbol	Description
10:0	Size	Size in bytes of the data buffer. This is the size of the frame or fragment as it needs to be fetched by the DMA manager. In most cases it will be equal to the byte size of the data buffer pointed to by the Packet field of the descriptor. Size is -1 encoded e.g. a buffer of 8 bytes is encoded as the Size value 7.
25:11	-	Unused
26	Override	Per frame override. If true, bits 30:27 will override the defaults from the MAC internal registers. If false, bits 30:27 will be ignored and the default values from the MAC will be used.
27	Huge	If true, enables huge frame, allowing unlimited frame sizes. When false, prevents transmission of more than the maximum frame length (MAXF[15:0]).
28	Pad	If true, pad short frames to 64 bytes.
29	CRC	If true, append a hardware CRC to the frame.
30	Last	If true, indicates that this is the descriptor for the last fragment in the transmit frame. If false, the fragment from the next descriptor should be appended.
31	Interrupt	If true, a TxDone interrupt will be generated when the data in this frame or frame fragment has been sent and the associated status information has been committed to memory.

Table 203 shows the one field transmit status.

Table 203. Transmit status fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Transmit status return flags, see Table 204.

The transmit status consists of one word which is the StatusInfo word. It contains flags returned by the MAC and flags generated by the transmit data path reflecting the status of the transmission. Table 204 lists the bit definitions in the StatusInfo word.

Table 204. Transmit status information word

Bit	Symbol	Description
20:0	-	Unused
24:21	CollisionCount	The number of collisions this packet incurred, up to the Retransmission Maximum.
25	Defer	This packet incurred deferral, because the medium was occupied. This is not an error unless excessive deferral occurs.
26	ExcessiveDefer	This packet incurred deferral beyond the maximum deferral limit and was aborted.
27	ExcessiveCollision	Indicates this packet exceeded the maximum collision limit and was aborted.
28	LateCollision	An Out of window Collision was seen, causing packet abort.
29	Underrun	A Tx underrun occurred due to the adapter not producing transmit data.
30	NoDescriptor	The transmit stream was interrupted because a descriptor was not available.
31	Error	An error occurred during transmission. This is a logical OR of Underrun, LateCollision, ExcessiveCollision, and ExcessiveDefer.

For multi-fragment frames, the value of the LateCollision, ExcessiveCollision, ExcessiveDefer, Defer and CollissionCount bits in all but the last fragment in the frame will be 0. The status of the last fragment in the frame will copy the value for these bits from the MAC. All fragment statuses will have valid Error, NoDescriptor and Underrun bits.

10.12 Ethernet block functional description

This section defines the functions of the DMA capable 10/100 Ethernet MAC. After introducing the DMA concepts of the Ethernet block, and a description of the basic transmit and receive functions, this section elaborates on advanced features such as flow control, receive filtering, etc.

10.12.1 Overview

The Ethernet block can transmit and receive Ethernet packets from an off-chip Ethernet PHY connected through the MII/RMII interface. MII or RMII mode is selected by software.

Typically during system start-up, the Ethernet block will be initialized. Software initialization of the Ethernet block should include initialization of the descriptor and status arrays as well as the receiver fragment buffers.

Remark: when initializing the Ethernet block, it is important to first configure the PHY and insure that reference clocks (ENET_REF_CLK signal in RMII mode, or both ENET_RX_CLK and ENET_TX_CLK signals in MII mode) are present at the external pins and connected to the EMAC module (selecting the appropriate pins using the IOCON registers) prior to continuing with Ethernet configuration. Otherwise the CPU can become locked and no further functionality will be possible. This will cause JTAG lose communication with the target, if debug mode is being used.

To transmit a packet the software driver has to set up the appropriate Control registers and a descriptor to point to the packet data buffer before transferring the packet to hardware by incrementing the TxProduceIndex register. After transmission, hardware will increment TxConsumeIndex and optionally generate an interrupt.

The hardware will receive packets from the PHY and apply filtering as configured by the software driver. While receiving a packet the hardware will read a descriptor from memory to find the location of the associated receiver data buffer. Receive data is written in the data buffer and receive status is returned in the receive descriptor status word. Optionally an interrupt can be generated to notify software that a packet has been received. Note that the DMA manager will prefetch and buffer up to three descriptors.

10.12.2 AHB interface

The registers of the Ethernet block connect to an AHB slave interface to allow access to the registers from the CPU.

The AHB interface has a 32-bit data path, which supports only word accesses and has an address aperture of 4 kB. Table 149 lists the registers of the Ethernet block.

All AHB write accesses to registers are posted except for accesses to the IntSet, IntClear and IntEnable registers. AHB write operations are executed in order.

If the PowerDown bit of the PowerDown register is set, all AHB read and write accesses will return a read or write error except for accesses to the PowerDown register.

Bus Errors

The Ethernet block generates errors for several conditions:

- The AHB interface will return a read error when there is an AHB read access to a write-only register; likewise a write error is returned when there is an AHB write access to the read-only register. An AHB read or write error will be returned on AHB read or write accesses to reserved registers. These errors are propagated back to the CPU. Registers defined as read-only and write-only are identified in Table 149.
- If the PowerDown bit is set all accesses to AHB registers will result in an error response except for accesses to the PowerDown register.

10.13 Interrupts

The Ethernet block has a single interrupt request output to the CPU (via the NVIC).

The interrupt service routine must read the IntStatus register to determine the origin of the interrupt. All interrupt statuses can be set by software writing to the IntSet register; statuses can be cleared by software writing to the IntClear register.

The transmit and receive data paths can only set interrupt statuses, they cannot clear statuses. The SoftInt interrupt cannot be set by hardware and can be used by software for test purposes.

10.13.1 Direct Memory Access (DMA)

Descriptor arrays

The Ethernet block includes two DMA managers. The DMA managers make it possible to transfer frames directly to and from memory with little support from the processor and without the need to trigger an interrupt for each frame.

The DMA managers work with arrays of frame descriptors and statuses that are stored in memory. The descriptors and statuses act as an interface between the Ethernet hardware and the device driver software. There is one descriptor array for receive frames and one descriptor array for transmit frames. Using buffering for frame descriptors, the memory traffic and memory bandwidth utilization of descriptors can be kept small.

Each frame descriptor contains two 32-bit fields: the first field is a pointer to a data buffer containing a frame or a fragment, whereas the second field is a control word related to that frame or fragment.

The software driver must write the base addresses of the descriptor and status arrays in the TxDescriptor/RxDescriptor and TxStatus/RxStatus registers. The number of descriptors/statuses in each array must be written in the TxDescriptorNumber/RxDescriptorNumber registers. The number of descriptors in an array corresponds to the number of statuses in the associated status array.

Transmit descriptor arrays, receive descriptor arrays and transmit status arrays must be aligned on a 4 byte (32bit)address boundary, while the receive status array must be aligned on a 8 byte (64bit) address boundary.

Ownership of descriptors

Both device driver software and Ethernet hardware can read and write the descriptor arrays at the same time in order to produce and consume descriptors. A descriptor is "owned" either by the device driver or by the Ethernet hardware. Only the owner of a descriptor reads or writes its value. Typically, the sequence of use and ownership of descriptors and statuses is as follows: a descriptor is owned and set up by the device driver; ownership of the descriptor/status is passed by the device driver to the Ethernet block, which reads the descriptor and writes information to the status field; the Ethernet block passes ownership of the descriptor back to the device driver, which uses the status information and then recycles the descriptor to be used for another frame. Software must pre-allocate the memory used to hold the descriptor arrays.

Software can hand over ownership of descriptors and statuses to the hardware by incrementing (and wrapping if on the array boundary) the TxProduceIndex/RxConsumeIndex registers. Hardware hands over descriptors and status to software by updating the TxConsumeIndex/ RxProduceIndex registers.

After handing over a descriptor to the receive and transmit DMA hardware, device driver software should not modify the descriptor or reclaim the descriptor by decrementing the TxProduceIndex/ RxConsumeIndex registers because descriptors may have been prefetched by the hardware. In this case the device driver software will have to wait until the frame has been transmitted or the device driver has to soft-reset the transmit and/or receive data paths which will also reset the descriptor arrays.

Sequential order with wrap-around

When descriptors are read from and statuses are written to the arrays, this is done in sequential order with wrap-around. Sequential order means that when the Ethernet block has finished reading/writing a descriptor/status, the next descriptor/status it reads/writes is the one at the next higher, adjacent memory address. Wrap around means that when the Ethernet block has finished reading/writing the last descriptor/status of the array (with the highest memory address), the next descriptor/status it reads/writes is the first descriptor/status of the array at the base address of the array.

Full and Empty state of descriptor arrays

The descriptor arrays can be empty, partially full or full. A descriptor array is empty when all descriptors are owned by the producer. A descriptor array is partially full if both producer and consumer own part of the descriptors and both are busy processing those descriptors. A descriptor array is full when all descriptors (except one) are owned by the consumer, so that the producer has no more room to process frames. Ownership of descriptors is indicated with the use of a consume index and a produce index. The produce index is the first element of the array owned by the producer. It is also the index of the array element that is next going to be used by the producer of frames (it may already be busy using it and subsequent elements). The consume index is the first element of the array that is owned by the consumer. It is also the number of the array element next to be consumed by the consumer of frames (it and subsequent elements may already be in the process of being consumed). If the consume index and the produce index are equal, the descriptor array is empty and all array elements are owned by the producer. If the consume index equals the produce index plus one, then the array is full and all array elements (except the one at the produce index) are owned by the consumer. With a full descriptor array, still one array element is kept empty, to be able to easily distinguish the full or empty state by looking at the value of the produce index and consume index. An array must have at least 2 elements to be able to indicate a full descriptor array with a produce index of value 0 and a consume index of value 1. The wrap around of the arrays is taken into account when determining if a descriptor array is full, so a produce index that indicates the last element in the array and a consume index that indicates the first element in the array, also means the descriptor array is full. When the produce index and the consume index are unequal and the consume index is not the produce index plus one (with wrap around taken into account), then the descriptor array is partially full and both the consumer and producer own enough descriptors to be able to operate actively on the descriptor array.

Interrupt bit

The descriptors have an Interrupt bit, which is programmed by software. When the Ethernet block is processing a descriptor and finds this bit set, it will allow triggering an interrupt (after committing status to memory) by passing the RxDoneInt or TxDoneInt bits in the IntStatus register to the interrupt output pin. If the Interrupt bit is not set in the descriptor, then the RxDoneInt or TxDoneInt are not set and no interrupt is triggered (note that the corresponding bits in IntEnable must also be set to trigger interrupts). This offers flexible ways of managing the descriptor arrays. For instance, the device driver could add 10 frames to the Tx descriptor array, and set the Interrupt bit in descriptor number 5 in the descriptor array. This would invoke the interrupt service routine before the transmit descriptor array is completely exhausted. The device driver could add another batch of frames to the descriptor array, without interrupting continuous transmission of frames.

Frame fragments

For maximum flexibility in frame storage, frames can be split up into multiple frame fragments with fragments located in different places in memory. In this case one descriptor is used for each frame fragment. So, a descriptor can point to a single frame or to a fragment of a frame. By using fragments, scatter/gather DMA can be done: transmit frames are gathered from multiple fragments in memory and receive frames can be scattered to multiple fragments in memory.

By stringing together fragments it is possible to create large frames from small memory areas. Another use of fragments is to be able to locate a frame header and frame payload in different places and to concatenate them without copy operations in the device driver.

For transmissions, the Last bit in the descriptor Control field indicates if the fragment is the last in a frame; for receive frames, the LastFrag bit in the StatusInfo field of the status words indicates if the fragment is the last in the frame. If the Last(Frag) bit is 0 the next descriptor belongs to the same Ethernet frame, If the Last(Frag) bit is 1 the next descriptor is a new Ethernet frame.

10.13.2 Initialization

After reset, the Ethernet software driver needs to initialize the Ethernet block. During initialization the software needs to:

- Remove the soft reset condition from the MAC.
- Configure the PHY via the MIIM interface of the MAC.

  Remark: it is important to configure the PHY and insure that reference clocks (ENET_REF_CLK signal in RMII mode, or both ENET_RX_CLK and ENET_TX_CLK signals in MII mode) are present at the external pins and connected to the EMAC module (selecting the appropriate pins using the IOCON registers) prior to continuing with Ethernet configuration. Otherwise the CPU can become locked and no further functionality will be possible. This will cause JTAG lose communication with the target, if debug mode is being used.
- Select MII or RMII mode
- Configure the transmit and receive DMA engines, including the descriptor arrays.
- Configure the host registers (MAC1,MAC2 etc.) in the MAC.
- Enable the receive and transmit data paths.

Depending on the PHY, the software needs to initialize registers in the PHY via the MII Management interface. The software can read and write PHY registers by programming the MCFG, MCMD, MADR registers of the MAC. Write data should be written to the MWTD register; read data and status information can be read from the MRDD and MIND registers.

The Ethernet block supports MII and RMII PHYs. During initialization software must select MII or RMII mode by programming the Command register.

Before switching to RMII mode the default soft reset (MAC1 register bit 15) has to be de-asserted. The clock(s) from the PHY must be running and internally connected during this operation.

Transmit and receive DMA engines should be initialized by the device driver by allocating the descriptor and status arrays in memory. Transmit and receive functions have their own dedicated descriptor and status arrays. The base addresses of these arrays need to be programmed in the TxDescriptor/TxStatus and RxDescriptor/RxStatus registers. The number of descriptors in an array matches the number of statuses in an array.

Please note that the transmit descriptors, receive descriptors and receive statuses are 8 bytes each while the transmit statuses are 4 bytes each. All descriptor arrays and transmit statuses need to be aligned on 4 byte boundaries; receive status arrays need to be aligned on 8 byte boundaries. The number of descriptors in the descriptor arrays needs to be written to the TxDescriptorNumber/RxDescriptorNumber registers using a -1 encoding i.e. the value in the registers is the number of descriptors minus one e.g. if the descriptor array has 4 descriptors the value of the number of descriptors register should be 3.

After setting up the descriptor arrays, frame buffers need to be allocated for the receive descriptors before enabling the receive data path. The Packet field of the receive descriptors needs to be filled with the base address of the frame buffer of that descriptor. Amongst others the Control field in the receive descriptor needs to contain the size of the data buffer using -1 encoding.

The receive data path has a configurable filtering function for discarding/ignoring specific Ethernet frames. The filtering function should also be configured during initialization.

After an assertion of the hardware reset, the soft reset bit in the MAC will be asserted. The soft reset condition must be removed before the Ethernet block can be enabled.

Enabling of the receive function is located in two places. The receive DMA manager needs to be enabled and the receive data path of the MAC needs to be enabled. To prevent overflow in the receive DMA engine the receive DMA engine should be enabled by setting the RxEnable bit in the Command register before enabling the receive data path in the MAC by setting the RECEIVE ENABLE bit in the MAC1 register.

The transmit DMA engine can be enabled at any time by setting the TxEnable bit in the Command register.

Before enabling the data paths, several options can be programmed in the MAC, such as automatic flow control, transmit to receive loop-back for verification, full/half duplex modes, etc.

Base addresses of descriptor arrays and descriptor array sizes cannot be modified without a (soft) reset of the receive and transmit data paths.

10.13.3 Transmit process

Overview

This section outlines the transmission process.

Device driver sets up descriptors and data

If the descriptor array is full the device driver should wait for the descriptor arrays to become not full before writing to a descriptor in the descriptor array. If the descriptor array is not full, the device driver should use the descriptor numbered TxProduceIndex of the array pointed to by TxDescriptor.

The Packet pointer in the descriptor is set to point to a data frame or frame fragment to be transmitted. The Size field in the Command field of the descriptor should be set to the number of bytes in the fragment buffer, -1 encoded. Additional control information can be indicated in the Control field in the descriptor (bits Interrupt, Last, CRC, Pad).

After writing the descriptor the descriptor needs to be handed over to the hardware by incrementing (and possibly wrapping) the TxProduceIndex register.

If the transmit data path is disabled, the device driver should not forget to enable the transmit data path by setting the TxEnable bit in the Command register.

When there is a multi-fragment transmission for fragments other than the last, the Last bit in the descriptor must be set to 0; for the last fragment the Last bit must be set to 1. To trigger an interrupt when the frame has been transmitted and transmission status has been committed to memory, set the Interrupt bit in the descriptor Control field to 1. To have the hardware add a CRC in the frame sequence control field of this Ethernet frame, set the CRC bit in the descriptor. This should be done if the CRC has not already been added by software. To enable automatic padding of small frames to the minimum required frame size, set the Pad bit in the Control field of the descriptor to 1. In typical applications bits CRC and Pad are both set to 1.

The device driver can set up interrupts using the IntEnable register to wait for a signal of completion from the hardware or can periodically inspect (poll) the progress of transmission. It can also add new frames at the end of the descriptor array, while hardware consumes descriptors at the start of the array.

The device driver can stop the transmit process by resetting the TxEnable bit in the Command register to 0. The transmission will not stop immediately; frames already being transmitted will be transmitted completely and the status will be committed to memory before deactivating the data path. The status of the transmit data path can be monitored by the device driver reading the TxStatus bit in the Status register.

As soon as the transmit data path is enabled and the corresponding TxConsumeIndex and TxProduceIndex are not equal i.e. the hardware still needs to process frames from the descriptor array, the TxStatus bit in the Status register will return to 1 (active).

Tx DMA manager reads the Tx descriptor array

When the TxEnable bit is set, the Tx DMA manager reads the descriptors from memory at the address determined by TxDescriptor and TxConsumeIndex. The number of descriptors requested is determined by the total number of descriptors owned by the hardware: TxProduceIndex - TxConsumeIndex. Block transferring descriptors minimizes memory loading. Read data returned from memory is buffered and consumed as needed.

Tx DMA manager transmits data

After reading the descriptor the transmit DMA engine reads the associated frame data from memory and transmits the frame. After transfer completion, the Tx DMA manager writes status information back to the StatusInfo and StatusHashCRC words of the status field. The value of the TxConsumeIndex is only updated after status information has been committed to memory, which is checked by an internal tag protocol in the memory interface. The Tx DMA manager continues to transmit frames until the descriptor array is empty. If the transmit descriptor array is empty the TxStatus bit in the Status register will return to 0 (inactive). If the descriptor array is empty the Ethernet hardware will set the TxFinishedInt bit of the IntStatus register. The transmit data path will still be enabled.

The Tx DMA manager inspects the Last bit of the descriptor Control field when loading the descriptor. If the Last bit is 0, this indicates that the frame consists of multiple fragments. The Tx DMA manager gathers all the fragments from the host memory, visiting a string of frame descriptors, and sends them out as one Ethernet frame on the Ethernet connection. When the Tx DMA manager finds a descriptor with the Last bit in the Control field set to 1, this indicates the last fragment of the frame and thus the end of the frame is found.

Update ConsumeIndex

Each time the Tx DMA manager commits a status word to memory it completes the transmission of a descriptor and it increments the TxConsumeIndex (taking wrap around into account) to hand the descriptor back to the device driver software. Software can re-use the descriptor for new transmissions after hardware has handed it back.

The device driver software can keep track of the progress of the DMA manager by reading the TxConsumeIndex register to see how far along the transmit process is. When the Tx descriptor array is emptied completely, the TxConsumeIndex register retains its last value.

Write transmission status

After the frame has been transmitted over the MII/RMII bus, the StatusInfo word of the frame descriptor is updated by the DMA manager.

If the descriptor is for the last fragment of a frame (or for the whole frame if there are no fragments), then depending on the success or failure of the frame transmission, error flags (Error, LateCollision, ExcessiveCollision, Underrun, ExcessiveDefer, Defer) are set in the status. The CollisionCount field is set to the number of collisions the frame incurred, up to the Retransmission Maximum programmed in the Collision window/retry register of the MAC.

Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx DMA manager. Even if the descriptor is for a frame fragment other than the last fragment, the error flags are returned via the AHB interface. If the Ethernet block detects a transmission error during transmission of a (multi-fragment) frame, all remaining fragments of the frame are still read via the AHB interface. After an error, the remaining transmit data is discarded by the Ethernet block. If there are errors

during transmission of a multi-fragment frame the error statuses will be repeated until the last fragment of the frame. Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx DMA manager. These may include error information if the error is detected early enough. The status for the last fragment in the frame will only be written after the transmission has completed on the Ethernet connection. Thus, the status for the last fragment will always reflect any error that occurred anywhere in the frame.

The status of the last frame transmission can also be inspected by reading the TSV0 and TSV1 registers. These registers do not report statuses on a fragment basis and do not store information of previously sent frames. They are provided primarily for debug purposes, because the communication between driver software and the Ethernet block takes place through the frame descriptors. The status registers are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

Transmission error handling

If an error occurs during the transmit process, the Tx DMA manager will report the error via the transmission StatusInfo word written in the Status array and the IntStatus interrupt status register.

The transmission can generate several types of errors: LateCollision, ExcessiveCollision, ExcessiveDefer, Underrun, and NoDescriptor. All have corresponding bits in the transmission StatusInfo word. In addition to the separate bits in the StatusInfo word, LateCollision, ExcessiveCollision, and ExcessiveDefer are ORed together into the Error bit of the Status. Errors are also propagated to the IntStatus register; the TxError bit in the IntStatus register is set in the case of a LateCollision, ExcessiveCollision, ExcessiveDefer, or NoDescriptor error; Underrun errors are reported in the TxUnderrun bit of the IntStatus register.

Underrun errors can have three causes:

- The next fragment in a multi-fragment transmission is not available. This is a nonfatal error. A NoDescriptor status will be returned on the previous fragment and the TxError bit in IntStatus will be set.
- The transmission fragment data is not available when the Ethernet block has already started sending the frame. This is a nonfatal error. An Underrun status will be returned on transfer and the TxError bit in IntStatus will be set.
- The flow of transmission statuses stalls and a new status has to be written while a previous status still waits to be transferred across the memory interface. This is a fatal error which can only be resolved by a soft reset of the hardware.

The first and second situations are nonfatal and the device driver has to re-send the frame or have upper software layers re-send the frame. In the third case the hardware is in an undefined state and needs to be soft reset by setting the TxReset bit in the Command register.

After reporting a LateCollision, ExcessiveCollision, ExcessiveDefer or Underrun error, the transmission of the erroneous frame will be aborted, remaining transmission data and frame fragments will be discarded and transmission will continue with the next frame in the descriptor array.

Device drivers should catch the transmission errors and take action.

Transmit triggers interrupts

The transmit data path can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set, the Tx DMA will set the TxDoneInt bit in the IntStatus register after sending the fragment and committing the associated transmission status to memory. Even if a descriptor (fragment) is not the last in a multi-fragment frame the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor array is empty while the Ethernet hardware is enabled the hardware will set the TxFinishedInt bit of the IntStatus register.
- If the AHB interface does not consume the transmission statuses at a sufficiently high bandwidth the transmission may underrun in which case the TxUnderrun bit will be set in the IntStatus register. This is a fatal error which requires a soft reset of the transmission queue.
- In the case of a transmission error (LateCollision, ExcessiveCollision, or ExcessiveDefer) or a multi-fragment frame where the device driver did provide the initial fragments but did not provide the rest of the fragments (NoDescriptor) or in the case of a nonfatal overrun, the hardware will set the TxErrorInt bit of the IntStatus register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the IntEnable register. Enabling or disabling does not affect the IntStatus register contents, only the propagation of the interrupt status to the CPU (via the NVIC).

The interrupts, either of individual frames or of the whole list, are a good means of communication between the DMA manager and the device driver, triggering the device driver to inspect the status words of descriptors that have been processed.

Transmit example

Figure 26 illustrates the transmit process in an example transmitting uses a frame header of 8 bytes and a frame payload of 12 bytes.

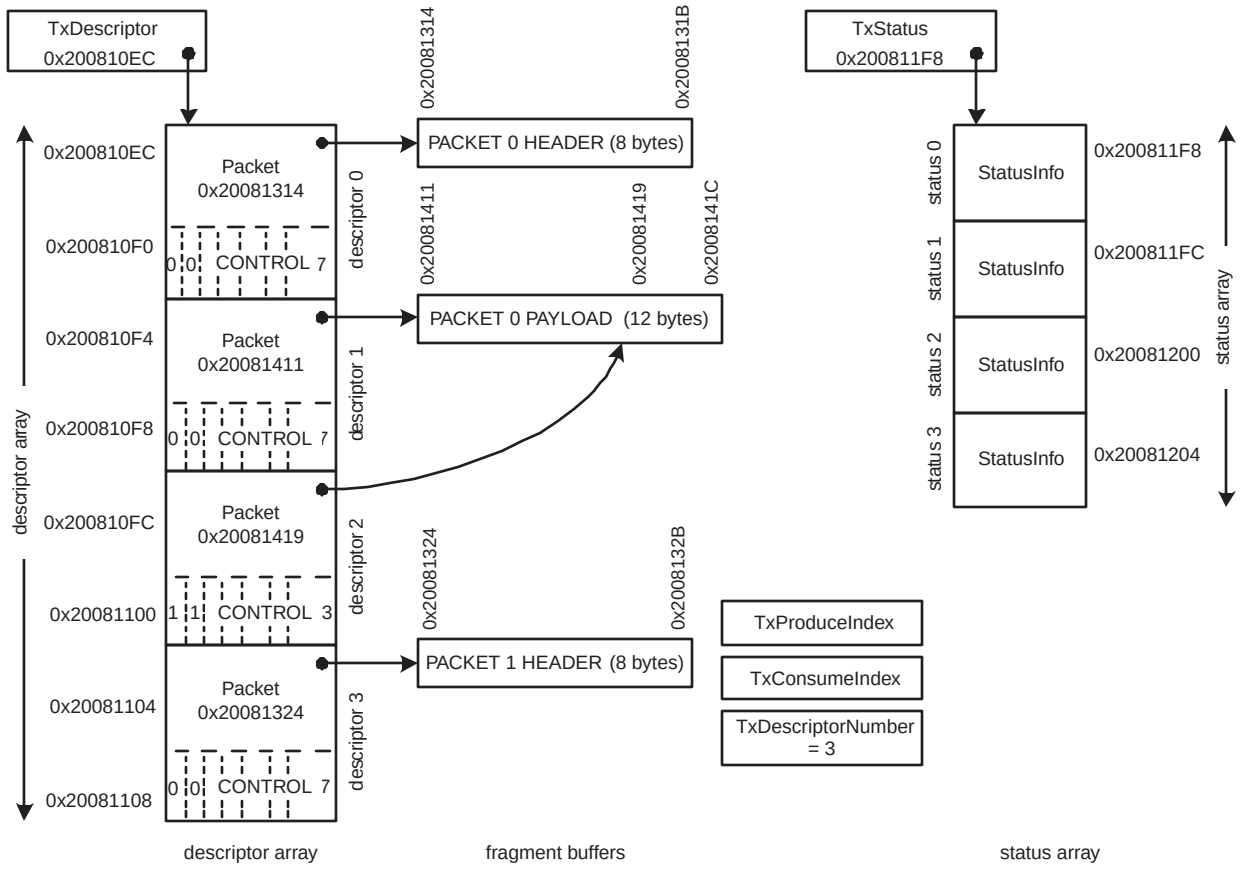


Fig 26. Transmit example memory and registers

After reset the values of the DMA registers will be zero. During initialization the device driver will allocate the descriptor and status array in memory. In this example, an array of four descriptors is allocated; the array is 4x2x4 bytes and aligned on a 4 byte address boundary. Since the number of descriptors matches the number of statuses the status array consists of four elements; the array is 4x1x4 bytes and aligned on a 4 byte address boundary. The device driver writes the base address of the descriptor array (0x2008 10EC) to the TxDescriptor register and the base address of the status array (0x2008 11F8) to the TxStatus register. The device driver writes the number of descriptors and statuses minus 1(3) to the TxDescriptorNumber register. The descriptors and statuses in the arrays need not be initialized, yet.

At this point, the transmit data path may be enabled by setting the TxEnable bit in the Command register. If the transmit data path is enabled while there are no further frames to send the TxFinishedInt interrupt flag will be set. To reduce the processor interrupt load only the desired interrupts can be enabled by setting the relevant bits in the IntEnable register.

Now suppose application software wants to transmit a frame of 12 bytes using a TCP/IP protocol (in real applications frames will be larger than 12 bytes). The TCP/IP stack will add a header to the frame. The frame header need not be immediately in front of the payload data in memory. The device driver can program the Tx DMA to collect header and payload data. To do so, the device driver will program the first descriptor to point at the

frame header; the Last flag in the descriptor will be set to false/0 to indicate a multi-fragment transmission. The device driver will program the next descriptor to point at the actual payload data. The maximum size of a payload buffer is 2 kB so a single descriptor suffices to describe the payload buffer. For the sake of the example though the payload is distributed across two descriptors. After the first descriptor in the array describing the header, the second descriptor in the array describes the initial 8 bytes of the payload; the third descriptor in the array describes the remaining 4 bytes of the frame. In the third descriptor the Last bit in the Control word is set to true/1 to indicate it is the last descriptor in the frame. In this example the Interrupt bit in the descriptor Control field is set in the last fragment of the frame in order to trigger an interrupt after the transmission completed. The Size field in the descriptor's Control word is set to the number of bytes in the fragment buffer, -1 encoded.

Note that in real device drivers, the payload will typically only be split across multiple descriptors if it is more than 2 kB. Also note that transmission payload data is forwarded to the hardware without the device driver copying it (zero copy device driver).

After setting up the descriptors for the transaction the device driver increments the TxProduceIndex register by 3 since three descriptors have been programmed. If the transmit data path was not enabled during initialization the device driver needs to enable the data path now.

If the transmit data path is enabled the Ethernet block will start transmitting the frame as soon as it detects the TxProduceIndex is not equal to TxConsumeIndex - both were zero after reset. The Tx DMA will start reading the descriptors from memory. The memory system will return the descriptors and the Ethernet block will accept them one by one while reading the transmit data fragments.

As soon as transmission read data is returned from memory, the Ethernet block will try to start transmission on the Ethernet connection via the MII/RMII interface.

After transmitting each fragment of the frame the Tx DMA will write the status of the fragment's transmission. Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx DMA manager. The status for the last fragment in the frame will only be written after the transmission has completed on the Ethernet connection.

Since the Interrupt bit in the descriptor of the last fragment is set, after committing the status of the last fragment to memory the Ethernet block will trigger a TxDoneInt interrupt, which triggers the device driver to inspect the status information.

In this example the device driver cannot add new descriptors as long as the Ethernet block has not incremented the TxConsumeIndex because the descriptor array is full (even though one descriptor is not programmed yet). Only after the hardware commits the status for the first fragment to memory and the TxConsumeIndex is set to 1 by the DMA manager can the device driver program the next (the fourth) descriptor. The fourth descriptor can already be programmed before completely transmitting the first frame.

In this example the hardware adds the CRC to the frame. If the device driver software adds the CRC, the CRC trailer can be considered another frame fragment which can be added by doing another gather DMA.

Each data byte is transmitted across the MII interface as two 4-bit values or the RMII interface as four 2-bit values. The Ethernet block adds the preamble, frame delimiter leader, and the CRC trailer if hardware CRC is enabled. Once transmission on the MII/RMII interface commences the transmission cannot be interrupted without generating an underrun error, which is why descriptors and data read commands are issued as soon as possible and pipelined.

Using an MII PHY, the data communication between the Ethernet block and the PHY is done at a 25 MHz rate. With an RMII PHY, the data communication between the Ethernet block and the PHY is at a 50 MHz rate. In 10 Mbps mode data will only be transmitted once every 10 clock cycles.

10.13.4 Receive process

This section outlines the receive process including the activities in the device driver software.

Device driver sets up descriptors

After initializing the receive descriptor and status arrays to receive frames from the Ethernet connection, the receive data path should be enabled in the MAC1 register and the Control register.

During initialization, each Packet pointer in the descriptors is set to point to a data fragment buffer. The size of the buffer is stored in the Size bits of the Control field of the descriptor. Additionally, the Control field in the descriptor has an Interrupt bit. The Interrupt bit allows generation of an interrupt after a fragment buffer has been filled and its status has been committed to memory.

After the initialization and enabling of the receive data path, all descriptors are owned by the receive hardware and should not be modified by the software unless hardware hands over the descriptor by incrementing the RxProduceIndex, indicating that a frame has been received. The device driver is allowed to modify the descriptors after a (soft) reset of the receive data path.

Rx DMA manager reads Rx descriptor arrays

When the RxEnable bit in the Command register is set, the Rx DMA manager reads the descriptors from memory at the address determined by RxDescriptor and RxProduceIndex. The Ethernet block will start reading descriptors even before actual receive data arrives on the MII/RMII interface (descriptor prefetching). The block size of the descriptors to be read is determined by the total number of descriptors owned by the hardware: RxConsumeIndex - RxProduceIndex - 1. Block transferring of descriptors minimizes memory load. Read data returned from memory is buffered and consumed as needed.

RX DMA manager receives data

After reading the descriptor, the receive DMA engine waits for the MAC to return receive data from the MII/RMII interface that passes the receive filter. Receive frames that do not match the filtering criteria are not passed to memory. Once a frame passes the receive filter, the data is written in the fragment buffer associated with the descriptor. The Rx DMA does not write beyond the size of the buffer. When a frame is received that is larger than a descriptor's fragment buffer, the frame will be written to multiple fragment buffers of

consecutive descriptors. In the case of a multi-fragment reception, all but the last fragment in the frame will return a status where the LastFrag bit is set to 0. Only on the last fragment of a frame the LastFrag bit in the status will be set to 1. If a fragment buffer is the last of a frame, the buffer may not be filled completely. The first receive data of the next frame will be written to the fragment buffer of the next descriptor.

After receiving a fragment, the Rx DMA manager writes status information back to the StatusInfo and StatusHashCRC words of the status. The Ethernet block writes the size in bytes of a descriptor's fragment buffer in the RxSize field of the Status word. The value of the RxProduceIndex is only updated after the fragment data and the fragment status information has been committed to memory, which is checked by an internal tag protocol in the memory interface. The Rx DMA manager continues to receive frames until the descriptor array is full. If the descriptor array is full, the Ethernet hardware will set the RxFinishedInt bit of the IntStatus register. The receive data path will still be enabled. If the receive descriptor array is full any new receive data will generate an overflow error and interrupt.

Update ProduceIndex

Each time the Rx DMA manager commits a data fragment and the associated status word to memory, it completes the reception of a descriptor and increments the RxProduceIndex (taking wrap around into account) in order to hand the descriptor back to the device driver software. Software can re-use the descriptor for new receptions by handing it back to hardware when the receive data has been processed.

The device driver software can keep track of the progress of the DMA manager by reading the RxProduceIndex register to see how far along the receive process is. When the Rx descriptor array is emptied completely, the RxProduceIndex retains its last value.

Write reception status

After the frame has been received from the MII/RMII bus, the StatusInfo and StatusHashCRC words of the frame descriptor are updated by the DMA manager.

If the descriptor is for the last fragment of a frame (or for the whole frame if there are no fragments), then depending on the success or failure of the frame reception, error flags (Error, NoDescriptor, Overrun, AlignmentError, RangeError, LengthError, SymbolError, or CRCError) are set in StatusInfo. The RxSize field is set to the number of bytes actually written to the fragment buffer, -1 encoded. For fragments not being the last in the frame the RxSize will match the size of the buffer. The hash CRCs of the destination and source addresses of a packet are calculated once for all the fragments belonging to the same packet and then stored in every StatusHashCRC word of the statuses associated with the corresponding fragments. If the reception reports an error, any remaining data in the receive frame is discarded and the LastFrag bit will be set in the receive status field, so the error flags in all but the last fragment of a frame will always be 0.

The status of the last received frame can also be inspected by reading the RSV register. The register does not report statuses on a fragment basis and does not store information of previously received frames. RSV is provided primarily for debug purposes, because the communication between driver software and the Ethernet block takes place through the frame descriptors.

Reception error handling

When an error occurs during the receive process, the Rx DMA manager will report the error via the receive StatusInfo written in the Status array and the IntStatus interrupt status register.

The receive process can generate several types of errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError, Overrun, and NoDescriptor. All have corresponding bits in the receive StatusInfo. In addition to the separate bits in the StatusInfo, AlignmentError, RangeError, LengthError, SymbolError, and CRCError are ORed together into the Error bit of the StatusInfo. Errors are also propagated to the IntStatus register; the RxError bit in the IntStatus register is set if there is an AlignmentError, RangeError, LengthError, SymbolError, CRCError, or NoDescriptor error; nonfatal overrun errors are reported in the RxError bit of the IntStatus register; fatal Overrun errors are report in the RxOverrun bit of the IntStatus register. On fatal overrun errors, the Rx data path needs to be soft reset by setting the RxReset bit in the Command register.

Overrun errors can have three causes:

- In the case of a multi-fragment reception, the next descriptor may be missing. In this case the NoDescriptor field is set in the status word of the previous descriptor and the RxError in the IntStatus register is set. This error is nonfatal.
- The data flow on the receiver data interface stalls, corrupting the packet. In this case the overrun bit in the status word is set and the RxError bit in the IntStatus register is set. This error is nonfatal.
- The flow of reception statuses stalls and a new status has to be written while a previous status still waits to be transferred across the memory interface. This error will corrupt the hardware state and requires the hardware to be soft reset. The error is detected and sets the Overrun bit in the IntStatus register.

The first overrun situation will result in an incomplete frame with a NoDescriptor status and the RxError bit in IntStatus set. Software should discard the partially received frame. In the second overrun situation the frame data will be corrupt which results in the Overrun status bit being set in the Status word while the IntError interrupt bit is set. In the third case receive errors cannot be reported in the receiver Status arrays which corrupts the hardware state; the errors will still be reported in the IntStatus register's Overrun bit. The RxReset bit in the Command register should be used to soft reset the hardware.

Device drivers should catch the above receive errors and take action.

Receive triggers interrupts

The receive data path can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set, the Rx DMA will set the RxDoneInt bit in the IntStatus register after receiving a fragment and committing the associated data and status to memory. Even if a descriptor (fragment) is not the last in a multi-fragment frame, the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor array is full while the Ethernet hardware is enabled, the hardware will set the RxFinishedInt bit of the IntStatus register.
- If the AHB interface does not consume receive statuses at a sufficiently high bandwidth, the receive status process may overrun, in which case the RxOverrun bit will be set in the IntStatus register.

- If there is a receive error (AlignmentError, RangeError, LengthError, SymbolError, or CRCError), or a multi-fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments, or if a nonfatal data Overrun occurred, the hardware will set the RxErrorInt bit of the IntStatus register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the IntEnable register. Enabling or disabling does not affect the IntStatus register contents, only the propagation of the interrupt status to the CPU (via the NVIC).

The interrupts, either of individual frames or of the whole list, are a good means of communication between the DMA manager and the device driver, triggering the device driver to inspect the status words of descriptors that have been processed.

Device driver processes receive data

As a response to status (e.g. RxDoneInt) interrupts or polling of the RxProduceIndex, the device driver can read the descriptors that have been handed over to it by the hardware (RxProduceIndex - RxConsumeIndex). The device driver should inspect the status words in the status array to check for multi-fragment receptions and receive errors.

The device driver can forward receive data and status to upper software layers. After processing of data and status, the descriptors, statuses and data buffers may be recycled and handed back to hardware by incrementing the RxConsumeIndex.

Receive example

Figure 27 illustrates the receive process in an example receiving a frame of 19 bytes.

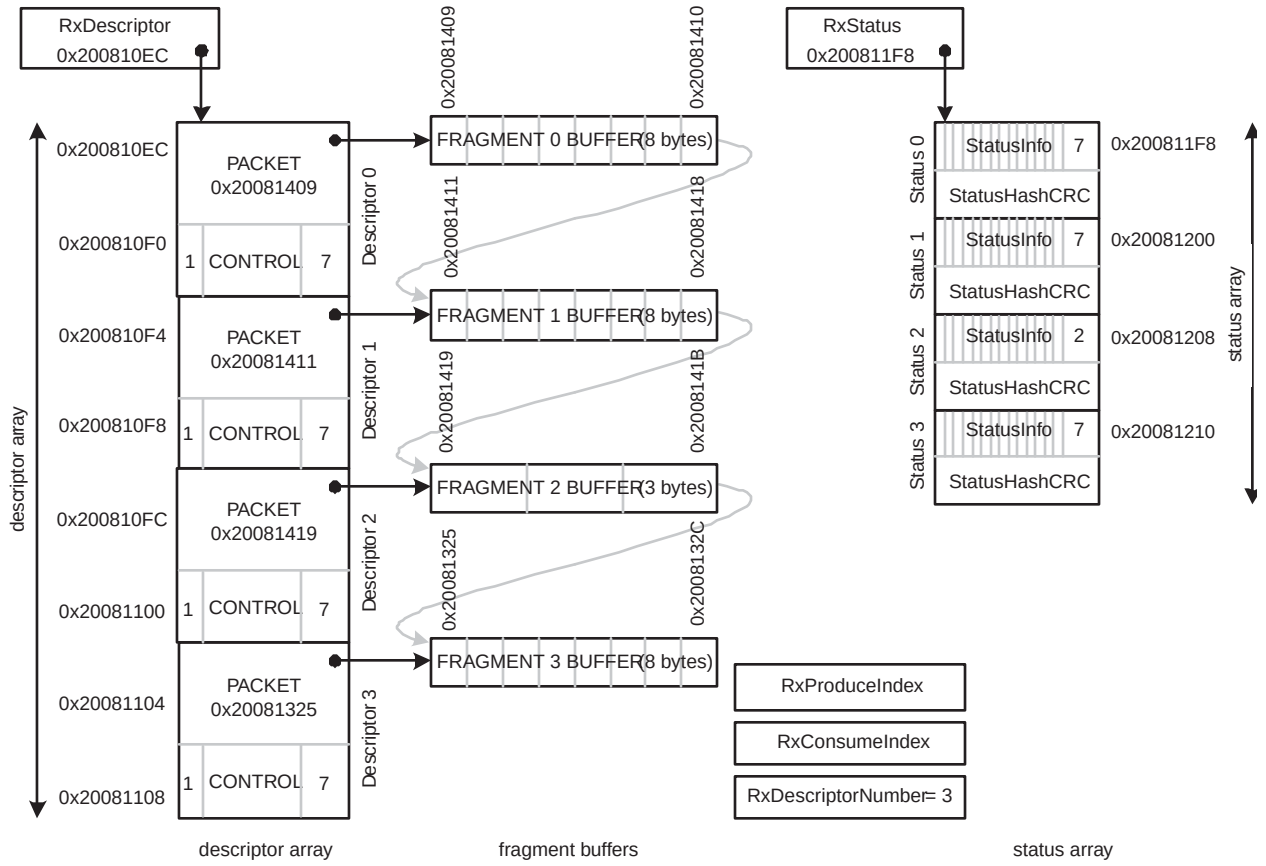


Fig 27. Receive Example Memory and Registers

After reset, the values of the DMA registers will be zero. During initialization, the device driver will allocate the descriptor and status array in memory. In this example, an array of four descriptors is allocated; the array is 4x2x4 bytes and aligned on a 4 byte address boundary. Since the number of descriptors matches the number of statuses, the status array consists of four elements; the array is 4x2x4 bytes and aligned on a 8 byte address boundary. The device driver writes the base address of the descriptor array (0x2008 10EC) in the RxDescriptor register, and the base address of the status array (0x2008 11F8) in the RxStatus register. The device driver writes the number of descriptors and statuses minus 1 (3) in the RxDescriptorNumber register. The descriptors and statuses in the arrays need not be initialized yet.

After allocating the descriptors, a fragment buffer needs to be allocated for each of the descriptors. Each fragment buffer can be between 1 byte and 2 k bytes. The base address of the fragment buffer is stored in the Packet field of the descriptors. The number of bytes in the fragment buffer is stored in the Size field of the descriptor Control word. The Interrupt field in the Control word of the descriptor can be set to generate an interrupt as soon as the descriptor has been filled by the receive process. In this example the fragment buffers are 8 bytes, so the value of the Size field in the Control word of the descriptor is set to 7. Note that in this example, the fragment buffers are actually a

continuous memory space; even when a frame is distributed over multiple fragments it will typically be in a linear, continuous memory space; when the descriptors wrap at the end of the descriptor array the frame will not be in a continuous memory space.

The device driver should enable the receive process by writing a 1 to the RxEnable bit of the Command register, after which the MAC needs to be enabled by writing a 1 to the 'RECEIVE ENABLE' bit of the MAC1 configuration register. The Ethernet block will now start receiving Ethernet frames. To reduce the processor interrupt load, some interrupts can be disabled by setting the relevant bits in the IntEnable register.

After the Rx DMA manager is enabled, it will start issuing descriptor read commands. In this example the number of descriptors is 4. Initially the RxProduceIndex and RxConsumeIndex are 0. Since the descriptor array is considered full if RxProduceIndex == RxConsumeIndex - 1, the Rx DMA manager can only read (RxConsumeIndex - RxProduceIndex - 1 =) 3 descriptors; note the wrapping.

After enabling the receive function in the MAC, data reception will begin starting at the next frame i.e. if the receive function is enabled while the MII/RMII interface is halfway through receiving a frame, the frame will be discarded and reception will start at the next frame. The Ethernet block will strip the preamble and start of frame delimiter from the frame. If the frame passes the receive filtering, the Rx DMA manager will start writing the frame to the first fragment buffer.

Suppose the frame is 19 bytes long. Due to the buffer sizes specified in this example, the frame will be distributed over three fragment buffers. After writing the initial 8 bytes in the first fragment buffer, the status for the first fragment buffer will be written and the Rx DMA will continue filling the second fragment buffer. Since this is a multi-fragment receive, the status of the first fragment will have a 0 for the LastFrag bit in the StatusInfo word; the RxSize field will be set to 7 (8, -1 encoded). After writing the 8 bytes in the second fragment the Rx DMA will continue writing the third fragment. The status of the second fragment will be like the status of the first fragment: LastFrag = 0, RxSize = 7. After writing the three bytes in the third fragment buffer, the end of the frame has been reached and the status of the third fragment is written. The third fragment's status will have the LastFrag bit set to 1 and the RxSize equal to 2 (3, -1 encoded).

The next frame received from the MII/RMII interface will be written to the fourth fragment buffer i.e. five bytes of the third buffer will be unused.

The Rx DMA manager uses an internal tag protocol in the memory interface to check that the receive data and status have been committed to memory. After the status of the fragments are committed to memory, an RxDoneInt interrupt will be triggered, which activates the device driver to inspect the status information. In this example, all descriptors have the Interrupt bit set in the Control word i.e. all descriptors will generate an interrupt after committing data and status to memory.

In this example the receive function cannot read new descriptors as long as the device driver does not increment the RxConsumeIndex, because the descriptor array is full (even though one descriptor is not programmed yet). Only after the device driver has forwarded the receive data to application software, and after the device driver has updated the RxConsumeIndex by incrementing it, will the Ethernet block can continue reading descriptors and receive data. The device driver will probably increment the RxConsumeIndex by 3, since the driver will forward the complete frame consisting of three fragments to the application, and hence free up three descriptors at the same time.

Each pair of nibbles transferred on the MII interface (or four pairs of bits for RMII) are transferred as a byte on the data write interface after being delayed by 128 or 136 cycles for filtering by the receive filter and buffer modules. The Ethernet block removes preamble, frame start delimiter, and CRC from the data and checks the CRC. To limit the buffer NoDescriptor error probability, three descriptors are buffered. The value of the RxProduceIndex is only updated after status information has been committed to memory, which is checked by an internal tag protocol in the memory interface. The software device driver will process the receive data, after which the device driver will update the RxConsumeIndex.

For an RMII PHY the data between the Ethernet block and the PHY is communicated at half the data-width and twice the clock frequency (50 MHz).

10.13.5 Transmission retry

If a collision on the Ethernet occurs, it usually takes place during the collision window spanning the first 64 bytes of a frame. If collision is detected, the Ethernet block will retry the transmission. For this purpose, the first 64 bytes of a frame are buffered, so that this data can be used during the retry. A transmission retry within the first 64 bytes in a frame is fully transparent to the application and device driver software.

When a collision occurs outside of the 64 byte collision window, a LateCollision error is triggered, and the transmission is aborted. After a LateCollision error, the remaining data in the transmit frame will be discarded. The Ethernet block will set the Error and LateCollision bits in the frame's status fields. The TxError bit in the IntStatus register will be set. If the corresponding bit in the IntEnable register is set, the TxError bit in the IntStatus register will be propagated to the CPU (via the NVIC). The device driver software should catch the interrupt and take appropriate actions.

The 'RETRANSMISSION MAXIMUM' field of the CLRT register can be used to configure the maximum number of retries before aborting the transmission.

10.13.6 Status hash CRC calculations

For each received frame, the Ethernet block is able to detect the destination address and source address and from them calculate the corresponding hash CRCs. To perform the computation, the Ethernet block features two internal blocks: one is a controller synchronized with the beginning and the end of each frame, the second block is the CRC calculator.

When a new frame is detected, internal signaling notifies the controller.The controller starts counting the incoming bytes of the frame, which correspond to the destination address bytes. When the sixth (and last) byte is counted, the controller notifies the calculator to store the corresponding 32-bit CRC into a first inner register. Then the controller repeats counting the next incoming bytes, in order to get synchronized with the source address. When the last byte of the source address is encountered, the controller again notifies the CRC calculator, which freezes until the next new frame. When the calculator receives this second notification, it stores the present 32-bit CRC into a second inner register. Then the CRCs remain frozen in their own registers until new notifications arise.

The destination address and source address hash CRCs being written in the StatusHashCRC word are the nine most significant bits of the 32-bit CRCs as calculated by the CRC calculator.

10.13.7 Duplex modes

The Ethernet block can operate in full duplex and half duplex mode. Half or full duplex mode needs to be configured by the device driver software during initialization.

For a full duplex connection the FullDuplex bit of the Command register needs to be set to 1 and the FULL-DUPLEX bit of the MAC2 configuration register needs to be set to 1; for half duplex the same bits need to be set to 0.

10.13.8 IEE 802.3/Clause 31 flow control

Overview

For full duplex connections, the Ethernet block supports IEEE 802.3/clause 31 flow control using pause frames. This type of flow control may be used in full-duplex point-to-point connections. Flow control allows a receiver to stall a transmitter e.g. when the receive buffers are (almost) full. For this purpose, the receiving side sends a pause frame to the transmitting side.

Pause frames use units of 512 bit times corresponding to 128 rx_clk/tx_clk cycles.

Receive flow control

In full-duplex mode, the Ethernet block will suspend its transmissions when the it receives a pause frame. Rx flow control is initiated by the receiving side of the transmission. It is enabled by setting the ‘RX FLOW CONTROL’ bit in the MAC1 configuration register. If the RX FLOW CONTROL’ bit is zero, then the Ethernet block ignores received pause control frames. When a pause frame is received on the Rx side of the Ethernet block, transmission on the Tx side will be interrupted after the currently transmitting frame has completed, for an amount of time as indicated in the received pause frame. The transmit data path will stop transmitting data for the number of 512 bit slot times encoded in the pause-timer field of the received pause control frame.

By default the received pause control frames are not forwarded to the device driver. To forward the receive flow control frames to the device driver, set the ‘PASS ALL RECEIVE FRAMES’ bit in the MAC1 configuration register.

Transmit flow control

If case device drivers need to stall the receive data e.g. because software buffers are full, the Ethernet block can transmit pause control frames. Transmit flow control needs to be initiated by the device driver software; there is no IEEE 802.3/31 flow control initiated by hardware, such as the DMA managers.

With software flow control, the device driver can detect a situation in which the process of receiving frames needs to be interrupted by sending out Tx pause frames. Note that due to Ethernet delays, a few frames can still be received before the flow control takes effect and the receive stream stops.

Transmit flow control is activated by writing 1 to the TxFlowControl bit of the Command register. When the Ethernet block operates in full duplex mode, this will result in transmission of IEEE 802.3/31 pause frames. The flow control continues until a 0 is written to TxFlowControl bit of the Command register.

If the MAC is operating in full-duplex mode, then setting the TxFlowControl bit of the Command register will start a pause frame transmission. The value inserted into the pause-timer value field of transmitted pause frames is programmed via the PauseTimer[15:0] bits in the FlowControlCounter register. When the TxFlowControl bit is de-asserted, another pause frame having a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission.

When flow control be in force for an extended time, a sequence of pause frames must be transmitted. This is supported with a mirror counter mechanism. To enable mirror counting, a nonzero value is written to the MirrorCounter[15:0] bits in the FlowControlCounter register. When the TxFlowControl bit is asserted, a pause frame is transmitted. After sending the pause frame, an internal mirror counter is initialized to zero. The internal mirror counter starts incrementing one every 512 bit-slot times. When the internal mirror counter reaches the MirrorCounter value, another pause frame is transmitted with pause-timer value equal to the PauseTimer field from the FlowControlCounter register, the internal mirror counter is reset to zero and restarts counting. The register MirrorCounter[15:0] is usually set to a smaller value than register PauseTimer[15:0] to ensure an early expiration of the mirror counter, allowing time to send a new pause frame before the transmission on the other side can resume. By continuing to send pause frames before the transmitting side finishes counting the pause timer, the pause can be extended as long as TxFlowControl is asserted. This continues until TxFlowControl is de-asserted when a final pause frame having a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission. To disable the mirror counter mechanism, write the value 0 to MirrorCounter field in the FlowControlCounter register. When using the mirror counter mechanism, account for time-of-flight delays, frame transmission time, queuing delays, crystal frequency tolerances, and response time delays by programming the MirrorCounter conservatively, typically about 80% of the PauseTimer value.

If the software device driver sets the MirrorCounter field of the FlowControlCounter register to zero, the Ethernet block will only send one pause control frame. After sending the pause frame an internal pause counter is initialized at zero; the internal pause counter is incremented by one every 512 bit-slot times. Once the internal pause counter reaches the value of the PauseTimer register, the TxFlowControl bit in the Command register will be reset. The software device driver can poll the TxFlowControl bit to detect when the pause completes.

The value of the internal counter in the flow control module can be read out via the FlowControlStatus register. If the MirrorCounter is nonzero, the FlowControlStatus register will return the value of the internal mirror counter; if the MirrorCounter is zero the FlowControlStatus register will return the value of the internal pause counter value.

The device driver is allowed to dynamically modify the MirrorCounter register value and switch between zero MirrorCounter and nonzero MirrorCounter modes.

Transmit flow control is enabled via the 'TX FLOW CONTROL' bit in the MAC1 configuration register. If the 'TX FLOW CONTROL' bit is zero, then the MAC will not transmit pause control frames, software must not initiate pause frame transmissions, and the TxFlowControl bit in the Command register should be zero.

Transmit flow control example

Figure 28 illustrates the transmit flow control.

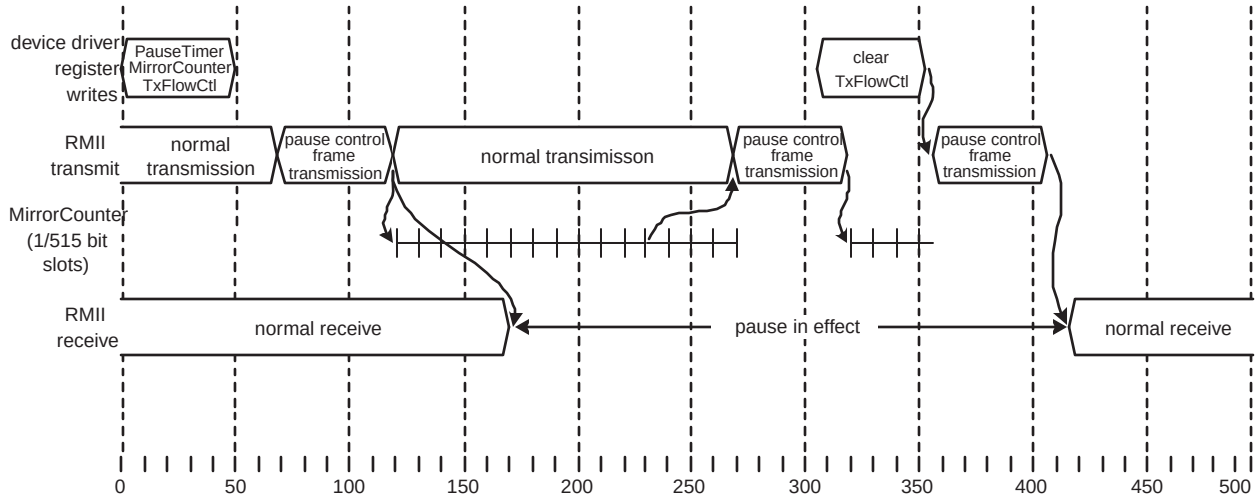


Fig 28. Transmit Flow Control

In this example, a frame is received while transmitting another frame (full duplex.) The device driver detects that some buffer might overrun and enables the transmit flow control by programming the PauseTimer and MirrorCounter fields of the FlowControlCounter register, after which it enables the transmit flow control by setting the TxFlowControl bit in the Command register.

As a response to the enabling of the flow control a pause control frame will be sent after the currently transmitting frame has been transmitted. When the pause frame transmission completes the internal mirror counter will start counting bit slots; as soon as the counter reaches the value in the MirrorCounter field another pause frame is transmitted. While counting the transmit data path will continue normal transmissions.

As soon as software disables transmit flow control a zero pause control frame is transmitted to resume the receive process.

10.13.9 Half-Duplex mode backpressure

When in half-duplex mode, backpressure can be generated to stall receive packets by sending continuous preamble that basically jams any other transmissions on the Ethernet medium. When the Ethernet block operates in half duplex mode, asserting the TxFlowControl bit in the Command register will result in applying continuous preamble on the Ethernet wire, effectively blocking traffic from any other Ethernet station on the same segment.

In half duplex mode, when the TxFlowControl bit goes high, continuous preamble is sent until TxFlowControl is de-asserted. If the medium is idle, the Ethernet block begins transmitting preamble, which raises carrier sense causing all other stations to defer. In the event the transmitting of preamble causes a collision, the backpressure 'rides through' the collision. The colliding station backs off and then defers to the backpressure. If during backpressure, the user wishes to send a frame, the backpressure is interrupted, the frame sent and then the backpressure resumed. If TxFlowControl is asserted for longer than 3.3 ms in 10 Mbps mode or 0.33 ms in 100 Mbps mode, backpressure will cease sending preamble for several byte times to avoid the jabber limit.

10.13.10 Receive filtering

Features of receive filtering

The Ethernet MAC has several receive packet filtering functions that can be configured from the software driver:

- Perfect address filter: allows packets with a perfectly matching station address to be identified and passed to the software driver.
- Hash table filter: allows imperfect filtering of packets based on the station address.
- Unicast/multicast/broadcast filtering: allows passing of all unicast, multicast, and/or broadcast packets.
- Magic packet filter: detection of magic packets to generate a Wake-on-LAN interrupt.

The filtering functions can be logically combined to create complex filtering functions. Furthermore, the Ethernet block can pass or reject runt packets smaller than 64 bytes; a promiscuous mode allows all packets to be passed to software.

Overview

The Ethernet block has the capability to filter out receive frames by analyzing the Ethernet destination address in the frame. This capability greatly reduces the load on the host system, because Ethernet frames that are addressed to other stations would otherwise need to be inspected and rejected by the device driver software, using up bandwidth, memory space, and host CPU time. Address filtering can be implemented using the perfect address filter or the (imperfect) hash filter. The latter produces a 6-bit hash code which can be used as an index into a 64 entry programmable hash table. Figure 29 depicts a functional view of the receive filter.

At the top of the diagram the Ethernet receive frame enters the filters. Each filter is controlled by signals from control registers; each filter produces a 'Ready' output and a 'Match' output. If 'Ready' is 0 then the Match value is 'don't care'; if a filter finishes filtering then it will assert its Ready output; if the filter finds a matching frame it will assert the Match output along with the Ready output. The results of the filters are combined by logic functions into a single RxAbort output. If the RxAbort output is asserted, the frame does not need to be received.

In order to reduce memory traffic, the receive data path has a buffer of 68 bytes. The Ethernet MAC will only start writing a frame to memory after 68 byte delays. If the RxAbort signal is asserted during the initial 68 bytes of the frame, the frame can be discarded and removed from the buffer and not stored to memory at all, not using up receive descriptors, etc. If the RxAbort signal is asserted after the initial 68 bytes in a frame (probably due to reception of a Magic Packet), part of the frame is already written to memory and the

Ethernet MAC will stop writing further data in the frame to memory; the FailFilter bit in the status word of the frame will be set to indicate that the software device driver can discard the frame immediately.

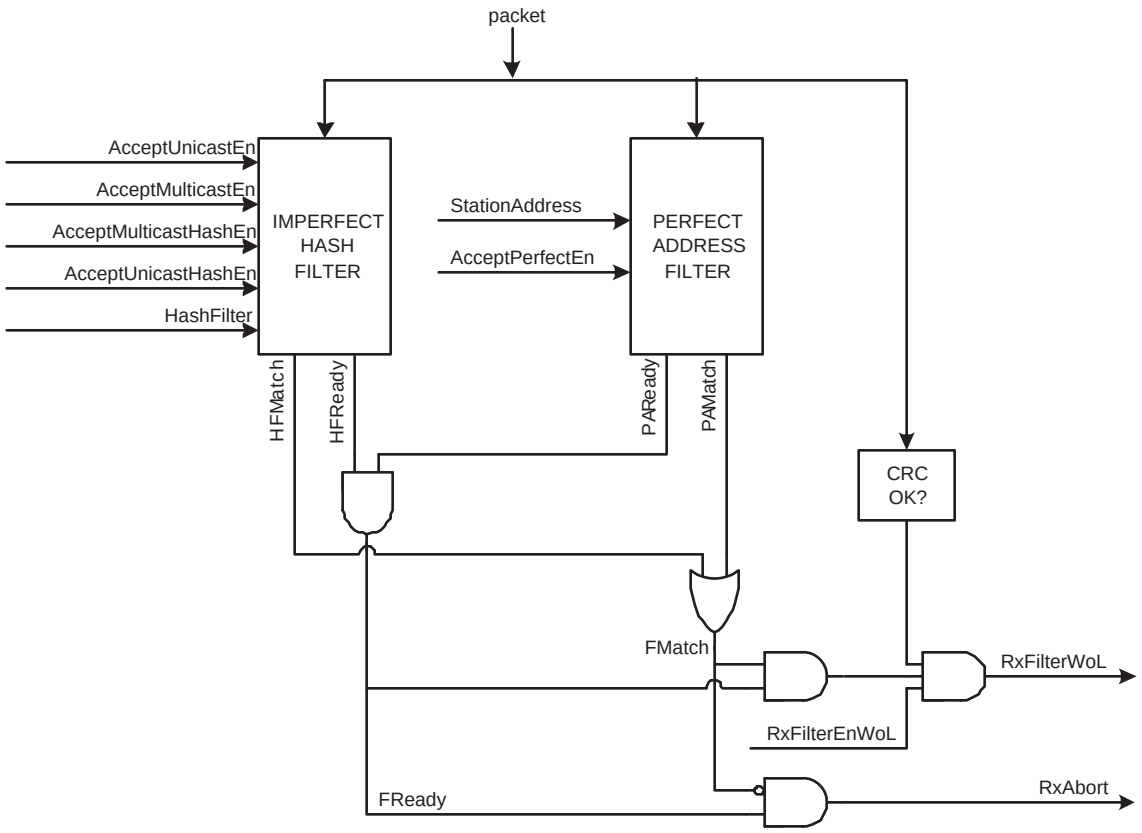


Fig 29. Receive filter block diagram

Unicast, broadcast and multicast

Generic filtering based on the type of frame (unicast, multicast or broadcast) can be programmed using the AcceptUnicastEn, AcceptMulticastEn, or AcceptBroadcastEn bits of the RxFilterCtrl register. Setting the AcceptUnicast, AcceptMulticast, and AcceptBroadcast bits causes all frames of types unicast, multicast and broadcast, respectively, to be accepted, ignoring the Ethernet destination address in the frame. To program promiscuous mode, i.e. to accept all frames, set all 3 bits to 1.

Perfect address match

When a frame with a unicast destination address is received, a perfect filter compares the destination address with the 6 byte station address programmed in the station address registers SA0, SA1, SA2. If the AcceptPerfectEn bit in the RxFilterCtrl register is set to 1, and the address matches, the frame is accepted.

Imperfect hash filtering

An imperfect filter is available, based on a hash mechanism. This filter applies a hash function to the destination address and uses the hash to access a table that indicates if the frame should be accepted. The advantage of this type of filter is that a small table can cover any possible address. The disadvantage is that the filtering is imperfect, i.e. sometimes frames are accepted that should have been discarded.

- Hash function:
  - The standard Ethernet cyclic redundancy check (CRC) function is calculated from the 6 byte destination address in the Ethernet frame (this CRC is calculated anyway as part of calculating the CRC of the whole frame), then bits [28:23] out of the 32-bit CRC result are taken to form the hash. The 6-bit hash is used to access the hash table: it is used as an index in the 64-bit HashFilter register that has been programmed with accept values. If the selected accept value is 1, the frame is accepted.
  - The device driver can initialize the hash filter table by writing to the registers HashFilterL and HashfilterH. HashFilterL contains bits 0 through 31 of the table and HashFilterH contains bit 32 through 63 of the table. So, hash value 0 corresponds to bit 0 of the HashfilterL register and hash value 63 corresponds to bit 31 of the HashFilterH register.
- Multicast and unicast
  - The imperfect hash filter can be applied to multicast addresses, by setting the AcceptMulticastHashEn bit in the RxFilter register to 1.
  - The same imperfect hash filter that is available for multicast addresses can also be used for unicast addresses. This is useful to be able to respond to a multitude of unicast addresses without enabling all unicast addresses. The hash filter can be applied to unicast addresses by setting the AcceptUnicastHashEn bit in the RxFilter register to 1.

Enabling and disabling filtering

The filters as defined in the sections above can be bypassed by setting the PassRxFilter bit in the Command register. When the PassRxFilter bit is set, all receive frames will be passed to memory. In this case the device driver software has to implement all filtering functionality in software. Setting the PassRxFilter bit does not affect the runt frame filtering as defined in the next section.

Runt frames

A frame with less than 64 bytes (or 68 bytes for VLAN frames) is shorter than the minimum Ethernet frame size and therefore considered erroneous; they might be collision fragments. The receive data path automatically filters and discards these runt frames without writing them to memory and using a receive descriptor.

When a runt frame has a correct CRC there is a possibility that it is intended to be useful. The device driver can receive the runt frames with correct CRC by setting the PassRuntFrame bit of the Command register to 1.

10.13.11 Power management

The Ethernet block supports power management by means of clock switching. All clocks in the Ethernet core can be switched off. If Wake-up on LAN is needed, the rx_clk should not be switched off.

10.13.12 Wake-up on LAN

Overview

The Ethernet block supports power management with remote wake-up over LAN. The host system can be powered down, even including part of the Ethernet block itself, while the Ethernet block continues to listen to packets on the LAN. Appropriately formed packets can be received and recognized by the Ethernet block and used to trigger the host system to wake up from its power-down state.

Wake-up of the system takes effect through an interrupt. When a wake-up event is detected, the WakeupInt bit in the IntStatus register is set. The interrupt status will trigger an interrupt if the corresponding WakeupIntEn bit in the IntEnable register is set. This interrupt should be used by system power management logic to wake up the system.

While in a power-down state the packet that generates a Wake-up on LAN event is lost.

There are two ways in which Ethernet packets can trigger wake-up events: generic Wake-up on LAN and Magic Packet. Magic Packet filtering uses an additional filter for Magic Packet detection. In both cases a Wake-up on LAN event is only triggered if the triggering packet has a valid CRC. Figure 29 shows the generation of the wake-up signal.

The RxFilterWoLStatus register can be read by the software to inspect the reason for a Wake-up event. Before going to power-down the power management software should clear the register by writing the RxFilterWolClear register.

NOTE: when entering in power-down mode, a receive frame might be not entirely stored into the Rx buffer. In this situation, after turning exiting power-down mode, the next receive frame is corrupted due to the data of the previous frame being added in front of the last received frame. Software drivers have to reset the receive data path just after exiting power-down mode.

The following subsections describe the two Wake-up on LAN mechanisms.

Filtering for WoL

The receive filter functionality can be used to generate Wake-up on LAN events. If the RxFilterEnWoL bit of the RxFilterCtrl register is set, the receive filter will set the WakeupInt bit of the IntStatus register if a frame is received that passes the filter. The interrupt will only be generated if the CRC of the frame is correct.

Magic Packet WoL

The Ethernet block supports wake-up using Magic Packet technology (see ‘Magic Packet technology’, Advanced Micro Devices). A Magic Packet is a specially formed packet solely intended for wake-up purposes. This packet can be received, analyzed and recognized by the Ethernet block and used to trigger a wake-up event.

A Magic Packet is a packet that contains in its data portion the station address repeated 16 times with no breaks or interruptions, preceded by 6 Magic Packet synchronization bytes with the value 0xFF. Other data may be surrounding the Magic Packet pattern in the data portion of the packet. The whole packet must be a well-formed Ethernet frame.

The magic packet detection unit analyzes the Ethernet packets, extracts the packet address and checks the payload for the Magic Packet pattern. The address from the packet is used for matching the pattern (not the address in the SA0/1/2 registers.) A magic packet only sets the wake-up interrupt status bit if the packet passes the receive filter as illustrated in Figure 29: the result of the receive filter is ANDed with the magic packet filter result to produce the result.

Magic Packet filtering is enabled by setting the MagicPacketEnWoL bit of the RxFilterCtrl register. Note that when doing Magic Packet WoL, the RxFilterEnWoL bit in the RxFilterCtrl register should be 0. Setting the RxFilterEnWoL bit to 1 would accept all packets for a matching address, not just the Magic Packets i.e. WoL using Magic Packets is more strict.

When a magic packet is detected, apart from the WakeupInt bit in the IntStatus register, the MagicPacketWoL bit is set in the RxFilterWoLStatus register. Software can reset the bit writing a 1 to the corresponding bit of the RxFilterWoLClear register.

Example: An example of a Magic Packet with station address 0x11 0x22 0x33 0x44 0x55 0x66 is the following (MISC indicates miscellaneous additional data bytes in the packet):

```
<DESTINATION> <SOURCE> <MISC>
FF FF FF FF FF FF
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
<MISC> <CRC>
```

10.13.13 Enabling and disabling receive and transmit

Enabling and disabling reception

After reset, the receive function of the Ethernet block is disabled. The receive function can be enabled by the device driver setting the RxEnable bit in the Command register and the “RECEIVE ENABLE’ bit in the MAC1 configuration register (in that order).

The status of the receive data path can be monitored by the device driver by reading the RxStatus bit of the Status register. Figure 30 illustrates the state machine for the generation of the RxStatus bit.

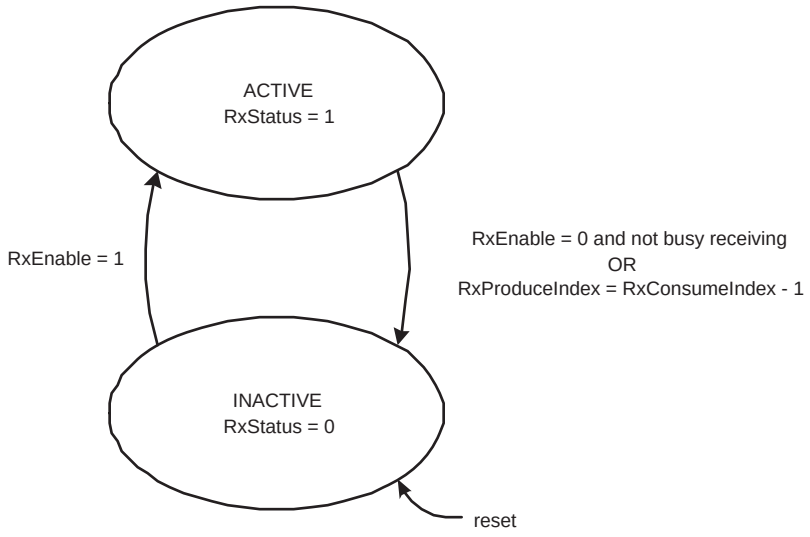


Fig 30. Receive Active/Inactive state machine

After a reset, the state machine is in the INACTIVE state. As soon as the RxEnable bit is set in the Command register, the state machine transitions to the ACTIVE state. As soon as the RxEnable bit is cleared, the state machine returns to the INACTIVE state. If the receive data path is busy receiving a packet while the receive data path gets disabled, the packet will be received completely, stored to memory along with its status before returning to the INACTIVE state. Also if the Receive descriptor array is full, the state machine will return to the INACTIVE state.

For the state machine in Figure 30, a soft reset is like a hardware reset assertion, i.e. after a soft reset the receive data path is inactive until the data path is re-enabled.

Enabling and disabling transmission

After reset, the transmit function of the Ethernet block is disabled. The Tx transmit data path can be enabled by the device driver setting the TxEnable bit in the Command register to 1.

The status of the transmit data paths can be monitored by the device driver reading the TxStatus bit of the Status register. Figure 31 illustrates the state machine for the generation of the TxStatus bit.

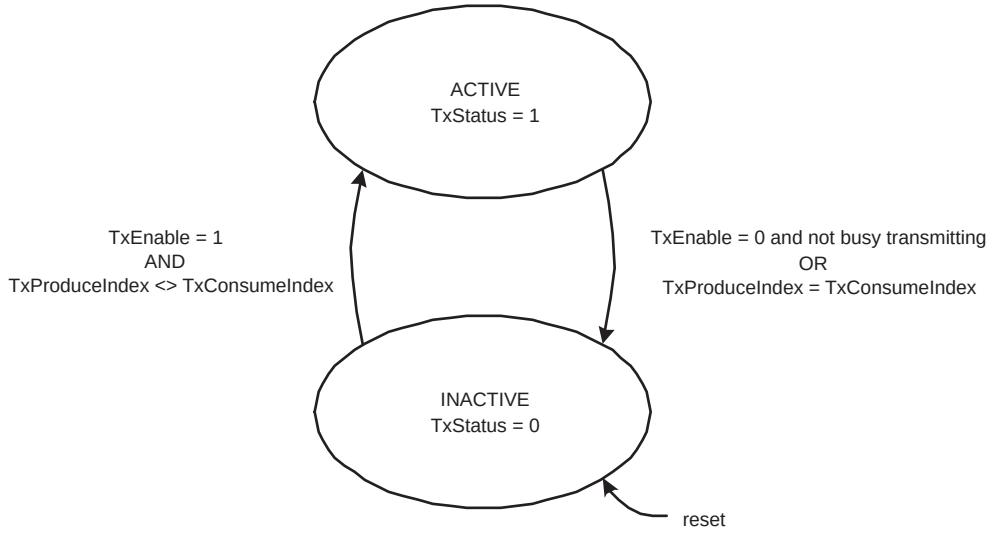


Fig 31. Transmit Active/Inactive state machine

After reset, the state machine is in the INACTIVE state. As soon as the TxEnable bit is set in the Command register and the Produce and Consume indices are not equal, the state machine transitions to the ACTIVE state. As soon as the TxEnable bit is cleared and the transmit data path has completed all pending transmissions, including committing the transmission status to memory, the state machine returns to the INACTIVE state. The state machine will also return to the INACTIVE state if the Produce and Consume indices are equal again i.e. all frames have been transmitted.

For the state machine in Figure 31, a soft reset is like a hardware reset assertion, i.e. after a soft reset the transmit data path is inactive until the data path is re-enabled.

10.13.14 Transmission padding and CRC

In the case of a frame of less than 60 bytes (or 64 bytes for VLAN frames), the Ethernet block can pad the frame to 64 or 68 bytes including a 4 bytes CRC Frame Check Sequence (FCS). Padding is affected by the value of the ‘AUTO DETECT PAD ENABLE’ (ADPEN), ‘VLAN PAD ENABLE’ (VLPEN) and ‘PAD/CRC ENABLE’ (PADEN) bits of the MAC2 configuration register, as well as the Override and Pad bits from the transmit descriptor Control word. CRC generation is affected by the ‘CRC ENABLE’ (CRCE) and ‘DELAYED CRC’ (DCRC) bits of the MAC2 configuration register, and the Override and CRC bits from the transmit descriptor Control word.

The effective pad enable (EPADEN) is equal to the ‘PAD/CRC ENABLE’ bit from the MAC2 register if the Override bit in the descriptor is 0. If the Override bit is 1, then EPADEN will be taken from the descriptor Pad bit. Likewise the effective CRC enable (ECRCE) equals CRCE if the Override bit is 0, otherwise it equal the CRC bit from the descriptor.

If padding is required and enabled, a CRC will always be appended to the padded frames. A CRC will only be appended to the non-padded frames if ECRCE is set.

If EPADEN is 0, the frame will not be padded and no CRC will be added unless ECRCE is set.

If EPADEN is 1, then small frames will be padded and a CRC will always be added to the padded frames. In this case if ADPEN and VLPEN are both 0, then the frames will be padded to 60 bytes and a CRC will be added creating 64 bytes frames; if VLPEN is 1, the frames will be padded to 64 bytes and a CRC will be added creating 68 bytes frames; if ADPEN is 1, while VLPEN is 0 VLAN frames will be padded to 64 bytes, non VLAN frames will be padded to 60 bytes, and a CRC will be added to padded frames, creating 64 or 68 bytes padded frames.

If CRC generation is enabled, CRC generation can be delayed by four bytes by setting the DELAYED CRC bit in the MAC2 register, in order to skip proprietary header information.

10.13.15 Huge frames and frame length checking

The ‘HUGE FRAME ENABLE’ bit in the MAC2 configuration register can be set to 1 to enable transmission and reception of frames of any length. Huge frame transmission can be enabled on a per frame basis by setting the Override and Huge bits in the transmit descriptor Control word.

When enabling huge frames, the Ethernet block will not check frame lengths and report frame length errors (RangeError and LengthError). If huge frames are enabled, the received byte count in the RSV register may be invalid because the frame may exceed the maximum size; the RxSize fields from the receive status arrays will be valid.

Frame lengths are checked by comparing the length/type field of the frame to the actual number of bytes in the frame. A LengthError is reported by setting the corresponding bit in the receive StatusInfo word.

The MAXF register allows the device driver to specify the maximum number of bytes in a frame. The Ethernet block will compare the actual receive frame to the MAXF value and report a RangeError in the receive StatusInfo word if the frame is larger.

10.13.16 Statistics counters

Generally, Ethernet applications maintain many counters that track Ethernet traffic statistics. There are a number of standards specifying such counters, such as IEEE std 802.3 / clause 30. Other standards are RFC 2665 and RFC 2233.

The approach taken here is that by default all counters are implemented in software. With the help of the StatusInfo field in frame statuses, many of the important statistics events listed in the standards can be counted by software.

10.13.17 MAC status vectors

Transmit and receive status information as detected by the MAC are available in registers TSV0, TSV1 and RSV so that software can poll them. These registers are normally of limited use because the communication between driver software and the Ethernet block takes place primarily through frame descriptors. Statistical events can be counted by software in the device driver. However, for debug purposes the transmit and receive status vectors are made visible. They are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

10.13.18 Reset

The Ethernet block has a hard reset input which is connected to the chip reset, as well as several soft resets which can be activated by setting the appropriate bits in registers. All registers in the Ethernet block have a value of 0 after a hard reset, unless otherwise specified.

Hard reset

After a hard reset, all registers will be set to their default value.

Soft reset

Parts of the Ethernet block can be soft reset by setting bits in the Command register and the MAC1 configuration register.The MAC1 register has six different reset bits:

- SOFT RESET: Setting this bit will put all modules in the MAC in reset, except for the MAC registers (at addresses 0x000 to 0x0FC). The value of the soft reset after a hardware reset assertion is 1, i.e. the soft reset needs to be cleared after a hardware reset.
- SIMULATION RESET: Resets the random number generator in the Transmit Function. The value after a hardware reset assertion is 0.
- RESET MCS/Rx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the receive function in the MAC. The value after a hardware reset assertion is 0.
- RESET Rx: Setting this bit will reset the receive function in the MAC. The value after a hardware reset assertion is 0.
- RESET MCS/Tx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the transmit function in the MAC. The value after a hardware reset assertion is 0.
- RESET Tx: Setting this bit will reset the transmit function of the MAC. The value after a hardware reset assertion is 0.

The above reset bits must be cleared by software.

The Command register has three different reset bits:

- TxReset: Writing a ‘1’ to the TxReset bit will reset the transmit data path, excluding the MAC portions, including all (read-only) registers in the transmit data path, as well as the TxProduceIndex register in the host registers module. A soft reset of the transmit data path will abort all AHB transactions of the transmit data path. The reset bit will be cleared autonomously by the Ethernet block. A soft reset of the Tx data path will clear the TxStatus bit in the Status register.
- RxReset: Writing a ‘1’ to the RxReset bit will reset the receive data path, excluding the MAC portions, including all (read-only) registers in the receive data path, as well as the RxConsumeIndex register in the host registers module. A soft reset of the receive data path will abort all AHB transactions of the receive data path. The reset bit will be cleared autonomously by the Ethernet block. A soft reset of the Rx data path will clear the RxStatus bit in the Status register.

- RegReset: Resets all of the data paths and registers in the host registers module, excluding the registers in the MAC. A soft reset of the registers will also abort all AHB transactions of the transmit and receive data path. The reset bit will be cleared autonomously by the Ethernet block.

To do a full soft reset of the Ethernet block, device driver software must:

- Set the ‘SOFT RESET’ bit in the MAC1 register to 1.
- Set the RegReset bit in the Command register, this bit clears automatically.
- Re-initialize the MAC registers (0x000 to 0x0FC).
- Reset the ‘SOFT RESET’ bit in the MAC1 register to 0.

To reset just the transmit data path, the device driver software has to:

- Set the ‘RESET MCS/Tx’ bit in the MAC1 register to 1.
- Disable the Tx DMA managers by setting the TxEnable bits in the Command register to 0.
- Set the TxReset bit in the Command register, this bit clears automatically.
- Reset the ‘RESET MCS/Tx’ bit in the MAC1 register to 0.

To reset just the receive data path, the device driver software has to:

- Disable the receive function by resetting the ‘RECEIVE ENABLE’ bit in the MAC1 configuration register and resetting of the RxEnable bit of the Command register.
- Set the ‘RESET MCS/Rx’ bit in the MAC1 register to 1.
- Set the RxReset bit in the Command register, this bit clears automatically.
- Reset the ‘RESET MCS/Rx’ bit in the MAC1 register to 0.

10.13.19 Ethernet errors

The Ethernet block generates errors for the following conditions:

- A reception can cause an error: AlignmentError, RangeError, LengthError, SymbolError, CRCError, NoDescriptor, or Overrun. These are reported back in the receive StatusInfo and in the interrupt status register (IntStatus).
- A transmission can cause an error: LateCollision, ExcessiveCollision, ExcessiveDefer, NoDescriptor, or Underrun. These are reported back in the transmission StatusInfo and in the interrupt status register (IntStatus).

10.14 AHB bandwidth

The Ethernet block is connected to an AHB bus which must carry all of the data and control information associated with all Ethernet traffic in addition to the CPU accesses required to operate the Ethernet block and deal with message contents.

10.14.1 DMA access

Assumptions

By making some assumptions, the bandwidth needed for each type of AHB transfer can be calculated and added in order to find the overall bandwidth requirement.

The flexibility of the descriptors used in the Ethernet block allows the possibility of defining memory buffers in a range of sizes. In order to analyze bus bandwidth requirements, some assumptions must be made about these buffers. The "worst case" is not addressed since that would involve all descriptors pointing to single byte buffers, with most of the memory occupied in holding descriptors and very little data. It can easily be shown that the AHB cannot handle the huge amount of bus traffic that would be caused by such a degenerate (and illogical) case.

For this analysis, an Ethernet packet is assumed to consist of a 64 byte frame. Continuous traffic is assumed on both the transmit and receive channels.

This analysis does not reflect the flow of Ethernet traffic over time, which would include inter-packet gaps in both the transmit and receive channels that reduce the bandwidth requirements over a larger time frame.

Types of DMA access and their bandwidth requirements

The interface to an external Ethernet PHY is via RMII. RMII operates at 50 MHz, transferring a byte in 4 clock cycles. The data transfer rate is 12.5 Mbps.

The interface to an external Ethernet PHY is via either MII or RMII. An interface MII operates at 25 MHz, transferring a byte in 2 clock cycles. An RMII interface operates at 50 MHz, transferring a byte in 4 clock cycles. The data transfer rate is the same in both cases: 12.5 Mbps.

The Ethernet block initiates DMA accesses for the following cases:

- Tx descriptor read:
  - Transmit descriptors occupy 2 words (8 bytes) of memory and are read once for each use of a descriptor.
  - Two word read happens once every 64 bytes (16 words) of transmitted data.
  - This gives 1/8th of the data rate, which = 1.5625 Mbps.
- Rx descriptor read:
  - Receive descriptors occupy 2 words (8 bytes) of memory and are read once for each use of a descriptor.
  - Two word read happens once every 64 bytes (16 words) of received data.
  - This gives 1/8th of the data rate, which = 1.5625 Mbps.
- Tx status write:

- Transmit status occupies 1 word (4 bytes) of memory and is written once for each use of a descriptor.
- One word write happens once every 64 bytes (16 words) of transmitted data.
- This gives 1/16th of the data rate, which = 0.7813 Mbps.

- Rx status write:
  - Receive status occupies 2 words (8 bytes) of memory and is written once for each use of a descriptor.
  - Two word write happens once every 64 bytes (16 words) of received data.
  - This gives 1/8 of the data rate, which = 1.5625 Mbps.
- Tx data read:
  - Data transmitted in an Ethernet frame, the size is variable.
  - Basic Ethernet rate = 12.5 Mbps.
- Rx data write:
  - Data to be received in an Ethernet frame, the size is variable.
  - Basic Ethernet rate = 12.5 Mbps.

This gives a total rate of 30.5 Mbps for the traffic generated by the Ethernet DMA function.

10.14.2 Types of CPU access

- Accesses that mirror each of the DMA access types:
  - All or part of status values must be read, and all or part of descriptors need to be written after each use, transmitted data must be stored in the memory by the CPU, and eventually received data must be retrieved from the memory by the CPU.
  - This gives roughly the same or slightly lower rate as the combined DMA functions, which = 30.5 Mbps.
- Access to registers in the Ethernet block:
  - The CPU must read the RxProduceIndex, TxConsumeIndex, and IntStatus registers, and both read and write the RxConsumeIndex and TxProduceIndex registers.
  - 7 word read/writes once every 64 bytes (16 words) of transmitted and received data.
  - This gives 7/16 of the data rate, which = 5.4688 Mbps.

This gives a total rate of 36 Mbps for the traffic generated by the Ethernet DMA function.

10.14.3 Overall bandwidth

Overall traffic on the AHB is the sum of DMA access rates and CPU access rates, which comes to approximately 66.5 MB/s.

The peak bandwidth requirement can be somewhat higher due to the use of small memory buffers, in order to hold often used addresses (e.g. the station address) for example. Driver software can determine how to build frames in an efficient manner that does not overutilize the AHB.

The bandwidth available on the AHB bus depends on the system clock frequency. As an example, assume that the system clock is set at 60 MHz. All or nearly all of bus accesses related to the Ethernet will be word transfers. The raw AHB bandwidth can be approximated as 4 bytes per two system clocks, which equals 2 times the system clock rate. With a 60 MHz system clock, the bandwidth is 120 MB/s, giving about 55% utilization for Ethernet traffic during simultaneous transmit and receive operations. This shows that it is not necessary to use the maximum CPU frequency for the Ethernet to work with plenty of bandwidth headroom.

10.15 CRC calculation

The calculation is used for several purposes:

- Generation the FCS at the end of the Ethernet frame.
- Generation of the hash table index for the hash table filtering.
- Generation of the destination and source address hash CRCs.

The C pseudocode function below calculates the CRC on a frame taking the frame (without FCS) and the number of bytes in the frame as arguments. The function returns the CRC as a 32-bit integer.

```
int crc_calc(char frame_no_fcs[], int frame_len) {
     int  i;   // iterator
     int  j;   // another iterator
     char byte;// current byte
     int  crc; // CRC result
     int q0, q1, q2, q3;  // temporary variables
     crc = 0xFFFFFFFF;
     for (i = 0; i < frame_len; i++) {
          byte = *frame_no_fcs++;
          for (j = 0; j < 2; j++) {
               if (((crc >> 28) ^ (byte >> 3)) & 0x00000001) {
                    q3 = 0x04C11DB7;
               } else {
                    q3 = 0x00000000;
               }
               if (((crc >> 29) ^ (byte >> 2)) & 0x00000001) {
                    q2 = 0x09823B6E;
               } else {
                    q2 = 0x00000000;
               }
               if (((crc >> 30) ^ (byte >> 1)) & 0x00000001) {
                    q1 = 0x130476DC;
               } else {
                    q1 = 0x00000000;
               }
               if (((crc >> 31) ^ (byte >> 0)) & 0x00000001) {
                    q0 = 0x2608EDB8;
               } else {
                    q0 = 0x00000000;
               }
               crc = (crc << 4) ^ q3 ^ q2 ^ q1 ^ q0;
               byte >>= 4;
          }
     }
     return crc;
}
```

For FCS calculation, this function is passed a pointer to the first byte of the frame and the length of the frame without the FCS.

For hash filtering, this function is passed a pointer to the destination address part of the frame and the CRC is only calculated on the 6 address bytes. The hash filter uses bits [28:23] for indexing the 64-bits { HashFilterH, HashFilterL } vector. If the corresponding bit is set the packet is passed, otherwise it is rejected by the hash filter.

For obtaining the destination and source address hash CRCs, this function calculates first both the 32-bit CRCs, then the nine most significant bits from each 32-bit CRC are extracted, concatenated, and written in every StatusHashCRC word of every fragment status.

11.1 How to read this chapter

The LCD controller is available on some LPC408x/407x devices, see Section 1.4 for details.

11.2 Basic configuration

The LCD controller is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCLCD.

   Remark: The LCD is disabled on reset (PCLCD = 0).

   Also see Section 11.6.12 for power-up procedure.
2. Clock: See Table 225 and Table 35.
3. Pins: Select LCD pins and pin modes through the relevant IOCON registers (Section 7.4.1).
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 3.3.7.3) and the CLKSEL bit in the LCD_POL register (Section 11.7.3).

11.3 Introduction

The LCD controller provides all of the necessary control signals to interface directly to a variety of color and monochrome LCD panels.

11.4 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 may be generated from the peripheral clock, or from a clock input pin.

11.4.1 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

11.4.2 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.

11.4.3 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.

11.4.4 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.

11.4.5 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.

11.4.6 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.

11.5 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 205, only the LCD related portion of the pin name is shown.

Remark: To connect the LCD controller to necessary pins, see Section 7.3.

Table 205. LCD controller pins

Pin name	Type	Function
LCD_PWR	output	LCD panel power enable.
LCD_DCLK	output	LCD panel clock.
LCD_ENAB_M	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.

11.5.1 Signal usage

The signals that are used for various display types are identified in the following sections.

11.5.1.1 Signals used for single panel STN displays

The signals used for single panel STN displays are shown in Table 206. UD refers to upper panel data.

Table 206. 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_ENAB_M	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]	-	-	-

11.5.1.2 Signals used for dual panel STN displays

The signals used for dual panel STN displays are shown in Table 207. UD refers to upper panel data, and LD refers to lower panel data.

Table 207. 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_ENAB_M	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]	-	-	-

11.5.1.3 Signals used for TFT displays

The signals used for TFT displays are shown in Table 208.

Table 208. 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_ENAB_M	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]

11.6 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 32 shows a simplified block diagram of the LCD controller.

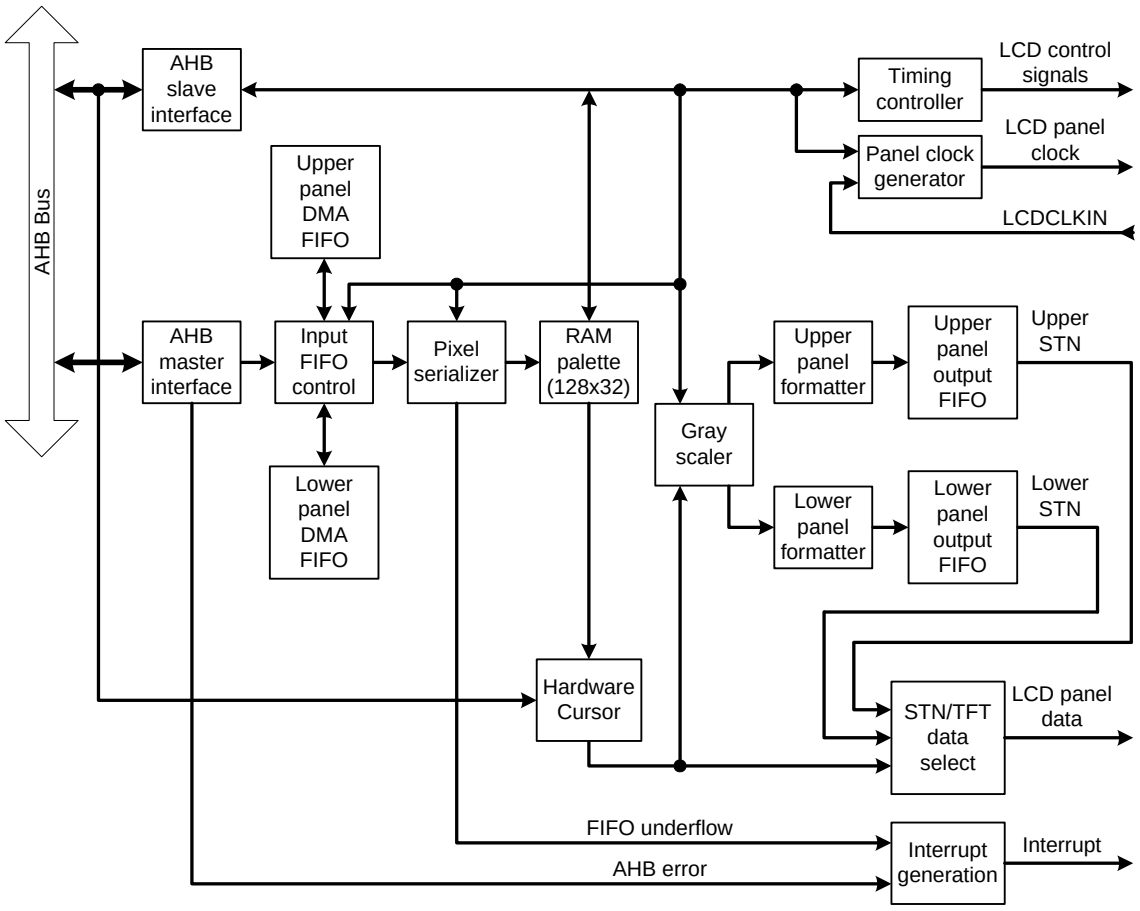


Fig 32. LCD controller block diagram

11.6.1 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.

11.6.1.1 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.

11.6.1.2 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.

11.6.2 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).

11.6.3 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 209 through Table 211 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 209. 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 210. 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 211. 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 212 shows the structure of the data in each DMA FIFO word in RGB mode.

Table 212. 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
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	-

11.6.4 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 213 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 213. 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 214 shows the bit representation of the palette data for the STN color modes.

Table 214. 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

Table 214. Palette data storage for STN color modes.

Bit(s)	Name (RGB format)	Description (RGB format)	Name (BGR format)	Description (BGR format)
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 215 shows the bit representation of the palette data for the STN monochrome mode.

Table 215. 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

11.6.5 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.

11.6.5.1 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.

11.6.5.2 Cursor sizes

Two cursor sizes are supported, as shown in Table 216.

Table 216. Palette data storage for STN monochrome mode.

X Pixels	Y Pixels	Bits per pixel	Words per line	Words in cursor image
32	32	2	2	64
64	64	2	4	256

11.6.5.3 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 33 shows how each pixel coordinate is assumed to be the top left corner of the pixel.

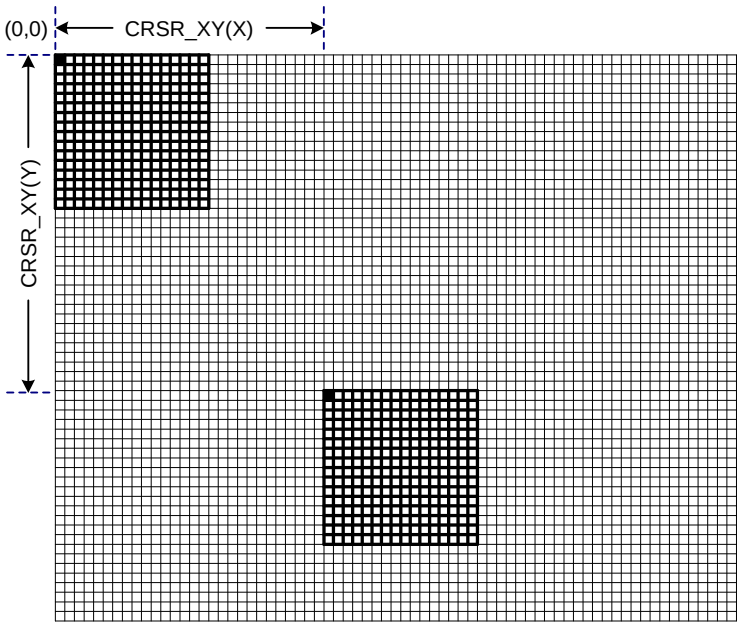


Fig 33. Cursor movement

11.6.5.4 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.

11.6.5.5 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 34). 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.

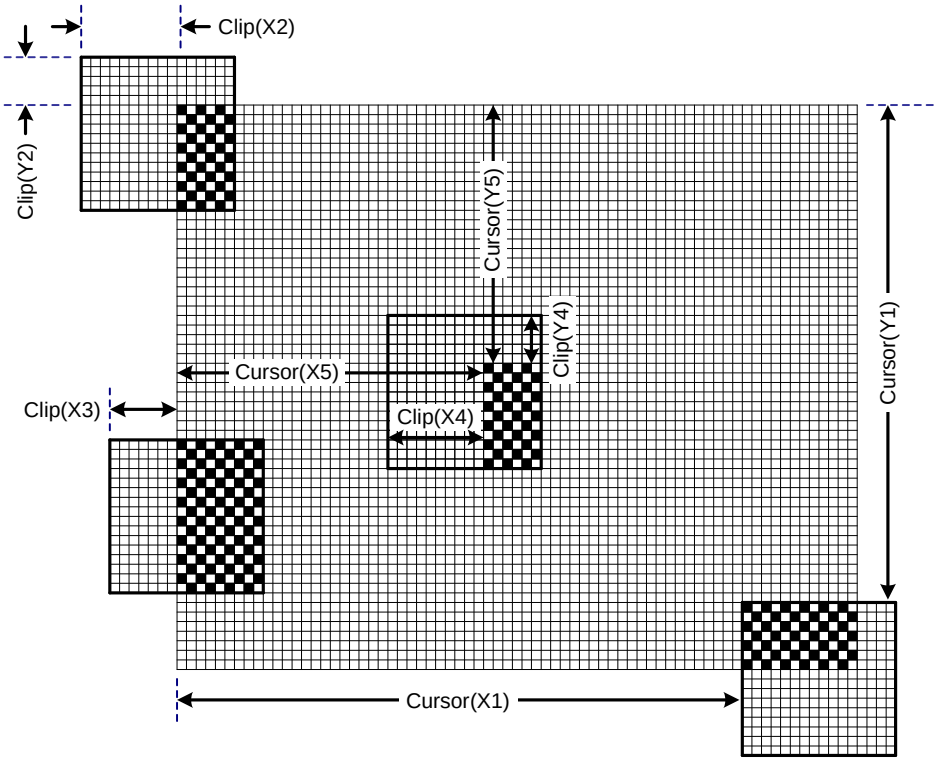


Fig 34. Cursor clipping

11.6.5.6 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 35.

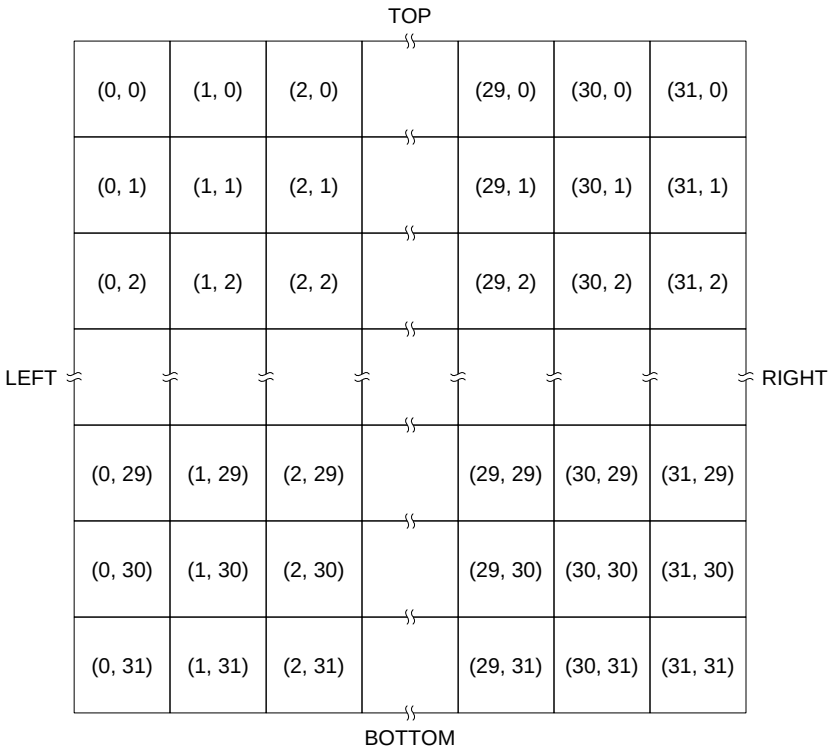


Fig 35. Cursor image format

32 by 32 pixel format

Four cursors are held in memory, each with the same pixel format. Table 217 lists the base addresses for the four cursors.

Table 217. Addresses for 32 x 32 cursors

Address	Description
0x2008 8800	Cursor 0 start address.
0x2008 8900	Cursor 1 start address.
0x2008 8A00	Cursor 2 start address.
0x2008 8B00	Cursor 3 start address.

Table 218 shows the buffer to pixel mapping for Cursor 0.

Table 218. 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)
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 219.

Table 219. 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 220 shows the pixel encoding bit assignments.

Table 220. 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.

11.6.6 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.

11.6.7 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 221 shows a color display driven with 2 2/3 pixels worth of data in a repeating sequence.

Table 221. 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.

11.6.8 Panel clock generator

The output of the panel clock generator block is the panel clock, pin LCD_DCLK. The panel clock can be based on either the peripheral clock for the LCD block or the external clock input for the LCD, pin LCD_CLKIN. Whichever source is selected can be divided down in order to produce the internal LCD clock, LCDCLK.

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.

11.6.9 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.

11.6.10 STN and TFT data select

Support is provided for passive Super Twisted Nematic (STN) and active Thin Film Transistor (TFT) LCD display types:

11.6.10.1 STN displays

STN display panels require algorithmic pixel pattern generation to provide pseudo gray scaling on monochrome displays, or color creation on color displays.

11.6.10.2 TFT displays

TFT display panels require the digital color value of each pixel to be applied to the display data inputs.

11.6.11 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.

11.6.11.1 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.

11.6.11.2 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.

11.6.11.2.1 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

11.6.11.2.2 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.

11.6.12 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_ENAB_M
- 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_ENAB_M
- 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 36 shows the power-up and power-down sequences.

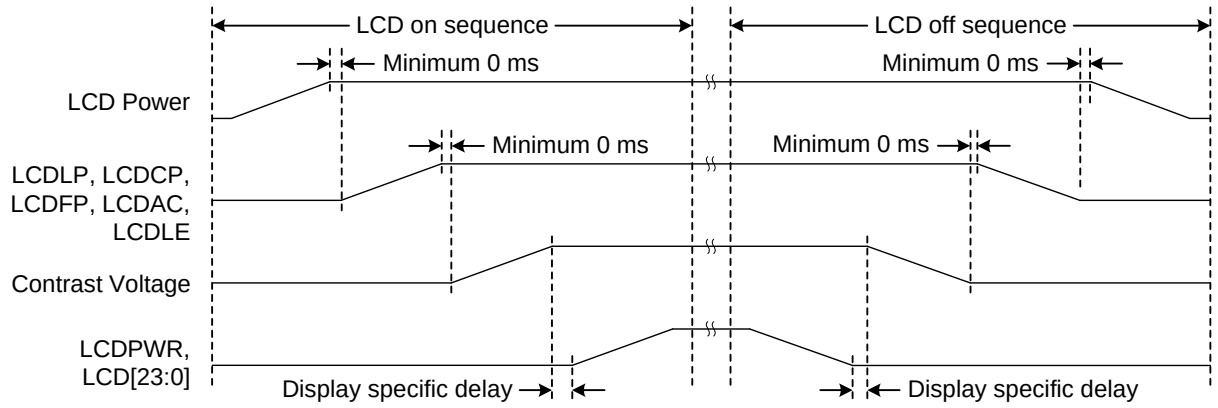


Fig 36. Power-up and power-down sequences

11.7 Register description

For LCD configuration and clocking control, see Table 35.

Table 222. Register overview: LCD controller (base address 0x2008 8000)

Name	Access	Address offset	Description	Reset value [1]	Table
TIMH	R/W	0x000	Horizontal Timing Control register	0	223
TIMV	R/W	0x004	Vertical Timing Control register	0	224
POL	R/W	0x008	Clock and Signal Polarity Control register	0	225
LE	R/W	0x00C	Line End Control register	0	226
UPBASE	R/W	0x010	Upper Panel Frame Base Address register	0	227
LPBASE	R/W	0x014	Lower Panel Frame Base Address register	0	228
CTRL	R/W	0x018	LCD Control register	0	229
INTMSK	R/W	0x01C	Interrupt Mask register	0	230
INTRAW	RO	0x020	Raw Interrupt Status register	0	231
INTSTAT	RO	0x024	Masked Interrupt Status register	0	232
INTCLR	WO	0x028	Interrupt Clear register	-	233
UPCURR	RO	0x02C	Upper Panel Current Address Value register	0	234
LPCURR	RO	0x030	Lower Panel Current Address Value register	0	235
PAL0	R/W	0x200	256x16-bit Color Palette registers	0	236
...		to			
PAL127		0x3FC	256x16-bit Color Palette registers	0	236
CRSR_IMG0	R/W	0x800	Cursor Image registers	0	237
...		to			
CRSR_IMG255		0xBFC	Cursor Image registers	0	237
CRSR_CTRL	R/W	0xC00	Cursor Control register	0	238
CRSR_CFG	R/W	0xC04	Cursor Configuration register	0	239
CRSR_PAL0	R/W	0xC08	Cursor Palette register 0	0	240
CRSR_PAL1	R/W	0xC0C	Cursor Palette register 1	0	241
CRSR_XY	R/W	0xC10	Cursor XY Position register	0	242
CRSR_CLIP	R/W	0xC14	Cursor Clip Position register	0	243
CRSR_INTMSK	R/W	0xC20	Cursor Interrupt Mask register	0	244
CRSR_INTCLR	WO	0xC24	Cursor Interrupt Clear register	-	245
CRSR_INTRAW	RO	0xC28	Cursor Raw Interrupt Status register	0	246
CRSR_INTSTAT	RO	0xC2C	Cursor Masked Interrupt Status register	0	247

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

11.7.1 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 223. Horizontal Timing register (TIMH, address 0x2008 8000) 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 pixel 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 pixel 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 pixel clocks to wait before asserting the line clock. HFP can generate a period of 1-256 pixel 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 pixel 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 pixel clocks to wait before starting the next display line. HBP can generate a delay of 1-256 pixel clock cycles. Program with desired value minus 1.	0

11.7.1.1 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 pixel clock cycles
- HBP = 5 pixel clock cycles
- HFP = 5 pixel clock cycles
- Panel Clock Divisor (PCD) = 1 (LCDCLK / 3)

Dual panel mode:

- HSW = 3 pixel clock cycles
- HBP = 5 pixel clock cycles
- HFP = 5 pixel 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.

11.7.2 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 224. Vertical Timing register (TIMV, address 0x2008 8004) 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

11.7.3 Clock and Signal Polarity register

The LCD_POL register controls various details of clock timing and signal polarity.

Table 225. Clock and Signal Polarity register (POL, address 0x2008 8008) 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 and eight down from the actual individual pixel clock rate. For color STN displays, 22/3 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	CLKSEL	Clock Select. This bit controls the selection of the source for LCDCLK. 0 = the clock source for the LCD block is CCLK. 1 = the clock source for the LCD block is LCD_CLKIN (external clock input for the LCD).	0
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_ENAB_M. This field has no effect if the LCD is operating in TFT mode, when the LCD_ENAB_M 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_ENAB_M 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_ENAB_M is in its active state. 0 = LCD_ENAB_M output pin is active HIGH in TFT mode. 1 = LCD_ENAB_M output pin is active LOW in TFT mode.	0
15	-	Reserved. Read value is undefined, only zero should be written.	-

Table 225. Clock and Signal Polarity register (POL, address 0x2008 8008) 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 pixel clock divider. Setting this to 1 bypasses the pixel 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

11.7.4 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 226. Line End Control register (LE, address 0x2008 800C) 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.	-

11.7.5 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 227. Upper Panel Frame Base register (UPBASE, address 0x2008 8010) 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

11.7.6 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 228.

Table 228. Lower Panel Frame Base register (LPBASE, address 0x2008 8014) 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

11.7.7 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 229.

Table 229. LCD Control register (CTRL, address 0x2008 8018) bit description

Bits	Symbol	Description	Reset value
0	LCDEN	LCD enable control bit. 0 = LCD disabled. Signals LCD_LP, LCD_DCLK, LCD_FP, LCD_ENAB_M, and LCD_LE are low. 1 = LCD enabled. Signals LCD_LP, LCD_DCLK, LCD_FP, LCD_ENAB_M, 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 229. LCD Control register (CTRL, address 0x2008 8018) 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.	-

11.7.8 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 230.

Table 230. Interrupt Mask register (INTMSK, address 0x2008 801C) 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.	-

11.7.9 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 231. Raw Interrupt Status register (INTRAW, address 0x2008 8020) 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.	-

11.7.10 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 232. Masked Interrupt Status register (INTSTAT, address 0x2008 8024) 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.	-

11.7.11 Interrupt Clear register

The LCD_INTCLR register is Write-Only. Writing a logic 1 to the relevant bit clears the corresponding interrupt.

Table 233. Interrupt Clear register (INTCLR, address 0x2008 8028) 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.

11.7.12 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 234. Upper Panel Current Address register (UPCURR, address 0x2008 802C) 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

11.7.13 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 235. Lower Panel Current Address register (LPCURR, address 0x2008 8030) 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

11.7.14 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 236. Color Palette registers (PAL[0:127], address 0x2008 8200 (PAL0) to 0x2008 83FC (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

11.7.15 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 11.6.5.6. 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 237. Cursor Image registers (CRSR_IMG[0:255], address 0x2008 8800 (CRSR_IMG0) to 0x2008 8BFC (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

11.7.16 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 238. Cursor Control register (CRSR_CTRL, address 0x2008 8C00) 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

11.7.17 Cursor Configuration register

The CRSR_CFG register provides overall configuration information for the hardware cursor.

Table 239. Cursor Configuration register (CRSR_CFG, address 0x2008 8C04) 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.	-

11.7.18 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 240. Cursor Palette register 0 (CRSR_PAL0, address 0x2008 8C08) 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.	-

11.7.19 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 241. Cursor Palette register 1 (CRSR_PAL1, address 0x2008 8C0C) 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.	-

11.7.20 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 242. Cursor XY Position register (CRSR_XY, address 0x2008 8C10) 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.	-

11.7.21 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 243.

Table 243. Cursor Clip Position register (CRSR_CLIP, address 0x2008 8C14) 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.	-

11.7.22 Cursor Interrupt Mask register

The CRSR_INTMSK register is used to enable or disable the cursor from interrupting the processor.

Table 244. Cursor Interrupt Mask register (CRSR_INTMSK, RW - 0x2008 8C20)

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.	-

11.7.23 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 245. Cursor Interrupt Clear register (CRSR_INTCLR, address 0x2008 8C24) 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.

11.7.24 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 246. Cursor Raw Interrupt Status register (CRSR_INTRAW, address 0x2008 8C28) 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.	-

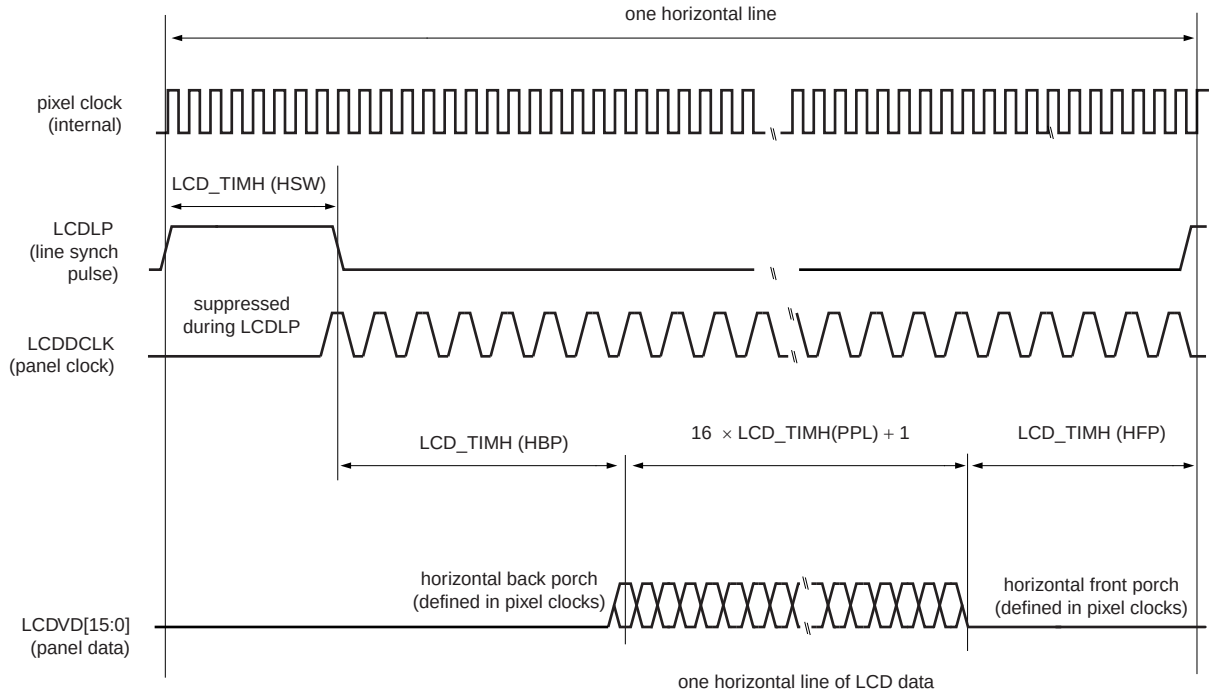
11.7.25 Cursor Masked Interrupt Status register

The CRSR_INTSTAT register is set to indicate a cursor interrupt providing that the interrupt is not masked in the CRSR_INTMSK register.

Table 247. Cursor Masked Interrupt Status register (CRSR_INTSTAT, address 0x2008 8C2C) 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.	-

11.8 LCD timing diagrams



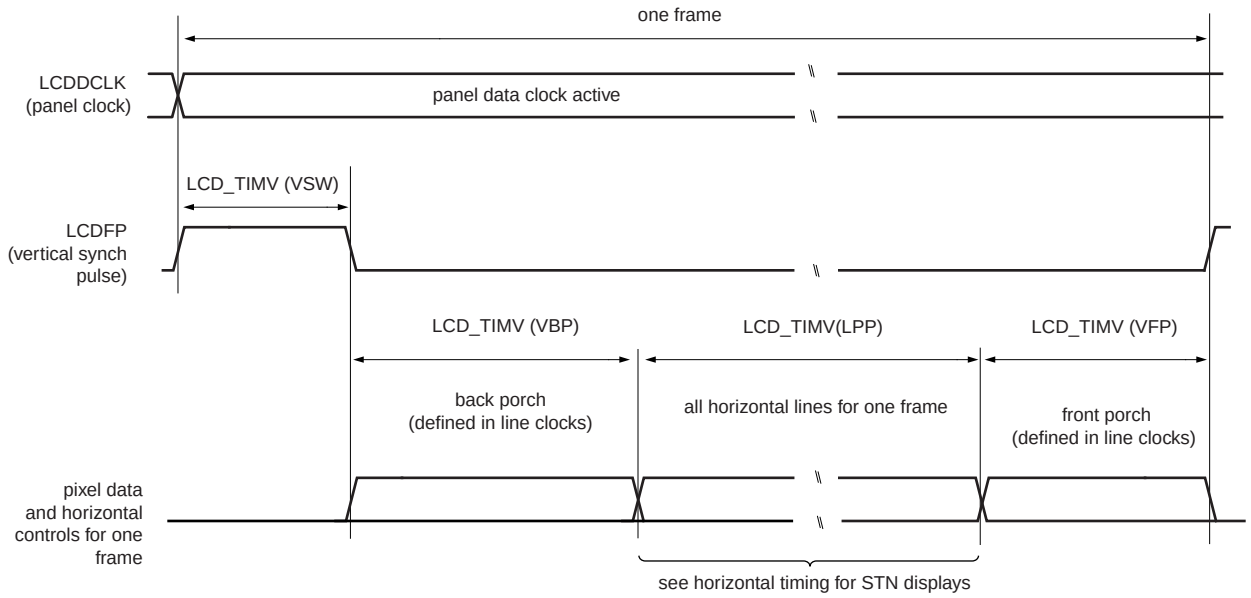
(1) The active data lines will vary with the type of STN panel (4-bit, 8-bit, color, mono) and with single or dual frames.

(2) The LCD panel clock is selected and scaled by the LCD controller and used to produce LCDCLK.

(3) The duration of the LCD_LP signal is controlled by the HSW field in the LCD_TIMH register.

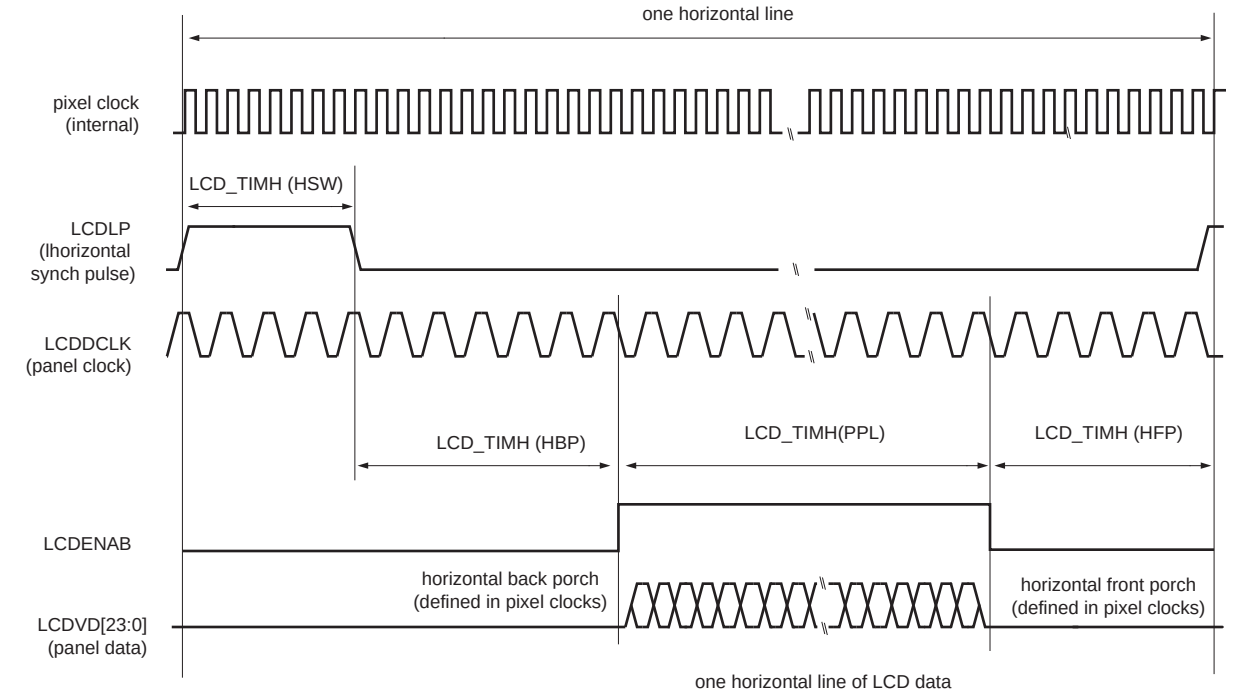
(4) The Polarity of the LCD_LP signal is determined by the IHS bit in the LCD_POL register.

Fig 37. Horizontal timing for STN displays



(1) Signal polarities may vary for some displays.

Fig 38. Vertical timing for STN displays



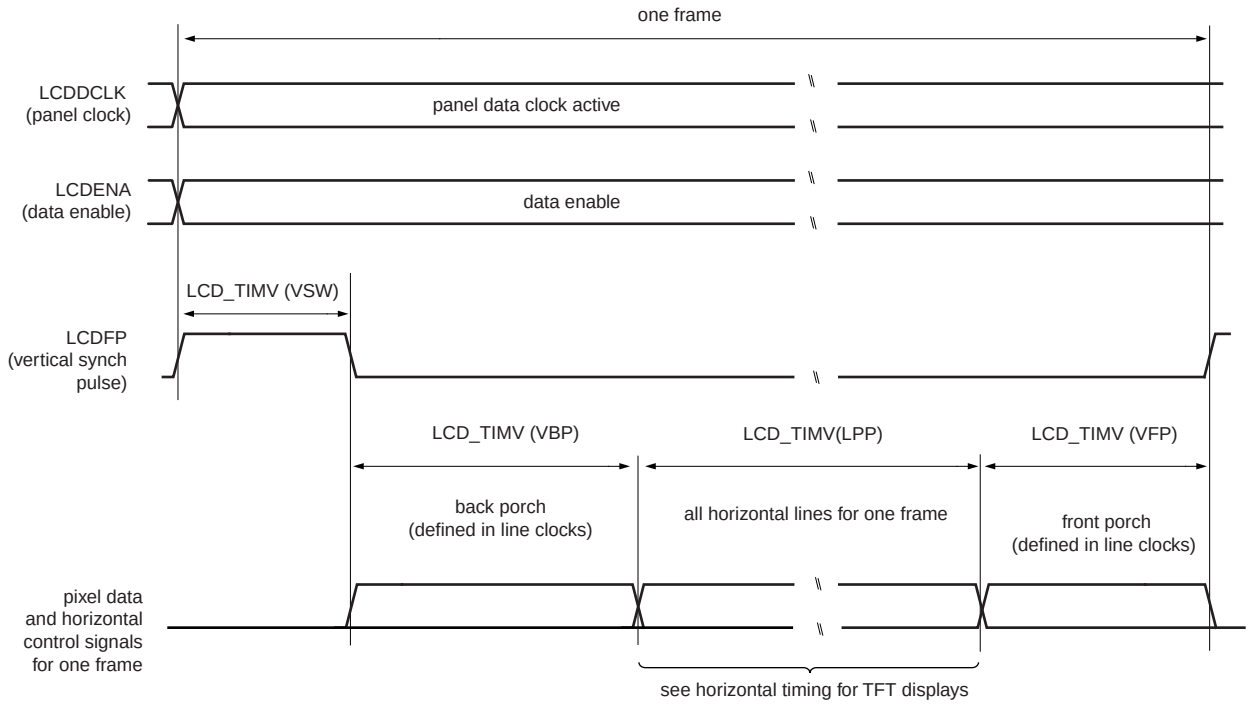
(1) The active data lines will vary with the type of TFT panel.

(2) The LCD panel clock is selected and scaled by the LCD controller and used to produce LCDCLK.

(3) The duration of the LCD_LP is controlled by the HSW field in the LCD_TIMH register.

(4) The polarity of the LCD_LP signal is determined by the IHS bit in the LCD_POL register.

Fig 39. Horizontal timing for TFT displays



(1) Polarities may vary for some displays.

Fig 40. Vertical timing for TFT displays

11.9 LCD panel signal usage

Table 248. 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_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M
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 249. 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_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M
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 250. 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_ ENAB_M	P2[4]	LCD_ ENAB_M	P2[4]	LCD_ ENAB_M	P2[4]	LCD_ ENAB_M	P2[4]	LCD_ ENAB_M
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

12.1 How to read this chapter

This chapter describes the USB device controller which is present on LPC408x/407x family devices. On some family devices, the USB controller can also be configured for Host or OTG operation (see Section 1.4 for details).

12.2 Basic configuration

The USB controller is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCUSB.

   Remark: On reset, the USB block is disabled (PCUSB = 0).
2. Clock: The USB block can be used with either the Main PLL (PLL0), or with the alternate PLL (PLL1) to obtain the USB clock. See Section 3.10.
3. Pins: Select the required USB pins and their modes in the relevant IOCON registers (Section 7.4.1).
4. Wake-up: Activity on the USB bus port can wake up the microcontroller from Power-down mode, see Section 3.12.8.
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. The USB global interrupt status is visible in the USBINTSTAT register (Table 38).
7. Initialization: See Section 12.13.

12.3 Introduction

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 16 logical or 32 physical 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 rate 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 USB device controller enables full-speed (12 Mb/s) data exchange with a USB host controller.

Table 251. USB related acronyms, abbreviations, and definitions used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-performance bus
ATLE	Auto Transfer Length Extraction
ATX	Analog Transceiver
DD	DMA Descriptor
DDP	DMA Description Pointer
DMA	Direct Memory Access
EOP	End-Of-Packet
EP	Endpoint
EP_RAM	Endpoint RAM
FS	Full Speed
LED	Light Emitting Diode
LS	Low Speed
MPS	Maximum Packet Size
NAK	Negative Acknowledge
PLL	Phase Locked Loop
RAM	Random Access Memory
SOF	Start-Of-Frame
SIE	Serial Interface Engine
SRAM	Synchronous RAM
UDCA	USB Device Communication Area
USB	Universal Serial Bus

12.4 Features

- Fully compliant with the USB 2.0 specification (full speed).
- Supports 32 physical (16 logical) endpoints.
- Supports Control, Bulk, Interrupt and Isochronous endpoints.
- Scalable realization of endpoints at run time.
- Endpoint maximum packet size selection (up to USB maximum specification) by software at run time.
- Supports SoftConnect and GoodLink features.
- Supports DMA transfers on all non-control endpoints.
- Allows dynamic switching between CPU controlled and DMA modes.
- Double buffer implementation for Bulk and Isochronous endpoints.

12.5 Fixed endpoint configuration

Table 252 shows the supported endpoint configurations. Endpoints are realized and configured at run time using the Endpoint realization registers, documented in Section 12.10.4 “Endpoint realization registers”.

Table 252. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Packet size (bytes)	Double buffer
0	0	Control	Out	8, 16, 32, 64	No
0	1	Control	In	8, 16, 32, 64	No
1	2	Interrupt	Out	1 to 64	No
1	3	Interrupt	In	1 to 64	No
2	4	Bulk	Out	8, 16, 32, 64	Yes
2	5	Bulk	In	8, 16, 32, 64	Yes
3	6	Isochronous	Out	1 to 1023	Yes
3	7	Isochronous	In	1 to 1023	Yes
4	8	Interrupt	Out	1 to 64	No
4	9	Interrupt	In	1 to 64	No
5	10	Bulk	Out	8, 16, 32, 64	Yes
5	11	Bulk	In	8, 16, 32, 64	Yes
6	12	Isochronous	Out	1 to 1023	Yes
6	13	Isochronous	In	1 to 1023	Yes
7	14	Interrupt	Out	1 to 64	No
7	15	Interrupt	In	1 to 64	No
8	16	Bulk	Out	8, 16, 32, 64	Yes
8	17	Bulk	In	8, 16, 32, 64	Yes
9	18	Isochronous	Out	1 to 1023	Yes
9	19	Isochronous	In	1 to 1023	Yes
10	20	Interrupt	Out	1 to 64	No
10	21	Interrupt	In	1 to 64	No
11	22	Bulk	Out	8, 16, 32, 64	Yes
11	23	Bulk	In	8, 16, 32, 64	Yes
12	24	Isochronous	Out	1 to 1023	Yes
12	25	Isochronous	In	1 to 1023	Yes
13	26	Interrupt	Out	1 to 64	No
13	27	Interrupt	In	1 to 64	No
14	28	Bulk	Out	8, 16, 32, 64	Yes
14	29	Bulk	In	8, 16, 32, 64	Yes
15	30	Bulk	Out	8, 16, 32, 64	Yes
15	31	Bulk	In	8, 16, 32, 64	Yes

12.6 Functional description

The architecture of the USB device controller is shown below in Figure 41.

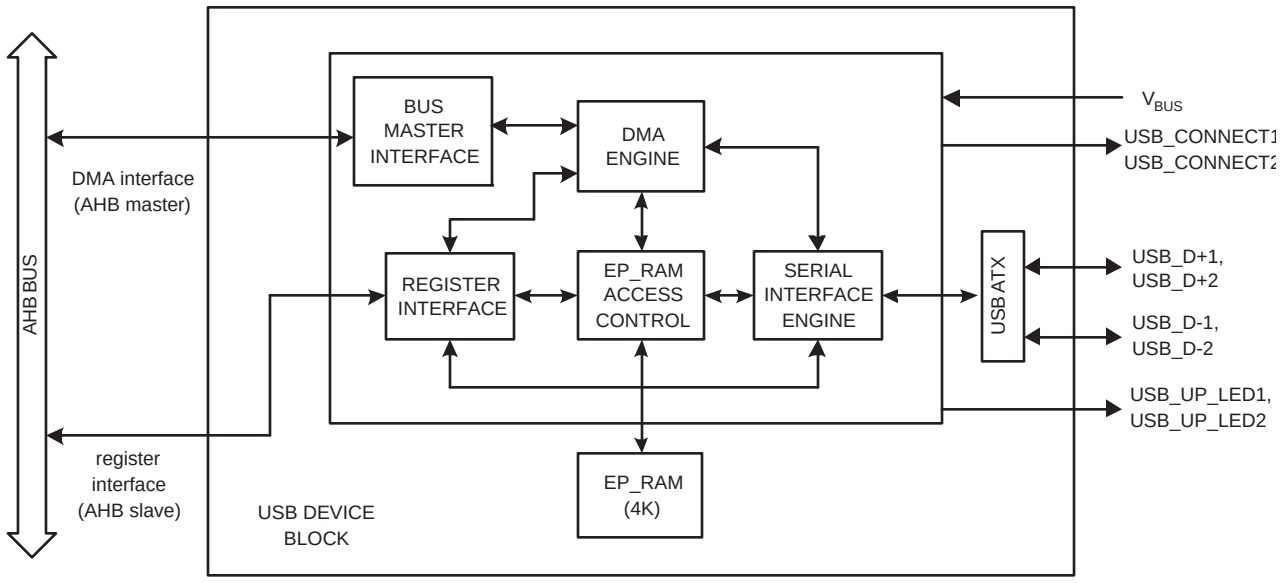


Fig 41. USB device controller block diagram

12.6.1 Analog transceiver

The USB Device Controller has a built-in analog transceiver (ATX). The USB ATX sends/receives the bidirectional D+ and D- signals of the USB bus.

12.6.2 Serial Interface Engine (SIE)

The SIE implements the full USB protocol layer. It is completely hardwired for speed and needs no firmware intervention. It handles transfer of data between the endpoint buffers in EP_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.

12.6.3 Endpoint RAM (EP_RAM)

Each endpoint buffer is implemented as an SRAM based FIFO. The SRAM dedicated for this purpose is called the EP_RAM. Each realized endpoint has a reserved space in the EP_RAM. The total EP_RAM space required depends on the number of realized endpoints, the maximum packet size of the endpoint, and whether the endpoint supports double buffering.

12.6.4 EP_RAM access control

The EP_RAM Access Control logic handles transfer of data from/to the EP_RAM and the three sources that can access it: the CPU (via the Register Interface), the SIE, and the DMA Engine.

12.6.5 DMA engine and bus master interface

When enabled for an endpoint, the DMA Engine transfers data between RAM on the AHB bus and the endpoint's buffer in EP_RAM. A single DMA channel is shared between all endpoints. When transferring data, the DMA Engine functions as a master on the AHB bus through the bus master interface.

12.6.6 Register interface

The Register Interface allows the CPU to control the operation of the USB Device Controller. It also provides a way to write transmit data to the controller and read receive data from the controller.

12.6.7 SoftConnect

The connection to the USB is accomplished by bringing D+ (for a full-speed device) HIGH through a 1.5 kOhm pull-up resistor. The SoftConnect feature can be used to allow software to finish its initialization sequence before deciding to establish connection to the USB. Re-initialization of the USB bus connection can also be performed without having to unplug the cable.

To use the SoftConnect feature, the CONNECT signal should control an external switch that connects the 1.5 kOhm resistor between D+ and +3.3V. Software can then control the CONNECT signal by writing to the CON bit using the SIE Set Device Status command.

12.6.8 GoodLink

Good USB connection indication is provided through GoodLink technology. When the device is successfully enumerated and configured, the LED indicator will be permanently ON. During suspend, the LED will be OFF.

This feature provides a user-friendly indicator on the status of the USB device. It is a useful field diagnostics tool to isolate faulty equipment.

To use the GoodLink feature the UP_LED signal should control an LED. The UP_LED signal is controlled using the SIE Configure Device command.

12.7 Operational overview

Transactions on the USB bus transfer data between device endpoints and the host. The direction of a transaction is defined with respect to the host. OUT transactions transfer data from the host to the device. IN transactions transfer data from the device to the host. All transactions are initiated by the host controller.

For an OUT transaction, the USB ATX receives the bidirectional D+ and D- signals of the USB bus. The Serial Interface Engine (SIE) receives the serial data from the ATX and converts it into a parallel data stream. The parallel data is written to the corresponding endpoint buffer in the EP_RAM.

For IN transactions, the SIE reads the parallel data from the endpoint buffer in EP_RAM, converts it into serial data, and transmits it onto the USB bus using the USB ATX.

Once data has been received or sent, the endpoint buffer can be read or written. How this is accomplished depends on the endpoint's type and operating mode. The two operating modes for each endpoint are Slave (CPU-controlled) mode, and DMA mode.

In Slave mode, the CPU transfers data between RAM and the endpoint buffer using the Register Interface. See Section 12.14 "Slave mode operation" for a detailed description of this mode.

In DMA mode, the DMA transfers data between RAM and the endpoint buffer. See Section 12.15 "DMA operation" for a detailed description of this mode.

12.8 Pin description

Table 253. USB external interface

Name	Direction	Description
V_{BUS}	I	V_{BUS} status input. When this input function is not enabled via the corresponding IOCON register, it is driven HIGH internally.
USB_CONNECT1, USB_CONNECT2	O	SoftConnect control signal.
USB_UP_LED1, USB_UP_LED2	O	GoodLink LED control signal.
USB_D+1, USB_D+2	I/O	Positive differential data.
USB_D-1, USB_D-2	I/O	Negative differential data.

12.9 Clocking and power management

This section describes the clocking and power management features of the USB Device Controller.

12.9.1 Power requirements

The USB protocol insists on power management by the device. This becomes very critical if the device draws power from the bus (bus-powered device). The following constraints should be met by a bus-powered device:

1. A device in the non-configured state should draw a maximum of 100 mA from the bus.
2. 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.
3. A suspended device can draw a maximum of 500 µA.

12.9.2 Clocks

The USB device controller clocks are shown in Table 254

Table 254. USB device controller clock sources

Clock source	Description
AHB master clock	Clock for the AHB master bus interface and DMA
AHB slave clock	Clock for the AHB slave interface
usbclk	48 MHz clock from the dedicated Alt PLL (PLL1) or the Main PLL (PLL0), used to recover the 12 MHz clock from the USB bus

12.9.3 Power management support

To help conserve power, the USB device controller automatically disables the AHB master clock and usbclk when not in use.

When the USB Device Controller goes into the suspend state (bus is idle for 3 ms), the usbclk input to the device controller is automatically disabled, helping to conserve power. However, if software wishes to access the device controller registers, usbclk must be active. To allow access to the device controller registers while in the suspend state, the USBClkCtrl and USBClkSt registers are provided.

When software wishes to access the device controller registers, it should first ensure usbclk is enabled by setting DEV_CLK_EN in the USBClkCtrl register, and then poll the corresponding DEV_CLK_ON bit in USBClkSt until set. Once set, usbclk will remain enabled until DEV_CLK_EN is cleared by software.

When a DMA transfer occurs, the device controller automatically turns on the AHB master clock. Once asserted, it remains active for a minimum of 2 ms (2 frames), to help ensure that DMA throughput is not affected by turning off the AHB master clock. 2 ms after the last DMA access, the AHB master clock is automatically disabled to help conserve power. If desired, software also has the capability of forcing this clock to remain enabled using the USBClkCtrl register.

Note that the AHB slave clock is always enabled as long as the PCUSB bit of PCONP is set. When the device controller is not in use, all of the device controller clocks may be disabled by clearing PCUSB.

The USB_NEED_CLK signal is used to facilitate going into and waking up from chip Power-down mode. USB_NEED_CLK is asserted if any of the bits of the USBClkSt register are asserted.

After entering the suspend state with DEV_CLK_EN and AHB_CLK_EN cleared, the DEV_CLK_ON and AHB_CLK_ON will be cleared when the corresponding clock turns off. When both bits are zero, USB_NEED_CLK will be low, indicating that the chip can be put into Power-down mode by writing to the PCON register. The status of USB_NEED_CLK can be read from the USBIntSt register.

Any bus activity in the suspend state will cause the USB_NEED_CLK signal to be asserted. When the chip is in Power-down mode and the USB interrupt is enabled, the assertion of USB_NEED_CLK causes the chip to wake up from Power-down mode.

12.9.4 Remote wake-up

The USB device controller supports software initiated remote wake-up. Remote wake-up involves resume signaling on the USB bus initiated from the device. This is done by clearing the SUS bit in the SIE Set Device Status register. Before writing into the register, all the clocks to the device controller have to be enabled using the USBClkCtrl register.

12.10 Register description

Table 255 shows the USB Device Controller registers directly accessible by the CPU. The Serial Interface Engine (SIE) has other registers that are indirectly accessible via the SIE command registers. See Section 12.12 “Serial interface engine command description” for more info.

The USB interrupt status is captured in the USBINTSTAT register in the syscon block.

Reading WO register will return an invalid value.

Table 255. Register overview: USB device controller (base address 0x2008 C000)

Name	Access	Address offset	Description	Reset value[1]	Table
Port select register					
PORTSEL	R/W	0x110	USB Port Select. This register is also used for OTG configuration. In device-only operations only bits 0 and 1 of this register are used to control the routing of USB pins to Port 1 or Port 2.	0	256
Device interrupt registers					
DEVINTST	RO	0x200	USB Device Interrupt Status	0x10	257
DEVINTEN	R/W	0x204	USB Device Interrupt Enable	0	258
DEVINTCLR	WO	0x208	USB Device Interrupt Clear	-	259
DEVINTSET	WO	0x20C	USB Device Interrupt Set	-	260
DEVINTPRI	WO	0x22C	USB Device Interrupt Priority	0	261
Endpoint interrupt registers					
EPINTST	RO	0x230	USB Endpoint Interrupt Status	0	263
EPINTEN	R/W	0x234	USB Endpoint Interrupt Enable	0	264
EPINTCLR	WO	0x238	USB Endpoint Interrupt Clear	-	265
EPINTSET	WO	0x23C	USB Endpoint Interrupt Set	-	266
EPINTPRI	WO	0x240	USB Endpoint Priority	0	267
Endpoint realization registers					
REEP	R/W	0x244	USB Realize Endpoint	0x3	268
EPIN	WO	0x248	USB Endpoint Index	0	269
MAXPSIZE	R/W	0x24C	USB MaxPacketSize	0x8	270
USB transfer registers					
RXDATA	RO	0x218	USB Receive Data	0	271
RXPLEN	RO	0220	USB Receive Packet Length	0	272
TXDATA	WO	0x21C	USB Transmit Data	-	273
TXPLEN	WO	0x224	USB Transmit Packet Length	0	274
CTRL	R/W	0x228	USB Control	0	275
SIE Command registers					
CMDCODE	WO	0x210	USB Command Code	0	276
CMDDATA	RO	0x214	USB Command Data	0	277
DMA registers					
DMARST	RO	0x250	USB DMA Request Status	0	278

Table 255. Register overview: USB device controller (base address 0x2008 C000)

Name	Access	Address offset	Description	Reset value[1]	Table
DMARCLR	WO	0x254	USB DMA Request Clear	-	279
DMARSET	WO	0x258	USB DMA Request Set	-	280
UDCAH	R/W	0x280	USB UDCA Head	0	281
EPDMAST	RO	0x284	USB Endpoint DMA Status	0	282
EPDMAEN	WO	0x288	USB Endpoint DMA Enable	-	283
EPDMADIS	WO	0x28C	USB Endpoint DMA Disable	-	284
DMAINTST	RO	0x290	USB DMA Interrupt Status	0	285
DMAINTEN	R/W	0x294	USB DMA Interrupt Enable	0	286
EOTINTST	RO	0x2A0	USB End of Transfer Interrupt Status	0	287
EOTINTCLR	WO	0x2A4	USB End of Transfer Interrupt Clear	-	288
EOTINTSET	WO	0x2A8	USB End of Transfer Interrupt Set	-	289
NDDRINTST	RO	0x2AC	USB New DD Request Interrupt Status	0	290
NDDRINTCLR	WO	0x2B0	USB New DD Request Interrupt Clear	-	291
NDDRINTSET	WO	0x2B4	USB New DD Request Interrupt Set	-	292
SYSERRINTST	RO	0x2B8	USB System Error Interrupt Status	0	293
SYSERRINTCLR	WO	0x2BC	USB System Error Interrupt Clear	-	294
SYSERRINTSET	WO	0x2C0	USB System Error Interrupt Set	-	295
Clock control registers					
CLKCTRL	R/W	0xFF4	USB Clock Control	0	296
CLKST	RO	0xFF8	USB Clock Status	0	297

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

12.10.1 Port select register

12.10.1.1 USB Port Select register

This register selects the USB port pins that the USB device signals are routed to. USBPortSel is a read/write register.

Table 256. USB Port Select register (PORTSEL - address 0x2008 C110) bit description

Bit	Symbol	Value	Description	Reset value
1:0	PORTSEL		Selects which USB port the device controller signals are mapped to. Other values are reserved.	0
		0x0	The USB device controller signals are mapped to the U1 port: USB_CONNECT1, USB_UP_LED1, USB_D+1, USB_D-1.	
		0x3	The USB device controller signals are mapped to the U2 port: USB_CONNECT2, USB_UP_LED2, USB_D+2, USB_D-2.	
31:2	-		Reserved. Read value is undefined, only zero should be written.	NA

12.10.2 Device interrupt registers

12.10.2.1 USB Device Interrupt Status register

The USBDevIntSt register holds the status of each interrupt. A 0 indicates no interrupt and 1 indicates the presence of the interrupt. USBDevIntSt is a read-only register.

Table 257. USB Device Interrupt Status register (DEVINTST - address 0x2008 C200) bit description

Bit	Symbol	Description	Reset value
0	FRAME	The frame interrupt occurs every 1 ms. This is used in isochronous packet transfers.	0
1	EP_FAST	Fast endpoint interrupt. If an Endpoint Interrupt Priority register (USBEpIntPri) bit is set, the corresponding endpoint interrupt will be routed to this bit.	0
2	EP_SLOW	Slow endpoints interrupt. If an Endpoint Interrupt Priority Register (USBEpIntPri) bit is not set, the corresponding endpoint interrupt will be routed to this bit.	0
3	DEV_STAT	Set when USB Bus reset, USB suspend change or Connect change event occurs. Refer to Section 12.12.6 “Set Device Status (Command: 0xFE, Data: write 1 byte)” on page 362.	0
4	CCEMPTY	The command code register (USBCmdCode) is empty (New command can be written).	1
5	CDFULL	Command data register (USBCmdData) is full (Data can be read now).	0
6	RxENDPKT	The current packet in the endpoint buffer is transferred to the CPU.	0
7	TxENDPKT	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen).	0
8	EP_RLZED	Endpoints realized. Set when Realize Endpoint register (USBReEp) or MaxPacketSize register (USBMaxPSize) is updated and the corresponding operation is completed.	0
9	ERR_INT	Error Interrupt. Any bus error interrupt from the USB device. Refer to Section 12.12.9 “Read Error Status (Command: 0xFB, Data: read 1 byte)” on page 364	0
31:10	-	Reserved. The value read from a reserved bit is not defined.	NA

12.10.2.2 USB Device Interrupt Enable register

Writing a one to a bit in this register enables the corresponding bit in USBDevIntSt to generate an interrupt on one of the interrupt lines when set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri. USBDevIntEn is a read/write register.

Table 258. USB Device Interrupt Enable register (DEVINTEN - address 0x2008 C204) bit description

Bit	Symbol	Description	Reset value
0	FRAMEEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
1	EP_FASTEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
2	EP_SLOWEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
3	DEV_STATEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
4	CCEMPTYEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
5	CDFULLEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
6	RxENDPKTEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
7	TxENDPKTEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0

Table 258. USB Device Interrupt Enable register (DEVINTEN - address 0x2008 C204) bit description

Bit	Symbol	Description	Reset value
8	EP_RLZEDEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257)) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
9	ERR_INTEN	0 = No interrupt is generated. 1 = An interrupt will be generated when the corresponding bit in the Device Interrupt Status (DevIntSt) register (Table 257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	0
31:10	-	Reserved	-

12.10.2.3 USB Device Interrupt Clear register

Writing one to a bit in this register clears the corresponding bit in USBDevIntSt. Writing a zero has no effect. USBDevIntClr is a write-only register.

Remark: Before clearing the EP_SLOW or EP_FAST interrupt bits, the corresponding endpoint interrupts in USBEpIntSt should be cleared.

Table 259. USB Device Interrupt Clear register (DEVINTCLR - address 0x2008 C208) bit description

Bit	Symbol	Description
0	FRAMECLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
1	EP_FASTCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
2	EP_SLOWCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
3	DEV_STATCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
4	CCEMPTYCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
5	CDFULLCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
6	RxENDPKTCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
7	TxENDPKTCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
8	EP_RLZEDCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
9	ERR_INTCLR	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is cleared.
31:10	-	Reserved

12.10.2.4 USB Device Interrupt Set register

Writing one to a bit in this register sets the corresponding bit in the USBDevIntSt. Writing a zero has no effect. USBDevIntSet is a write-only register.

Table 260. USB Device Interrupt Set register (DEVINTSET - address 0x2008 C20C) bit description

Bit	Symbol	Description
0	FRAMESET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
1	EP_FASTSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
2	EP_SLOWSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
3	DEV_STATSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
4	CCEMPTYSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
5	CDFULLSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
6	RxENDPKTSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
7	TxENDPKTSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
8	EP_RLZEDSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
9	ERR_INTSET	0 = No effect. 1 = The corresponding bit in USBDevIntSt (Section 12.10.2.1) is set.
31:10	-	Reserved

12.10.2.5 USB Device Interrupt Priority register

Writing one to a bit in this register causes the corresponding interrupt to be routed to the USB_INT_REQ_HP interrupt line. Writing zero causes the interrupt to be routed to the USB_INT_REQ_LP interrupt line. Either the EP_FAST or FRAME interrupt can be routed to USB_INT_REQ_HP, but not both. If the software attempts to set both bits to one, no interrupt will be routed to USB_INT_REQ_HP. USBDevIntPri is a write-only register.

Table 261. USB Device Interrupt Priority register (DEVINTPRI - address 0x2008 C22C) bit description

Bit	Symbol	Value	Description	Reset value
0	FRAME		Frame interrupt routing	0
		0	FRAME interrupt is routed to USB_INT_REQ_LP.	
		1	FRAME interrupt is routed to USB_INT_REQ_HP.	
1	EP_FAST		Fast endpoint interrupt routing	0
		0	EP_FAST interrupt is routed to USB_INT_REQ_LP.	
		1	EP_FAST interrupt is routed to USB_INT_REQ_HP.	
31:2	-		Reserved. Read value is undefined, only zero should be written.	-

12.10.3 Endpoint interrupt registers

The registers in this group facilitate handling of endpoint interrupts. Endpoint interrupts are used in Slave mode operation.

Table 262. USB Endpoint registers bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	EPx31 = EP15TX	EPx30 = EP15RX	EPx29 = EP14TX	EPx28 = EP14RX	EPx27 = EP13TX	EPx26 = EP13RX	EPx25 = EP12TX	EPx24 = EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EPx23 = EP11TX	EPx22 = EP11RX	EPx21 = EP10TX	EPx20 = EP10RX	EPx19 = EP9TX	EPx18 = EP9RX	EPx17 = EP8TX	EPx16 = EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EPx15 = EP7TX	EPx14 = EP7RX	EPx13 = EP6TX	EPx12 = EP6RX	EPx11 = EP5TX	EPx10 = EP5RX	EPx9 = EP4TX	EPx8 = EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EPx7 = EP3TX	EPx6 = EP3RX	EPx5 = EP2TX	EPx4 = EP2RX	EPx3 = EP1TX	EPx2 = EP1RX	EPx1 = EP0TX	EPx0 = EP0RX

12.10.3.1 USB Endpoint Interrupt Status register

Each physical non-isochronous endpoint is represented by a bit in this register to indicate that it has generated an interrupt. All non-isochronous OUT endpoints generate an interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet is successfully transmitted, or when a NAK handshake is sent on the bus and the interrupt on NAK feature is enabled (see Section 12.12.3 “Set Mode (Command: 0xF3, Data: write 1 byte)” on page 360). A bit set to one in this register causes either the EP_FAST or EP_SLOW bit of USBDevIntSt to be set depending on the value of the corresponding bit of USBEpDevIntPri. USBEpIntSt is a read-only register.

Note that for Isochronous endpoints, handling of packet data is done when the FRAME interrupt occurs.

Table 263. USB Endpoint Interrupt Status register (EPINTST - address 0x2008 C230) bit description

Bit	Symbol	Description	Reset value
31:0	EPST	1 = Endpoint Data Received (bits 0, 2, 4, ..., 30) or Transmitted (bits 1, 3, 5, ..., 31) Interrupt received.	0

12.10.3.2 USB Endpoint Interrupt Enable register

Setting a bit to 1 in this register causes the corresponding bit in USBEpIntSt to be set when an interrupt occurs for the associated endpoint. Setting a bit to 0 causes the corresponding bit in USBDMARSt to be set when an interrupt occurs for the associated endpoint. USBEpIntEn is a read/write register.

Table 264. USB Endpoint Interrupt Enable register (EPINTEN - address 0x2008 C234) bit description

Bit	Symbol	Description	Reset value
31:0	EPEN	0= The corresponding bit in USBDMARSt is set when an interrupt occurs for this endpoint. 1 = The corresponding bit in USBEpIntSt is set when an interrupt occurs for this endpoint. Implies Slave mode for this endpoint.	0

12.10.3.3 USB Endpoint Interrupt Clear register

Writing a one to this a bit in this register causes the SIE Select Endpoint/Clear Interrupt command to be executed (Table 305) for the corresponding physical endpoint. Writing zero has no effect. Before executing the Select Endpoint/Clear Interrupt command, the CDFULL bit in USBDevIntSt is cleared by hardware. On completion of the command, the CDFULL bit is set, USBCmdData contains the status of the endpoint, and the corresponding bit in USBEpIntSt is cleared.

Notes:

- When clearing interrupts using USBEpIntClr, software should wait for CDFULL to be set to ensure the corresponding interrupt has been cleared before proceeding.
- While setting multiple bits in USBEpIntClr simultaneously is possible, it is not recommended; only the status of the endpoint corresponding to the least significant interrupt bit cleared will be available at the end of the operation.
- Alternatively, the SIE Select Endpoint/Clear Interrupt command can be directly invoked using the SIE command registers, but using USBEpIntClr is recommended because of its ease of use.

Each physical endpoint has its own reserved bit in this register. The bit field definition is the same as that of EpIntSt shown in Table 263. EpIntClr is a write-only register.

Table 265. USB Endpoint Interrupt Clear register (EPINTCLR - address 0x2008 C238) bit description

Bit	Symbol	Description
31:0	EPCLR	0 = No effect. 1 = Clears the corresponding bit in USBEpIntSt, by executing the SIE Select Endpoint/Clear Interrupt command for this endpoint.

12.10.3.4 USB Endpoint Interrupt Set register

Writing a one to a bit in this register sets the corresponding bit in USBEpIntSt. Writing zero has no effect. Each endpoint has its own bit in this register. USBEpIntSet is a write-only register.

Table 266. USB Endpoint Interrupt Set register (EPINTSET - address 0x2008 C23C) bit description

Bit	Symbol	Description
31:0	EPSET	0 = No effect. 1 = Sets the corresponding bit in USBEpIntSt.

12.10.3.5 USB Endpoint Interrupt Priority register

This register determines whether an endpoint interrupt is routed to the EP_FAST or EP_SLOW bits of USBDevIntSt. If a bit in this register is set to one, the interrupt is routed to EP_FAST, if zero it is routed to EP_SLOW. Routing of multiple endpoints to EP_FAST or EP_SLOW is possible.

Note that the USBDevIntPri register determines whether the EP_FAST interrupt is routed to the USB_INT_REQ_HP or USB_INT_REQ_LP interrupt line.

USBEpIntPri is a write-only register.

Table 267. USB Endpoint Interrupt Priority register (EPINTPRI - address 0x2008 C240) bit description

Bit	Symbol	Description	Reset value
31:0	EPPRI	0 = The corresponding interrupt is routed to the EP_SLOW bit of USBDevIntSt 1 = The corresponding interrupt is routed to the EP_FAST bit of USBDevIntSt	0

12.10.4 Endpoint realization registers

The registers in this group allow realization and configuration of endpoints at run time.

12.10.4.1 EP RAM requirements

The USB device controller uses a RAM based FIFO for each endpoint buffer. The RAM dedicated for this purpose is called the Endpoint RAM (EP_RAM). Each endpoint has space reserved in the EP_RAM. The EP_RAM space required for an endpoint depends on its MaxPacketSize and whether it is double buffered. 32 words of EP_RAM are used by the device for storing the endpoint buffer pointers. The EP_RAM is word aligned but the MaxPacketSize is defined in bytes hence the RAM depth has to be adjusted to the next word boundary. Also, each buffer has one word header showing the size of the packet length received.

The EP_ RAM space (in words) required for the physical endpoint can be expressed as

$$EPRAMspace = \left(\frac{MaxPacketSize + 3}{4} + 1\right) \times dbstatus$$

where dbstatus = 1 for a single buffered endpoint and 2 for double a buffered endpoint.

Since all the realized endpoints occupy EP_RAM space, the total EP_RAM requirement is

$$TotalEPRAMspace = 32 + \sum_{n=0}^{N} EPRAMspace(n)$$

where N is the number of realized endpoints. Total EP_RAM space should not exceed 4096 bytes (4 kB, 1 kwords).

12.10.4.2 USB Realize Endpoint register

Writing one to a bit in this register causes the corresponding endpoint to be realized. Writing zeros causes it to be unrealized. This register returns to its reset state when a bus reset occurs. USBReEp is a read/write register.

Table 268. USB Realize Endpoint register (REEP - address 0x2008 C244) bit description

Bit	Symbol	Description	Reset value
31:0	EPR	0 = Endpoint EPxx is not realized. 1 = Endpoint EPxx is realized.	0

On reset, only the control endpoints are realized. Other endpoints, if required, are realized by programming the corresponding bits in USBReEp. To calculate the required EP_RAM space for the realized endpoints, see Section 12.10.4.1.

Realization of endpoints is a multi-cycle operation. Pseudo code for endpoint realization is shown below.

```
Clear EP_RLZED bit in USBDevIntSt;

for every endpoint to be realized,
{
     /* OR with the existing value of the Realize Endpoint register */
     USBReEp |= (UInt32) ((0x1 << endpt));
     /* Load Endpoint index Reg with physical endpoint no.*/
     USBEpIn = (UInt32) endpointnumber;

     /* load the max packet size Register */
     USBEpMaxPSize = MPS;

     /* check whether the EP_RLZED bit in the Device Interrupt Status register is set
     */
     while (!(USBDevIntSt & EP_RLZED))
     {
          /* wait until endpoint realization is complete */
     }
     /* Clear the EP_RLZED bit */
     Clear EP_RLZED bit in USBDevIntSt;
}
```

The device will not respond to any transactions to unrealized endpoints. The SIE Configure Device command will only cause realized and enabled endpoints to respond to transactions. For details see Table 300.

12.10.4.3 USB Endpoint Index register

Each endpoint has a register carrying the MaxPacketSize value for that endpoint. This is in fact a register array. Hence before writing, this register is addressed through the USBEpIn register.

The USBEpIn register will hold the physical endpoint number. Writing to USBMaxPSize will set the array element pointed to by USBEpIn. USBEpIn is a write-only register.

Table 269. USB Endpoint Index register (EPIN - address 0x2008 C248) bit description

Bit	Symbol	Description	Reset value
4:0	PHY_EP	Physical endpoint number (0-31)	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	NA

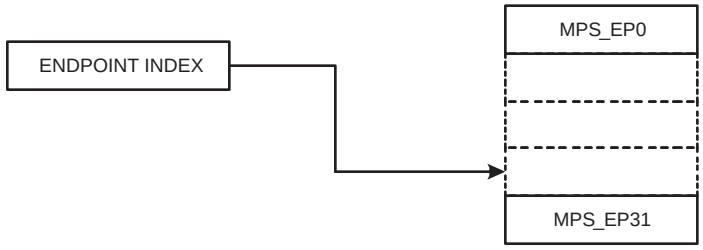
12.10.4.4 USB MaxPacketSize register

On reset, the control endpoint is assigned the maximum packet size of 8 bytes. Other endpoints are assigned 0. Modifying USBMaxPSize will cause the endpoint buffer addresses within the EP_RAM to be recalculated. This is a multi-cycle process. At the end, the EP_RLZED bit will be set in USBDevIntSt (Table 257). USBMaxPSize array indexing is shown in Figure 42. USBMaxPSize is a read/write register.

Table 270. USB MaxPacketSize register (MAXPSIZE - address 0x2008 C24C) bit description

Bit	Symbol	Description	Reset value
9:0	MPS	The maximum packet size value.	0x008[1]
31:10	-	Reserved. Read value is undefined, only zero should be written.	NA

[1] Reset value for EP0 and EP1. All other endpoints have a reset value of 0x0.



The Endpoint Index is set via the USBEpIn register. MPS_EP0 to MPS_EP31 are accessed via the USBMaxPSize register.

Fig 42. USB MaxPacketSize register array indexing

12.10.5 USB transfer registers

The registers in this group are used for transferring data between endpoint buffers and RAM in Slave mode operation. See Section 12.14 “Slave mode operation”.

12.10.5.1 USB Receive Data register

For an OUT transaction, the CPU reads the endpoint buffer data from this register. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. On reading this register, data from the selected endpoint buffer is fetched. The data is in little endian format: the first byte received from the USB bus will be available in the least significant byte of USBRxData. USBRxData is a read-only register.

Table 271. USB Receive Data register (RXDATA - address 0x2008 C218) bit description

Bit	Symbol	Description	Reset value
31:0	RX_DATA	Data received.	0

12.10.5.2 USB Receive Packet Length register

This register contains the number of bytes remaining in the endpoint buffer for the current packet being read via the USBRxData register, and a bit indicating whether the packet is valid or not. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. This register is updated on each read of the USBRxData register. USBRxPLen is a read-only register.

Table 272. USB Receive Packet Length register (RXPLEN - address 0x2008 C220) bit description

Bit	Symbol	Value	Description	Reset value
9:0	PKT_LNGTH	-	The remaining number of bytes to be read from the currently selected endpoint’s buffer. When this field decrements to 0, the RxENDPKT bit will be set in USBDevIntSt.	0
10	DV		Data valid. This bit is useful for isochronous endpoints. Non-isochronous endpoints do not raise an interrupt when an erroneous data packet is received. But invalid data packet can be produced with a bus reset. For isochronous endpoints, data transfer will happen even if an erroneous packet is received. In this case DV bit will not be set for the packet.	0
		0	Data is invalid.	
		1	Data is valid.	
11	PKT_RDY		The PKT_LNGTH field is valid and the packet is ready for reading.	0
31:12	-		Reserved. The value read from a reserved bit is not defined.	NA

12.10.5.3 USB Transmit Data register

For an IN transaction, the CPU writes the endpoint data into this register. Before writing to this register, the WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately, and the packet length should be written to the USBTxPlen register. On writing this register, the data is written to the selected endpoint buffer. The data is in little endian format: the first byte sent on the USB bus will be the least significant byte of USBTxData. USBTxData is a write-only register.

Table 273. USB Transmit Data register (TXDATA - address 0x2008 C21C) bit description

Bit	Symbol	Description
31:0	TX_DATA	Transmit Data.

12.10.5.4 USB Transmit Packet Length register

This register contains the number of bytes transferred from the CPU to the selected endpoint buffer. Before writing data to USBTxData, software should first write the packet length (≤ MaxPacketSize) to this register. After each write to USBTxData, hardware decrements USBTxPLen by 4. The WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set to select the desired endpoint buffer before starting this process.

For data buffers larger than the endpoint's MaxPacketSize, software should submit data in packets of MaxPacketSize, and send the remaining extra bytes in the last packet. For example, if the MaxPacketSize is 64 bytes and the data buffer to be transferred is of length 130 bytes, then the software sends two 64-byte packets and the remaining 2 bytes in the last packet. So, a total of 3 packets are sent on USB. USBTxPLen is a write-only register.

Table 274. USB Transmit Packet Length register (TXPLEN - address 0x2008 C224) bit description

Bit	Symbol	Description	Reset value
9:0	PKT_LNGTH	The remaining number of bytes to be written to the selected endpoint buffer. This field is decremented by 4 by hardware after each write to USBTxData. When this field decrements to 0, the TxENDPKT bit will be set in USBDevIntSt.	0
31:10	-	Reserved. Read value is undefined, only zero should be written.	NA

12.10.5.5 USB Control register

This register controls the data transfer operation of the USB device. It selects the endpoint buffer that is accessed by the USBRxData and USBTxData registers, and enables reading and writing them. USBCtrl is a read/write register.

Table 275. USB Control register (CTRL - address 0x2008 C228) bit description

Bit	Symbol	Value	Description	Reset value
0	RD_EN		Read mode control. Enables reading data from the OUT endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBRxData register. This bit is cleared by hardware when the last word of the current packet is read from USBRxData.	0
		0	Disabled.	
		1	Enabled.	
1	WR_EN		Write mode control. Enables writing data to the IN endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBTxData register. This bit is cleared by hardware when the number of bytes in USBTxLen have been sent.	0
		0	Disabled.	
		1	Enabled.	
5:2	LOG_ENDPOINT		Logical Endpoint number.	0
31:6	-		Reserved. Read value is undefined, only zero should be written.	NA

12.10.6 SIE command code registers

The SIE command code registers are used for communicating with the Serial Interface Engine. See Section 12.12 “Serial interface engine command description” for more information.

12.10.6.1 USB Command Code register

This register is used for sending the command and write data to the SIE. The commands written here are propagated to the SIE and executed there. After executing the command, the register is empty, and the CCEMPTY bit of USBDevIntSt register is set. See Section 12.12 for details. USBCmdCode is a write-only register.

Table 276. USB Command Code register (CMDCODE - address 0x2008 C210) bit description

Bit	Symbol	Value	Description
7:0	-		Reserved. Read value is undefined, only zero should be written.
15:8	CMD_PHASE		The command phase:
		0x02	Read
		0x01	Write
		0x05	Command
23:16	CMD_CODE_WDATA		This is a multi-purpose field. When CMD_PHASE is Command or Read, this field contains the code for the command (CMD_CODE). When CMD_PHASE is Write, this field contains the command write data (CMD_WDATA).
31:24	-		Reserved. Read value is undefined, only zero should be written.

12.10.6.2 USB Command Data register

This register contains the data retrieved after executing a SIE command. When the data is ready to be read, the CD_FULL bit of the USBDevIntSt register is set. See Table 257 for details. USBCmdData is a read-only register.

Table 277. USB Command Data register (CMDDATA - address 0x2008 C214) bit description

Bit	Symbol	Description	Reset value
7:0	CMD_RDATA	Command Read Data.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

12.10.7 DMA registers

The registers in this group are used for the DMA mode of operation (see Section 12.15 “DMA operation”)

12.10.7.1 USB DMA Request Status register

A bit in this register associated with a non-isochronous endpoint is set by hardware when an endpoint interrupt occurs (see the description of USBEpIntSt) and the corresponding bit in USBEpIntEn is 0. A bit associated with an isochronous endpoint is set when the corresponding bit in USBEpIntEn is 0 and a FRAME interrupt occurs. A set bit serves as a flag for the DMA engine to start the data transfer if the DMA is enabled for the corresponding endpoint in the USBEpDMASt register. The DMA cannot be enabled for control endpoints (EP0 and EP1). USBDMARSt is a read-only register.

Table 278. USB DMA Request Status register (DMARST - address 0x2008 C250) bit description

Bit	Symbol	Description	Reset value
0	EPRST0	Control endpoint OUT (DMA cannot be enabled for this endpoint and EP0 bit must be 0).	0
1	EPRST1	Control endpoint IN (DMA cannot be enabled for this endpoint and EP1 bit must be 0).	0
31:2	EPRST	Endpoint xx ($2 \leq xx \leq 31$) DMA request. 0 = DMA not requested by endpoint xx. 1 = DMA requested by endpoint xx.	0

[1] DMA can not be enabled for this endpoint and the corresponding bit in the USBDMARSt must be 0.

12.10.7.2 USB DMA Request Clear register

Writing one to a bit in this register will clear the corresponding bit in the USBDMARSt register. Writing zero has no effect.

This register is intended for initialization prior to enabling the DMA for an endpoint. When the DMA is enabled for an endpoint, hardware clears the corresponding bit in USBDMARSt on completion of a packet transfer. Therefore, software should not clear the bit using this register while the endpoint is enabled for DMA operation.

USBDMARClr is a write-only register.

The USBDMARClr bit allocation is identical to the USBDMARSt register (Table 278).

Table 279. USB DMA Request Clear register (DMARCLR - address 0x2008 C254) bit description

Bit	Symbol	Description
0	EPRCLR0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).
1	EPRCLR1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).
31:2	EPRCLR	Clear the endpoint xx ($2 \leq xx \leq 31$) DMA request. 0 = No effect 1 = Clear the corresponding bit in USBDMARSt.

12.10.7.3 USB DMA Request Set register

Writing one to a bit in this register sets the corresponding bit in the USBDMARSt register. Writing zero has no effect.

This register allows software to raise a DMA request. This can be useful when switching from Slave to DMA mode of operation for an endpoint: if a packet to be processed in DMA mode arrives before the corresponding bit of USBEpIntEn is cleared, the DMA request is not raised by hardware. Software can then use this register to manually start the DMA transfer.

Software can also use this register to initiate a DMA transfer to proactively fill an IN endpoint buffer before an IN token packet is received from the host.

USBDMARSet is a write-only register.

The USBDMARSet bit allocation is identical to the USBDMARSt register (Table 278).

Table 280. USB DMA Request Set register (DMARSET - address 0x2008 C258) bit description

Bit	Symbol	Description
0	EPRSET0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).
1	EPRSET1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).
31:2	EPRSET	Set the endpoint xx ($2 \leq xx \leq 31$) DMA request. 0 = No effect 1 = Set the corresponding bit in DMARSt.

12.10.7.4 USB UDCA Head register

The UDCA (USB Device Communication Area) Head register maintains the address where the UDCA is located in the RAM. Refer to Section 12.15.2 “USB device communication area” and Section 12.15.4 “The DMA descriptor” for more details on the UDCA and DMA descriptors. UDCAH is a read/write register.

Table 281. USB UDCA Head register (UDCAH - address 0x2008 C280) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved. Read value is undefined, only zero should be written. The UDCA is aligned to 128-byte boundaries.	0
31:7	UDCA_ADDR	Start address of the UDCA.	0

12.10.7.5 USB EP DMA Status register

Bits in this register indicate whether DMA operation is enabled for the corresponding endpoint. A DMA transfer for an endpoint can start only if the corresponding bit is set in this register. EpDMASt is a read-only register.

Table 282. USB EP DMA Status register (EPDMAST - address 0x2008 C284) bit description

Bit	Symbol	Description	Reset value
0	EP_DMA_ST0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_ENABLE bit must be 0).	0
1	EP_DMA_ST1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_ENABLE bit must be 0).	0
31:2	EP_DMA_ST	Endpoint xx ($2 \leq xx \leq 31$) DMA enabled bit. 0 = The DMA for endpoint EPxx is disabled. 1 = The DMA for endpoint EPxx is enabled.	0

12.10.7.6 USB EP DMA Enable register

Writing one to a bit to this register will enable the DMA operation for the corresponding endpoint. Writing zero has no effect.The DMA cannot be enabled for control endpoints EP0 and EP1. EpDMAEn is a write-only register.

Table 283. USB EP DMA Enable register (EPDMAEN - address 0x2008 C288) bit description

Bit	Symbol	Description
0	EP_DMA_EN0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_ENABLE bit value must be 0).
1	EP_DMA_EN1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_ENABLE bit must be 0).
31:2	EP_DMA_EN	Endpoint xx($2 \leq xx \leq 31$) DMA enable control bit. 0 = No effect. 1 = Enable the DMA operation for endpoint EPxx.

12.10.7.7 USB EP DMA Disable register

Writing a one to a bit in this register clears the corresponding bit in EpDMASt. Writing zero has no effect on the corresponding bit of EpDMASt. Any write to this register clears the internal DMA_PROCEED flag. Refer to Section 12.15.5.4 “Optimizing descriptor fetch” for more information on the DMA_PROCEED flag. If a DMA transfer is in progress for an endpoint when its corresponding bit is cleared, the transfer is completed before the DMA is disabled. When an error condition is detected during a DMA transfer, the corresponding bit is cleared by hardware. EpDMADis is a write-only register.

Table 284. USB EP DMA Disable register (EPDMADIS - address 0x2008 C28C) bit description

Bit	Symbol	Description
0	EP_DMA_DIS0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_DISABLE bit value must be 0).
1	EP_DMA_DIS1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_DISABLE bit value must be 0).
31:2	EP_DMA_DIS	Endpoint xx ($2 \leq xx \leq 31$) DMA disable control bit. 0 = No effect. 1 = Disable the DMA operation for endpoint EPxx.

12.10.7.8 USB DMA Interrupt Status register

Each bit of this register reflects whether any of the 32 bits in the corresponding interrupt status register are set. DMAIntSt is a read-only register.

Table 285. USB DMA Interrupt Status register (DMAINTST - address 0x2008 C290) bit description

Bit	Symbol	Value	Description	Reset value
0	EOT		End of Transfer Interrupt bit.	0
		0	All bits in the EoTIntSt register are 0.	
		1	At least one bit in the EoTIntSt is set.	
1	NDDR		New DD Request Interrupt bit.	0
		0	All bits in the NDDRIntSt register are 0.	
		1	At least one bit in the NDDRIntSt is set.	
2	ERR		System Error Interrupt bit.	0
		0	All bits in the SysErrIntSt register are 0.	
		1	At least one bit in the SysErrIntSt is set.	
31:3	-		Reserved. The value read from a reserved bit is not defined.	NA

12.10.7.9 USB DMA Interrupt Enable register

Writing a one to a bit in this register enables the corresponding bit in DMAIntSt to generate an interrupt on the USB_INT_REQ_DMA interrupt line when set. DMAIntEn is a read/write register.

Table 286. USB DMA Interrupt Enable register (DMAINTEN - address 0x2008 C294) bit description

Bit	Symbol	Value	Description	Reset value
0	EOT		End of Transfer Interrupt enable bit.	0
		0	Disabled.	
		1	Enabled.	
1	NDDR		New DD Request Interrupt enable bit.	0
		0	Disabled.	
		1	Enabled.	
2	ERR		System Error Interrupt enable bit.	0
		0	Disabled.	
		1	Enabled.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	NA

12.10.7.10 USB End of Transfer Interrupt Status register

When the DMA transfer completes for the current DMA descriptor, either normally (descriptor is retired) or because of an error, the bit corresponding to the endpoint is set in this register. The cause of the interrupt is recorded in the DD_status field of the descriptor. EoTIntSt is a read-only register.

Table 287. USB End of Transfer Interrupt Status register (EOTINTST - address 0x2008 C2A0) bit description

Bit	Symbol	Description	Reset value
31:0	EPTXINTST	Endpoint xx ($2 \le xx \le 31$) End of Transfer Interrupt request. 0 = There is no End of Transfer interrupt request for endpoint xx. 1 = There is an End of Transfer Interrupt request for endpoint xx.	0

12.10.7.11 USB End of Transfer Interrupt Clear register

Writing one to a bit in this register clears the corresponding bit in the EoTIntSt register. Writing zero has no effect. EoTIntClr is a write-only register.

Table 288. USB End of Transfer Interrupt Clear register (EOTINTCLR - address 0x2008 C2A4) bit description

Bit	Symbol	Description
31:0	EPTXINTCLR	Clear endpoint xx ($2 \le xx \le 31$) End of Transfer Interrupt request. 0 = No effect. 1 = Clear the EPxx End of Transfer Interrupt request in the EoTIntSt register.

12.10.7.12 USB End of Transfer Interrupt Set register

Writing one to a bit in this register sets the corresponding bit in the EoTIntSt register. Writing zero has no effect. EoTIntSet is a write-only register.

Table 289. USB End of Transfer Interrupt Set register (EOTINTSET - address 0x2008 C2A8) bit description

Bit	Symbol	Description
31:0	EPTXINTSET	Set endpoint xx (2 ≤ xx ≤ 31) End of Transfer Interrupt request. 0 = No effect. 1 = Set the EPxx End of Transfer Interrupt request in the EoTIntSt register.

12.10.7.13 USB New DD Request Interrupt Status register

A bit in this register is set when a transfer is requested from the USB device and no valid DD is detected for the corresponding endpoint. NDDRIntSt is a read-only register.

Table 290. USB New DD Request Interrupt Status register (NDDRINTST - address 0x2008 C2AC) bit description

Bit	Symbol	Description	Reset value
31:0	EPNDDINTST	Endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request. 0 = There is no new DD interrupt request for endpoint xx. 1 = There is a new DD interrupt request for endpoint xx.	0

12.10.7.14 USB New DD Request Interrupt Clear register

Writing one to a bit in this register clears the corresponding bit in the NDDRIntSt register. Writing zero has no effect. NDDRIntClr is a write-only register.

Table 291. USB New DD Request Interrupt Clear register (NDDRINTCLR - address 0x2008 C2B0) bit description

Bit	Symbol	Description
31:0	EPNDDINTCLR	Clear endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request. 0 = No effect. 1 = Clear the EPxx new DD interrupt request in the NDDRIntSt register.

12.10.7.15 USB New DD Request Interrupt Set register

Writing one to a bit in this register sets the corresponding bit in the NDDRIntSt register. Writing zero has no effect. NDDRIntSet is a write-only register

Table 292. USB New DD Request Interrupt Set register (NDDRINTSET - address 0x2008 C2B4) bit description

Bit	Symbol	Description
31:0	EPNDDINTSET	Set endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request. 0 = No effect. 1 = Set the EPxx new DD interrupt request in the NDDRIntSt register.

12.10.7.16 USB System Error Interrupt Status register

If a system error (AHB bus error) occurs when transferring the data or when fetching or updating the DD the corresponding bit is set in this register. SysErrIntSt is a read-only register.

Table 293. USB System Error Interrupt Status register (SYSERRINTST - address 0x2008 C2B8) bit description

Bit	Symbol	Description	Reset value
31:0	EPERRINTST	Endpoint xx ($2 \leq xx \leq 31$) System Error Interrupt request. 0 = There is no System Error Interrupt request for endpoint xx. 1 = There is a System Error Interrupt request for endpoint xx.	0

12.10.7.17 USB System Error Interrupt Clear register

Writing one to a bit in this register clears the corresponding bit in the SysErrIntSt register. Writing zero has no effect. SysErrIntClr is a write-only register.

Table 294. USB System Error Interrupt Clear register (SYSERRINTCLR - address 0x2008 C2BC) bit description

Bit	Symbol	Description
31:0	EPERRINTCLR	Clear endpoint xx ($2 \leq xx \leq 31$) System Error Interrupt request. 0 = No effect. 1 = Clear the EPxx System Error Interrupt request in the SysErrIntSt register.

12.10.7.18 USB System Error Interrupt Set register

Writing one to a bit in this register sets the corresponding bit in the SysErrIntSt register. Writing zero has no effect. SysErrIntSet is a write-only register.

Table 295. USB System Error Interrupt Set register (SYSERRINTSET - address 0x2008 C2C0) bit description

Bit	Symbol	Description
31:0	EPERRINTSET	Set endpoint xx ($2 \leq xx \leq 31$) System Error Interrupt request. 0 = No effect. 1 = Set the EPxx System Error Interrupt request in the SysErrIntSt register.

12.10.8 Clock control registers

12.10.8.1 USB Clock Control register

This register controls the clocking of the USB Device Controller. Whenever software wants to access the device controller registers, both DEV_CLK_EN and AHB_CLK_EN must be set. The PORTSEL_CLK_EN bit need only be set when accessing the PortSel register.

The software does not have to repeat this exercise for every register access, provided that the corresponding ClkCtrl bits are already set. Note that this register is functional only when the PCUSB bit of PCONP is set; when PCUSB is cleared, all clocks to the device controller are disabled irrespective of the contents of this register. ClkCtrl is a read/write register.

Table 296. ClkCtrl register (CLKCTRL - address 0x2008 CFF4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only zero should be written.	NA
1	DEV_CLK_EN	Device clock enable. Enables the usbclk input to the device controller	0
2	-	Reserved. Read value is undefined, only zero should be written.	NA
3	PORTSEL_CLK_EN	Port select register clock enable.	NA
4	AHB_CLK_EN	AHB clock enable	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	NA

12.10.8.2 USB Clock Status register

This register holds the clock availability status. The bits of this register are ORed together to form the USB_NEED_CLK signal. When enabling a clock via ClkCtrl, software should poll the corresponding bit in ClkSt. If it is set, then software can go ahead with the register access. Software does not have to repeat this exercise for every access, provided that the ClkCtrl bits are not disturbed. ClkSt is a read-only register.

Table 297. USB Clock Status register (CLKST - address 0x2008 CFF8) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only zero should be written.	NA
1	DEV_CLK_ON	Device clock on. The usbclk input to the device controller is active.	0
2	-	Reserved. Read value is undefined, only zero should be written.	NA
3	PORTSEL_CLK_ON	Port select register clock on.	NA
4	AHB_CLK_ON	AHB clock on.	0
31:5	-	Reserved. The value read from a reserved bit is not defined.	NA

12.11 Interrupt handling

This section describes how an interrupt event on any of the endpoints is routed to the Nested Vectored Interrupt Controller (NVIC). For a diagram showing interrupt event handling, see Figure 43.

All non-isochronous OUT endpoints (control, bulk, and interrupt endpoints) generate an interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet has been successfully transmitted or when a NAK signal is sent and interrupts on NAK are enabled by the SIE Set Mode command, see Section 12.12.3. For isochronous endpoints, a frame interrupt is generated every 1 ms.

The interrupt handling is different for Slave and DMA mode.

Slave mode

If an interrupt event occurs on an endpoint and the endpoint interrupt is enabled in the EpIntEn register, the corresponding status bit in the EpIntSt is set. For non-isochronous endpoints, all endpoint interrupt events are divided into two types by the corresponding EpIntPri[n] registers: fast endpoint interrupt events and slow endpoint interrupt events. All fast endpoint interrupt events are ORed and routed to bit EP_FAST in the DevIntSt register. All slow endpoint interrupt events are ORed and routed to the EP_SLOW bit in DevIntSt.

For isochronous endpoints, the FRAME bit in DevIntSt is set every 1 ms.

The DevIntSt register holds the status of all endpoint interrupt events as well as the status of various other interrupts (see Section 12.10.2.1). By default, all interrupts (if enabled in DevIntEn) are routed to the USB_INT_REQ_LP bit in the IntSt register to request low priority interrupt handling. However, the DevIntPri register can route either the FRAME or the EP_FAST bit to the USB_INT_REQ_HP bit in the IntSt register.

Only one of the EP_FAST and FRAME interrupt events can be routed to the USB_INT_REQ_HP bit. If routing both bits to USB_INT_REQ_HP is attempted, both interrupt events are routed to USB_INT_REQ_LP.

Slow endpoint interrupt events are always routed directly to the USB_INT_REQ_LP bit for low priority interrupt handling by software.

The final interrupt signal to the NVIC is gated by the EN_USB_INTS bit in the IntSt register. The USB interrupts are routed to the NVIC only if EN_USB_INTS is set.

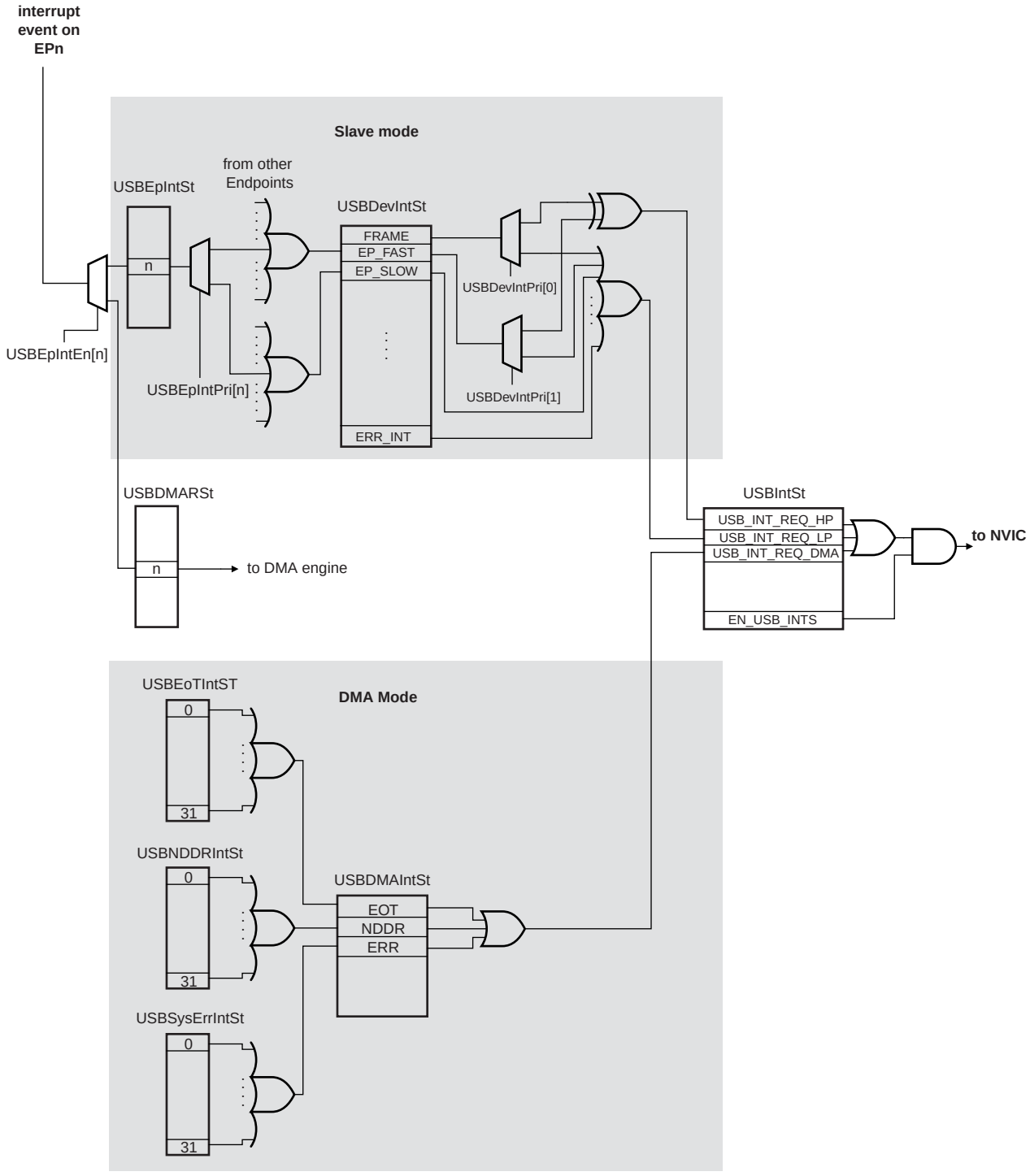
DMA mode

If an interrupt event occurs on a non-control endpoint and the endpoint interrupt is not enabled in the EpIntEn register, the corresponding status bit in the DMARSt is set by hardware. This serves as a flag for the DMA engine to transfer data if DMA transfer is enabled for the corresponding endpoint in the EpDMASt register.

Three types of interrupts can occur for each endpoint for data transfers in DMA mode: End of transfer interrupt, new DD request interrupt, and system error interrupt. These interrupt events set a bit for each endpoint in the respective registers EoTIntSt, NDDRIntSt, and SysErrIntSt. The End of transfer interrupts from all endpoints are then ORed and routed to

the EOT bit in DMAIntSt. Likewise, all New DD request interrupts and system error interrupt events are routed to the NDDR and ERR bits respectively in the DMAStInt register.

The EOT, NDDR, and ERR bits (if enabled in DMAIntEn) are ORed to set the USB_INT_REQ_DMA bit in the IntSt register. If the EN_USB_INTS bit is set in IntSt, the interrupt is routed to the NVIC.



For simplicity, DevIntEn and DMAIntEn are not shown.

Fig 43. Interrupt event handling

12.12 Serial interface engine command description

The functions and registers of the Serial Interface Engine (SIE) are accessed using commands, which consist of a command code followed by optional data bytes (read or write action). The CmdCode (Table 276) and CmdData (Table 277) registers are used for these accesses.

A complete access consists of two phases:

1. **Command phase:** the CmdCode register is written with the CMD_PHASE field set to the value 0x05 (Command), and the CMD_CODE field set to the desired command code. On completion of the command, the CCEMPTY bit of DevIntSt is set.
2. **Data phase (optional):** for writes, the CmdCode register is written with the CMD_PHASE field set to the value 0x01 (Write), and the CMD_WDATA field set with the desired write data. On completion of the write, the CCEMPTY bit of DevIntSt is set. For reads, CmdCode register is written with the CMD_PHASE field set to the value 0x02 (Read), and the CMD_CODE field set with command code the read corresponds to. On completion of the read, the CDFULL bit of DevInSt will be set, indicating the data is available for reading in the CmdData register. In the case of multi-byte registers, the least significant byte is accessed first.

An overview of the available commands is given in Table 298.

Here is an example of the Read Current Frame Number command (reading 2 bytes):

```
DevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL
CmdCode = 0x00F50500;       // CMD_CODE=0xF5, CMD_PHASE=0x05(Command)
while (!(DevIntSt & 0x10)); // Wait for CCEMPTY.
DevIntClr = 0x10;           // Clear CCEMPTY interrupt bit.
CmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(DevIntSt & 0x20)); // Wait for CDFULL.
DevIntClr = 0x20;           // Clear CDFULL.
CurFrameNum = CmdData;      // Read Frame number LSB byte.
CmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(DevIntSt & 0x20)); // Wait for CDFULL.
Temp = CmdData;             // Read Frame number MSB byte
DevIntClr = 0x20;           // Clear CDFULL interrupt bit.
CurFrameNum = CurFrameNum | (Temp << 8);
```

Here is an example of the Set Address command (writing 1 byte):

```
DevIntClr = 0x10;           // Clear CCEMPTY.
CmdCode = 0x00D00500;       // CMD_CODE=0xD0, CMD_PHASE=0x05(Command)
while (!(DevIntSt & 0x10)); // Wait for CCEMPTY.
DevIntClr = 0x10;           // Clear CCEMPTY.
CmdCode = 0x008A0100;       // CMD_WDATA=0x8A(DEV_EN=1, DEV_ADDR=0xA),
                               // CMD_PHASE=0x01(Write)
while (!(DevIntSt & 0x10)); // Wait for CCEMPTY.
DevIntClr = 0x10;           // Clear CCEMPTY.
```

Table 298. SIE command code table

Command name	Recipient	Code (Hex)	Data phase
Device commands			
Set Address	Device	D0	Write 1 byte
Configure Device	Device	D8	Write 1 byte
Set Mode	Device	F3	Write 1 byte
Read Current Frame Number	Device	F5	Read 1 or 2 bytes
Read Test Register	Device	FD	Read 2 bytes
Set Device Status	Device	FE	Write 1 byte
Get Device Status	Device	FE	Read 1 byte
Get Error Code	Device	FF	Read 1 byte
Read Error Status	Device	FB	Read 1 byte
Endpoint Commands			
Select Endpoint	Endpoint 0	00	Read 1 byte (optional)
	Endpoint 1	01	Read 1 byte (optional)
	Endpoint xx	xx	Read 1 byte (optional)
Select Endpoint/Clear Interrupt	Endpoint 0	40	Read 1 byte
	Endpoint 1	41	Read 1 byte
	Endpoint xx	xx + 40	Read 1 byte
Set Endpoint Status	Endpoint 0	40	Write 1 byte
	Endpoint 1	41	Write 1 byte
	Endpoint xx	xx + 40	Write 1 byte
Clear Buffer	Selected Endpoint	F2	Read 1 byte (optional)
Validate Buffer	Selected Endpoint	FA	None

12.12.1 Set Address (Command: 0xD0, Data: write 1 byte)

The Set Address command is used to set the USB assigned address and enable the (embedded) function. The address set in the device will take effect after the status stage of the control transaction. After a bus reset, DEV_ADDR is set to 0x00, and DEV_EN is set to 1. The device will respond to packets for function address 0x00, endpoint 0 (default endpoint).

Table 299. Set Address command bit description

Bit	Symbol	Description	Reset value
6:0	DEV_ADDR	Device address set by the software. After a bus reset this field is set to 0x00.	0
7	DEV_EN	Device Enable. After a bus reset this bit is set to 1. 0: Device will not respond to any packets. 1: Device will respond to packets for function address DEV_ADDR.	0

12.12.2 Configure Device (Command: 0xD8, Data: write 1 byte)

A value of 1 written to the register indicates that the device is configured and all the enabled non-control endpoints will respond. Control endpoints are always enabled and respond even if the device is not configured, in the default state.

Table 300. Configure Device command bit description

Bit	Symbol	Description	Reset value
0	CONF_DEVICE	Device is configured. All enabled non-control endpoints will respond. This bit is cleared by hardware when a bus reset occurs. When set, the UP_LED signal is driven LOW if the device is not in the suspended state (SUS=0).	0
7:1	-	Reserved. Read value is undefined, only zero should be written.	NA

12.12.3 Set Mode (Command: 0xF3, Data: write 1 byte)

Table 301. Set Mode command bit description

Bit	Symbol	Value	Description	Reset value
0	AP_CLK		Always PLL Clock.	0
		0	USB_NEED_CLK is functional; the 48 MHz clock can be stopped when the device enters suspend state.	
		1	USB_NEED_CLK is fixed to 1; the 48 MHz clock cannot be stopped when the device enters suspend state.	
1	INAK_CI		Interrupt on NAK for Control IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
2	INAK_CO		Interrupt on NAK for Control OUT endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
3	INAK_II		Interrupt on NAK for Interrupt IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
4	INAK_IO[1]		Interrupt on NAK for Interrupt OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
5	INAK_BI		Interrupt on NAK for Bulk IN endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
6	INAK_BO[2]		Interrupt on NAK for Bulk OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
7	-		Reserved. Read value is undefined, only zero should be written.	NA

[1] This bit should be reset to 0 if the DMA is enabled for any of the Interrupt OUT endpoints.

[2] This bit should be reset to 0 if the DMA is enabled for any of the Bulk OUT endpoints.

12.12.4 Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes)

Returns the frame number of the last successfully received SOF. The frame number is eleven bits wide. The frame number returns least significant byte first. In case the user is only interested in the lower 8 bits of the frame number, only the first byte needs to be read.

- 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.

12.12.5 Read Test Register (Command: 0xFD, Data: read 2 bytes)

The test register is 16 bits wide. It returns the value of 0xA50F if the USB clocks (usbclk and AHB slave clock) are running.

12.12.6 Set Device Status (Command: 0xFE, Data: write 1 byte)

The Set Device Status command sets bits in the Device Status Register.

Table 302. Set Device Status command bit description

Bit	Symbol	Value	Description	Reset value
0	CON		The Connect bit indicates the current connect status of the device. It controls the CONNECT output pin, used for SoftConnect. Reading the connect bit returns the current connect status. This bit is cleared by hardware when the V_{BUS} status input is LOW for more than 3 ms. The 3 ms delay filters out temporary dips in the V_{BUS} voltage.	0
		0	Writing a 0 will make the CONNECT pin go HIGH.	
		1	Writing a 1 will make the CONNECT pin go LOW.	
1	CON_CH		Connect Change.	0
		0	This bit is cleared when read.	
		1	This bit is set when the device's pull-up resistor is disconnected because V_{BUS} disappeared. The DEV_STAT interrupt is generated when this bit is 1.	
2	SUS		Suspend: The Suspend bit represents the current suspend state. When the device is suspended (SUS = 1) and the CPU writes a 0 into it, the device will generate a remote wake-up. This will only happen when the device is connected (CON = 1). When the device is not connected or not suspended, writing a 0 has no effect. Writing a 1 to this bit has no effect.	0
		0	This bit is reset to 0 on any activity.	
		1	This bit is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 ms.	
3	SUS_CH		Suspend (SUS) bit change indicator. The SUS bit can toggle because: • The device goes into the suspended state. • The device is disconnected. • The device receives resume signalling on its upstream port. This bit is cleared when read.	0
		0	SUS bit not changed.	
		1	SUS bit changed. At the same time a DEV_STAT interrupt is generated.	
4	RST		Bus Reset bit. On a bus reset, the device will automatically go to the default state. In the default state: • Device is unconfigured. • Will respond to address 0. • Control endpoint will be in the Stalled state. • All endpoints are unrealized except control endpoints EP0 and EP1. • Data toggling is reset for all endpoints. • All buffers are cleared. • There is no change to the endpoint interrupt status. • DEV_STAT interrupt is generated. Note: Bus resets are ignored when the device is not connected (CON=0).	0
		0	This bit is cleared when read.	
		1	This bit is set when the device receives a bus reset. A DEV_STAT interrupt is generated.	
7:5	-		Reserved. Read value is undefined, only zero should be written.	NA

12.12.7 Get Device Status (Command: 0xFE, Data: read 1 byte)

The Get Device Status command returns the Device Status Register. Reading the device status returns 1 byte of data. The bit field definition is same as the Set Device Status Register as shown in Table 302.

Remark: To ensure correct operation, the DEV_STAT bit of DevIntSt must be cleared before executing the Get Device Status command.

12.12.8 Get Error Code (Command: 0xFF, Data: read 1 byte)

Different error conditions can arise inside the SIE. The Get Error Code command returns the last error code that occurred. The 4 least significant bits form the error code.

Table 303. Get Error Code command bit description

Bit	Symbol	Value	Description	Reset value
3:0	EC		Error Code.	0
		0000	No Error.	
		0001	PID Encoding Error.	
		0010	Unknown PID.	
		0011	Unexpected Packet - any packet sequence violation from the specification.	
		0100	Error in Token CRC.	
		0101	Error in Data CRC.	
		0110	Time Out Error.	
		0111	Babble.	
		1000	Error in End of Packet.	
		1001	Sent/Received NAK.	
		1010	Sent Stall.	
		1011	Buffer Overrun Error.	
		1100	Sent Empty Packet (ISO Endpoints only).	
		1101	Bitstuff Error.	
		1110	Error in Sync.	
		1111	Wrong Toggle Bit in Data PID, ignored data.	
4	EA	-	The Error Active bit will be reset once this register is read.	
7:5	-		Reserved. Read value is undefined, only zero should be written.	NA

12.12.9 Read Error Status (Command: 0xFB, Data: read 1 byte)

This command reads the 8-bit Error register from the USB device. This register records which error events have recently occurred in the SIE. If any of these bits are set, the ERR_INT bit of DevIntSt is set. The error bits are cleared after reading this register.

Table 304. Read Error Status command bit description

Bit	Symbol	Description	Reset value
0	PID_ERR	PID encoding error or Unknown PID or Token CRC.	0
1	UEPKT	Unexpected Packet - any packet sequence violation from the specification.	0
2	DCRC	Data CRC error.	0
3	TIMEOUT	Time out error.	0
4	EOP	End of packet error.	0
5	B_OVRN	Buffer Overrun.	0
6	BTSTF	Bit stuff error.	0
7	TGL_ERR	Wrong toggle bit in data PID, ignored data.	0

12.12.10 Select Endpoint (Command: 0x00 - 0x1F, Data: read 1 byte (optional))

The Select Endpoint command initializes an internal pointer to the start of the selected buffer in EP_RAM. Optionally, this command can be followed by a data read, which returns some additional information on the packet(s) in the endpoint buffer(s). The command code of the Select Endpoint command is equal to the physical endpoint number. In the case of a single buffered endpoint the B_2_FULL bit is not valid.

Table 305. Select Endpoint command bit description

Bit	Symbol	Value	Description	Reset value
0	FE		Full/Empty. This bit indicates the full or empty status of the endpoint buffer(s). For IN endpoints, the FE bit gives the ANDed result of the B_1_FULL and B_2_FULL bits. For OUT endpoints, the FE bit gives ORed result of the B_1_FULL and B_2_FULL bits. For single buffered endpoints, this bit simply reflects the status of B_1_FULL.	0
		0	For an IN endpoint, at least one write endpoint buffer is empty.	
		1	For an OUT endpoint, at least one endpoint read buffer is full.	
1	ST		Stalled endpoint indicator.	0
		0	The selected endpoint is not stalled.	
		1	The selected endpoint is stalled.	
2	STP		SETUP bit: the value of this bit is updated after each successfully received packet (i.e. an ACKed package on that particular physical endpoint).	0
		0	The STP bit is cleared by doing a Select Endpoint/Clear Interrupt on this endpoint.	
		1	The last received packet for the selected endpoint was a SETUP packet.	
3	PO		Packet over-written bit.	0
		0	The PO bit is cleared by the ‘Select Endpoint/Clear Interrupt’ command.	
		1	The previously received packet was over-written by a SETUP packet.	
4	EPN		EP NAKed bit indicates sending of a NAK. If the host sends an OUT packet to a filled OUT buffer, the device returns NAK. If the host sends an IN token packet to an empty IN buffer, the device returns NAK.	0
		0	The EPN bit is reset after the device has sent an ACK after an OUT packet or when the device has seen an ACK after sending an IN packet.	
		1	The EPN bit is set when a NAK is sent and the interrupt on NAK feature is enabled.	
5	B_1_FULL		The buffer 1 status.	0
		0	Buffer 1 is empty.	
		1	Buffer 1 is full.	
6	B_2_FULL		The buffer 2 status.	0
		0	Buffer 2 is empty.	
		1	Buffer 2 is full.	
7	-		Reserved. Read value is undefined, only zero should be written.	NA

12.12.11 Select Endpoint/Clear Interrupt (Command: 0x40 - 0x5F, Data: read 1 byte)

Commands 0x40 to 0x5F are identical to their Select Endpoint equivalents, with the following differences:

- They clear the bit corresponding to the endpoint in the EpIntSt register.
- In case of a control OUT endpoint, they clear the STP and PO bits in the corresponding Select Endpoint Register.
- Reading one byte is obligatory.

Remark: This command may be invoked by using the CmdCode and CmdData registers, or by setting the corresponding bit in EpIntClr. For ease of use, using the EpIntClr register is recommended.

12.12.12 Set Endpoint Status (Command: 0x40 - 0x55, Data: write 1 byte (optional))

The Set Endpoint Status command sets status bits 7:5 and 0 of the endpoint. The Command Code of Set Endpoint Status is equal to the sum of 0x40 and the physical endpoint number in hex. Not all bits can be set for all types of endpoints.

Table 306. Set Endpoint Status command bit description

Bit	Symbol	Value	Description	Reset value
0	ST		Stalled endpoint bit. A Stalled control endpoint is automatically unstalled when it receives a SETUP token, regardless of the content of the packet. If the endpoint should stay in its stalled state, the CPU can stall it again by setting this bit. When a stalled endpoint is unstalled - either by the Set Endpoint Status command or by receiving a SETUP token - it is also re-initialized. This flushes the buffer: in case of an OUT buffer it waits for a DATA 0 PID; in case of an IN buffer it writes a DATA 0 PID. There is no change of the interrupt status of the endpoint. When already unstalled, writing a zero to this bit initializes the endpoint. When an endpoint is stalled by the Set Endpoint Status command, it is also re-initialized.	0
		0	The endpoint is unstalled.	
		1	The endpoint is stalled.	
4:1	-		Reserved. Read value is undefined, only zero should be written.	NA
5	DA		Disabled endpoint bit.	0
		0	The endpoint is enabled.	
		1	The endpoint is disabled.	
6	RF_MO		Rate Feedback Mode.	0
		0	Interrupt endpoint is in the Toggle mode.	
		1	Interrupt endpoint is in the Rate Feedback mode. This means that transfer takes place without data toggle bit.	
7	CND_ST		Conditional Stall bit.	0
		0	Unstalls both control endpoints.	
		1	Stall both control endpoints, unless the STP bit is set in the Select Endpoint register. It is defined only for control OUT endpoints.	

12.12.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional))

When an OUT packet sent by the host has been received successfully, an internal hardware FIFO status Buffer_Full flag is set. All subsequent packets will be refused by returning a NAK. When the device software has read the data, it should free the buffer by issuing the Clear Buffer command. This clears the internal Buffer_Full flag. When the buffer is cleared, new packets will be accepted.

When bit 0 of the optional data byte is 1, the previously received packet was over-written by a SETUP packet. The Packet over-written bit is used only in control transfers. According to the USB specification, a SETUP packet should be accepted irrespective of the buffer status. The software should always check the status of the PO bit after reading the SETUP data. If it is set then it should discard the previously read data, clear the PO bit by issuing a Select Endpoint/Clear Interrupt command, read the new SETUP data and again check the status of the PO bit.

See Section 12.14 “Slave mode operation” for a description of when this command is used.

Table 307. Clear Buffer command bit description

Bit	Symbol	Value	Description	Reset value
0	PO		Packet over-written bit. This bit is only applicable to the control endpoint EP0.	0
		0	The previously received packet is intact.	
		1	The previously received packet was over-written by a later SETUP packet.	
7:1	-		Reserved. Read value is undefined, only zero should be written.	NA

12.12.14 Validate Buffer (Command: 0xFA, Data: none)

When the CPU has written data into an IN buffer, software should issue a Validate Buffer command. This tells hardware that the buffer is ready for sending on the USB bus. Hardware will send the contents of the buffer when the next IN token packet is received.

Internally, there is a hardware FIFO status flag called Buffer_Full. This flag is set by the Validate Buffer command and cleared when the data has been sent on the USB bus and the buffer is empty.

A control IN buffer cannot be validated when its corresponding OUT buffer has the Packet Over-written (PO) bit (see the Clear Buffer Register) set or contains a pending SETUP packet. For the control endpoint the validated buffer will be invalidated when a SETUP packet is received.

See Section 12.14 “Slave mode operation” for a description of when this command is used.

12.13 USB device controller initialization

The USB device controller initialization includes the following steps:

1. Enable the device controller by setting the PCUSB bit of PCONP.
2. Configure and enable the PLL and Clock Dividers to provide 48 MHz for usbclk and the desired frequency for cclk. For the procedure for determining the PLL setting and configuration, see Section 3.10.5 “Procedure for determining PLL settings”.
3. Enable the device controller clocks by setting DEV_CLK_EN and AHB_CLK_EN bits in the ClkCtrl register. Poll the respective clock bits in the ClkSt register until they are set.
4. Select the desired USB port pins using the PortSel register. The PORTSEL_CLK_EN bit must be set in ClkCtrl before accessing PortSel and should be cleared after accessing PortSel.
5. Enable the USB pin functions by writing to the corresponding IOCON registers.
6. Disable the pull-ups and pull-downs on the V_{BUS} pin using the corresponding IOCON register by putting the pin in the “plain-input” mode. See Section 7.4.1 “I/O configuration register contents (IOCON)”.
7. Set EpIn and MaxPSize registers for EP0 and EP1, and wait until the EP_RLZED bit in DevIntSt is set so that EP0 and EP1 are realized.
8. Enable endpoint interrupts (Slave mode):
   - Clear all endpoint interrupts using EpIntClr.
   - Clear any device interrupts using DevIntClr.
   - Enable Slave mode for the desired endpoints by setting the corresponding bits in EpIntEn.
   - Set the priority of each enabled interrupt using EpIntPri.
   - Configure the desired interrupt mode using the SIE Set Mode command.
   - Enable device interrupts using DevIntEn (normally DEV_STAT, EP_SLOW, and possibly EP_FAST).
9. Configure the DMA (DMA mode):
   - Disable DMA operation for all endpoints using EpDMADis.
   - Clear any pending DMA requests using DMARClr.
   - Clear all DMA interrupts using EoTIntClr, NDDRIntClr, and SysErrIntClr.
   - Prepare the UDCA in system memory.
   - Write the desired address for the UDCA to UDCAH.
   - Enable the desired endpoints for DMA operation using EpDMAEn.
   - Set EOT, DDR, and ERR bits in DMAIntEn.
10. Install USB interrupt handler in the NVIC by writing its address to the appropriate vector table location and enabling the USB interrupt in the NVIC.
11. Set default USB address to 0x0 and DEV_EN to 1 using the SIE Set Address command. A bus reset will also cause this to happen.
12. Set CON bit to 1 to make CONNECT active using the SIE Set Device Status command.

The configuration of the endpoints varies depending on the software application. By default, all the endpoints are disabled except control endpoints EP0 and EP1. Additional endpoints are enabled and configured by software after a SET_CONFIGURATION or SET_INTERFACE device request is received from the host.

12.14 Slave mode operation

In Slave mode, the CPU transfers data between RAM and the endpoint buffer using the Register Interface.

12.14.1 Interrupt generation

In slave mode, data packet transfer between RAM and an endpoint buffer can be initiated in response to an endpoint interrupt. Endpoint interrupts are enabled using the EpIntEn register, and are observable in the EpIntSt register.

All non-isochronous OUT endpoints generate an endpoint interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet is successfully transmitted, or when a NAK handshake is sent on the bus and the interrupt on NAK feature is enabled.

For Isochronous endpoints, transfer of data is done when the FRAME interrupt (in DevIntSt) occurs.

12.14.2 Data transfer for OUT endpoints

When the software wants to read the data from an endpoint buffer it should set the RD_EN bit and program LOG_ENDPOINT with the desired endpoint number in the Ctrl register. The control logic will fetch the packet length to the RxPLen register, and set the PKT_RDY bit (Table 272).

Software can now start reading the data from the RxData register (Table 271). When the end of packet is reached, the RD_EN bit is cleared, and the RxENDPKT bit is set in the DevSt register. Software now issues a Clear Buffer (refer to Table 307) command. The endpoint is now ready to accept the next packet. For OUT isochronous endpoints, the next packet will be received irrespective of whether the buffer has been cleared. Any data not read from the buffer before the end of the frame is lost. See Section 12.16 “Double buffered endpoint operation” for more details.

If the software clears RD_EN before the entire packet is read, reading is terminated, and the data remains in the endpoint’s buffer. When RD_EN is set again for this endpoint, the data will be read from the beginning.

12.14.3 Data transfer for IN endpoints

When writing data to an endpoint buffer, WR_EN (Section 12.10.5.5 “USB Control register”) is set and software writes to the number of bytes it is going to send in the packet to the TxPLen register (Section 12.10.5.4). It can then write data continuously in the TxData register.

When the number of bytes programmed in TxPLen have been written to TxData, the WR_EN bit is cleared, and the TxENDPKT bit is set in the DevIntSt register. Software issues a Validate Buffer (Section 12.12.14 “Validate Buffer (Command: 0xFA, Data: none)”) command. The endpoint is now ready to send the packet. For IN isochronous endpoints, the data in the buffer will be sent only if the buffer is validated before the next FRAME interrupt occurs; otherwise, an empty packet will be sent in the next frame. If the software clears WR_EN before the entire packet is written, writing will start again from the beginning the next time WR_EN is set for this endpoint.

Both RD_EN and WR_EN can be high at the same time for the same logical endpoint. Interleaved read and write operation is possible.

12.15 DMA operation

In DMA mode, the DMA transfers data between RAM and the endpoint buffer.

The following sections discuss DMA mode operation. Background information is given in sections Section 12.15.2 “USB device communication area” and Section 12.15.3 “Triggering the DMA engine”. The fields of the DMA Descriptor are described in Section 12.15.4 “The DMA descriptor”. The last three sections describe DMA operation: Section 12.15.5 “Non-isochronous endpoint operation”, Section 12.15.6 “Isochronous endpoint operation”, and Section 12.15.7 “Auto Length Transfer Extraction (ATLE) mode operation”.

12.15.1 Transfer terminology

Within this section three types of transfers are mentioned:

1. USB transfers – transfer of data over the USB bus. The USB 2.0 specification refers to these simply as transfers. Within this section they are referred to as USB transfers to distinguish them from DMA transfers. A USB transfer is composed of transactions. Each transaction is composed of packets.
2. DMA transfers – the transfer of data between an endpoint buffer and system memory (RAM).
3. Packet transfers – in this section, a packet transfer refers to the transfer of a packet of data between an endpoint buffer and system memory (RAM). A DMA transfer is composed of one or more packet transfers.

12.15.2 USB device communication area

The CPU and DMA controller communicate through a common area of memory, called the USB Device Communication Area, or UDCA. The UDCA is a 32-word array of DMA Descriptor Pointers (DDPs), each of which corresponds to a physical endpoint. Each DDP points to the start address of a DMA Descriptor, if one is defined for the endpoint. DDPs for unrealized endpoints and endpoints disabled for DMA operation are ignored and can be set to a NULL (0x0) value.

The start address of the UDCA is stored in the UDCAH register. The UDCA can reside at any 128-byte boundary of RAM that is accessible to both the CPU and DMA controller.

Figure 44 illustrates the UDCA and its relationship to the UDCA Head (UDCAH) register and DMA Descriptors.

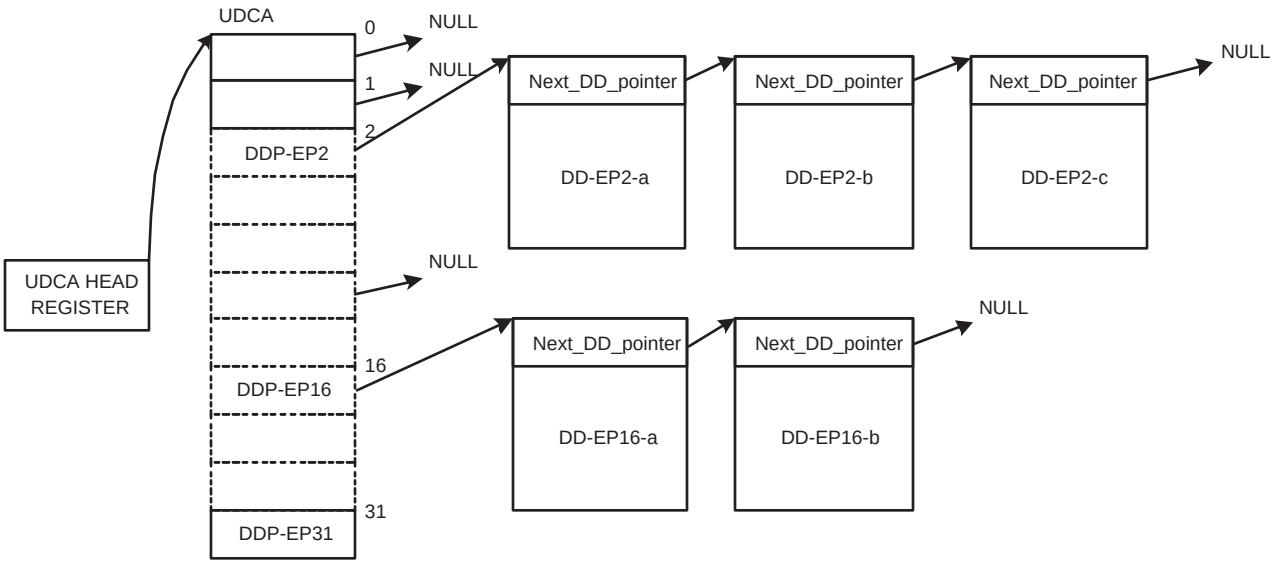


Fig 44. UDCA Head register and DMA Descriptors

12.15.3 Triggering the DMA engine

An endpoint raises a DMA request when Slave mode is disabled by setting the corresponding bit in the EpIntEn register to 0 (Section 12.10.3.2) and an endpoint interrupt occurs (see Section 12.10.7.1 “USB DMA Request Status register”).

A DMA transfer for an endpoint starts when the endpoint is enabled for DMA operation in EpDMASt, the corresponding bit in DMARSt is set, and a valid DD is found for the endpoint.

All endpoints share a single DMA channel to minimize hardware overhead. If more than one DMA request is active in DMARSt, the endpoint with the lowest physical endpoint number is processed first.

In DMA mode, the bits corresponding to Interrupt on NAK for Bulk OUT and Interrupt OUT endpoints (INAK_BO and INAK_IO) should be set to 0 using the SIE Set Mode command (Section 12.12.3).

12.15.4 The DMA descriptor

DMA transfers are described by a data structure called the DMA Descriptor (DD).

DDs are placed in RAM. These descriptors can be located anywhere in on-chip RAM at word-aligned addresses.

DDs for non-isochronous endpoints are four words long. DDs for isochronous endpoints are five words long.

The parameters associated with a DMA transfer are:

- The start address of the DMA buffer
- The length of the DMA buffer

- The start address of the next DMA descriptor
- Control information
- Count information (number of bytes transferred)
- Status information

Table 308 lists the DMA descriptor fields.

Table 308. DMA descriptor

Word position	Access (H/W)	Access (S/W)	Bit position	Description
0	R	R/W	31:0	Next_DD_pointer
1	R	R/W	1:0	DMA_mode (00 -Normal; 01 - ATLE)
	R	R/W	2	Next_DD_valid (1 - valid; 0 - invalid)
	-	-	3	Reserved. Read value is undefined, only zero should be written.
	R	R/W	4	Isochronous_endpoint (1 - isochronous; 0 - non-isochronous)
	R	R/W	15:5	Max_packet_size
	R/W[1]	R/W	31:16	DMA_buffer_length This value is specified in bytes for non-isochronous endpoints and in number of packets for isochronous endpoints.
2	R/W	R/W	31:0	DMA_buffer_start_addr
3	R/W	R/I	0	DD_retired (To be initialized to 0)
	W	R/I	4:1	DD_status (To be initialized to 0000): 0000 - NotServiced 0001 - BeingServiced 0010 - NormalCompletion 0011 - DataUnderrun (short packet) 1000 - DataOverrun 1001 - SystemError
	W	R/I	5	Packet_valid (To be initialized to 0)
	W	R/I	6	LS_byte_extracted (ATLE mode) (To be initialized to 0)
	W	R/I	7	MS_byte_extracted (ATLE mode) (To be initialized to 0)
	R	W	13:8	Message_length_position (ATLE mode)
	-	-	15:14	Reserved. Read value is undefined, only zero should be written.
	R/W	R/I	31:16	Present_DMA_count (To be initialized to 0)
4	R/W	R/W	31:0	Isochronous_packetsize_memory_address

[1] Write-only in ATLE mode

Legend: R - Read; W - Write; I - Initialize

12.15.4.1 Next_DD_pointer

Pointer to the memory location from where the next DMA descriptor will be fetched.

12.15.4.2 DMA_mode

Specifies the DMA mode of operation. Two modes have been defined: Normal and Automatic Transfer Length Extraction (ATLE) mode. In normal mode, software initializes the DMA_buffer_length for OUT endpoints. In ATLE mode, the DMA_buffer_length is

extracted from the incoming data. See Section 12.15.7 “Auto Length Transfer Extraction (ATLE) mode operation” on page 380 for more details.

12.15.4.3 Next_DD_valid

This bit indicates whether the software has prepared the next DMA descriptor. If set, the DMA engine fetches the new descriptor when it is finished with the current one.

12.15.4.4 Isochronous_endpoint

When set, this bit indicates that the descriptor belongs to an isochronous endpoint. Hence 5 words have to be read when fetching it.

12.15.4.5 Max_packet_size

The maximum packet size of the endpoint. This parameter is used while transferring the data for IN endpoints from the memory. It is used for OUT endpoints to detect the short packet. This is applicable to non-isochronous endpoints only. This field should be set to the same MPS value that is assigned for the endpoint using the MaxPSize register.

12.15.4.6 DMA_buffer_length

This indicates the depth of the DMA buffer allocated for transferring the data. The DMA engine will stop using this descriptor when this limit is reached and will look for the next descriptor.

In Normal mode operation, software sets this value for both IN and OUT endpoints. In ATLE mode operation, software sets this value for IN endpoints only. For OUT endpoints, hardware sets this value using the extracted length of the data stream.

For isochronous endpoints, DMA_buffer_length is specified in number of packets, for non-isochronous endpoints in bytes.

12.15.4.7 DMA_buffer_start_addr

The address where the data is read from or written to. This field is updated each time the DMA engine finishes transferring a packet.

12.15.4.8 DD_retired

This bit is set by hardware when the DMA engine finishes the current descriptor. This happens when the end of the buffer is reached, a short packet is transferred (non-isochronous endpoints), or an error condition is detected.

12.15.4.9 DD_status

The status of the DMA transfer is encoded in this field. The following codes are defined:

- **NotServiced** - No packet has been transferred yet.
- **BeingServiced** - At least one packet is transferred.
- **NormalCompletion** - The DD is retired because the end of the buffer is reached and there were no errors. The DD_retired bit is also set.
- **DataUnderrun** - Before reaching the end of the DMA buffer, the USB transfer is terminated because a short packet is received. The DD_retired bit is also set.

- **DataOverrun** - The end of the DMA buffer is reached in the middle of a packet transfer. This is an error situation. The DD_retired bit is set. The present DMA count field is equal to the value of DMA_buffer_length. The packet must be re-transmitted from the endpoint buffer in another DMA transfer. The corresponding EPxx_DMA_ENABLE bit in EpDMASt is cleared.
- **SystemError** - The DMA transfer being serviced is terminated because of an error on the AHB bus. The DD_retired bit is not set in this case. The corresponding EPxx_DMA_ENABLE in EpDMASt is cleared. Since a system error can happen while updating the DD, the DD fields in RAM may be unreliable.

12.15.4.10 Packet_valid

This bit is used for isochronous endpoints. It indicates whether the last packet transferred to the memory is received with errors or not. This bit is set if the packet is valid, i.e., it was received without errors. See Section 12.15.6 “Isochronous endpoint operation” on page 378 for isochronous endpoint operation.

This bit is unnecessary for non-isochronous endpoints because a DMA request is generated only for packets without errors, and thus Packet_valid will always be set when the request is generated.

12.15.4.11 LS_byte_extracted

Used in ATLE mode. When set, this bit indicates that the Least Significant Byte (LSB) of the transfer length has been extracted. The extracted size is reflected in the DMA_buffer_length field, bits 23:16.

12.15.4.12 MS_byte_extracted

Used in ATLE mode. When set, this bit indicates that the Most Significant Byte (MSB) of the transfer size has been extracted. The size extracted is reflected in the DMA_buffer_length field, bits 31:24. Extraction stops when LS_Byte_extracted and MS_byte_extracted bits are set.

12.15.4.13 Present_DMA_count

The number of bytes transferred by the DMA engine. The DMA engine updates this field after completing each packet transfer.

For isochronous endpoints, Present_DMA_count is the number of packets transferred; for non-isochronous endpoints, Present_DMA_count is the number of bytes.

12.15.4.14 Message_length_position

Used in ATLE mode. This field gives the offset of the message length position embedded in the incoming data packets. This is applicable only for OUT endpoints. Offset 0 indicates that the message length starts from the first byte of the first packet.

12.15.4.15 Isochronous_packetsize_memory_address

The memory buffer address where the packet size information along with the frame number has to be transferred or fetched. See Figure 45. This is applicable to isochronous endpoints only.

12.15.5 Non-isochronous endpoint operation

12.15.5.1 Setting up DMA transfers

Software prepares the DMA Descriptors (DDs) for those physical endpoints to be enabled for DMA transfer. These DDs are present in on-chip RAM. The start address of the first DD is programmed into the DMA Description pointer (DDP) location for the corresponding endpoint in the UDCA. Software then sets the EPxx_DMA_ENABLE bit for this endpoint in the EpDMAEn register (Section 12.10.7.6).The DMA_mode bit field in the descriptor is set to '00' for normal mode operation. All other DD fields are initialized as specified in Table 308.

DMA operation is not supported for physical endpoints 0 and 1 (default control endpoints).

12.15.5.2 Finding DMA Descriptor

When there is a trigger for a DMA transfer for an endpoint, the DMA engine will first determine whether a new descriptor has to the fetched or not. A new descriptor does not have to be fetched if the last packet transferred was for the same endpoint and the DD is not yet in the retired state. An internal flag called DMA_PROCEED is used to identify this condition (see Section 12.15.5.4 "Optimizing descriptor fetch" on page 377).

If a new descriptor has to be read, the DMA engine will calculate the location of the DDP for this endpoint and will fetch the start address of the DD from this location. A DD start address at location zero is considered invalid. In this case the NDDR interrupt is raised. All other word-aligned addresses are considered valid.

When the DD is fetched, the DD status word (word 3) is read first and the status of the DD_retired bit is checked. If not set, DDP points to a valid DD. If DD_retired is set, the DMA engine will read the control word (word 1) of the DD.

If Next_DD_valid bit is set, the DMA engine will fetch the Next_DD_pointer field (word 0) of the DD and load it to the DDP. The new DDP is written to the UDCA area.

The full DD (4 words) will then be fetched from the address in the DDP. The DD will give the details of the DMA transfer to be done. The DMA engine will load its hardware resources with the information fetched from the DD (start address, DMA count etc.).

If Next_DD_valid is not set and DD_retired bit is set, the DMA engine raises the NDDR interrupt for this endpoint and clears the corresponding EPxx_DMA_ENABLE bit.

12.15.5.3 Transferring the data

For OUT endpoints, the current packet is read from the EP_RAM by the DMA Engine and transferred to on-chip RAM memory locations starting from DMA_buffer_start_addr. For IN endpoints, the data is fetched from on-chip RAM at DMA_buffer_start_addr and written to the EP_RAM. The DMA_buffer_start_addr and Present_DMA_count fields are updated after each packet is transferred.

12.15.5.4 Optimizing descriptor fetch

A DMA transfer normally involves multiple packet transfers. Hardware will not re-fetch a new DD from memory unless the endpoint changes. To indicate an ongoing multi-packet transfer, hardware sets an internal flag called DMA_PROCEED.

The DMA_PROCEED flag is cleared after the required number of bytes specified in the DMA_buffer_length field is transferred. It is also cleared when the software writes into the EpDMADis register. The ability to clear the DMA_PROCEED flag allows software to force the DD to be re-fetched for the next packet transfer. Writing all zeros into the EpDMADis register clears the DMA_PROCEED flag without disabling DMA operation for any endpoint.

12.15.5.5 Ending the packet transfer

On completing a packet transfer, the DMA engine writes back the DD with updated status information to the same memory location from where it was read. The DMA_buffer_start_addr, Present_DMA_count, and the DD_status fields in the DD are updated.

A DD can have the following types of completion:

Normal completion - If the current packet is fully transferred and the Present_DMA_count field equals the DMA_buffer_length, the DD has completed normally. The DD will be written back to memory with DD_retired set and DD_status set to NormalCompletion. The EOT interrupt is raised for this endpoint.

USB transfer end completion - If the current packet is fully transferred and its size is less than the Max_packet_size field, and the end of the DMA buffer is still not reached, the USB transfer end completion occurs. The DD will be written back to the memory with DD_retired set and DD_Status set to the DataUnderrun completion code. The EOT interrupt is raised for this endpoint.

Error completion - If the current packet is partially transferred i.e. the end of the DMA buffer is reached in the middle of the packet transfer, an error situation occurs. The DD is written back with DD_retired set and DD_status set to the DataOverrun status code. The EOT interrupt is raised for this endpoint and the corresponding bit in EpDMASt register is cleared. The packet will be re-sent from the endpoint buffer to memory when the corresponding EPxx_DMA_ENABLE bit is set again using the EpDMAEn register.

12.15.5.6 No_Packet DD

For an IN transfer, if the system does not have any data to send for a while, it can respond to an NDDR interrupt by programming a No_Packet DD. This is done by setting both the Max_packet_size and DMA_buffer_length fields in the DD to 0. On processing a No_Packet DD, the DMA engine clears the DMA request bit in DMARSt corresponding to the endpoint without transferring a packet. The DD is retired with a status code of NormalCompletion. This can be repeated as often as necessary. The device will respond to IN token packets on the USB bus with a NAK until a DD with a data packet is programmed and the DMA transfers the packet into the endpoint buffer.

12.15.6 Isochronous endpoint operation

For isochronous endpoints, the packet size can vary for each packet. There is one packet per isochronous endpoint for each frame.

12.15.6.1 Setting up DMA transfers

Software sets the isochronous endpoint bit to 1 in the DD, and programs the initial value of the Isochronous_packetsize_memory_address field. All other fields are initialized the same as for non-isochronous endpoints.

For isochronous endpoints, the DMA_buffer_length and Present_DMA_count fields are in frames rather than bytes.

12.15.6.2 Finding the DMA Descriptor

Finding the descriptors is done in the same way as that for a non-isochronous endpoint.

A DMA request will be placed for DMA-enabled isochronous endpoints on every FRAME interrupt. On processing the request, the DMA engine will fetch the descriptor and if Isochronous_endpoint is set, will fetch the Isochronous_packetsize_memory_address from the fifth word of the DD.

12.15.6.3 Transferring the Data

The data is transferred to or from the memory location DMA_buffer_start_addr. After the end of the packet transfer the Present_DMA_count value is incremented by 1.

The isochronous packet size is stored in memory as shown in Figure 45. Each word in the packet size memory shown is divided into fields: Frame_number (bits 31 to 17), Packet_valid (bit 16), and Packet_length (bits 15 to 0). The space allocated for the packet size memory for a given DD should be DMA_buffer_length words in size – one word for each packet to transfer.

OUT endpoints

At the completion of each frame, the packet size is written to the address location in Isochronous_packet_size_memory_address, and Isochronous_packet_size_memory_address is incremented by 4.

IN endpoints

Only the Packet_length field of the isochronous packet size word is used. For each frame, an isochronous data packet of size specified by this field is transferred from the USB device to the host, and Isochronous_packet_size_memory_address is incremented by 4 at the end of the packet transfer. If Packet_length is zero, an empty packet will be sent by the USB device.

12.15.6.4 DMA descriptor completion

DDs for isochronous endpoints can only end with a status code of NormalCompletion since there is no short packet on Isochronous endpoints, and the USB transfer continues indefinitely until a SystemError occurs. There is no DataOverrun detection for isochronous endpoints.

12.15.6.5 Isochronous OUT Endpoint Operation Example

Assume that an isochronous endpoint is programmed for the transfer of 10 frames and that the transfer begins when the frame number is 21. After transferring four frames with packet sizes of 10,15, 8 and 20 bytes without errors, the descriptor and memory map appear as shown in Figure 45.

The_total_number_of_bytes_transferred = 0x0A + 0x0F + 0x08 + 0x14 = 0x35.

The Packet_valid bit (bit 16) of all the words in the packet length memory is set to 1.

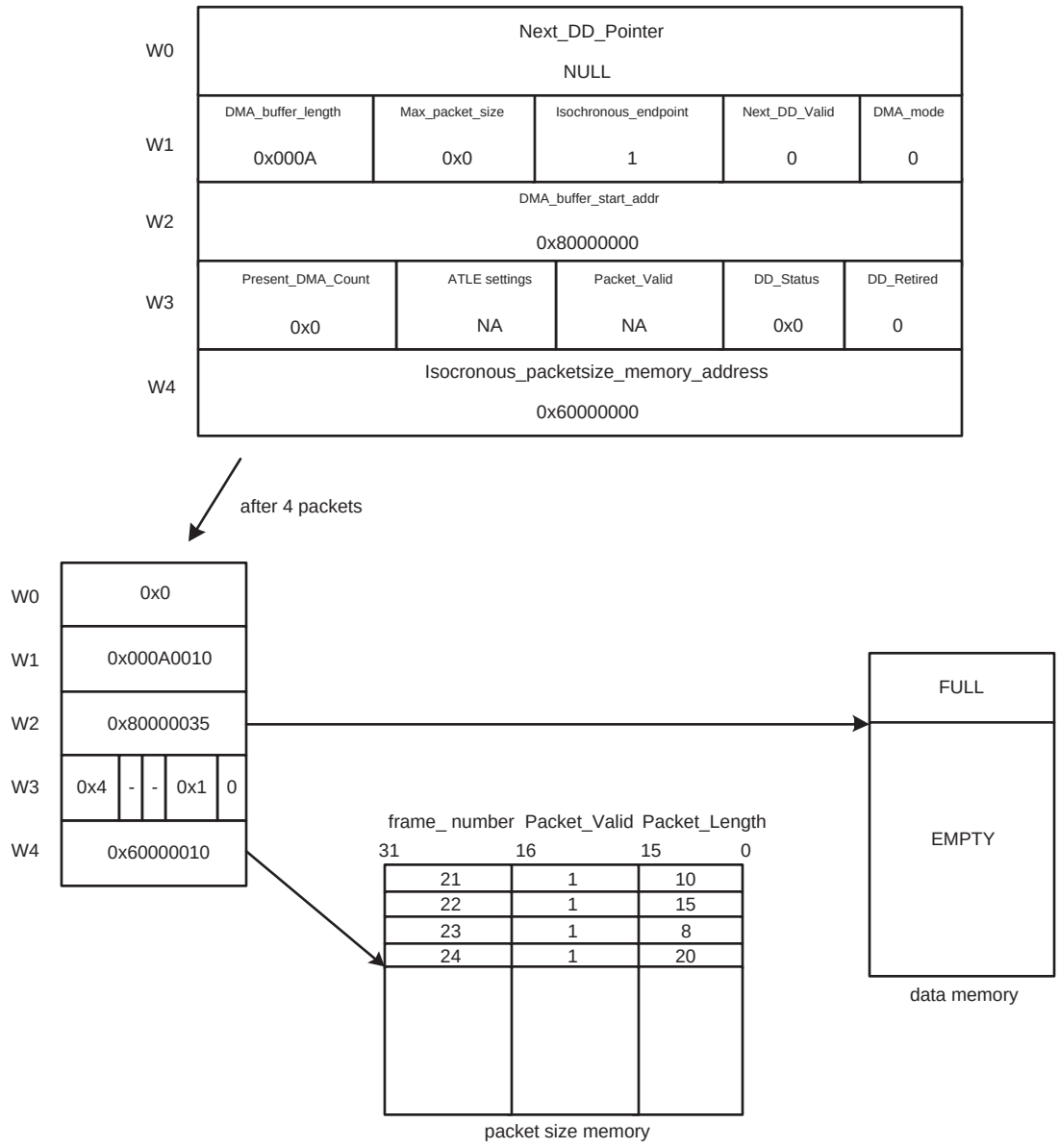


Fig 45. Isochronous OUT endpoint operation example

12.15.7 Auto Length Transfer Extraction (ATLE) mode operation

Some host drivers such as NDIS (Network Driver Interface Specification) host drivers are capable of concatenating small USB transfers (delta transfers) to form a single large USB transfer. For OUT USB transfers, the device hardware has to break up this concatenated transfer back into the original delta transfers and transfer them to separate DMA buffers. This is achieved by setting the DMA mode to Auto Transfer Length Extraction (ATLE) mode in the DMA descriptor. ATLE mode is supported for Bulk endpoints only.

OUT transfers in ATLE mode

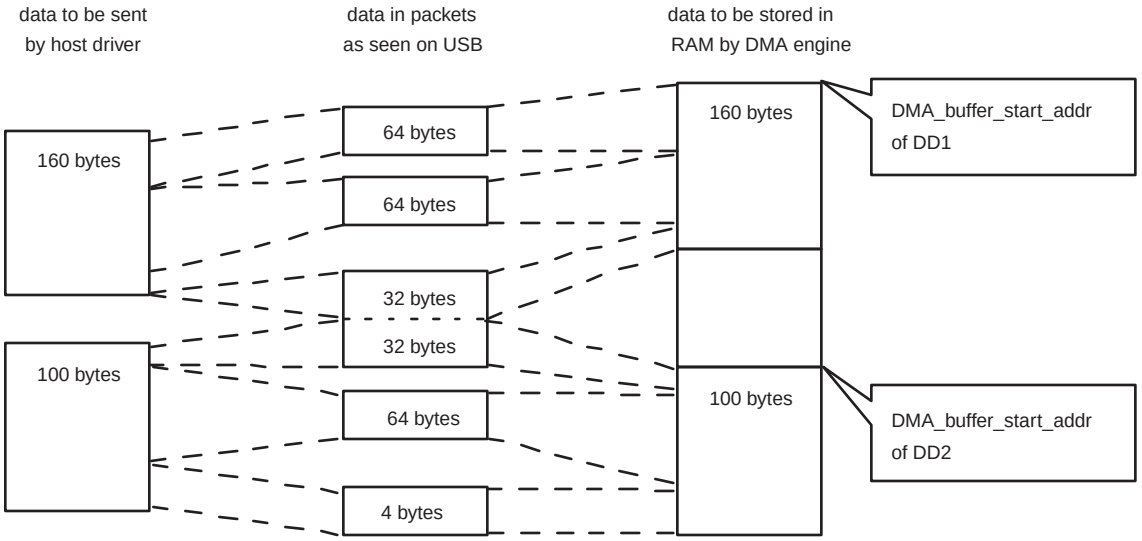


Fig 46. Data transfer in ATLE mode

Figure 46 shows a typical OUT USB transfer in ATLE mode, where the host concatenates two USB transfers of 160 bytes and 100 bytes, respectively. Given a MaxPacketSize of 64, the device hardware interprets this USB transfer as four packets of 64 bytes and a short packet of 4 bytes. The third and fourth packets are concatenated. Note that in Normal mode, the USB transfer would be interpreted as packets of 64, 64, 32, and 64 and 36 bytes.

It is now the responsibility of the DMA engine to separate these two USB transfers and put them in the memory locations in the DMA_buffer_start_addr field of DMA Descriptor 1 (DD1) and DMA Descriptor 2 (DD2).

Hardware reads the two-byte-wide DMA_buffer_length at the offset (from the start of the USB transfer) specified by Message_length_position from the incoming data packets and writes it in the DMA_buffer_length field of the DD. To ensure that both bytes of the DMA_buffer_length are extracted in the event they are split between two packets, the flags LS_byte_extracted and MS_byte_extracted are set by hardware after the respective byte is extracted. After the extraction of the MS byte, the DMA transfer continues as in the normal mode.

The flags LS_byte_extracted and MS_byte_extracted are set to 0 by software when preparing a new DD. Therefore, once a DD is retired, the transfer length is extracted again for the next DD.

If DD1 is retired during the transfer of a concatenated packet (such as the third packet in Figure 46), and DD2 is not programmed (Next_DD_valid field of DD1 is 0), then DD1 is retired with DD_status set to the DataOverrun status code. This is treated as an error condition and the corresponding EPxx_DMA_ENABLE bit of EpDMASt is cleared by hardware.

In ATLE mode, the last buffer length to be transferred always ends with a short or empty packet indicating the end of the USB transfer. If the concatenated transfer lengths are such that the USB transfer ends on a MaxPacketSize packet boundary, the (NDIS) host will send an empty packet to mark the end of the USB transfer.

IN transfers in ATLE mode

For IN USB transfers from the device to the host, DMA_buffer_length is set by the device software as in normal mode.

In ATLE mode, the device concatenates data from multiple DDs to form a single USB transfer. If a DD is retired in the middle of a packet (packet size is less than MaxPacketSize), the next DD referenced by Next_DD_pointer is fetched, and the remaining bytes to form a packet of MaxPacketSize are transferred from the next DD's buffer.

If the next DD is not programmed (i.e. Next_DD_valid field in DD is 0), and the DMA buffer length for the current DD has completed before the MaxPacketSize packet boundary, then the available bytes from current DD are sent as a short packet on USB, which marks the end of the USB transfer for the host.

If the last buffer length completes on a MaxPacketSize packet boundary, the device software must program the next DD with DMA_buffer_length field 0, so that an empty packet is sent by the device to mark the end of the USB transfer for the host.

12.15.7.1 Setting up the DMA transfer

For OUT endpoints, the host hardware needs to set the field Message_length_position in the DD. This indicates the start location of the message length in the incoming data packets. Also the device software has to set the DMA_buffer_length field to 0 for OUT endpoints because this field is updated by the device hardware after the extraction of the buffer length.

For IN endpoints, descriptors are set in the same way as in normal mode operation.

Since a single packet can be split between two DDs, software should always keep two DDs ready, except for the last DMA transfer which ends with a short or empty packet.

12.15.7.2 Finding the DMA Descriptor

DMA descriptors are found in the same way as the normal mode operation.

12.15.7.3 Transferring the Data

OUT endpoints

If the LS_byte_extracted or MS_byte_extracted bit in the status field is not set, the hardware will extract the transfer length from the data stream and program DMA_buffer_length. Once the extraction is complete both the LS_byte_extracted and MS_byte_extracted bits will be set.

IN endpoints

The DMA transfer proceeds as in normal mode and continues until the number of bytes transferred equals the DMA_buffer_length.

12.15.7.4 Ending the packet transfer

The DMA engine proceeds with the transfer until the number of bytes specified in the field DMA_buffer_length is transferred to or from on-chip RAM. Then the EOT interrupt will be generated. If this happens in the middle of the packet, the linked DD will get loaded and the remaining part of the packet gets transferred to or from the address pointed by the new DD.

OUT endpoints

If the linked DD is not valid and the packet is partially transferred to memory, the DD ends with DataOverrun status code set, and the DMA will be disabled for this endpoint. Otherwise DD_status will be updated with the NormalCompletion status code.

IN endpoints

If the linked DD is not valid and the packet is partially transferred to USB, the DD ends with a status code of NormalCompletion in the DD_status field. This situation corresponds to the end of the USB transfer, and the packet will be sent as a short packet. Also, when the linked DD is valid and buffer length is 0, an empty packet will be sent to indicate the end of the USB transfer.

12.16 Double buffered endpoint operation

The Bulk and Isochronous endpoints of the USB Device Controller are double buffered to increase data throughput.

When a double-buffered endpoint is realized, enough space for both endpoint buffers is automatically allocated in the EP_RAM. See Section 12.10.4.1.

For the following discussion, the endpoint buffer currently accessible to the CPU or DMA engine for reading or writing is said to be the active buffer.

12.16.1 Bulk endpoints

For Bulk endpoints, the active endpoint buffer is switched by the SIE Clear Buffer or Validate Buffer commands.

The following example illustrates how double buffering works for a Bulk OUT endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty, and that the active buffer is B_1.

1. The host sends a data packet to the endpoint. The device hardware puts the packet into B_1, and generates an endpoint interrupt.
2. Software clears the endpoint interrupt and begins reading the packet data from B_1. While B_1 is still being read, the host sends a second packet, which device hardware places in B_2, and generates an endpoint interrupt.
3. Software is still reading from B_1 when the host attempts to send a third packet. Since both B_1 and B_2 are full, the device hardware responds with a NAK.
4. Software finishes reading the first packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.
5. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. Software finds that the active buffer (B_2) has data (FE=1). Software clears the endpoint interrupt and begins reading the contents of B_2.
6. The host re-sends the third packet which device hardware places in B_1. An endpoint interrupt is generated.
7. Software finishes reading the second packet from B_2 and sends a SIE Clear Buffer command to free B_2 to receive another packet. B_1 becomes the active buffer. Software waits for the next endpoint interrupt to occur (it already has been generated back in step 6).
8. Software responds to the endpoint interrupt by clearing it and begins reading the third packet from B_1.
9. Software finishes reading the third packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.
10. Software tests the FE bit and finds that the active buffer (B_2) is empty (FE=0).
11. Both B_1 and B_2 are empty. Software waits for the next endpoint interrupt to occur. The active buffer is now B_2. The next data packet sent by the host will be placed in B_2.

The following example illustrates how double buffering works for a Bulk IN endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty and that the active buffer is B_1. The interrupt on NAK feature is enabled.

1. The host requests a data packet by sending an IN token packet. The device responds with a NAK and generates an endpoint interrupt.
2. Software clears the endpoint interrupt. The device has three packets to send. Software fills B_1 with the first packet and sends a SIE Validate Buffer command. The active buffer is switched to B_2.
3. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. It finds that B_2 is empty (FE=0) and fills B_2 with the second packet. Software sends a SIE Validate Buffer command, and the active buffer is switched to B_1.
4. Software waits for the endpoint interrupt to occur.
5. The device successfully sends the packet in B_1 and clears the buffer. An endpoint interrupt occurs.
6. Software clears the endpoint interrupt. Software fills B_1 with the third packet and validates it using the SIE Validate Buffer command. The active buffer is switched to B_2.
7. The device successfully sends the second packet from B_2 and generates an endpoint interrupt.
8. Software has no more packets to send, so it simply clears the interrupt.
9. The device successfully sends the third packet from B_1 and generates an endpoint interrupt.
10. Software has no more packets to send, so it simply clears the interrupt.
11. Both B_1 and B_2 are empty, and the active buffer is B_2. The next packet written by software will go into B_2.

In DMA mode, switching of the active buffer is handled automatically in hardware. For Bulk IN endpoints, proactively filling an endpoint buffer to take advantage of the double buffering can be accomplished by manually starting a packet transfer using the DMARSet register.

12.16.2 Isochronous endpoints

For isochronous endpoints, the active data buffer is switched by hardware when the FRAME interrupt occurs. The SIE Clear Buffer and Validate Buffer commands do not cause the active buffer to be switched.

Double buffering allows the software to make full use of the frame interval writing or reading a packet to or from the active buffer, while the packet in the other buffer is being sent or received on the bus.

For an OUT isochronous endpoint, any data not read from the active buffer before the end of the frame is lost when it switches.

For an IN isochronous endpoint, if the active buffer is not validated before the end of the frame, an empty packet is sent on the bus when the active buffer is switched, and its contents will be overwritten when it becomes active again.

13.1 How to read this chapter

This chapter describes the USB host controller which is present on some LPC408x/407x devices (see Section 1.4 for details). On these devices, the USB controller can be configured for device, Host, or OTG operation.

13.2 Basic configuration

The USB controller is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCUSB.

   Remark: On reset, the USB block is disabled (PCUSB = 0).
2. Clock: The USB block can be used with the Alt PLL (PLL1) to obtain the USB clock or with the Main PLL (PLL0). See Section 3.10.
3. Pins: Select USB pins and their modes in the relevant IOCON registers (Section 7.4.1).
4. Wake-up: Activity on the USB bus port can wake up the microcontroller from Power-down mode, see Section 3.12.8.
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. Initialization: see Section 14.11.

13.3 Introduction

This section describes the host portion of the USB 2.0 OTG dual role core which integrates the host controller (OHCI compliant), device controller, and I^2C interface. The I^2C interface controls the external OTG ATX.

The USB is a 4 wire bus that supports communication between a host and a number (127 max.) 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 309. USB (OHCI) related acronyms and abbreviations used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-Performance Bus
ATX	Analog Transceiver
DMA	Direct Memory Access
FS	Full Speed
LS	Low Speed
OHCI	Open Host Controller Interface
USB	Universal Serial Bus

13.3.1 Features

- OHCI compliant.
- OpenHCI specifies the operation and interface of the USB Host Controller and 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.
- The Host Controller has four USB states visible to the SW Driver.
- HCCA register points to Interrupt and Isochronous Descriptors List.
- ControlHeadED and BulkHeadED registers point to Control and Bulk Descriptors List.

13.3.2 Architecture

The architecture of the USB host controller is shown below in Figure 47.

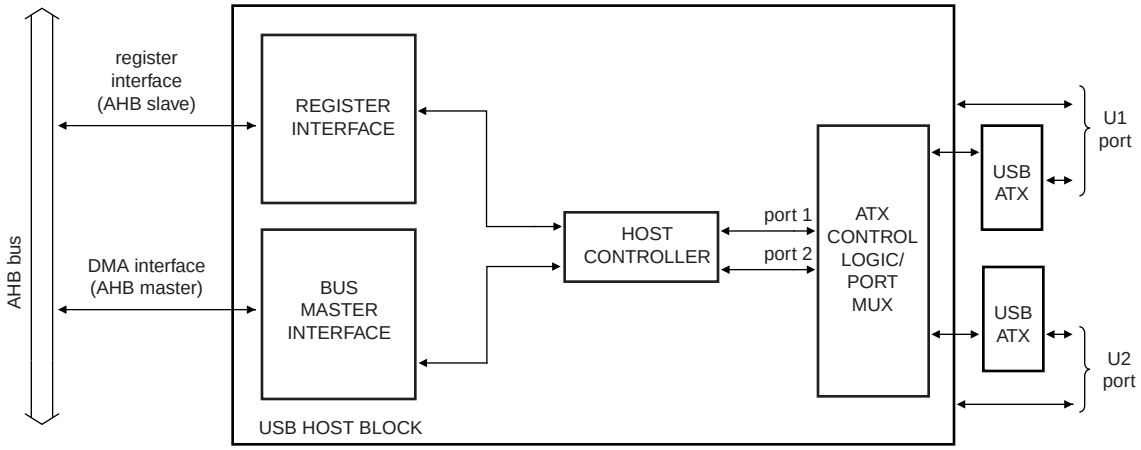


Fig 47. USB Host controller block diagram

13.4 Interfaces

The OTG controller has two USB ports indicated by suffixes 1 and 2 in the USB pin names and referred to as USB port 1 (U1) and USB port 2 (U2) in the following text.

13.4.1 Pin description

Table 310. USB OTG port pins

Pin name	Direction	Description	Pin category
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding IOCON register, it is driven HIGH internally.	USB Connector
Port U1			
USB_D+1	I/O	Positive differential data	USB Connector
USB_D−1	I/O	Negative differential data	USB Connector
USB_CONNECT1	O	SoftConnect control signal	Control
USB_UP_LED1	O	GoodLink LED control signal	Control
$\overline{\text{USB\_INT1}}$	I	OTG ATX interrupt	External OTG transceiver
USB_SCL1	I/O	I^2C serial clock	External OTG transceiver
USB_SDA1	I/O	I^2C serial data	External OTG transceiver
$\overline{\text{USB\_TX\_E1}}$	O	Transmit enable	External OTG transceiver
USB_TX_DP1	O	D+ transmit data	External OTG transceiver
USB_TX_DM1	O	D− transmit data	External OTG transceiver
USB_RCV1	I	Differential receive data	External OTG transceiver
USB_RX_DP1	I	D+ receive data	External OTG transceiver
USB_RX_DM1	I	D− receive data	External OTG transceiver
$\overline{\text{USB\_LS1}}$	O	Low speed status (applies to host functionality only)	External OTG transceiver
$\overline{\text{USB\_SSPND1}}$	O	Bus suspend status	External OTG transceiver
$\overline{\text{USB\_PPWR1}}$	O	Port power enable	Host power switch
USB_PWRD1	I	Port power status	Host power switch
$\overline{\text{USB\_OVRCR1}}$	I	Over-current status	Host power switch
$\overline{\text{USB\_HSTEN1}}$	O	Host enabled status	
Port U2			
USB_D+2	I/O	Positive differential data	USB Connector
USB_D−2	I/O	Negative differential data	USB Connector
USB_CONNECT2	O	SoftConnect control signal	Control
USB_UP_LED2	O	GoodLink LED control signal	Control
$\overline{\text{USB\_PPWR2}}$	O	Port power enable	Host power switch
USB_PWRD2	I	Port power status	Host power switch
$\overline{\text{USB\_OVRCR2}}$	I	Over-current status	Host power switch
$\overline{\text{USB\_HSTEN2}}$	O	Host enabled status	Control

13.4.1.1 USB host usage note

Both ports can be configured as USB hosts. For details on how to connect the USB ports, see the USB OTG chapter, Section 14.7.

The USB device/host/OTG controller is disabled after RESET and must be enabled by writing a 1 to the PCUSB bit in the PCONP register, see Section 3.3.2.2.

13.4.2 Software interface

The software interface of the USB host block consists of a register view and the format definitions for the endpoint descriptors. For details on these two aspects see the OHCI specification. The register map is shown in the next subsection.

13.4.2.1 Register map

The following registers are located in the AHB clock 'cclk' domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned in the word address boundaries.

Table 311. USB Host register address definitions

Name	Function	Address	R/W[1]	Reset value
HcRevision	BCD representation of the version of the HCI specification that is implemented by the Host Controller.	0x2008 C000	R	0x10
HcControl	Defines the operating modes of the HC.	0x2008 C004	R/W	0
HcCommandStatus	This register is used to receive the commands from the Host Controller Driver (HCD). It also indicates the status of the HC.	0x2008 C008	R/W	0
HcInterruptStatus	Indicates the status on various events that cause hardware interrupts by setting the appropriate bits.	0x2008 C00C	R/W	0
HcInterruptEnable	Controls the bits in the HcInterruptStatus register and indicates which events will generate a hardware interrupt.	0x2008 C010	R/W	0
HcInterruptDisable	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.	0x2008 C014	R/W	0
HcHCCA	Contains the physical address of the host controller communication area.	0x2008 C018	R/W	0
HcPeriodCurrentED	Contains the physical address of the current isochronous or interrupt endpoint descriptor.	0x2008 C01C	R	0
HcControlHeadED	Contains the physical address of the first endpoint descriptor of the control list.	0x2008 C020	R/W	0
HcControlCurrentED	Contains the physical address of the current endpoint descriptor of the control list	0x2008 C024	R/W	0
HcBulkHeadED	Contains the physical address of the first endpoint descriptor of the bulk list.	0x2008 C028	R/W	0
HcBulkCurrentED	Contains the physical address of the current endpoint descriptor of the bulk list.	0x2008 C02C	R/W	0
HcDoneHead	Contains the physical address of the last transfer descriptor added to the 'Done' queue.	0x2008 C030	R	0
HcFmInterval	Defines the bit time interval in a frame and the full speed maximum packet size which would not cause an overrun.	0x2008 C034	R/W	0x2EDF
HcFmRemaining	A 14-bit counter showing the bit time remaining in the current frame.	0x2008 C038	R	0

Table 311. USB Host register address definitions *…continued*

Name	Function	Address	R/W[1]	Reset value
HcFmNumber	Contains a 16-bit counter and provides the timing reference among events happening in the HC and the HCD.	0x2008 C03C	R	0
HcPeriodicStart	Contains a programmable 14-bit value which determines the earliest time HC should start processing a periodic list.	0x2008 C040	R/W	0
HcLSThreshold	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.	0x2008 C044	R/W	0x628h
HcRhDescriptorA	First of the two registers which describes the characteristics of the root hub.	0x2008 C048	R/W	0xFF000902
HcRhDescriptorB	Second of the two registers which describes the characteristics of the Root Hub.	0x2008 C04C	R/W	0x60000h
HcRhStatus	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.	0x2008 C050	R/W	0
HcRhPortStatus[1]	Controls and reports the port events on a per-port basis.	0x2008 C054	R/W	0
HcRhPortStatus[2]	Controls and reports the port events on a per port basis.	0x2008 C058	R/W	0
Module_ID/ Ver_Rev_ID	IP number, where yy (0x00) is unique version number and zz (0x00) is a unique revision number.	0x2008 C0FC	R	0x3505yyzz

[1] The R/W column lists the accessibility of the register:

a) Registers marked ‘R’ for access will return their current value when read.

b) Registers marked ‘R/W’ allow both read and write.

13.4.2.2 USB Host Register Definitions

Refer to the OHCI specification document on the Compaq website for register definitions.

14.1 How to read this chapter

This chapter describes the USB OTG controller which is present on some LPC408x/407x devices (see Section 1.4 for details). On these devices, the USB controller can be configured for device, Host, or OTG operation.

14.2 Basic configuration

The USB controller is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCUSB.

   Remark: On reset, the USB block is disabled (PCUSB = 0).
2. Clock: The USB clock can generated using the Alt PLL (PLL1) or with the Main PLL (PLL0). See Section 3.10.
3. Pins: Select USB pins and their modes in the relevant IOCON registers (Section 7.4.1).
4. Wake-up: Activity on the USB bus port can wake up the microcontroller from Power-down mode (see Section 14.10.2 and Section 3.12.8).
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. The USB global interrupt status is visible in the USBINTSTAT register (Table 38).
7. Initialization: see Section 14.11.

14.3 Introduction

This chapter describes the OTG and I^2C portions of the USB 2.0 OTG dual role device controller which integrates the (OHCI) host controller, device controller, and I^2C. The I^2C interface that is part of the USB block is intended to control an external OTG transceiver, and is not the same as the I^2C peripherals described in Section 22.1.

USB OTG (On-The-Go) is a supplement to the USB 2.0 specification that augments the capability of existing mobile devices and USB peripherals by adding host functionality for connection to USB peripherals. The specification and more information on USB OTG can be found on the USB Implementers Forum web site.

14.4 Features

- Fully compliant with On-The-Go supplement to the *USB 2.0 Specification, Revision 1.0a*.
- Hardware support for Host Negotiation Protocol (HNP).
- Includes a programmable timer required for HNP and SRP.
- Supports any OTG transceiver compliant with the *OTG Transceiver Specification (CEA-2011), Rev. 1.0*.

14.5 Architecture

The architecture of the USB OTG controller is shown below in the block diagram.

The host, device, OTG, and I^2C controllers can be programmed through the register interface. The OTG controller enables dynamic switching between host and device roles through the HNP protocol. One port may be connected to an external OTG transceiver to support an OTG connection. The communication between the register interface and an external OTG transceiver is handled through an I^2C interface and through the external OTG transceiver interrupt signal.

For USB connections that use the device or host controller only (not OTG), the ports use an embedded USB Analog Transceiver (ATX).

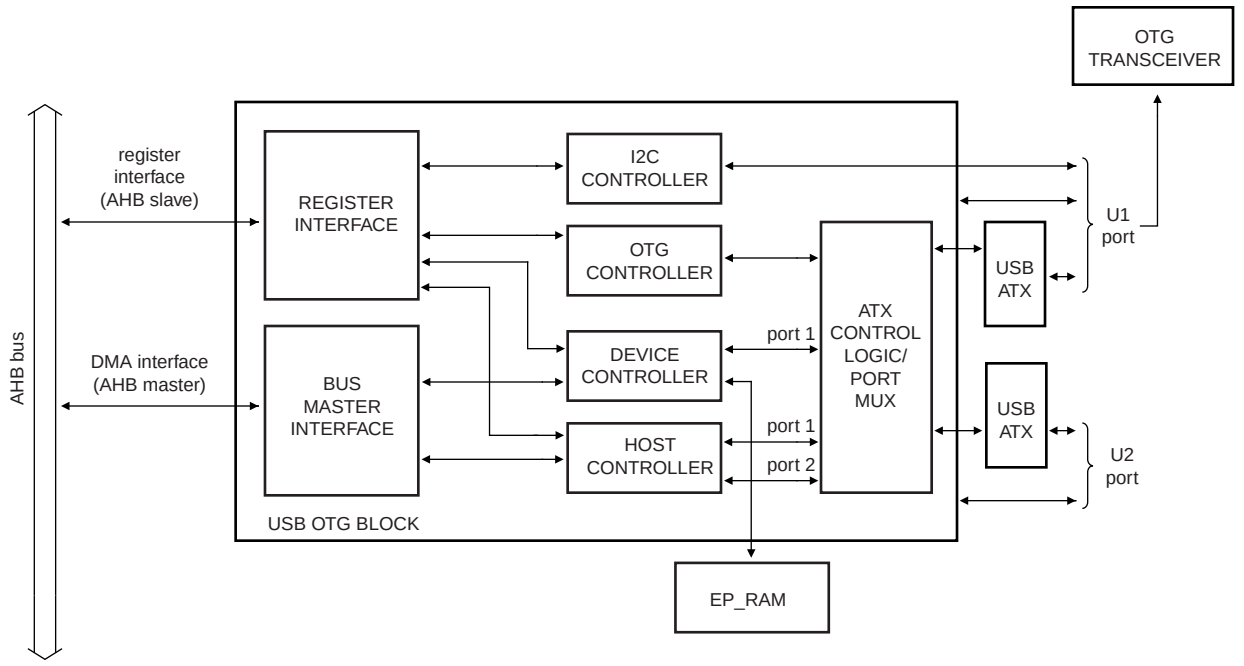


Fig 48. USB OTG controller block diagram

14.6 Modes of operation

The OTG controller is capable of operating in the following modes:

- One dual-role OTG port and optionally another Host port (see Figure 49 and Figure 50)
- Two Host ports (see Figure 51)
- One Host port and one Device port (see Figure 52)

14.7 Pin configuration

The OTG controller has two USB ports indicated by suffixes 1 and 2 in the USB pin names and referred to as USB port 1 (U1) and USB port 2 (U2) in the following text.

Table 312. USB OTG port 1 pins

Pin name	Direction	Description	Pin category
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding IOCON register, it is driven HIGH internally.	USB Connector
Port U1			
USB_D+1	I/O	Positive differential data	USB Connector
USB_D−1	I/O	Negative differential data	USB Connector
USB_CONNECT1	O	SoftConnect control signal	Control
USB_UP_LED1	O	GoodLink LED control signal	Control
$\overline{\text{USB\_INT1}}$	I	OTG ATX interrupt	External OTG transceiver
USB_SCL1	I/O	I^2C serial clock	External OTG transceiver
USB_SDA1	I/O	I^2C serial data	External OTG transceiver
$\overline{\text{USB\_TX\_E1}}$	O	Transmit enable	External OTG transceiver
USB_TX_DP1	O	D+ transmit data	External OTG transceiver
USB_TX_DM1	O	D− transmit data	External OTG transceiver
USB_RCV1	I	Differential receive data	External OTG transceiver
USB_RX_DP1	I	D+ receive data	External OTG transceiver
USB_RX_DM1	I	D− receive data	External OTG transceiver
$\overline{\text{USB\_LS1}}$	O	Low speed status (applies to host functionality only)	External OTG transceiver
$\overline{\text{USB\_SSPND1}}$	O	Bus suspend status	External OTG transceiver
$\overline{\text{USB\_PPWR1}}$	O	Port power enable	Host power switch
USB_PWRD1	I	Port power status	Host power switch
$\overline{\text{USB\_OVRCR1}}$	I	Over-current status	Host power switch
$\overline{\text{USB\_HSTEN1}}$	O	Host enabled status	
Port U2			
USB_D+2	I/O	Positive differential data	USB Connector
USB_D−2	I/O	Negative differential data	USB Connector
USB_CONNECT2	O	SoftConnect control signal	Control
USB_UP_LED2	O	GoodLink LED control signal	Control
$\overline{\text{USB\_PPWR2}}$	O	Port power enable	Host power switch
USB_PWRD2	I	Port power status	Host power switch
$\overline{\text{USB\_OVRCR2}}$	I	Over-current status	Host power switch
$\overline{\text{USB\_HSTEN2}}$	O	Host enabled status	Control

14.7.1 Using port U1 for OTG operation

The following figures show different ways to realize connections to a USB device using ports U1 and U2. The example described here uses an ISP1302 (ST-Ericsson) for the external OTG transceiver and the USB Host power switch LM3526-L (National Semiconductors). There are two ways to connect the OTG transceiver:

1. Use the internal USB transceiver for USB signalling and use the external OTG transceiver for OTG functionality only (see Figure 49). This option uses the internal transceiver in VP/VM mode.
2. Use the external OTG transceiver in VP/VM mode for OTG functionality and USB signalling (see Figure 50).

In both cases port U2 is connected as a host. Solution one uses fewer pins.

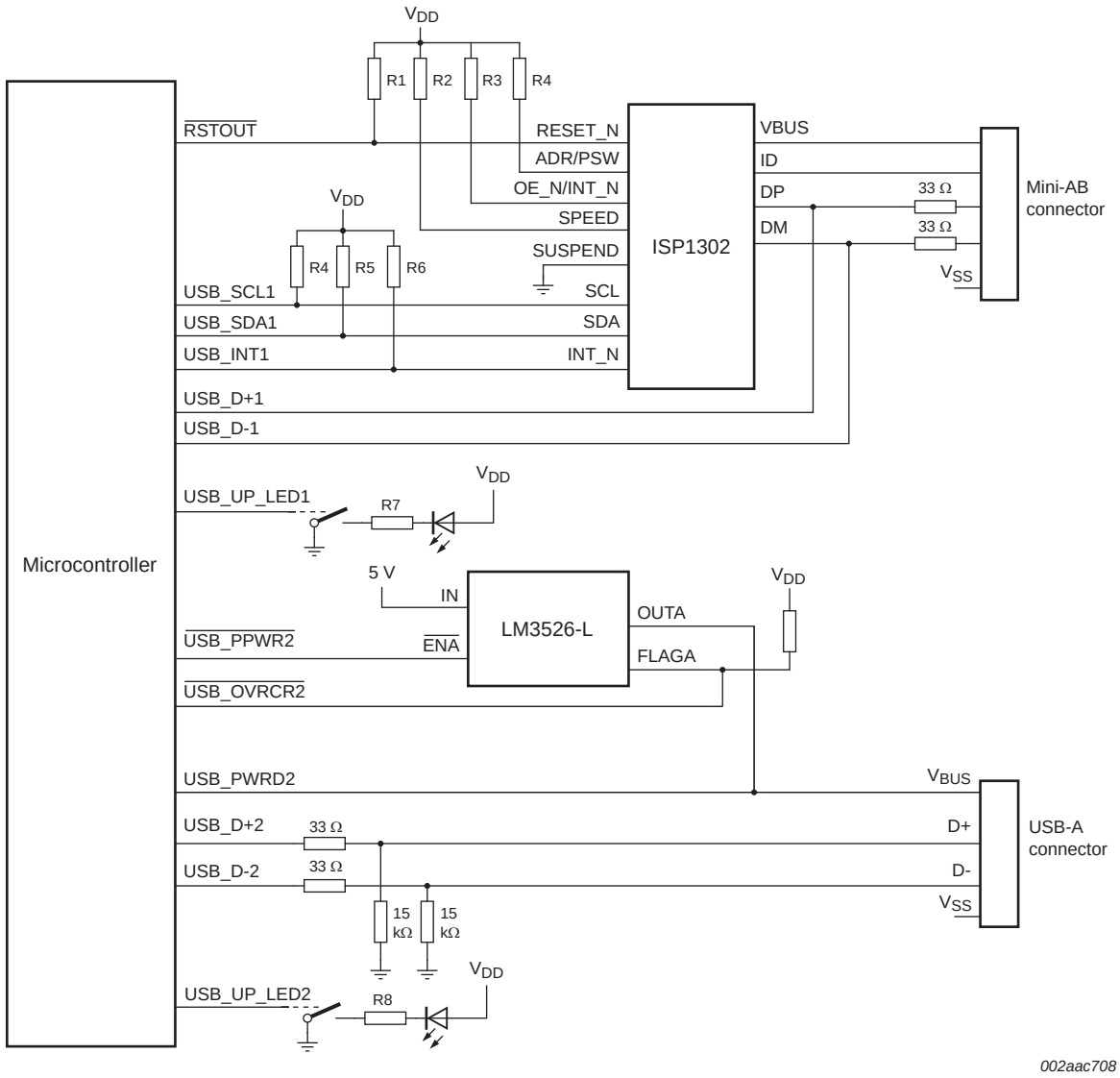


Fig 49. USB OTG port configuration: port U1 OTG dual-role device, port U2 host

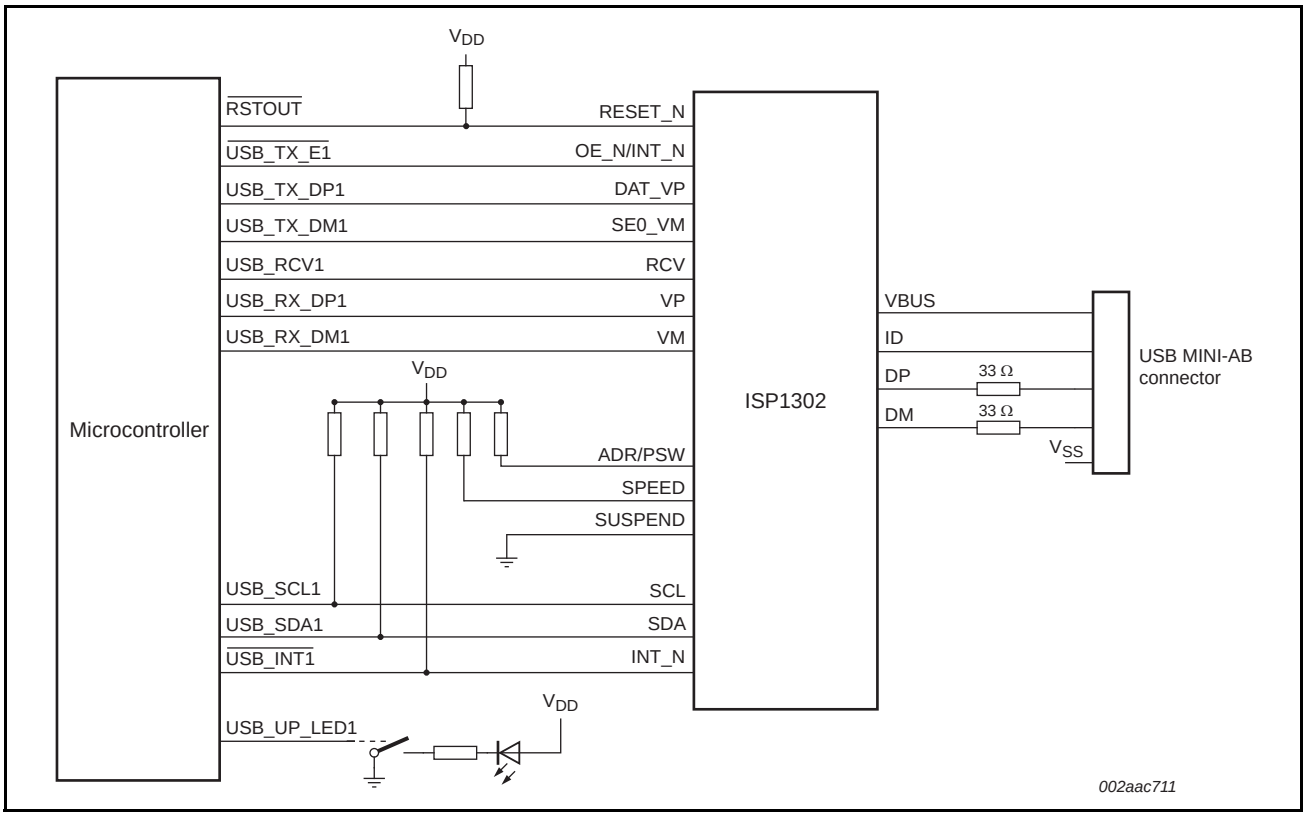


Fig 50. USB OTG port configuration: VP_VM mode

14.7.2 Using both ports U1 and U2 for host operation

Both ports U1 and U2 are connected as hosts using an embedded USB transceiver. There is no OTG functionality on the port.

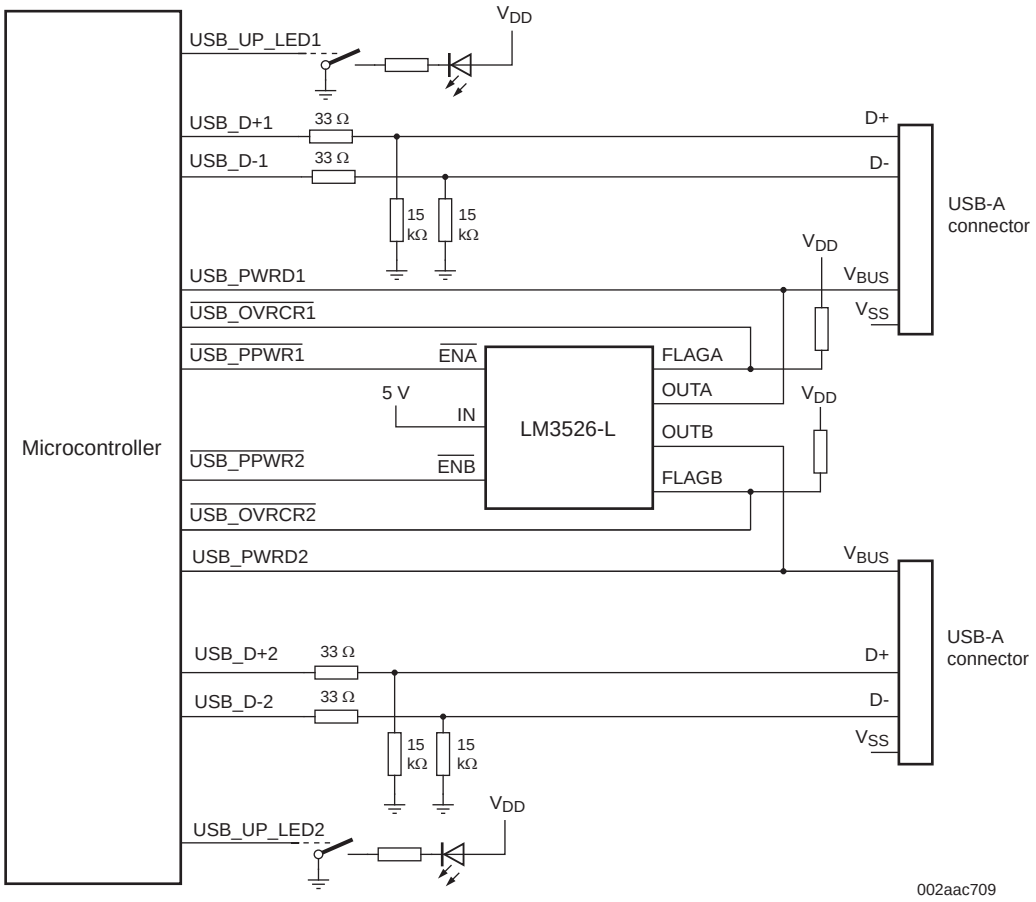


Fig 51. USB host port configuration: port U1 and U2 as hosts

14.7.3 Using U1 for host operation and U2 for device operation

Port U2 is connected as device, and port U1 is connected as host. Both ports use embedded USB transceivers. There is no OTG functionality on either USB port.

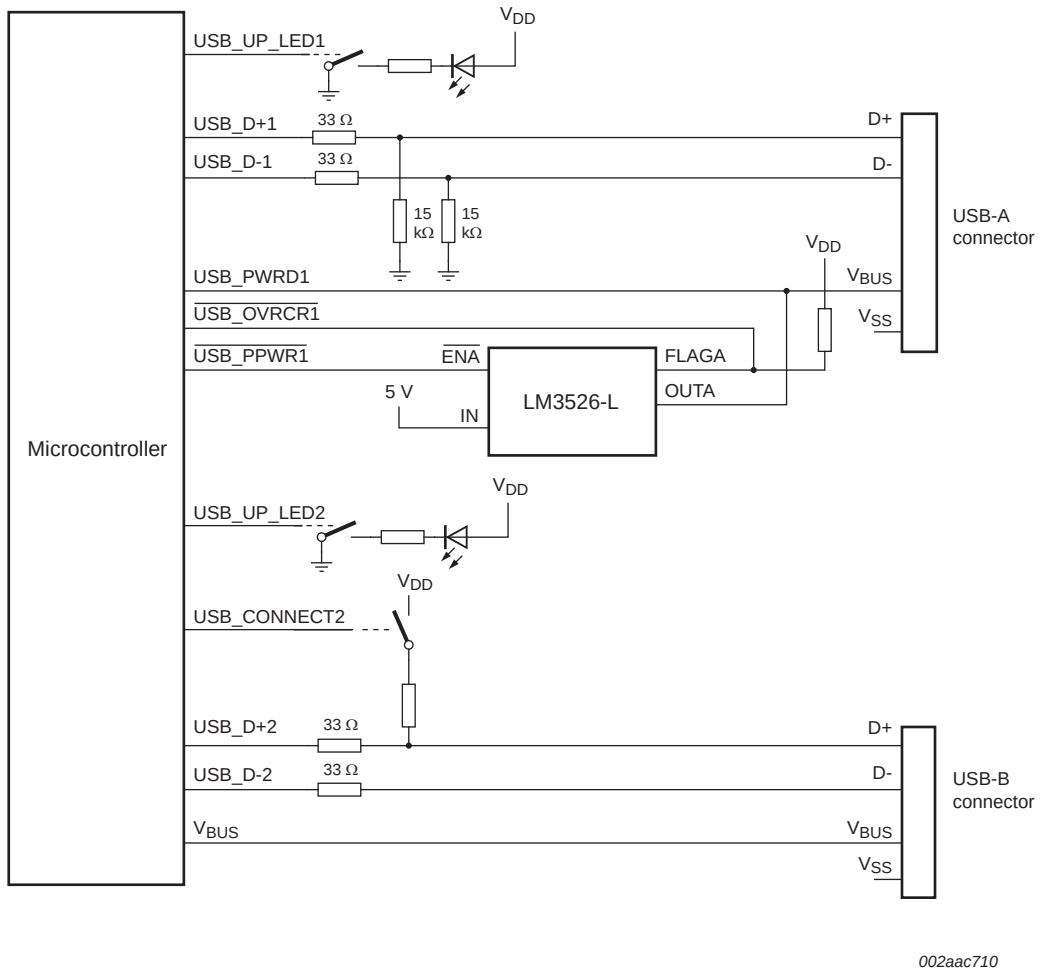


Fig 52. USB device port configuration: port U1 host and port U2 device

14.8 Register description

The OTG and I²C registers are summarized in the following table.

The Device and Host registers are explained in Table 255 and Table 311 in the USB Device Controller and USB Host (OHCI) Controller chapters. All registers are 32 bits wide and aligned to word address boundaries.

The USB interrupt status is captured in the USBINTSTAT register in the syscon block.

Bits 0 and 1 of the StCtrl register are used to control the routing of the USB pins to ports 1 and 2 in device-only applications (see Section 12.10.1).

Table 313. Register overview: USB OTG controller (base address 0x2008 C000)

Name	Access	Address offset	Description	Reset value	Table
OTG registers					
INTST	RO	0x100	OTG Interrupt Status	0	314
INTEN	R/W	0x104	OTG Interrupt Enable	0	314
INTSET	WO	0x108	OTG Interrupt Set	NA	314
INTCLR	WO	0x10C	OTG Interrupt Clear	NA	314
PORTSEL	R/W	0x110	OTG Status and Control and USB port select	0	318
TMR	R/W	0x114	OTG Timer	0xFFFF	319
I²C registers					
I2C_RX	RO	0x300	I²C Receive	NA	320
I2C_TX	WO	0x300	I²C Transmit	NA	321
I2C_STS	RO	0x304	I²C Status	0x0A00	322
I2C_CTL	R/W	0x308	I²C Control	0	323
I2C_CLKHI	R/W	0x30C	I²C Clock High	0xB9	324
I2C_CLKLO	R/W	0x310	I²C Clock Low	0xB9	325
Clock control registers					
CLKCTRL	R/W	0xFF4	OTG clock controller	0	326
CLKST	RO	0xFF8	OTG clock status	0	327

14.8.1 OTG Interrupt Status Register

Bits in this register are set by hardware when the interrupt event occurs during the HNP handoff sequence. See Section 14.9 for more information on when these bits are set.

Table 314. OTG Interrupt Status register (INTST - address 0x2008 C100) bit description

Bit	Symbol	Description	Reset Value
0	TMR	Timer time-out.	0
1	REMOVE_PU	Remove pull-up. This bit is set by hardware to indicate that software needs to disable the D+ pull-up resistor.	0

Table 314. OTG Interrupt Status register (INTST - address 0x2008 C100) bit description

Bit	Symbol	Description	Reset Value
2	HNP_FAILURE	HNP failed. This bit is set by hardware to indicate that the HNP switching has failed.	0
3	HNP_SUCCESS	HNP succeeded. This bit is set by hardware to indicate that the HNP switching has succeeded.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.2 OTG Interrupt Enable Register

Writing a one to a bit in this register enables the corresponding bit in OTGIntSt to generate an interrupt on one of the interrupt lines. The interrupt is routed to the USB_OTG_INT interrupt line in the USBIntSt register.

The bit allocation and reset value of OTGIntEn is the same as OTGIntSt.

Table 315. OTG Interrupt enable register (INTEN - address 0x2008 C104) bit description

Bit	Symbol	Description	Reset Value
0	TMR_EN	1 = enable the corresponding bit in the IntSt register.	0
1	REMOVE_PU_EN	1 = enable the corresponding bit in the IntSt register.	0
2	HNP_FAILURE_EN	1 = enable the corresponding bit in the IntSt register.	0
3	HNP_SUCCES_EN	1 = enable the corresponding bit in the IntSt register.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.3 OTG Interrupt Set Register

Writing a one to a bit in this register will set the corresponding bit in the OTGIntSt register. Writing a zero has no effect. The bit allocation of OTGIntSet is the same as in OTGIntSt.

Table 316. OTG Interrupt enable register (INTSET - address 0x2008 C108) bit description

Bit	Symbol	Description	Reset Value
0	TMR_SET	0 = no effect. 1 = set the corresponding bit in the IntSt register.	0
1	REMOVE_PU_SET	0 = no effect. 1 = set the corresponding bit in the IntSt register.	0
2	HNP_FAILURE_SET	0 = no effect. 1 = set the corresponding bit in the IntSt register.	0
3	HNP_SUCCES_SET	0 = no effect. 1 = set the corresponding bit in the IntSt register.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.4 OTG Interrupt Clear Register

Writing a one to a bit in this register will clear the corresponding bit in the OTGIntSt register. Writing a zero has no effect. The bit allocation of OTGIntClr is the same as in OTGIntSt.

Table 317. OTG Interrupt enable register (INTCLR - address 0x2008 C10C) bit description

Bit	Symbol	Description	Reset Value
0	TMR_CLR	0 = no effect. 1 = clear the corresponding bit in the IntSt register.	0
1	REMOVE_PU_CLR	0 = no effect. 1 = clear the corresponding bit in the IntSt register.	0
2	HNP_FAILURE_CLR	0 = no effect. 1 = clear the corresponding bit in the IntSt register.	0
3	HNP_SUCCES_CLR	0 = no effect. 1 = clear the corresponding bit in the IntSt register.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.5 OTG Status and Control Register

The OTGStCtrl register allows enabling hardware tracking during the HNP hand over sequence, controlling the OTG timer, monitoring the timer count, and controlling the functions mapped to port U1 and U2.

Time critical events during the switching sequence are controlled by the OTG timer. The timer can operate in two modes:

1. Monoshot mode: an interrupt is generated at the end of TIMEOUT_CNT (see Section 14.8.6 “OTG Timer Register”), the TMR bit is set in OTGIntSt, and the timer will be disabled.
2. Free running mode: an interrupt is generated at the end of TIMEOUT_CNT (see Section 14.8.6 “OTG Timer Register”), the TMR bit is set, and the timer value is reloaded into the counter. The timer is not disabled in this mode.

Table 318. OTG Status Control register (STCTRL - address 0x2008 C110) bit description

Bit	Symbol	Description	Reset Value
1:0	PORT_FUNC	Controls connection of USB functions (see Figure 53). Bit 0 is set or cleared by hardware when B_HNP_TRACK or A_HNP_TRACK is set and HNP succeeds. See Section 14.9. 00: U1 = device (OTG), U2 = host 01: U1 = host (OTG), U2 = host 10: Reserved 11: U1 = host, U2 = device	-
3:2	TMR_SCALE	Timer scale selection. This field determines the duration of each timer count. 00: 10 μs (100 KHz) 01: 100 μs (10 KHz) 10: 1000 μs (1 KHz) 11: Reserved	0
4	TMR_MODE	Timer mode selection. 0: monoshot 1: free running	0
5	TMR_EN	Timer enable. When set, TMR_CNT increments. When cleared, TMR_CNT is reset to 0.	0
6	TMR_RST	Timer reset. Writing one to this bit resets TMR_CNT to 0. This provides a single bit control for the software to restart the timer when the timer is enabled.	0
7	-	Reserved. Read value is undefined, only zero should be written.	NA
8	B_HNP_TRACK	Enable HNP tracking for B-device (peripheral), see Section 14.9. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
9	A_HNP_TRACK	Enable HNP tracking for A-device (host), see Section 14.9. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
10	PU_REMOVED	When the B-device changes its role from peripheral to host, software sets this bit when it removes the D+ pull-up, see Section 14.9. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
15:11	-	Reserved. Read value is undefined, only zero should be written.	NA
31:16	TMR_CNT	Current timer count value.	0

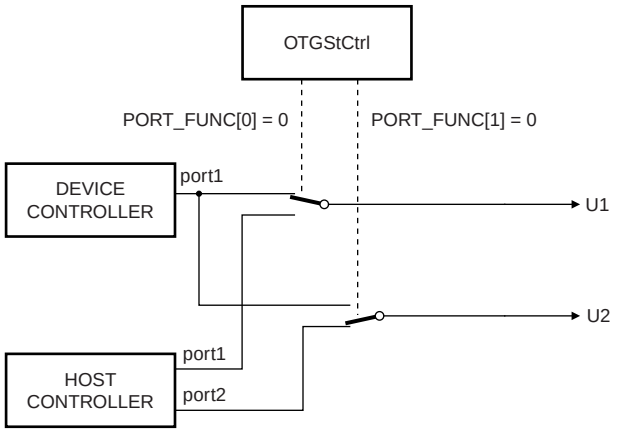


Fig 53. Port selection for PORT_FUNC bit 0 = 0 and PORT_FUNC bit 1 = 0

14.8.6 OTG Timer Register

Table 319. OTG Timer register (TMR - address 0x2008 C114) bit description

Bit	Symbol	Description	Reset Value
15:0	TIMEOUT_CNT	The TMR interrupt is set when TMR_CNT reaches this value.	0xFFFF
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.7 I²C Receive Register

This register is the top byte of the receive FIFO. The receive FIFO is 4 bytes deep. The Rx FIFO is flushed by a hard reset or by a soft reset (I2C_CTL bit 7). Reading an empty FIFO gives unpredictable data results.

Table 320. I²C Receive register (I2C_RX - address 0x2008 C300) bit description

Bit	Symbol	Description	Reset Value
7:0	RX Data	Receive data.	-

14.8.8 I²C Transmit Register

This register is the top byte of the transmit FIFO. The transmit FIFO is 4 bytes deep.

The Tx FIFO is flushed by a hard reset, soft reset (I2C_CTL bit 7) or if an arbitration failure occurs (I2C_STS bit 3). Data writes to a full FIFO are ignored.

I2C_TX must be written for both write and read operations to transfer each byte. Bits [7:0] are ignored for master-receive operations. The master-receiver must write a dummy byte to the TX FIFO for each byte it expects to receive in the RX FIFO. When the STOP bit is set or the START bit is set to cause a RESTART condition on a byte written to the TX FIFO (master-receiver), then the byte read from the slave is not acknowledged. That is, the last byte of a master-receive operation is not acknowledged.

Table 321. I²C Transmit register (I2C_TX - address 0x2008 C300) bit description

Bit	Symbol	Description	Reset Value
7:0	TX Data	Transmit data.	-
8	START	When 1, issue a START condition before transmitting this byte.	-
9	STOP	When 1, issue a STOP condition after transmitting this byte.	-
31:10	-	Reserved. Read value is undefined, only zero should be written.	-

14.8.9 I²C Status Register

The I2C_STS register provides status information on the TX and RX blocks as well as the current state of the external buses. Individual bits are enabled as interrupts by the I2C_CTL register and routed to the I2C_USB_INT bit in USBIntSt.

Table 322. I²C status register (I2C_STS - address 0x2008 C304) bit description

Bit	Symbol	Value	Description	Reset Value
0	TDI		Transaction Done Interrupt. This flag is set if a transaction completes successfully. It is cleared by writing a one to bit 0 of the status register. It is unaffected by slave transactions.	0
		0	Transaction has not completed.	
		1	Transaction completed.	
1	AFI		Arbitration Failure Interrupt. When transmitting, if the SDA is low when SDAOUT is high, then this I²C has lost the arbitration to another device on the bus. The Arbitration Failure bit is set when this happens. It is cleared by writing a one to bit 1 of the status register.	0
		0	No arbitration failure on last transmission.	
		1	Arbitration failure occurred on last transmission.	
2	NAI		No Acknowledge Interrupt. After every byte of data is sent, the transmitter expects an acknowledge from the receiver. This bit is set if the acknowledge is not received. It is cleared when a byte is written to the master TX FIFO.	0
		0	Last transmission received an acknowledge.	
		1	Last transmission did not receive an acknowledge.	

Table 322. I^2C status register (I2C_STS - address 0x2008 C304) bit description

Bit	Symbol	Value	Description	Reset Value
3	DRMI		Master Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a stop condition or it will hold SCL low until more data is available. The Master Data Request bit is set when the master transmitter is data-starved. If the master TX FIFO is empty and the last byte did not have a STOP condition flag, then SCL is held low until the CPU writes another byte to transmit. This bit is cleared when a byte is written to the master TX FIFO.	0
		0	Master transmitter does not need data.	
		1	Master transmitter needs data.	
4	DRSI		Slave Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a STOP condition or it will hold SCL low until more data is available. The Slave Data Request bit is set when the slave transmitter is data-starved. If the slave TX FIFO is empty and the last byte transmitted was acknowledged, then SCL is held low until the CPU writes another byte to transmit. This bit is cleared when a byte is written to the slave Tx FIFO.	0
		0	Slave transmitter does not need data.	
		1	Slave transmitter needs data.	
5	Active		Indicates whether the bus is busy. This bit is set when a START condition has been seen. It is cleared when a STOP condition is seen..	0
6	SCL		The current value of the SCL signal.	-
7	SDA		The current value of the SDA signal.	-
8	RFF		Receive FIFO Full (RFF). This bit is set when the RX FIFO is full and cannot accept any more data. It is cleared when the RX FIFO is not full. If a byte arrives when the Receive FIFO is full, the SCL is held low until the CPU reads the RX FIFO and makes room for it.	0
		0	RX FIFO is not full	
		1	RX FIFO is full	
9	RFE		Receive FIFO Empty. RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data.	1
		0	RX FIFO contains data.	
		1	RX FIFO is empty	
10	TFF		Transmit FIFO Full. TFF is set when the TX FIFO is full and is cleared when the TX FIFO is not full.	0
		0	TX FIFO is not full.	
		1	TX FIFO is full	
11	TFE		Transmit FIFO Empty. TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data.	1
		0	TX FIFO contains valid data.	
		1	TX FIFO is empty	
31:12	-		Reserved. Read value is undefined, only zero should be written.	NA

14.8.10 I²C Control Register

The I2C_CTL register is used to enable interrupts and reset the I²C state machine. Enabled interrupts cause the USB_I2C_INT interrupt output line to be asserted when set.

Table 323. I²C Control register (I2C_CTL - address 0x2008 C308) bit description

Bit	Symbol	Value	Description	Reset Value
0	TDIE		Transmit Done Interrupt Enable. This enables the TDI interrupt signalling that this I²C issued a STOP condition.	0
		0	Disable the TDI interrupt.	
		1	Enable the TDI interrupt.	
1	AFIE		Transmitter Arbitration Failure Interrupt Enable. This enables the AFI interrupt which is asserted during transmission when trying to set SDA high, but the bus is driven low by another device.	0
		0	Disable the AFI.	
		1	Enable the AFI.	
2	NAIE		Transmitter No Acknowledge Interrupt Enable. This enables the NAI interrupt signalling that transmitted byte was not acknowledged.	0
		0	Disable the NAI.	
		1	Enable the NAI.	
3	DRMIE		Master Transmitter Data Request Interrupt Enable. This enables the DRMI interrupt which signals that the master transmitter has run out of data, has not issued a STOP, and is holding the SCL line low.	0
		0	Disable the DRMI interrupt.	
		1	Enable the DRMI interrupt.	
4	DRSIE		Slave Transmitter Data Request Interrupt Enable. This enables the DRSI interrupt which signals that the slave transmitter has run out of data and the last byte was acknowledged, so the SCL line is being held low.	0
		0	Disable the DRSI interrupt.	
		1	Enable the DRSI interrupt.	
5	REFIE		Receive FIFO Full Interrupt Enable. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data.	0
		0	Disable the RFFI.	
		1	Enable the RFFI.	
6	RFDAIE		Receive Data Available Interrupt Enable. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty).	0
		0	Disable the DAI.	
		1	Enable the DAI.	
7	TFFIE		Transmit FIFO Not Full Interrupt Enable. This enables the Transmit FIFO Not Full interrupt to indicate that the more data can be written to the transmit FIFO. Note that this is not full. It is intended help the CPU to write to the I²C block only when there is room in the FIFO and do this without polling the status register.	0
		0	Disable the TFFI.	
		1	Enable the TFFI.	

Table 323. I²C Control register (I2C_CTL - address 0x2008 C308) bit description

Bit	Symbol	Value	Description	Reset Value
8	SRST		Soft reset. This is only needed in unusual circumstances. If a device issues a start condition without issuing a stop condition. A system timer may be used to reset the I²C if the bus remains busy longer than the time-out period. On a soft reset, the Tx and Rx FIFOs are flushed, I2C_STS register is cleared, and all internal state machines are reset to appear idle. The I2C_CLKHI, I2C_CLKLO and I2C_CTL (except Soft Reset Bit) are NOT modified by a soft reset.	0
		0	See the text.	
		1	Reset the I²C to idle state. Self clearing.	
31:9	-		Reserved. Read value is undefined, only zero should be written.	NA

14.8.11 I²C Clock High Register

The CLK register holds a terminal count for counting 48 MHz clock cycles to create the high period of the slower I²C serial clock, SCL.

Table 324. I²C_CLKHI register (I2C_CLKHI - address 0x2008 C30C) bit description

Bit	Symbol	Description	Reset Value
7:0	CDHI	Clock divisor high. This value is the number of 48 MHz clocks the serial clock (SCL) will be high.	0xB9
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.12 I²C Clock Low Register

The CLK register holds a terminal count for counting 48 MHz clock cycles to create the low period of the slower I²C serial clock, SCL.

Table 325. I²C_CLKLO register (I2C_CLKLO - address 0x2008 C310) bit description

Bit	Symbol	Description	Reset Value
7:0	CDLO	Clock divisor low. This value is the number of 48 MHz clocks the serial clock (SCL) will be low.	0xB9
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

14.8.13 OTG Clock Control Register

This register controls the clocking of the OTG controller. Whenever software wants to access the registers, the corresponding clock control bit needs to be set. The software does not have to repeat this exercise for every register access, provided that the corresponding OTGClkCtrl bits are already set.

Table 326. OTG clock control register (CLKCTRL - address 0x2008 CFF4) bit description

Bit	Symbol	Value	Description	Reset Value
0	HOST_CLK_EN		Host clock enable	0
		0	Disable the Host clock.	
		1	Enable the Host clock.	
1	DEV_CLK_EN		Device clock enable	0
		0	Disable the Device clock.	
		1	Enable the Device clock.	
2	I2C_CLK_EN		I^2C clock enable	0
		0	Disable the I^2C clock.	
		1	Enable the I^2C clock.	
3	OTG_CLK_EN		OTG clock enable. In device-only applications, this bit enables access to the PORTSEL register.	0
		0	Disable the OTG clock.	
		1	Enable the OTG clock.	
4	AHB_CLK_EN		AHB master clock enable	0
		0	Disable the AHB clock.	
		1	Enable the AHB clock.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	NA

14.8.14 OTG Clock Status Register

This register holds the clock availability status. When enabling a clock via OTGClkCtrl, software should poll the corresponding bit in this register. If it is set, then software can go ahead with the register access. Software does not have to repeat this exercise for every access, provided that the OTGClkCtrl bits are not disturbed.

Table 327. OTG clock status register (CLKST - address 0x2008 CFF8) bit description

Bit	Symbol	Value	Description	Reset Value
0	HOST_CLK_ON		Host clock status.	0
		0	Host clock is not available.	
		1	Host clock is available.	
1	DEV_CLK_ON		Device clock status.	0
		0	Device clock is not available.	
		1	Device clock is available.	
2	I2C_CLK_ON		I^2C clock status.	0
		0	I^2C clock is not available.	
		1	I^2C clock is available.	
3	OTG_CLK_ON		OTG clock status.	0
		0	OTG clock is not available.	
		1	OTG clock is available.	
4	AHB_CLK_ON		AHB master clock status.	0
		0	AHB clock is not available.	
		1	AHB clock is available.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	NA

14.8.15 Interrupt handling

The interrupts set in the OTGIntSt register are set and cleared during HNP switching. All OTG related interrupts, if enabled, are routed to the USB_OTG_INT bit in the USBIntSt register.

I^2C related interrupts are set in the I2C_STS register and routed, if enabled by I2C_CTL, to the USB_I2C_INT bit.

For more details on the interrupts created by device controller, see the USB device chapter. For interrupts created by the host controllers, see the OHCI specification.

The EN_USB_INTS bit in the USBIntSt register enables the routing of any of the USB related interrupts to the NVIC controller (see Figure 54).

Remark: During the HNP switching between host and device with the OTG stack active, an action may raise several levels of interrupts. It is advised to let the OTG stack initiate any actions based on interrupts and ignore device and host level interrupts. This means that during HNP switching, the OTG stack provides the communication to the host and device controllers.

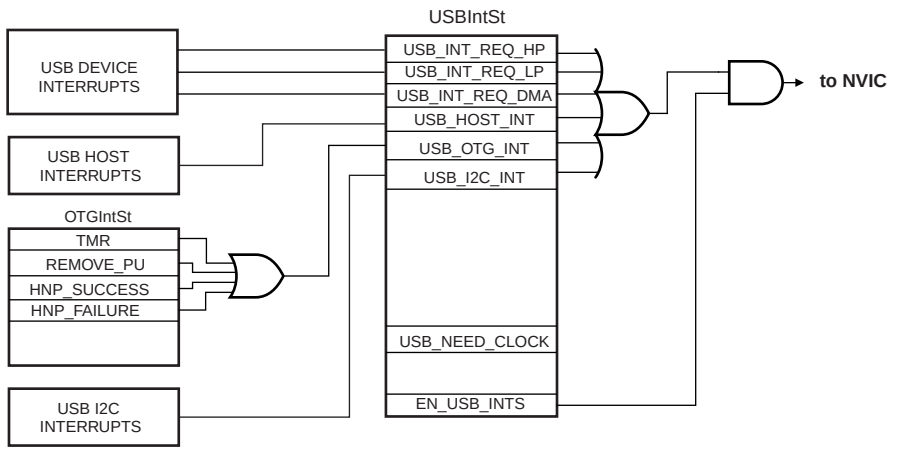


Fig 54. USB OTG interrupt handling

14.9 HNP support

This section describes the hardware support for the Host Negotiation Protocol (HNP) provided by the OTG controller.

When two dual-role OTG devices are connected to each other, the plug inserted into the mini-AB receptacle determines the default role of each device. The device with the mini-A plug inserted becomes the default Host (A-device), and the device with the mini-B plug inserted becomes the default Peripheral (B-device).

Once connected, the default Host (A-device) and the default Peripheral (B-device) can switch Host and Peripheral roles using HNP.

The context of the OTG controller operation is shown in Figure 55. Each controller (Host, Device, or OTG) communicates with its software stack through a set of status and control registers and interrupts. In addition, the OTG software stack communicates with the external OTG transceiver through the I^2C interface and the external transceiver interrupt signal.

The OTG software stack is responsible for implementing the HNP state machines as described in the On-The-Go Supplement to the USB 2.0 Specification.

The OTG controller hardware provides support for some of the state transitions in the HNP state machines as described in the following subsections.

The USB state machines, the HNP switching, and the communications between the USB controllers are described in more detail in the following documentation:

- USB OHCI specification
- USB OTG supplement, version 1.2
- USB 2.0 specification
- ISP1302 data sheet and user manual

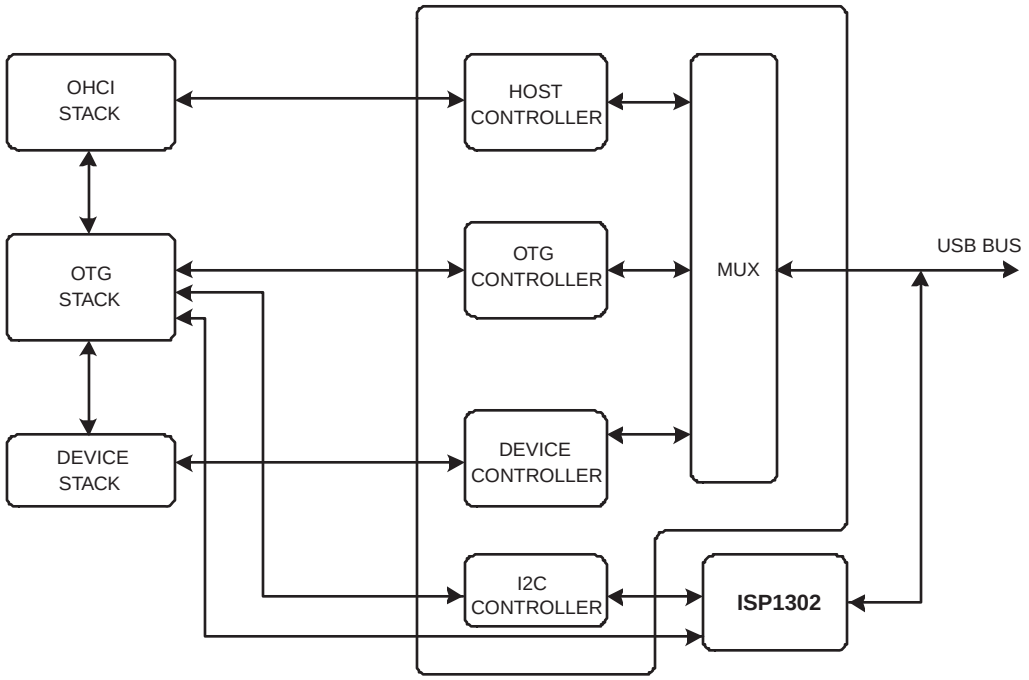


Fig 55. USB OTG controller with software stack

14.9.1 B-device: peripheral to host switching

In this case, the default role of the OTG controller is peripheral (B-device), and it switches roles from Peripheral to Host.

The On-The-Go Supplement defines the behavior of a dual-role B-device during HNP using a state machine diagram. The OTG software stack is responsible for implementing all of the states in the Dual-Role B-Device State Diagram.

The OTG controller hardware provides support for the state transitions between the states b_peripheral, b_wait_acon, and b_host in the Dual-Role B-Device state diagram. Setting B_HNP_TRACK in the OTGStCtrl register enables hardware support for the B-device switching from peripheral to host. The hardware actions after setting this bit are shown in Figure 56.

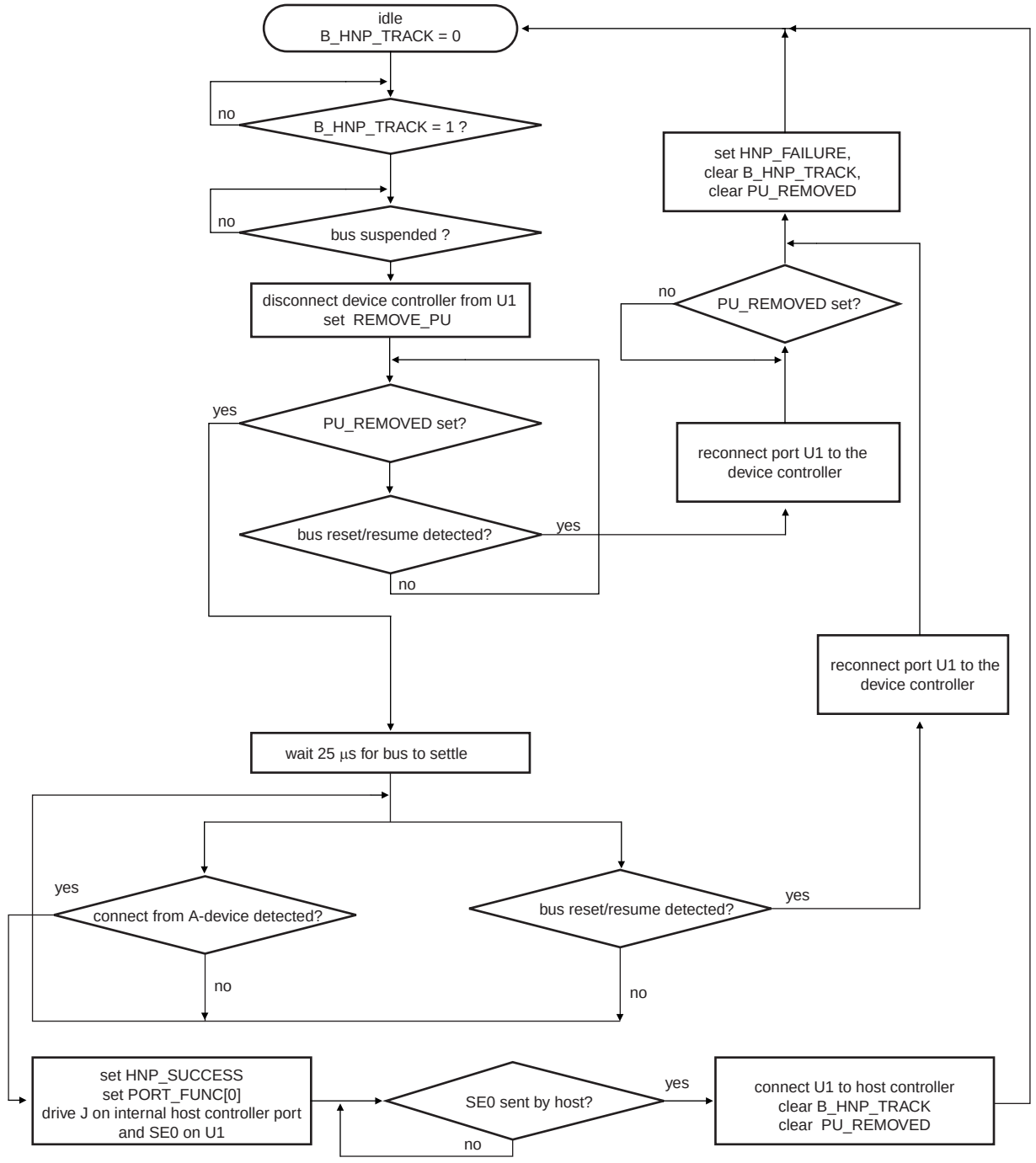


Fig 56. Hardware support for B-device switching from peripheral state to host state

Figure 57 shows the actions that the OTG software stack should take in response to the hardware actions setting REMOVE_PU, HNP_SUCCESS, AND HNP_FAILURE. The relationship of the software actions to the Dual-Role B-Device states is also shown. B-device states are in bold font with a circle around them.

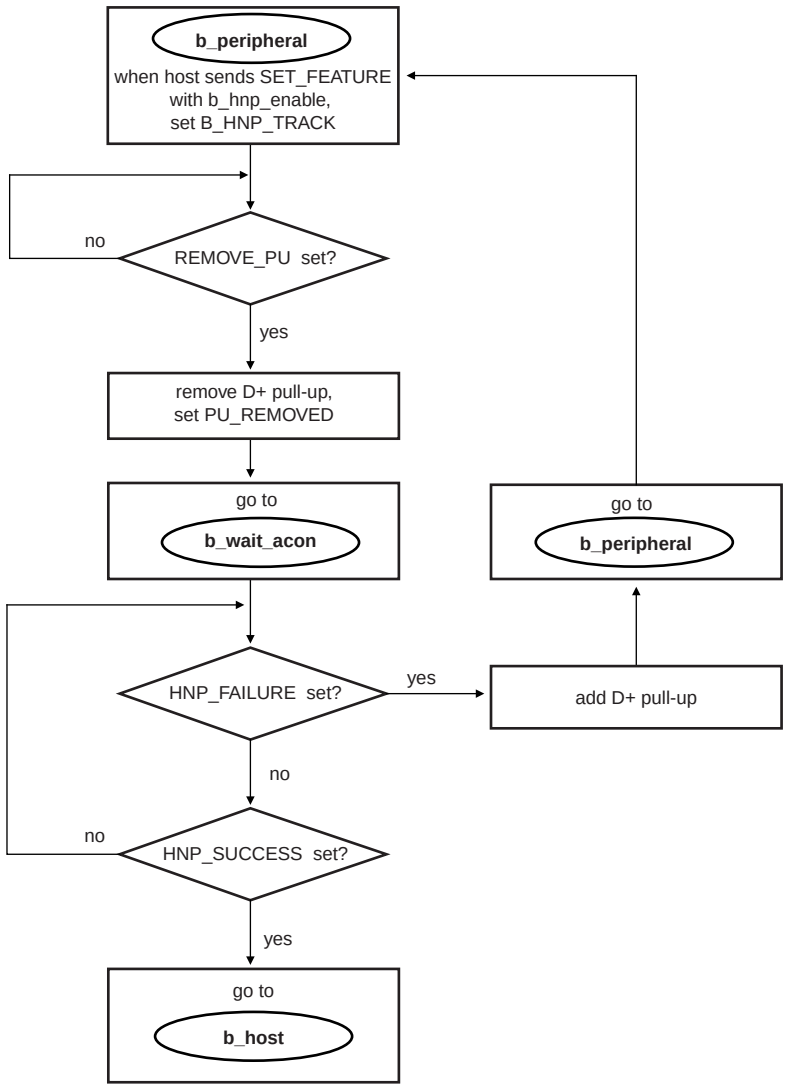


Fig 57. State transitions implemented in software during B-device switching from peripheral to host

Note that only the subset of B-device HNP states and state transitions supported by hardware are shown. Software is responsible for implementing all of the HNP states.

Figure 57 may appear to imply that the interrupt bits such as REMOVE_PU should be polled, but this is not necessary if the corresponding interrupt is enabled.

Following are code examples that show how the actions in Figure 57 are accomplished. The examples assume that ISP1302 is being used as the external OTG transceiver.

Remove D+ pull-up

```
/* Remove D+ pull-up through ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
```

```
OTG_I2C_TX = 0x201; // Clear DP_PULLUP bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Add D+ pull-up

```
/* Add D+ pull-up through ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x201; // Set DP_PULLUP bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

14.9.2 A-device: host to peripheral HNP switching

In this case, the role of the OTG controller is host (A-device), and the A-device switches roles from host to peripheral.

The On-The-Go Supplement defines the behavior of a dual-role A-device during HNP using a state machine diagram. The OTG software stack is responsible for implementing all of the states in the Dual-Role A-Device State Diagram.

The OTG controller hardware provides support for the state transitions between a_host, a_suspend, a_wait_vfall, and a_peripheral in the Dual-Role A-Device state diagram. Setting A_HNP_TRACK in the OTGStCtrl register enables hardware support for switching the A-device from the host state to the device state. The hardware actions after setting this bit are shown in Figure 58.

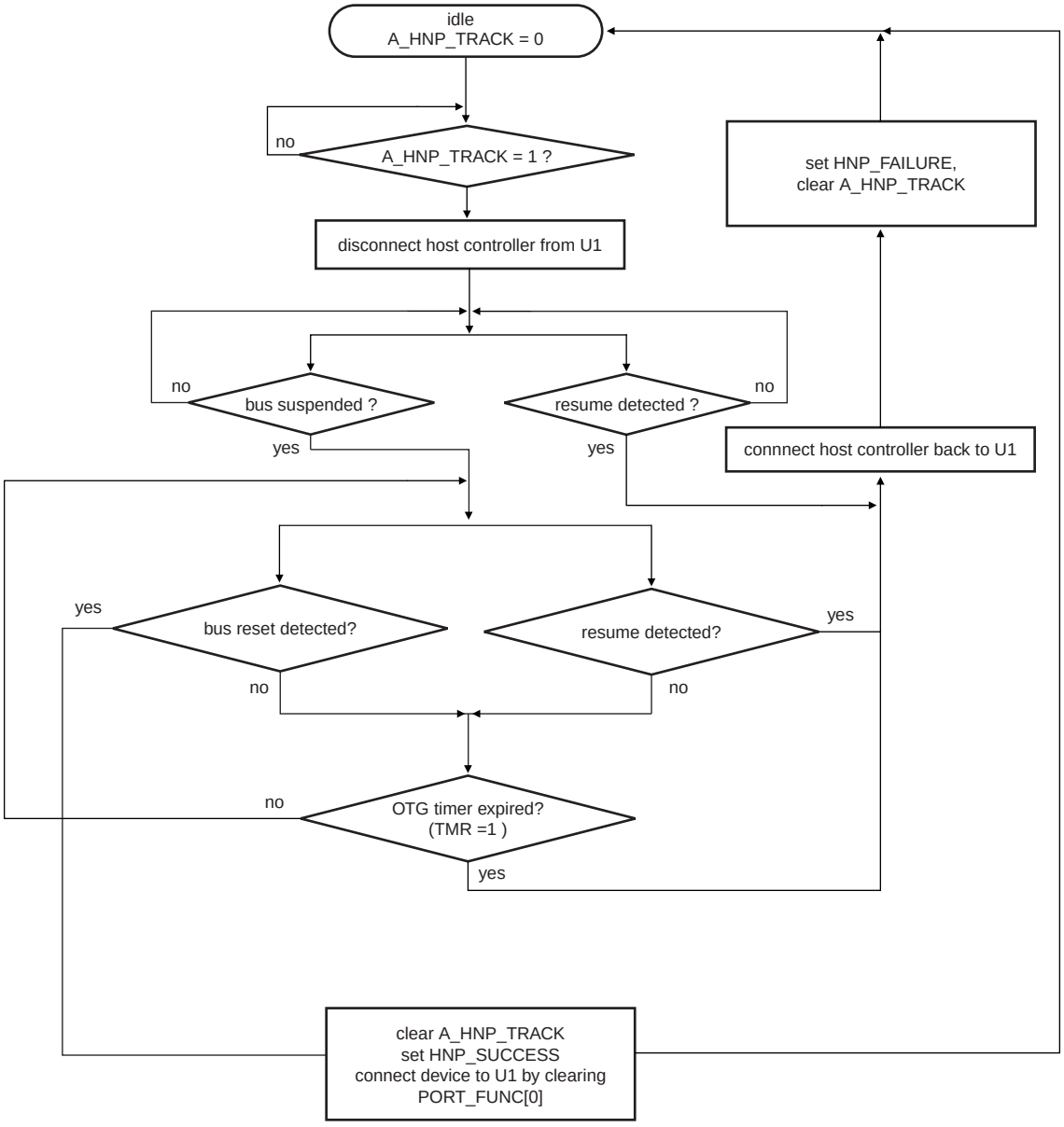


Fig 58. Hardware support for A-device switching from host state to peripheral state

Figure 59 shows the actions that the OTG software stack should take in response to the hardware actions setting TMR, HNP_SUCCESS, and HNP_FAILURE. The relationship of the software actions to the Dual-Role A-Device states is also shown. A-device states are shown in bold font with a circle around them.

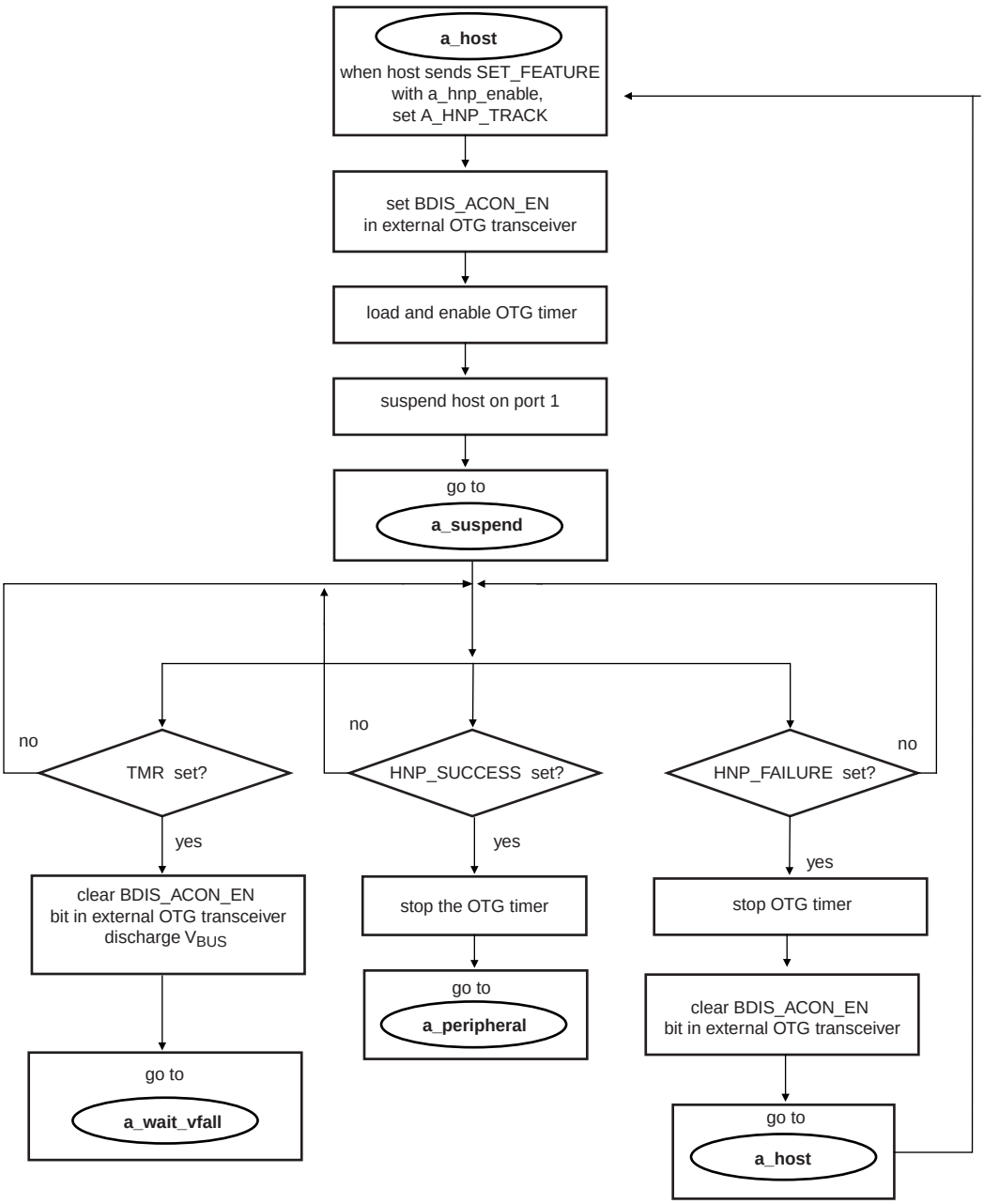


Fig 59. State transitions implemented in software during A-device switching from host to peripheral

Note that only the subset of A-device HNP states and state transitions supported by hardware are shown. Software is responsible for implementing all of the HNP states.

Figure 59 may appear to imply that the interrupt bits such as TMR should be polled, but this is not necessary if the corresponding interrupt is enabled.

Following are code examples that show how the actions in Figure 59 are accomplished. The examples assume that ISP1302 is being used as the external OTG transceiver.

Set BDIS_ACON_EN in external OTG transceiver

```
/* Set BDIS_ACON_EN in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x004; // Send Mode Control 1 (Set) register address
OTG_I2C_TX = 0x210; // Set BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Clear BDIS_ACON_EN in external OTG transceiver

```
/* Set BDIS_ACON_EN in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x005; // Send Mode Control 1 (Clear) register address
OTG_I2C_TX = 0x210; // Clear BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Discharge V_{BUS}

```
/* Clear the VBUS_DRV bit in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x220; // Clear VBUS_DRV bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;

/* Set the VBUS_DISCHRG bit in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x240; // Set VBUS_DISCHRG bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Load and enable OTG timer

```
/* The following assumes that the OTG timer has previously been  */
/* configured for a time scale of 1 ms (TMR_SCALE = “10”)        */
/* and monoshot mode (TMR_MODE = 0)                              */

/* Load the timeout value to implement the a_aidl_bdis_tmr timer */
/*     the minimum value is 200 ms                               */
OTG_TIMER = 200;

/* Enable the timer */
OTG_STAT_CTRL |= TMR_EN;
```

Stop OTG timer

```
/* Disable the timer - causes TMR_CNT to be reset to 0  */
OTG_STAT_CTRL &= ~TMR_EN;

/* Clear TMR interrupt */
OTG_INT_CLR = TMR;
```

Suspend host on port 1

```
/* Write to PortSuspendStatus bit to suspend host port 1 -               */
/* this example demonstrates the low-level action software needs to take. */
/* The host stack code where this is done will be somewhat more involved. */
HC_RH_PORT_STAT1 = PSS;
```

14.10 Clocking and power management

The OTG controller clocking is shown in Figure 60.

A clock switch controls each clock with the exception of ahb_slave_clk. When the enable of the clock switch is asserted, its clock output is turned on and its CLK_ON output is asserted. The CLK_ON signals are observable in the OTGClkSt register.

To conserve power, the clocks to the Device, Host, OTG, and I^2C controllers can be disabled when not in use by clearing the respective CLK_EN bit in the OTGClkCtrl register. When the entire USB block is not in use, all of its clocks can be disabled by clearing the PCUSB bit in the PCONP register.

When software wishes to access registers in one of the controllers, it should first ensure that the respective controller's 48 MHz clock is enabled by setting its CLK_EN bit in the OTGClkCtrl register and then poll the corresponding CLK_ON bit in OTGClkSt until set. Once set, the controller's clock will remain enabled until CLK_EN is cleared by software. Accessing the register of a controller when its 48 MHz clock is not enabled will result in a data abort exception.

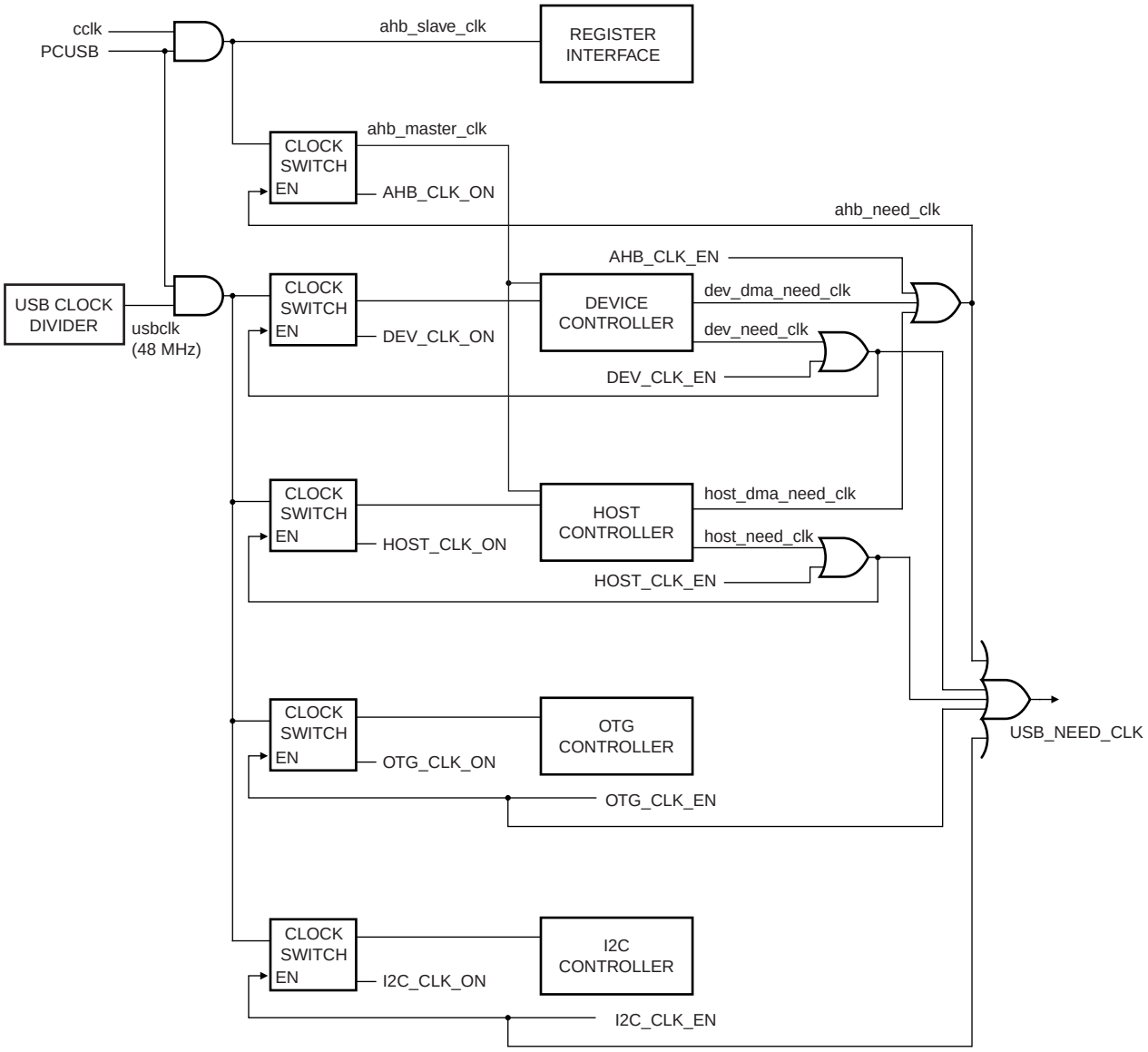


Fig 60. Clocking and power control

14.10.1 Device clock request signals

The Device controller has two clock request signals, dev_need_clk and dev_dma_need_clk. When asserted, these signals turn on the device's 48 MHz clock and ahb_master_clk respectively.

The dev_need_clk signal is asserted while the device is not in the suspend state, or if the device is in the suspend state and activity is detected on the USB bus. The dev_need_clk signal is de-asserted if a disconnect is detected (CON bit is cleared in the SIE Get Device Status register – Section 12.10.6). This signal allows DEV_CLK_EN to be cleared during normal operation when software does not need to access the Device controller registers – the Device will continue to function normally and automatically shut off its clock when it is suspended or disconnected.

The dev_dma_need_clk signal is asserted on any Device controller DMA access to memory. Once asserted, it remains active for 2 ms (2 frames), to help assure that DMA throughput is not affected by any latency associated with re-enabling ahb_master_clk. 2 ms after the last DMA access, dev_dma_need_clk is de-asserted to help conserve power. This signal allows AHB_CLK_EN to be cleared during normal operation.

14.10.1.1 Host clock request signals

The Host controller has two clock request signals, host_need_clk and host_dma_need_clk. When asserted, these signals turn on the host's 48 MHz clock and ahb_master_clk respectively.

The host_need_clk signal is asserted while the Host controller functional state is not UsbSuspend, or if the functional state is UsbSuspend and resume signaling or a disconnect is detected on the USB bus. This signal allows HOST_CLK_EN to be cleared during normal operation when software does not need to access the Host controller registers – the Host will continue to function normally and automatically shut off its clock when it goes into the UsbSuspend state.

The host_dma_need_clk signal is asserted on any Host controller DMA access to memory. Once asserted, it remains active for 2 ms (2 frames), to help assure that DMA throughput is not affected by any latency associated with re-enabling ahb_master_clk. 2 ms after the last DMA access, host_dma_need_clk is de-asserted to help conserve power. This signal allows AHB_CLK_EN to be cleared during normal operation.

14.10.2 Power-down mode support

The CPU can be configured to wake up from Power-down mode on any USB bus activity. When the chip is in Power-down mode and the USB interrupt is enabled, the assertion of USB_NEED_CLK causes the chip to wake up from Power-down mode.

Before Power-down mode can be entered when the USB activity interrupt is enabled, USB_NEED_CLK must be de-asserted. This is accomplished by clearing all of the CLK_EN bits in OTGClkCtrl and putting the Host controller into the UsbSuspend functional state. If it is necessary to wait for either of the dma_need_clk signals or the dev_need_clk to be de-asserted, the status of USB_NEED_CLK can be polled in the USBIntSt register to determine when they have all been de-asserted.

14.11 USB OTG controller initialization

The OTG device controller initialization includes the following steps:

1. Enable the device controller by setting the PCUSB bit of PCONP.
2. Configure and enable the Alt PLL (PLL1) or Main PLL (PLL0) to provide 48 MHz for usbclk and the desired frequency for cclk. For the procedure for determining the PLL setting and configuration, see Section 3.10.5.
3. Enable the desired controller clocks by setting their respective CLK_EN bits in the USBClkCtrl register. Poll the corresponding CLK_ON bits in the USBClkSt register until they are set.
4. Enable the desired USB pin functions by writing to the corresponding IOCON registers.
5. Follow the appropriate steps in Section 12.13 “USB device controller initialization” to initialize the device controller.
6. Follow the guidelines given in the OpenHCI specification for initializing the host controller.

15.1 Basic configuration

The SPIFI peripheral is configured using the following registers:

1. Power: In the PCONP register (see Section 3.3.2.2), set bit PCSPIFI.

   Remark: On reset, the SPIFI is disabled (PCSPIFI = 0).
2. SPIFI clock: see Section 3.3.3.6.
3. Pins: Select SPIFI pins and pin modes through the relevant IOCON registers (Section 7.4.1).

15.2 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 (see Table 330).
- Using the SPIFI, as described in this chapter, accomplished with a driver library available from NXP Semiconductors.

Remark: The SPIFI software library with the SPIFI API are available on LPCWare.com.

15.3 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.

A driver API included in on-chip ROM handles setup, programming and erasure. After an initialize call to the SPIFI driver, the flash content is accessible as normal memory using byte, halfword, and word accesses by the processor and/or DMA.

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 328).

Table 328. SPIFI flash memory map

Memory	Address
SPIFI data	0x2800 0000 to 0x28FF FFFF
	Remark: This is the address space allocated to the SPIFI. The area allocated allows a maximum of 16 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

15.4 Pin description

Table 329. SPIFI Pin description

Pin function	Direction	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.
$\overline{\text{SPIFI\_CS}}$	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_MOSI or IO0	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 $\overline{\text{CS}}$ goes high.
SPIFI_MISO or IO1	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 $\overline{\text{CS}}$ goes high.
SPIFI_SIO[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.

15.5 Supported devices

Serial flash devices with the following features are supported:

- Read JDEC ID
- Page programming
- at least one command with uniform erase size throughout the device

Table 330 shows a list of vendor QSPI devices which are verified to support the SPIFI API available on LPCWare.com. Other devices can be used and will run in basic single SPI mode at lower speed.

Remark: All QSPI devices have been tested at an operating voltage of 3.3 V.

Table 330. Supported QSPI devices

Manufacturer	Device name
AMIC	A25L512, A25L010, A25L020, A25L040, A25L080, A25L016, A25L032, A25LQ032
Atmel	AT25F512B, AT25DF021, AT25DF041A, AT25DF081A, AT25DF161, AT25DQ161, AT25DF321A, AT25DF641
Chingis	Pm25LD256, Pm25LD512, Pm25LD010, Pm25LD020, Pm25LD040, Pm25LQ032
Elite (ESMT)	F25L08P, F25L16P, F25L32P, F25L32Q
Eon	EN25F10, EN25F20, EN25F40, EN25Q40, EN25F80, EN25Q80, EN25QH16, EN25Q32, EN25Q64, EN25Q128
Gigadevice	GD25Q512, GD25Q10, GD25Q20, GD25Q40, GD25Q80, GD25Q16, GD25Q32, GD25Q64
Macronix	MX25L8006, MX25L8035, MX25L8036, MX25U8035[1], MX25L1606, MX25L1633, MX25L1635, MX25L1636, MX25U1635[1], MX25L3206, MX25L3235, MX25L3236, MX25U3235[1], MX25L6436, MX25L6445, MX25L6465, MX25L12836, MX25L12845, MX25L12865, MX25L25635, MX25L25735
Numonyx	M25P10, M25P20, M25P40, M25P80, M25PX80, M25P16, M25PX16, M25P32, M25PX32, M25P64, M25PX64, N25Q032, N25Q064, N25Q128
Spansion	S25FL004K, S25FL008K, S25FL016K, S25FL032K, S25FL032P, S25FL064K, S25FL064P, S25FL129P
SST	SST26VF016, SST26VF032, SST25VF064
Winbond	W25Q40, W25Q80, W25Q16, W25Q32, W25Q64

[1] Level translation circuitry, which might affect performance, is required for these parts.

The following devices lack one or more of these features and are not supported:

- Elite: F25L004, F25L008, F25L016.
- Eon: 25B64.
- SST: 25VF512, 25WF512, 25VF010, 25WF010, 25LF020, 25VF020, 25WF020, 25VF040, 25WF040, 25VF080, 25WF080, 25VF016, 25VF032.

15.6 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 a software library available on LPCWare.com or using the register interface.

15.7 Register description

The SPIFI register interface supports word accesses.

Table 331. Register overview: SPIFI (base address 0x2009 4000)

Name	Access	Address offset	Description	Reset value	Reference
CTRL	R/W	0x000	SPIFI control register	0x400F FFFF	Table 332
CMD	R/W	0x004	SPIFI command register	0x0000 0000	Table 333
ADDR	R/W	0x008	SPIFI address register	0x0000 0000	Table 334
IDATA	R/W	0x00C	SPIFI intermediate data register	0x0000 0000	Table 335
CLIMIT	R/W	0x010	SPIFI cache limit register	0x0800 0000	Table 336
DATA	R/W	0x014	SPIFI data register	0x0000 0000	Table 337
MCMD	R/W	0x018	SPIFI memory command register	0x0000 0000	Table 338
STAT	R/W	0x01C	SPIFI status register	0x0200 0000	Table 339

15.7.1 SPIFI control register

The SPIFI control register controls the overall operation of the SPIFI and should be written before any commands are initiated.

Table 332. SPIFI control register (CTRL, address 0x2009 4000) 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.	1111
20	-		Reserved.	-
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

Table 332. SPIFI control register (CTRL, address 0x2009 4000) bit description

Bit	Symbol	Value	Description	Reset value
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 $\overline{CS}$ is HIGH.	
		1	SCK HIGH. the SPIFI keeps SCK high after the rising edge for the last bit of each command and while $\overline{CS}$ is HIGH, and drives it low after it drives $\overline{CS}$ 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 rising 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 rising 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 rising 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. DRQEN should only be used in Command mode.	0

15.7.2 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 333. SPIFI command register (CMD, address 0x2009 4004) 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 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.	

Table 333. SPIFI command register (CMD, address 0x2009 4004) 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, 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

15.7.3 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 334. SPIFI address register (ADDR, address 0x2009 4008) bit description

Bit	Symbol	Description	Reset value
31:0	ADDRESS	Address.	0

15.7.4 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 335. SPIFI intermediate data register (IDATA, address 0x2009 400C) bit description

Bit	Symbol	Description	Reset value
31:0	IDATA	Value of intermediate bytes.	0

15.7.5 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 336. SPIFI cache limit register (CLIMIT, address 0x2009 4010) bit description

Bit	Symbol	Description	Reset value
31:0	CLIMIT	Zero-based upper limit of cacheable memory	0x0800 0000

15.7.6 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 337. SPIFI Data register (DATA, address 0x2009 4014) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	Input or output data	0

15.7.7 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 338. SPIFI memory command register (MCMD, address 0x2009 4018) 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.	
23:21	FRAMEFORM		This field controls the opcode and address fields.	0
		0x0	Reserved.	
		0x1	Reserved.	
		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

15.7.8 SPIFI status register

This register indicates the state of the SPIFI.

Table 339. SPIFI status register (STAT, address 0x2009 401C) 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 $\overline{CS}$.	5
23:6	-	Reserved	0
31:24	VERSION	-	0x02

15.8 Functional description

15.8.1 Data transfer

Serial SPI uses the signals SPIFI_SCK, $\overline{SPIFI\_CS}$,SPIFI_MISO, and SPIFI_MISO, while quad mode adds the two IO signals SPIFI_SIO[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 61 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.

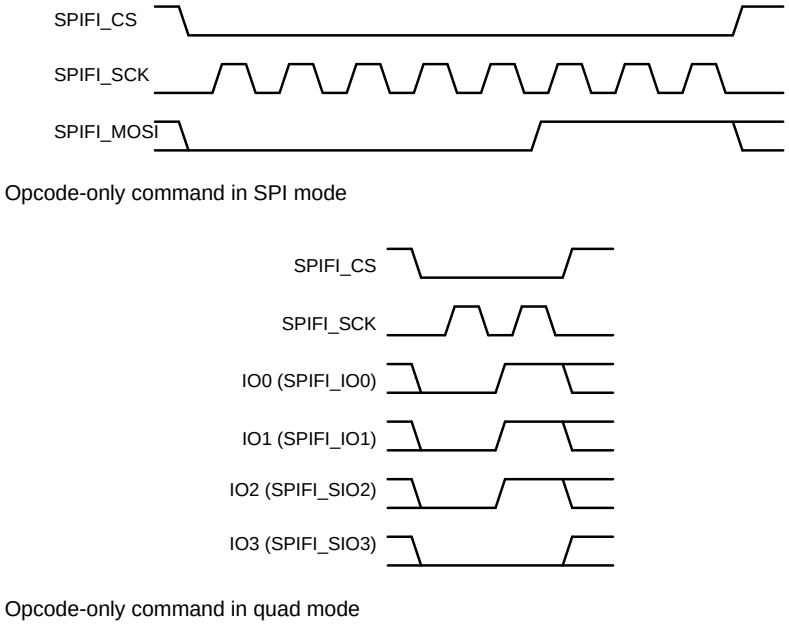


Fig 61. Opcode only commands

Figure 62 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.

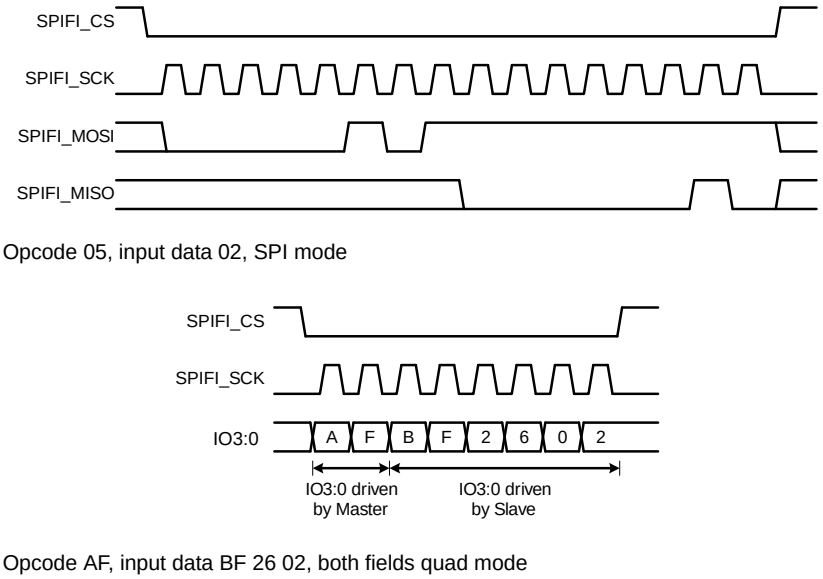


Fig 62. Read commands

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.

15.8.2 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.

The SPIFI includes a cache to maximize performance for accesses to the serial flash region of the memory map. The cache is only used in Memory mode and can be disabled.

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.

15.8.3 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:

- DRQEN 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.

16.1 How to read this chapter

The SD card interface is available on most LPC408x/407x devices, see Section 1.4 for details.

16.2 Basic configuration

The SD card interface (also known as MCI or Multimedia card interface) is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PC_SD.

   Remark: On reset, the SD card interface is disabled (PCSD = 0).
2. Peripheral clock: The SD card interface operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Select SD card interface pins and their modes through the relevant IOCON registers (Section 7.4.1).
4. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

16.3 Introduction

The Secure Digital card interface is an interface between the Advanced Peripheral Bus (APB) system bus and multimedia and/or secure digital memory cards. It consists of two parts:

- The SD card interface provides all functions specific to the Secure Digital memory card, such as the clock generation unit, power management control, command and data transfer. The interface also supports the Multimedia Card Interface.
- The APB interface accesses SD card interface registers, and generates interrupt and DMA request signals.

16.4 Features

The following features are provided by the SD card interface:

- Conformance to *Secure Digital Memory Card Physical Layer Specification, v0.96*.
- Conformance to *Multimedia Card Specification v2.11*.
- Use as a multimedia card bus or a secure digital memory card bus host. It can be connected to several multimedia cards, or a single secure digital memory card.
- DMA supported through the General Purpose DMA Controller.

16.5 Pin description

Table 340. SD/MMC card interface pin description

Pin Name	Type	Description
SD_CLK	Output	Clock output
SD_CMD	Input	Command input/output.
SD_DAT[3:0]	Output	Data lines. Only SD_DAT[0] is used for Multimedia cards.
SD_PWR	Output	Power Supply Enable for external card power supply.

16.6 Functional overview

The SD card interface may be used as a secure digital memory card bus host (see Section 16.6.1 “Secure digital memory card”) or as a multimedia card bus host (see Section 16.6.2 “Multimedia card”). A single secure digital memory card or up to 4 multimedia cards (depending on board loading) may be connected.

16.6.1 Secure digital memory card

Figure 63 shows the secure digital memory card connection.

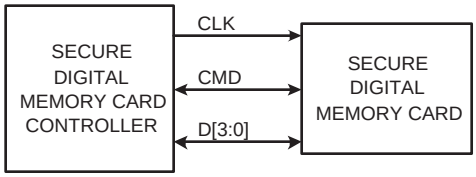


Fig 63. Secure digital memory card connection

16.6.1.1 Secure digital memory card bus signals

The following signals are used on the secure digital memory card bus:

- SD_CLK Host to card clock signal
- SD_CMD Bidirectional command/response signal
- SD_DAT[3:0] Bidirectional data signals

16.6.2 Multimedia card

Figure 64 shows the multimedia card system.

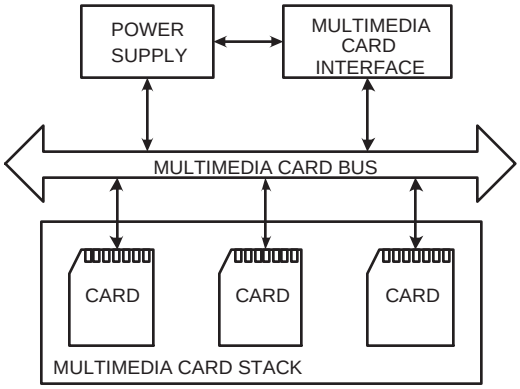


Fig 64. Multimedia card system

Multimedia cards are grouped into three types according to their function:

- Read Only Memory (ROM) cards, containing pre-programmed data
- Read/Write (R/W) cards, used for mass storage
- Input/Output (I/O) cards, used for communication

The multimedia card system transfers commands and data using three signal lines:

- CLK: One bit is transferred on both command and data lines with each clock cycle. The clock frequency varies between 0 MHz and 20 MHz (for a multimedia card) or 0 MHz and 25 MHz (for a secure digital memory card).
- CMD: Bidirectional command channel that initializes a card and transfers commands. CMD has two operational modes:
  - Open-drain for initialization
  - Push-pull for command transfer
- DAT: Bidirectional data channel, operating in push-pull mode

16.6.3 SD card interface details

Figure 65 shows a simplified block diagram of the SD card interface.

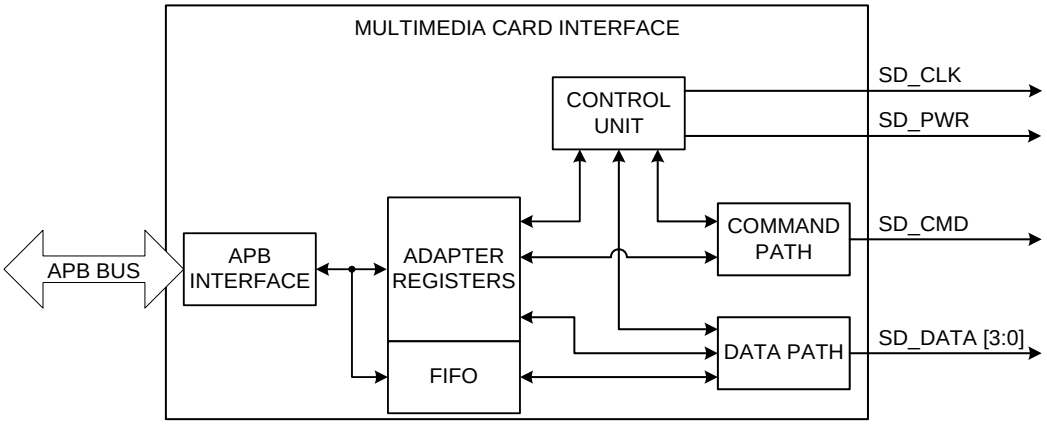


Fig 65. SD card interface

The SD card interface is a secure digital/multimedia memory card bus master that provides an interface to a multimedia card stack or to a secure digital memory card. It consists of five subunits:

- Adapter register block
- Control unit
- Command path
- Data path
- Data FIFO

16.6.3.1 Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when 1 is written into the corresponding bit location of the MCIClear register.

16.6.3.2 Control unit

The control unit contains the power management functions and the clock divider for the memory card clock.

There are three power phases:

- Power-off
- Power-up
- Power-on

The power management logic controls an external power supply unit, and disables the card bus output signals during the power-off or power-up phases. The power-up phase is a transition phase between the power-off and power-on phases, and allows an external power supply to reach the card bus operating voltage. A device driver is used to ensure that the interface remains in the power-up phase until the external power supply reaches the operating voltage.

The clock management logic generates and controls the SD_CLK signal. The SD_CLK output can use either a clock divide or clock bypass mode. The clock output is inactive:

- after reset
- during the power-off or power-up phases
- if the power saving mode is enabled and the card bus is in the IDLE state (eight clock periods after both the command and data path subunits enter the IDLE phase)

16.6.3.3 Command path

The command path subunit sends commands to and receives responses from the cards.

16.6.3.4 Command path state machine

When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the Command Path State Machine (CPSM) sets the status flags and enters the IDLE state if a response is not required. If a response is required, it waits for the response (see Figure 66). When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set.

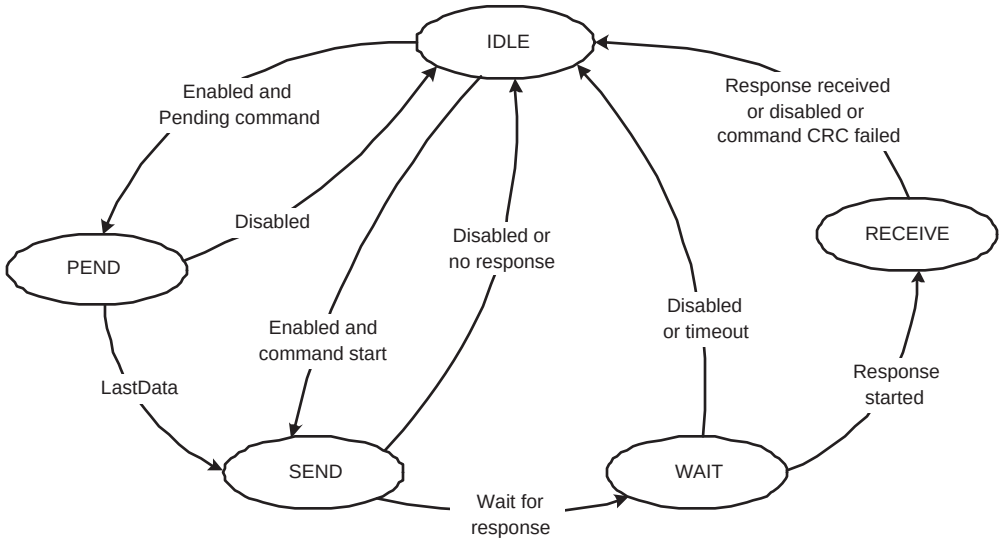


Fig 66. Command path state machine

When the WAIT state is entered, the command timer starts running. If the timeout[1] is reached before the CPSM moves to the RECEIVE state, the timeout flag is set and the IDLE[2] state is entered.

If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, the CPSM enters the PEND state, and waits for a CmdPend signal from the data path subunit. When CmdPend is detected, the CPSM moves to the SEND state. This enables the data counter to trigger the stop command transmission.

1. The timeout period has a fixed value of 64 SD_CLK clocks period.
2. The CPSM remains in the IDLE state for at least eight SD_CLK periods to meet Ncc and Nrc timing constraints.

Figure 67 shows the command transfer.

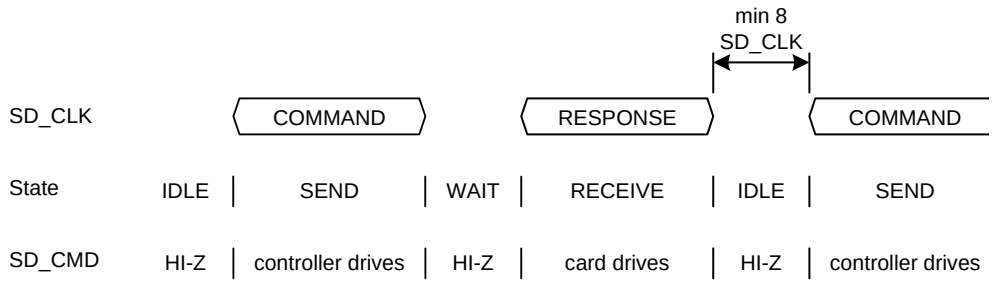


Fig 67. Command transfer

16.6.3.5 Command format

The command path operates in a half-duplex mode, so that commands and responses can either be sent or received. If the CPSM is not in the SEND state, the SD_CMD output is in HI-Z state, as shown in Figure 67. Data on SD_CMD is synchronous to the rising SD_CLK edge. All commands have a fixed length of 48 bits. Table 341 shows the command format.

Table 341. Command format

Bit Position	Width	Value	Description
0	1	1	End bit.
7:1	7	-	CRC7
39:8	32	-	Argument.
45:40	6	-	Command index.
46	1	1	Transmission bit.
47	1	0	Stat bit.

The SD card interface supports two response types. Both use CRC error checking:

- 48 bit short response (see Table 342)
- 136 bit long response (see Table 343)

Note: If the response does not contain CRC (CMD1 response), the device driver must ignore the CRC failed status.

Table 342. Simple response format

Bit Position	Width	Value	Description
0	1	1	End bit.
7:1	7	-	CRC7 (or 1111111).
39:8	32	-	Argument.
45:40	6	-	Command index.
46	1	0	Transmission bit.
47	1	0	Start bit.

Table 343. Long response format

Bit Position	Width	Value	Description
0	1	1	End bit.
127:1	127	-	CID or CSD (including internal CRC7).
133:128	6	111111	Reserved.
134	1	1	Transmission bit.
135	1	0	Start bit.

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response, and whether the response is 48 or 136 bits long (see Section 16.7.4 “Command Register” for more information). The command path implements the status flags shown in Table 344 (see Section 16.7.11 “Status Register” for more information).

Table 344. Command path status flags

Flag	Description
CmdRespEnd	Set if response CRC is OK.
CmdCrcFail	Set if response CRC fails.
CmdSent	Set when command (that does not require response) is sent.
CmdTimeOut	Response timeout.
CmdActive	Command transfer in progress.

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status). The CRC checksum is calculated for the first 120 bits of CID or CSD for the long response format. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7 bit value:

$$CRC[6:0] = \text{Remainder } [(M(x) \times x_7) / G(x)]$$

$$G(x) = x_7 + x_3 + 1$$

$$M(x) = (\text{start bit}) \times x_{39} + \ldots + (\text{last bit before CRC}) \times x_0 \text{ , or}$$

$$M(x) = (\text{start bit}) \times x_{119} + \ldots + (\text{last bit before CRC}) \times x_0$$

16.6.3.6 Data path

The card data bus width can be programmed using the clock control register. If the wide bus mode is enabled, data is transferred at four bits per clock cycle over all four data signals (SD_DAT[3:0]). If the wide bus mode is not enabled, only one bit per clock cycle is transferred over SD_DAT[0].

Depending on the transfer direction (send or receive), the Data Path State Machine (DPSM) moves to the WAIT_S or WAIT_R state when it is enabled:

- Send: The DPSM moves to the WAIT_S state. If there is data in the send FIFO, the DPSM moves to the SEND state, and the data path subunit starts sending data to a card.

- Receive: The DPSM moves to the WAIT_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the RECEIVE state, and the data path subunit starts receiving data from a card.

16.6.3.7 Data path state machine

The DPSM operates at SD_CLK frequency. Data on the card bus signals is synchronous to the rising edge of SD_CLK. The DPSM has six states, as shown in Figure 68.

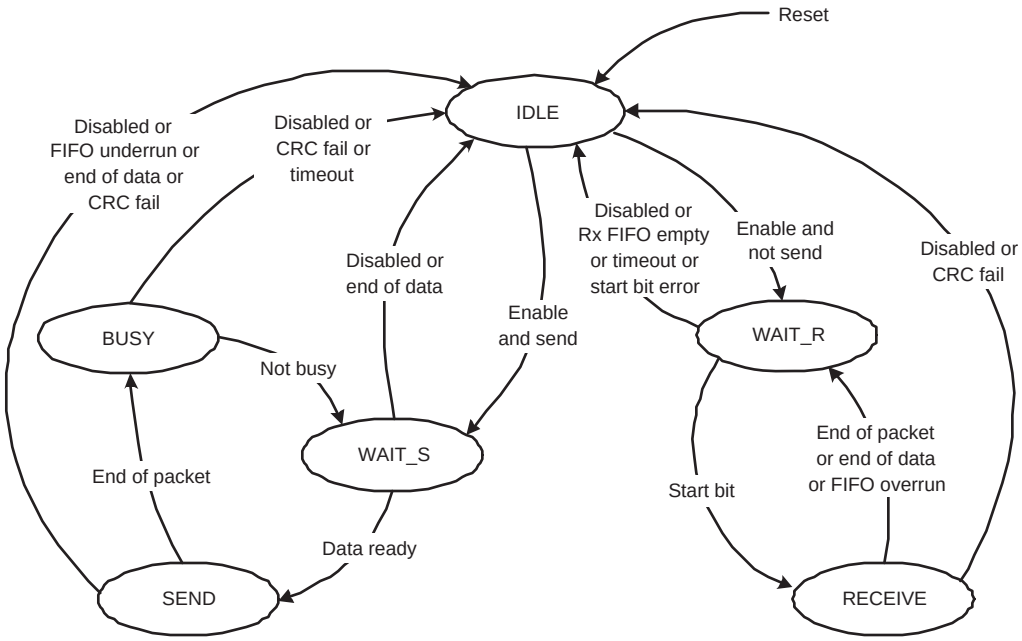


Fig 68. Data path state machine

- IDLE: The data path is inactive, and the SD_DAT[3:0] outputs are in HI-Z. When the data control register is written and the enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the WAIT_S or WAIT_R state.

  WAIT_R: If the data counter equals zero, the DPSM moves to the IDLE state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on SD_DAT.

The DPSM moves to the RECEIVE state if it receives a start bit before a timeout, and loads the data block counter. If it reaches a timeout before it detects a start bit, or a start bit error occurs, it moves to the IDLE state and sets the timeout status flag.

- RECEIVE: Serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - In block mode, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to the WAIT_R state. If not, the CRC fail status flag is set and the DPSM moves to the IDLE state.

- In stream mode, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the WAIT_R state.

If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to the WAIT_R state.

- WAIT_S: The DPSM moves to the IDLE state if the data counter is zero. If not, it waits until the data FIFO empty flag is deasserted, and moves to the SEND state.

Note: The DPSM remains in the WAIT_S state for at least two clock periods to meet Nwr timing constraints.

- SEND: The DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - In block mode, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit, and moves to the BUSY state.
  - In stream mode, the DPSM sends data to a card while the enable bit is HIGH and the data counter is not zero. It then moves to the IDLE state.

If a FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the IDLE state.

- BUSY: The DPSM waits for the CRC status flag:
  - If it does not receive a positive CRC status, it moves to the IDLE state and sets the CRC fail status flag.
  - If it receives a positive CRC status, it moves to the WAIT_S state if SD_DAT[0] is not LOW (the card is not busy).

If a timeout occurs while the DPSM is in the BUSY state, it sets the data timeout flag and moves to the IDLE state.

The data timer is enabled when the DPSM is in the WAIT_R or BUSY state, and generates the data timeout error:

- When transmitting data, the timeout occurs if the DPSM stays in the BUSY state for longer than the programmed timeout period
- When receiving data, the timeout occurs if the end of the data is not true, and if the DPSM stays in the WAIT_R state for longer than the programmed timeout period.

16.6.3.8 Data counter

The data counter has two functions:

- To stop a data transfer when it reaches zero. This is the end of the data condition.
- To start transferring a pending command (see Figure 69). This is used to send the stop command for a stream data transfer.

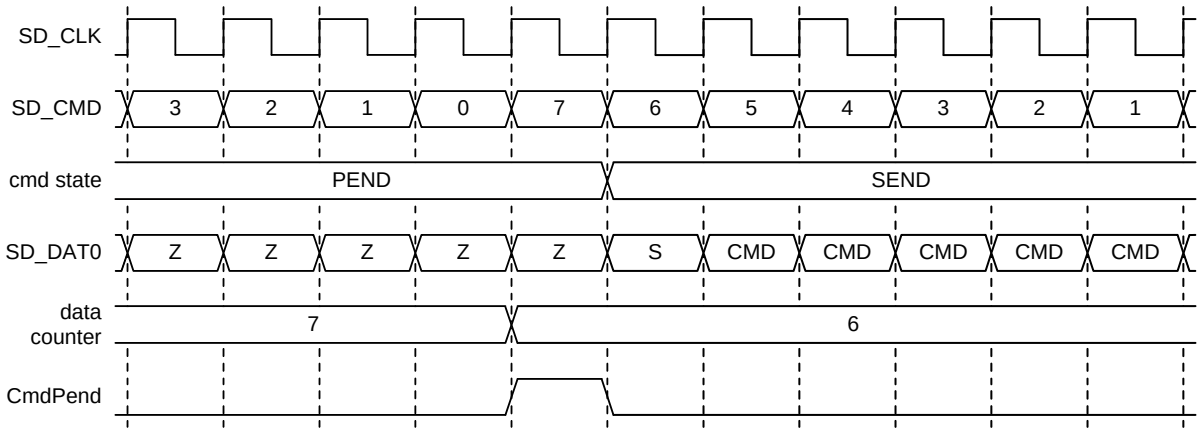


Fig 69. Pending command start

The data block counter determines the end of a data block. If the counter is zero, the end-of-data condition is TRUE (see Section 16.7.9 “Data Control Register” for more information).

16.6.3.9 Bus mode

In wide bus mode, all four data signals (SD_DAT[3:0]) are used to transfer data, and the CRC code is calculated separately for each data signal. While transmitting data blocks to a card, only SD_DAT[0] is used for the CRC token and busy signalling. The start bit must be transmitted on all four data signals at the same time (during the same clock period). If the start bit is not detected on all data signals on the same clock edge while receiving data, the DPSM sets the start bit error flag and moves to the IDLE state.

The data path also operates in half-duplex mode, where data is either sent to a card or received from a card. While not being transferred, SD_DAT[3:0] are in the HI-Z state.

Data on these signals is synchronous to the rising edge of the clock period.

If standard bus mode is selected the SD_DAT[3:1] outputs are always in HI-Z state and only the SD_DAT[0] output is driven LOW when data is transmitted.

Design note: If wide mode is selected, all data outputs enabled at the same time. If not, the SD_DAT[3:1] outputs are always off, and only the SD_DAT[0] output is driven LOW when data is transmitted.

16.6.3.10 CRC Token status

The CRC token status follows each write data block, and determines whether a card has received the data block correctly. When the token has been received, the card asserts a busy signal by driving SD_DAT[0] LOW. Table 345 shows the CRC token status values.

Table 345. CRC token status

Token	Description
010	Card has received error-free data block.
101	Card has detected a CRC error.

16.6.3.11 Status flags

Table 346 lists the data path status flags (see Section 16.7.11 “Status Register” on page 457 for more information).

Table 346. Data path status flags

Flag	Description
TxFifoFull	Transmit FIFO is full.
TxFifoEmpty	Transmit FIFO is empty.
TxFifoHalfEmpty	Transmit FIFO is half full.
TxDataAvlbl	Transmit FIFO data available.
TxUnderrun	Transmit FIFO underrun error.
RxFifoFull	Receive FIFO is full.
RxFifoEmpty	Receive FIFO is empty.
RxFifoHalfFull	Receive FIFO is half full.
RxDataAvlbl	Receive FIFO data available.
RxOverrun	Receive FIFO overrun error.
DataBlockEnd	Data block sent/received.
StartBitErr	Start bit not detected on all data signals in wide bus mode.
DataCrcFail	Data packet CRC failed.
DataEnd	Data end (data counter is zero).
DataTimeOut	Data timeout.
TxActive	Data transmission in progress.
RxActive	Data reception in progress.

16.6.3.12 CRC generator

The CRC generator calculates the CRC checksum only for the data bits in a single block, and is bypassed in data stream mode. The checksum is a 16 bit value:

$$CRC[15:0] = \text{Remainder}\ [(M(x) \times x^{15}) / G(x)]$$

$$G(x) = x^{16} + x^{12} + x^{5} + 1$$

$$M(x) - (\text{first data bit}) \times x^{n} + \ldots + (\text{last data bit}) \times x^{0}$$

16.6.3.13 Data FIFO

The data FIFO (first-in-first-out) subunit is a data buffer with transmit and receive logic.

The FIFO contains a 32 bit wide, 16-word deep data buffer, and transmit and receive logic. Because the data FIFO operates in the APB clock domain (PCLK), all signals from the subunits in the SD card interface clock domain (MCLK) are re-synchronized.

Depending on TxActive and RxActive, the FIFO can be disabled, transmit enabled, or receive enabled. TxActive and RxActive are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when TxActive is asserted (see Section 16.6.3.14 “Transmit FIFO”)

- The receive FIFO refers to the receive logic and data buffer when RxActive is asserted (see Section 16.6.3.15 “Receive FIFO”).

16.6.3.14 Transmit FIFO

Data can be written to the transmit FIFO through the APB interface once the SD card interface is enabled for transmission.

The transmit FIFO is accessible via 16 sequential addresses (see Section 16.7.15 “Data FIFO Register”). The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out.

If the transmit FIFO is disabled, all status flags are deasserted. The data path subunit asserts TxActive when it transmits data. Table 347 lists the transmit FIFO status flags.

Table 347. Transmit FIFO status flags

Flag	Description
TxFifoFull	Set to HIGH when all 16 transmit FIFO words contain valid data.
TxFifoEmpty	Set to HIGH when the transmit FIFO does not contain valid data.
TxHalfEmpty	Set to HIGH when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TxDataAvlbl	Set to HIGH when the transmit FIFO contains valid data. This flag is the inverse of the TxFifoEmpty flag.
TxUnderrun	Set to HIGH when an underrun error occurs. This flag is cleared by writing to the MCIClear register.

16.6.3.15 Receive FIFO

When the data path subunit receives a word of data, it drives data on the write data bus and asserts the write enable signal. This signal is synchronized to the PCLK domain. The write pointer is incremented after the write is completed, and the receive FIFO control logic asserts RxWrDone, that then deasserts the write enable signal.

On the read side, the content of the FIFO word pointed to by the current value of the read pointer is driven on the read data bus. The read pointer is incremented when the APB bus interface asserts RxRdPrtInc.

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RxActive when it receives data. Table 353 lists the receive FIFO status flags.

The receive FIFO is accessible via 16 sequential addresses (see Section 16.7.15 “Data FIFO Register”).

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RxActive when it receives data. Table 348 lists the receive FIFO status flags.

Table 348. Receive FIFO status flags

Symbol	Description
RxFifoFull	Set to HIGH when all 16 receive FIFO words contain valid data.
RxFifoEmpty	Set to HIGH when the receive FIFO does not contain valid data.

Table 348. Receive FIFO status flags

Symbol	Description
RxHalfFull	Set to HIGH when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request.
RxDataAvlbl	Set to HIGH when the receive FIFO is not empty. This flag is the inverse of the RxFifoEmpty flag.
RxOverrun	Set to HIGH when an overrun error occurs. This flag is cleared by writing to the MCIClear register.

16.6.3.16 APB interfaces

The APB interface generates the interrupt and DMA requests, and accesses the SD card interface registers and the data FIFO. It consists of a data path, register decoder, and interrupt/DMA logic. DMA is controlled by the General Purpose DMA controller, see that chapter for details.

16.6.3.17 Interrupt logic

The interrupt logic generates an interrupt request signal that is asserted when at least one of the selected status flags is HIGH. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set.

16.7 Register description

Table 349. Register overview: SD card interface (base address 0x400C 0000)

Name	Access	Address offset	Description	Reset value[1]	Table
PWR	R/W	0x000	Power control register.	0	350
CLOCK	R/W	0x004	Clock control register.	0	351
ARGUMENT	R/W	0x008	Argument register.	0	352
COMMAND	R/W	0x00C	Command register.	0	353
RESPCMD	RO	0x010	Response command register.	0	355
RESPONSE0	RO	0x014	Response register.	0	356
RESPONSE1	RO	0x018	Response register.	0	356
RESPONSE2	RO	0x01C	Response register.	0	356
RESPONSE3	RO	0x020	Response register.	0	356
DATATIMER	R/W	0x024	Data Timer.	0	358
DATALENGTH	R/W	0x028	Data length register.	0	359
DATACTRL	R/W	0x02C	Data control register.	0	360
DATACNT	RO	0x030	Data counter.	0	362
STATUS	RO	0x034	Status register.	0	363
CLEAR	WO	0x038	Clear register.	-	364
MASK0	R/W	003C	Interrupt 0 mask register.	0	365
FIFOCNT	RO	0x4048	FIFO Counter.	0	366
FIFO	R/W	0x080 to 0x0BC	Data FIFO Register.	0	367

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

16.7.1 Power Control Register

The PWR register controls an external power supply. Power can be switched on and off, and adjust the output voltage. Table 350 shows the bit assignment of the Power register.

The active level of the SD_PWR pin can be selected by bit 3 of the SCS register (see Section 3.3.7.1 “System Controls and Status register” on page 45 for details).

Table 350: Power Control register (PWR - address 0x400C 0000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CTRL		Power control	0
		0x0	Power-off	
		0x1	Reserved	
		0x2	Power-up	
		0x3	Power-on	
5:2	-		Reserved. Read value is undefined, only zero should be written.	NA
6	OPENDRAIN		SD_CMD output control.	0
7	ROD		Rod control.	0
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

When the external power supply is switched on, the software first enters the power-up phase, and waits until the supply output is stable before moving to the power-on phase. During the power-up phase, SD_PWR is set HIGH. The card bus outlets are disabled during both phases.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

16.7.2 Clock Control Register

The Clock register controls the SD_CLK output. Table 351 shows the bit assignment of the clock control register.

Table 351: MCI Clock Control register (CLOCK - address 0x400C 0004) bit description

Bit	Symbol	Value	Description	Reset Value
7:0	CLKDIV		Bus clock period: SD_CLK frequency = MCLK / [2×(ClkDiv+1)].	0
8	ENABLE		Enable SD card bus clock:	0
		0	Clock disabled.	
		1	Clock enabled.	
9	PWRSAVE		Disable SD_CLK output when bus is idle:	0
		0	Always enabled.	
		1	Clock enabled when bus is active.	
10	BYPASS		Enable bypass of clock divide logic:	0
		0	Disable bypass.	
		1	Enable bypass. MCLK driven to card bus output (SD_CLK).	
11	WIDEBUS		Enable wide bus mode.	0
		0	Standard bus mode (only SD_DAT[0] used).	
		1	Wide bus mode (SD_DAT[3:0] used)	
31:12	-		Reserved. Read value is undefined, only zero should be written.	NA

While the SD card interface is in identification mode, the SD_CLK frequency must be less than 400 kHz. The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

16.7.3 Argument Register

The Argument register contains a 32 bit command argument, which is sent to a card as part of a command message. Table 352 shows the bit assignment of the Argument register.

Table 352: MCI Argument register (ARGUMENT - address 0x400C 0008) bit description

Bit	Symbol	Description	Reset Value
31:0	CmdArg	Command argument	0x0000 0000

If a command contains an argument, it must be loaded into the argument register before writing a command to the command register.

16.7.4 Command Register

The Command register contains the command index and command type bits:

- The command index is sent to a card as part of a command message.
- The command type bits control the Command Path State Machine (CPSM). Writing 1 to the enable bit starts the command send operation, while clearing the bit disables the CPSM.

Table 353 shows the bit assignment of the Command register.

Table 353: MCI Command register (COMMAND - address 0x400C 000C) bit description

Bit	Symbol	Description	Reset Value
5:0	CmdIndex	Command index.	0
6	Response	If set, CPSM waits for a response.	0
7	LongRsp	If set, CPSM receives a 136 bit long response.	0
8	Interrupt	If set, CPSM disables command timer and waits for interrupt request.	0
9	Pending	If set, CPSM waits for CmdPend before it starts sending a command.	0
10	Enable	If set, CPSM is enabled.	0
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Table 354 shows the response types.

Table 354: Command Response Types

Response	Long Response	Description
0	0	No response, expect CmdSent flag.
0	1	No response, expect CmdSent flag.
1	0	Short response, expect CmdRespEnd or CmdCrcFail flag.
1	1	Long response, expect CmdRespEnd or CmdCrcFail flag.

16.7.5 Command Response Register

The RespCommand register contains the command index field of the last command response received. Table 353 shows the bit assignment of the RespCommand register.

Table 355: MCI Command Response register (RESPCMD - address 0x400C 0010) bit description

Bit	Symbol	Description	Reset Value
5:0	RESPCMD	Response command index	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	NA

If the command response transmission does not contain the command index field (long response), the RespCmd field is unknown, although it must contain 111111 (the value of the reserved field from the response).

16.7.6 Response Registers

The Response0-3 registers contain the status of a card, which is part of the received response. Table 356 shows the bit assignment of the Response0-3 registers.

Table 356: MCI Response registers (RESPONSE[0:3] - addresses 0x400C 0014, 0x400C 0018, 0x400C 001C and 0x400C 0020) bit description

Bit	Symbol	Description	Reset Value
31:0	STATUS	Card status	0

The card status size can be 32 or 127 bits, depending on the response type (see Table 357).

Table 357: Response Register Type

Description	Short Response	Long Response
Response0	Card status [31:0]	Card status [127:96]
Response1	Unused	Card status [95:64]
Response2	Unused	Card status [63:32]
Response3	Unused	Card status [31:1]

The most significant bit of the card status is received first. The Response3 register LSBit is always 0.

16.7.7 Data Timer Register

The DataTimer register contains the data time-out period, in card bus clock periods. Table 358 shows the bit assignment of the DataTimer register.

Table 358: MCI Data Timer register (DATATIMER - address 0x400C 0024) bit description

Bit	Symbol	Description	Reset Value
31:0	DATATIME	Data timeout period.	0

A counter loads the value from the data timer register, and starts decrementing when the Data Path State Machine (DPSM) enters the WAIT_R or BUSY state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

A data transfer must be written to the data timer register and the data length register before being written to the data control register.

16.7.8 Data Length Register

The DataLength register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts. Table 359 shows the bit assignment of the DataLength register.

Table 359: MCI Data Length register (DATALENGTH - address 0x400C 0028) bit description

Bit	Symbol	Description	Reset Value
15:0	DATALENGTH	Data length value	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

For a block data transfer, the value in the data length register must be a multiple of the block size (see Section 16.7.9 “Data Control Register”).

To initiate a data transfer, write to the data timer register and the data length register before writing to the data control register.

16.7.9 Data Control Register

The DataCtrl register controls the DPSM. Table 360 shows the bit assignment of the DataCtrl register.

Table 360: Data Control register (DATACTRL - address 0x400C 002C) bit description

Bit	Symbol	Value	Description	Reset Value
0	ENABLE		Data transfer enable.	0
1	DIRECTION		Data transfer direction	0
		0	From controller to card.	
		1	From card to controller.	
2	MODE		Data transfer mode	0
		0	Block data transfer.	
		1	Stream data transfer.	
3	DMAENABLE		Enable DMA	0
		0	DMA disabled.	
		1	DMA enabled.	
7:4	BLOCKSIZE		Data block length	0
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Data transfer starts if 1 is written to the enable bit. Depending on the direction bit, the DPSM moves to the WAIT_S or WAIT_R state. It is not necessary to clear the enable bit after the data transfer. BlockSize controls the data block length if Mode is 0, as shown in Table 361.

Table 361: Data Block Length

Block Size	Block Length
0	2^0= 1 byte.
1	2^1 = 2 bytes.
:	:
11	2^{11} = 2048 bytes.
12:15	Reserved.

16.7.10 Data Counter Register

The DataCnt register loads the value from the data length register (see Section 16.7.8 "Data Length Register") when the DPSM moves from the IDLE state to the WAIT_R or WAIT_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the IDLE state and the data status end flag is set. Table 362 shows the bit assignment of the DataCnt register.

Table 362: MCI Data Counter register (DATACNT - address 0x400C 0030) bit description

Bit	Symbol	Description	Reset Value
15:0	DATACOUNT	Remaining data	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

Note: This register should be read only when the data transfer is complete.

16.7.11 Status Register

The Status register is a read-only register. It contains two types of flag:

- Static [10:0]: These remain asserted until they are cleared by writing to the Clear register (see Section 16.7.12 “Clear Register”).
- Dynamic [21:11]: These change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO).

Table 363 shows the bit assignment of the Status register.

Table 363: MCI Status register (STATUS - address 0x400C 0034) bit description

Bit	Symbol	Description	Reset Value
0	CMDCRCFAIL	Command response received (CRC check failed).	0
1	DATACRCFAIL	Data block sent/received (CRC check failed).	0
2	CMDTIMEOUT	Command response timeout.	0
3	DATATIMEOUT	Data timeout.	0
4	TXUNDERRUN	Transmit FIFO underrun error.	0
5	RXOVERRUN	Receive FIFO overrun error.	0
6	CMDRESPEND	Command response received (CRC check passed).	0
7	CMDSENT	Command sent (no response required).	0
8	DATAEND	Data end (data counter is zero).	0
9	STARTBITERR	Start bit not detected on all data signals in wide bus mode.	0
10	DATABLOCKEND	Data block sent/received (CRC check passed).	0
11	CMDACTIVE	Command transfer in progress.	0
12	TXACTIVE	Data transmit in progress.	0
13	RXACTIVE	Data receive in progress.	0
14	TXFIFOHALFEMPTY	Transmit FIFO half empty.	0
15	RXFIFOHALFFULL	Receive FIFO half full.	0
16	TXFIFOFULL	Transmit FIFO full.	0
17	RXFIFOFULL	Receive FIFO full.	0
18	TXFIFOEMPTY	Transmit FIFO empty.	0
19	RXFIFOEMPTY	Receive FIFO empty.	0
20	TXDATAAVLBL	Data available in transmit FIFO.	0
21	RXDATAAVLBL	Data available in receive FIFO.	0
31:22	-	Reserved. The value read from a reserved bit is not defined.	NA

16.7.12 Clear Register

The Clear register is a write-only register. The corresponding static status flags can be cleared by writing a 1 to the corresponding bit in the register. Table 364 shows the bit assignment of the Clear register.

Table 364: MCI Clear register (CLEAR - address 0x400C 0038) bit description

Bit	Symbol	Description
0	CMDCRCFAILCLR	Clears CmdCrcFail flag.
1	DATACRCFAILCLR	Clears DataCrcFail flag.
2	CMDTIMEOUTCLR	Clears CmdTimeOut flag.
3	DATATIMEOUTCLR	Clears DataTimeOut flag.
4	TXUNDERRUNCLR	Clears TxUnderrun flag.
5	RXOVERRUNCLR	Clears RxOverrun flag.
6	CMDRESPENDCLR	Clears CmdRespEnd flag.
7	CMDSENTCLR	Clears CmdSent flag.
8	DATAENDCLR	Clears DataEnd flag.
9	STARTBITERRCLR	Clears StartBitErr flag.
10	DATABLOCKENDCLR	Clears DataBlockEnd flag.
31:11	-	Reserved. Read value is undefined, only zero should be written.

16.7.13 Interrupt Mask Registers

The interrupt mask registers determine which status flags generate an interrupt request by setting the corresponding bit to 1. Table 365 shows the bit assignment of the Maskx registers.

Table 365: MCI Interrupt Mask registers (MASK0 - address 0x400C 003C) bit description

Bit	Symbol	Description	Reset Value
0	MASK0	Mask CmdCrcFail flag.	0
1	MASK1	Mask DataCrcFail flag.	0
2	MASK2	Mask CmdTimeOut flag.	0
3	MASK3	Mask DataTimeOut flag.	0
4	MASK4	Mask TxUnderrun flag.	0
5	MASK5	Mask RxOverrun flag.	0
6	MASK6	Mask CmdRespEnd flag.	0
7	MASK7	Mask CmdSent flag.	0
8	MASK8	Mask DataEnd flag.	0
9	MASK9	Mask StartBitErr flag.	0
10	MASK10	Mask DataBlockEnd flag.	0
11	MASK11	Mask CmdActive flag.	0
12	MASK12	Mask TxActive flag.	0
13	MASK13	Mask RxActive flag.	0
14	MASK14	Mask TxFifoHalfEmpty flag.	0
15	MASK15	Mask RxFifoHalfFull flag.	0
16	MASK16	Mask TxFifoFull flag.	0
17	MASK17	Mask RxFifoFull flag.	0
18	MASK18	Mask TxFifoEmpty flag.	0
19	MASK19	Mask RxFifoEmpty flag.	0
20	MASK20	Mask TxDataAvlbl flag.	0
21	MASK21	Mask RxDataAvlbl flag.	0
31:22	-	Reserved. Read value is undefined, only zero should be written.	NA

16.7.14 FIFO Counter Register

The FifoCnt register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register (see Section 16.7.8 “Data Length Register”) when the Enable bit is set in the data control register. If the data length is not word aligned (multiple of 4), the remaining 1 to 3 bytes are regarded as a word. Table 366 shows the bit assignment of the FifoCnt register.

Table 366: MCI FIFO Counter register (FIFOCNT - address 0x400C 0048) bit description

Bit	Symbol	Description	Reset Value
14:0	DATACOUNT	Remaining data	0
31:15	-	Reserved. Read value is undefined, only zero should be written.	NA

16.7.15 Data FIFO Register

The receive and transmit FIFOs can be read or written as 32 bit wide registers. The FIFOs contain 16 entries on 16 sequential addresses. This allows the microprocessor to use its load and store multiple operands to read/write to the FIFO. Table 367 shows the bit assignment of the FIFO register.

Table 367: MCI Data FIFO register (FIFO - address 0x400C 0080 to 0x400C 00BC) bit description

Bit	Symbol	Description	Reset Value
31:0	DATA	FIFO data.	0

17.1 Basic configuration

The UART1 peripheral is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bits PCUART1.

   Remark: On reset, UART1 is enabled (PCUART1 = 1).
2. Peripheral clock: UART1 operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Baud rate: In register U1LCR (Table 378), set bit DLAB =1. This enables access to registers DLL (Table 372) and DLM (Table 373) for setting the baud rate. Also, if needed, set the fractional baud rate in the fractional divider register (Table 385).
4. UART FIFO: Use bit FIFO enable (bit 0) in register U1FCR (Table 377) to enable the FIFOs.
5. Pins: Select UART pins and pin modes through the in the relevant IOCON registers (Section 7.4.1).

   Remark: UART receive pins should not have pull-down resistors enabled.
6. Interrupts: To enable UART interrupts set bit DLAB =0 in register U1LCR (Table 378). This enables access to U1IER (Table 374). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
7. DMA: UART1 transmit and receive functions can operated with the GPDMA controller (see Table 696).

17.2 Features

- Full modem control handshaking available
- Data sizes of 5, 6, 7, and 8 bits.
- Parity generation and checking: odd, even mark, space or none.
- One or two stop bits.
- 16 byte Receive and Transmit FIFOs.
- Built-in baud rate generator, including a fractional rate divider for great versatility.
- Supports DMA for both transmit and receive.
- Auto-baud capability
- Break generation and detection.
- Multiprocessor addressing mode.
- RS-485/EIA-485 support.

17.3 Architecture

The architecture of the UART1 is shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART1.

The UART1 receiver block, U1RX, monitors the serial input line, RXD1, for valid input. The UART1 RX Shift Register (U1RSR) accepts valid characters via RXD1. After a valid character is assembled in the U1RSR, it is passed to the UART1 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART1 transmitter block, U1TX, accepts data written by the CPU or host and buffers the data in the UART1 TX Holding Register FIFO (U1THR). The UART1 TX Shift Register (U1TSR) reads the data stored in the U1THR and assembles the data to transmit via the serial output pin, TXD1.

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 TX block. The U1BRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the U1DLL and U1DLM registers. This divided down clock is the 16x oversample clock.

The modem interface contains registers U1MCR and U1MSR. This interface is responsible for handshaking between a modem peripheral and the UART1.

The interrupt interface contains registers U1IER and U1IIR. The interrupt interface receives several one clock wide enables from the U1TX and U1RX blocks.

Status information from the U1TX and U1RX is stored in the U1LSR. Control information for the U1TX and U1RX is stored in the U1LCR.

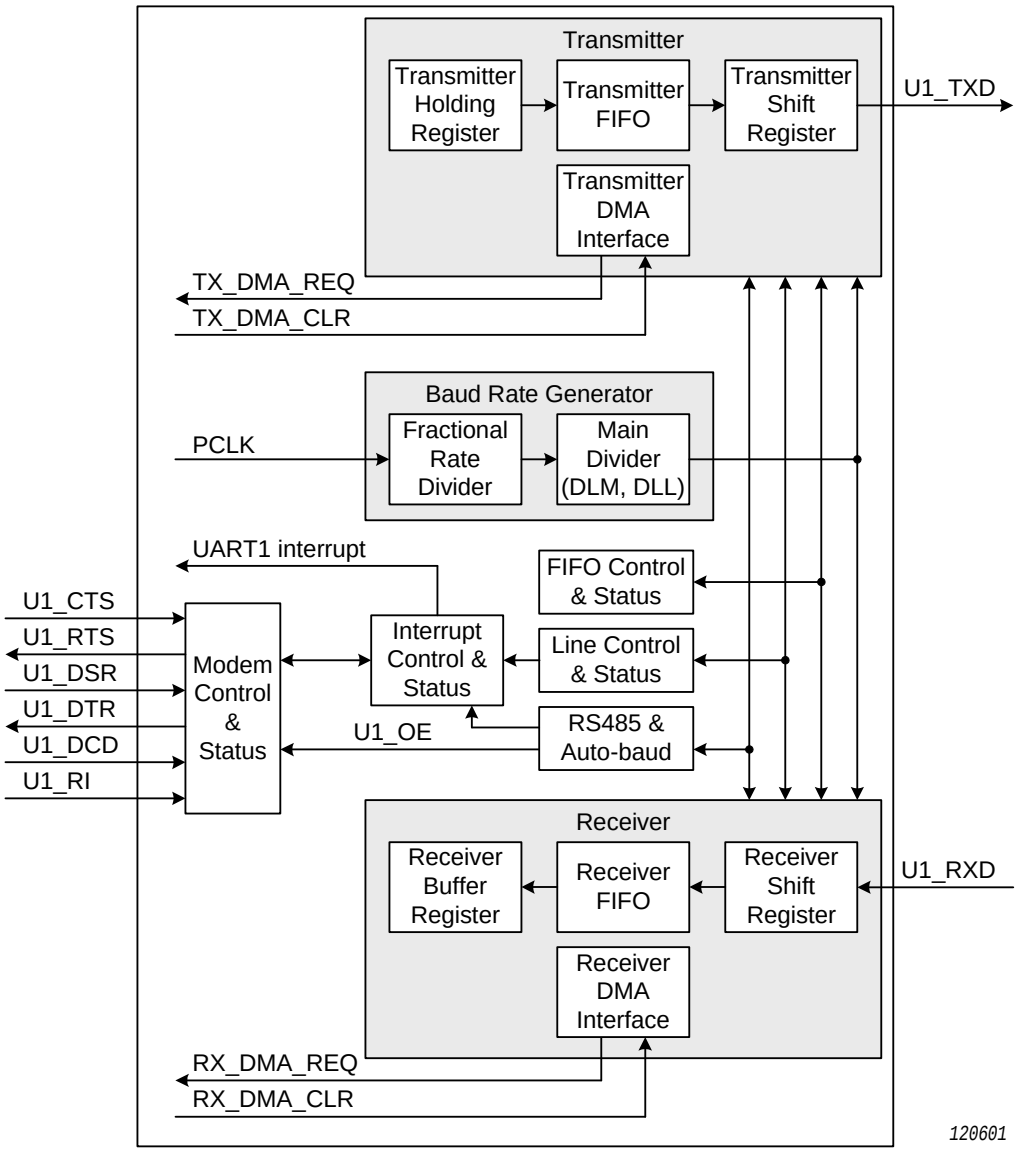


Fig 70. UART1 block diagram

17.4 Pin description

Table 368: UART1 Pin Description

Pin	Type	Description
U1_RXD	Input	**Serial Input.** Serial receive data.
U1_TXD	Output	**Serial Output.** Serial transmit data.
U1_CTS	Input	**Clear To Send.** Active low signal indicates if the external modem is ready to accept transmitted data via TXD1 from the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[4]. State change information is stored in U1MSR[0] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1). Clear to send. CTS1 is an asynchronous, active low modem status signal. Its condition can be checked by reading bit 4 (CTS) of the modem status register. Bit 0 (DCTS) of the Modem Status Register (MSR) indicates that CTS1 has changed states since the last read from the MSR. If the modem status interrupt is enabled when CTS1 changes levels and the auto-cts mode is not enabled, an interrupt is generated. CTS1 is also used in the auto-cts mode to control the transmitter.
U1_DCD	Input	**Data Carrier Detect.** Active low signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged. In normal operation of the modem interface (U1MCR[4]=0), the complement value of this signal is stored in U1MSR[7]. State change information is stored in U1MSR3 and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
U1_DSR	Input	**Data Set Ready.** Active low signal indicates if the external modem is ready to establish a communications link with the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[5]. State change information is stored in U1MSR[1] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
U1_DTR	Output	Data Terminal Ready. Active low signal indicates that the UART1 is ready to establish connection with external modem. The complement value of this signal is stored in U1MCR[0]. The DTR pin can also be used as an RS-485/EIA-485 output enable signal.
U1_RI	Input	**Ring Indicator.** Active low signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[6]. State change information is stored in U1MSR[2] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
U1_RTS	Output	**Request To Send.** Active low signal indicates that the UART1 would like to transmit data to the external modem. The complement value of this signal is stored in U1MCR[1]. In auto-rts mode, RTS1 is used to control the transmitter FIFO threshold logic. Request to send. RTS1 is an active low signal informing the modem or data set that the UART is ready to receive data. RTS1 is set to the active (low) level by setting the RTS modem control register bit and is set to the inactive (high) level either as a result of a system reset or during loop-back mode operations or by clearing bit 1 (RTS) of the MCR. In the auto-rts mode, RTS1 is controlled by the transmitter FIFO threshold logic. The RTS pin can also be used as an RS-485/EIA-485 output enable signal.

17.5 Register description

The Divisor Latch Access Bit (DLAB) is contained in U1LCR[7] and enables access to the Divisor Latches.

Table 369: Register overview: UART1 (base address 0x4001 0000)

Name	Access	Address offset	Description	Reset Value[1]	Table
RBR	RO	0x000	DLAB =0. Receiver Buffer Register. Contains the next received character to be read.	NA	370
THR	WO	0x000	DLAB =0. Transmit Holding Register. The next character to be transmitted is written here.	NA	371
DLL	R/W	0x000	DLAB =1. 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.	0x01	372
DLM	R/W	0x004	DLAB =1. 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.	0	373
IER	R/W	0x004	DLAB =0. Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART1 interrupts.	0	374
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01	375
FCR	WO	0x008	FIFO Control Register. Controls UART1 FIFO usage and modes.	0	377
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0	378
MCR	R/W	0x010	Modem Control Register. Contains controls for flow control handshaking and loopback mode.	0	379
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60	381
MSR	RO	0x018	Modem Status Register. Contains handshake signal status flags.	0	382
SCR	R/W	0x01C	Scratch Pad Register. 8-bit temporary storage for software.	0	383
ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0	384
FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10	385
TER	R/W	0x030	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	0x80	387
RS485CTRL	R/W	0x04C	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	0	388
RSADRMATCH	R/W	0x050	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	0	389
RS485DLY	R/W	0x054	RS-485/EIA-485 direction control delay.	0	390

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

17.5.1 UART1 Receiver Buffer Register

The U1RBR is the top byte of the UART1 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 zeroes.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1RBR. The U1RBR 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 U1LSR register, and then to read a byte from the U1RBR.

Table 370: UART1 Receiver Buffer Register when DLAB = 0 (RBR - address 0x4001 0000) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UART1 Receiver Buffer Register contains the oldest received byte in the UART1 RX FIFO.	undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

17.5.2 UART1 Transmitter Holding Register

The write-only U1THR is the top byte of the UART1 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 U1LCR must be zero in order to access the U1THR. The U1THR is write-only.

Table 371: UART1 Transmitter Holding Register when DLAB = 0 (THR - address 0x4001 0000) bit description

Bit	Symbol	Description
7:0	THR	Writing to the UART1 Transmit Holding Register causes the data to be stored in the UART1 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.
31:8	-	Reserved. Read value is undefined, only zero should be written.

17.5.3 UART1 Divisor Latch LSB and MSB Registers

The UART1 Divisor Latch is part of the UART1 Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16x the desired baud rate. The U1DLL and U1DLM registers together form a 16-bit divisor where U1DLL contains the lower 8 bits of the divisor and U1DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U1LCR must be one in order to access the UART1 Divisor Latches. Details on how to select the right value for U1DLL and U1DLM can be found later in this chapter, see Section 17.5.16.

Table 372: UART1 Divisor Latch LSB Register when DLAB = 1 (DLL - address 0x4001 0000) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UART1 Divisor Latch LSB Register, along with the U1DLM register, determines the baud rate of the UART1.	0x01
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 373: UART1 Divisor Latch MSB Register when DLAB = 1 (DLM - address 0x4001 0004) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UART1 Divisor Latch MSB Register, along with the U1DLL register, determines the baud rate of the UART1.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

17.5.4 UART1 Interrupt Enable Register

The U1IER is used to enable the four UART1 interrupt sources.

Table 374: UART1 Interrupt Enable Register when DLAB = 0 (IER - address 0x4001 0004) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for UART1. 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 UART1. The status of this interrupt can be read from LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXIE		RX Line Interrupt Enable. Enables the UART1 RX line status interrupts. The status of this interrupt can be read from LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
3	MSIE		Modem Status Interrupt Enable. Enables the modem interrupt. The status of this interrupt can be read from MSR[3:0].	0
		0	Disable the modem interrupt.	
		1	Enable the modem interrupt.	
6:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	CTSIE		CTS Interrupt Enable. If auto-cts mode is enabled this bit enables/disables the modem status interrupt generation on a CTS1 signal transition. If auto-cts mode is disabled a CTS1 transition will generate an interrupt if Modem Status Interrupt Enable (IER[3]) is set. In normal operation a CTS1 signal transition will generate a Modem Status Interrupt unless the interrupt has been disabled by clearing the IER[3] bit in the IER register. In auto-cts mode a transition on the CTS1 bit will trigger an interrupt only if both the IER[3] and IER[7] bits are set.	0
		0	Disable the CTS interrupt.	
		1	Enable the CTS interrupt.	
8	ABEOIE		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	
9	ABTOIE		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

17.5.5 UART1 Interrupt Identification Register

The U1IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U1IIR access. If an interrupt occurs during an U1IIR access, the interrupt is recorded for the next U1IIR access.

Table 375: UART1 Interrupt Identification Register (IIR - address 0x4001 0008) bit description

Bit	Symbol	Value	Description	Reset Value
0	INTSTATUS		Interrupt status. Note that IIR[0] is active low. The pending interrupt can be determined by evaluating IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. IER[3:1] identifies an interrupt corresponding to the UART1 RX or TX FIFO. All other combinations of IER[3:1] not listed below are reserved (100,101,111).	0
		0x3	1 - Receive Line Status (RLS).	
		0x2	2a - Receive Data Available (RDA).	
		0x6	2b - Character Time-out Indicator (CTI).	
		0x1	3 - THRE Interrupt.	
		0x0	4 - Modem Interrupt.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFOENABLE		Copies of FCR[0].	0
8	ABEOINT		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOINT		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, the value read from a reserved bit is not defined.	NA

Bit U1IIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. 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 376. Given the status of U1IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U1IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART1 RLS interrupt (U1IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART1RX input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART1 Rx error condition that set the interrupt can be observed via U1LSR[4:1]. The interrupt is cleared upon an U1LSR read.

The UART1 RDA interrupt (U1IIR[3:1] = 010) shares the second level priority with the CTI interrupt (U1IIR[3:1] = 110). The RDA is activated when the UART1 Rx FIFO reaches the trigger level defined in U1FCR7:6 and is reset when the UART1 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 (U1IIR[3:1] = 110) is a second level interrupt and is set when the UART1 Rx FIFO contains at least one character and no UART1 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART1 Rx FIFO activity (read or write of UART1 RSR) will clear the interrupt. This interrupt is intended to flush the UART1 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 376: UART1 Interrupt Handling

U1IIR[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]	U1LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U1FCR0=1)	U1RBR Read[3] or UART1 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	U1RBR Read[3]
0010	Third	THRE	THRE[2]	U1IIR Read[4] (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 17.5.10 “UART1 Line Status Register”

[3] For details see Section 17.5.1 “UART1 Receiver Buffer Register”

[4] For details see Section 17.5.5 “UART1 Interrupt Identification Register” and Section 17.5.2 “UART1 Transmitter Holding Register”

The UART1 THRE interrupt (U1IIR[3:1] = 001) is a third level interrupt and is activated when the UART1 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART1 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 U1THR at one time

since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U1THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART1 THR FIFO has held two or more characters at one time and currently, the U1THR is empty. The THRE interrupt is reset when a U1THR write occurs or a read of the U1IIR occurs and the THRE is the highest interrupt (U1IIR[3:1] = 001).

It is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a low to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining U1MSR[3:0]. A U1MSR read will clear the modem interrupt.

17.5.6 UART1 FIFO Control Register

The write-only U1FCR controls the operation of the UART1 RX and TX FIFOs.

Table 377: UART1 FIFO Control Register (FCR - address 0x4001 0008) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFOEN		FIFO enable.	0
		0	Must not be used in the application.	
		1	Active high enable for both UART1 RX and TX FIFOs and FCR[7:1] access. This bit must be set for proper UART1 operation. Any transition on this bit will automatically clear the UART1 FIFOs.	
1	RXFIFORES		RX FIFO Reset.	0
		0	No impact on either of UART1 FIFOs.	
		1	Writing a logic 1 to FCR[1] will clear all bytes in UART1 RX FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset.	0
		0	No impact on either of UART1 FIFOs.	
		1	Writing a logic 1 to FCR[2] will clear all bytes in UART1 TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMAMODE		DMA Mode Select. When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 17.5.6.1.	0
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RXTRIGLVL		RX Trigger Level. These two bits determine how many receiver UART1 FIFO characters must be written before an interrupt 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, user software should not write ones to reserved bits.	NA

17.5.6.1 DMA Operation

The user can optionally operate the UART 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.

UART 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.

UART 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.

17.5.7 UART1 Line Control Register

The U1LCR determines the format of the data character that is to be transmitted or received.

Table 378: UART1 Line Control Register (LCR - address 0x4001 000C) 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 LCR[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	BC		Break Control.	0
		0	Disable break transmission.	
		1	Enable break transmission. Output pin UART1 TXD is forced to logic 0 when LCR[6] is active high.	
7	DLAB		Divisor Latch Access Bit (DLAB)	0
		0	Disable access to Divisor Latches.	
		1	Enable access to Divisor Latches.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

17.5.8 UART1 Modem Control Register

The U1MCR enables the modem loopback mode and controls the modem output signals.

Table 379: UART1 Modem Control Register (MCR - address 0x4001 0010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTRCTRL	-	DTR Control. Source for modem output pin, DTR. This bit reads as 0 when modem loopback mode is active.	0
1	RTSCTRL	-	RTS Control. Source for modem output pin RTS. This bit reads as 0 when modem loopback mode is active.	0
3:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
4	LMS		Loopback Mode Select. The modem loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RXD1, has no effect on loopback and output pin, TXD1 is held in marking state. The 4 modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the 4 modem outputs are connected to the 4 modem inputs. As a result of these connections, the upper 4 bits of the MSR will be driven by the lower 4 bits of the MCR rather than the 4 modem inputs in normal mode. This permits modem status interrupts to be generated in loopback mode by writing the lower 4 bits of MCR.	0
		0	Disable modem loopback mode.	
		1	Enable modem loopback mode.	
5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
6	RTSEN		RTS enable.	0
		0	Disable auto-rts flow control.	
		1	Enable auto-rts flow control.	
7	CTSEN		CTS enable.	0
		0	Disable auto-cts flow control.	
		1	Enable auto-cts flow control.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

17.5.9 Auto-flow control

If auto-RTS mode is enabled the UART1‘s receiver FIFO hardware controls the RTS1 output of the UART1. If the auto-CTS mode is enabled the UART1‘s U1TSR hardware will only start transmitting if the CTS1 input signal is asserted.

17.5.9.1 Auto-RTS

The auto-RTS function is enabled by setting the RTSen bit. Auto-RTS data flow control originates in the U1RBR module and is linked to the programmed receiver FIFO trigger level. If auto-RTS is enabled, the data-flow is controlled as follows:

When the receiver FIFO level reaches the programmed trigger level, RTS1 is de-asserted (to a high value). It is possible that the sending UART sends an additional byte after the trigger level is reached (assuming the sending UART has another byte to send) because it might not recognize the de-assertion of RTS1 until after it has begun sending the additional byte. RTS1 is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The re-assertion of RTS1 signals to the sending UART to continue transmitting data.

If Auto-RTS mode is disabled, the RTSen bit controls the RTS1 output of the UART1. If Auto-RTS mode is enabled, hardware controls the RTS1 output, and the actual value of RTS1 will be copied in the RTS Control bit of the UART1. As long as Auto-RTS is enabled, the value of the RTS Control bit is read-only for software.

Example: Suppose the UART1 operating in '550 mode has trigger level in U1FCR set to 0x2 then if Auto-RTS is enabled the UART1 will de-assert the RTS1 output as soon as the receive FIFO contains 8 bytes (Table 377 on page 472). The RTS1 output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

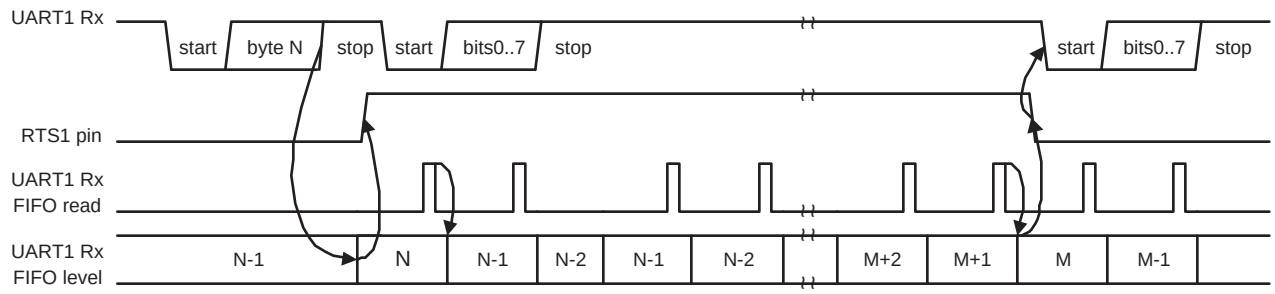


Fig 71. Auto-RTS Functional Timing

17.5.9.2 Auto-CTS

The Auto-CTS function is enabled by setting the CTSen bit. If Auto-CTS is enabled the transmitter circuitry in the U1TSR module checks CTS1 input before sending the next data byte. When CTS1 is active (low), the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS1 must be released before the middle of the last stop bit that is currently being sent. In Auto-CTS mode a change of the CTS1 signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, Delta CTS bit in the U1MSR will be set though. Table 380 lists the conditions for generating a Modem Status interrupt.

Table 380: Modem status interrupt generation

Enable Modem Status Interrupt (U1ER[3])	CTSen (U1MCR[7])	CTS Interrupt Enable (U1IER[7])	Delta CTS (U1MSR[0])	Delta DCD or Trailing Edge RI or Delta DSR (U1MSR[3] or U1MSR[2] or U1MSR[1])	Modem Status Interrupt
0	x	x	x	x	No
1	0	x	0	0	No
1	0	x	1	x	Yes
1	0	x	x	1	Yes
1	1	0	x	0	No
1	1	0	x	1	Yes

Table 380: Modem status interrupt generation

Enable Modem Status Interrupt (U1ER[3])	CTSen (U1MCR[7])	CTS Interrupt Enable (U1IER[7])	Delta CTS (U1MSR[0])	Delta DCD or Trailing Edge RI or Delta DSR (U1MSR[3] or U1MSR[2] or U1MSR[1])	Modem Status Interrupt
1	1	1	0	0	No
1	1	1	1	x	Yes
1	1	1	x	1	Yes

The auto-CTS function reduces interrupts to the host system. When flow control is enabled, a CTS1 state change does not trigger host interrupts because the device automatically controls its own transmitter. Without Auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. Figure 72 illustrates the Auto-CTS functional timing.

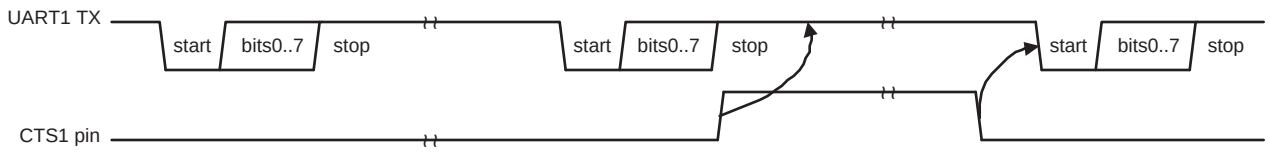


Fig 72. Auto-CTS Functional Timing

While starting transmission of the initial character the CTS1 signal is asserted. Transmission will stall as soon as the pending transmission has completed. The UART will continue transmitting a 1 bit as long as CTS1 is de-asserted (high). As soon as CTS1 gets de-asserted transmission resumes and a start bit is sent followed by the data bits of the next character.

17.5.10 UART1 Line Status Register

The U1LSR is a read-only register that provides status information on the UART1 TX and RX blocks.

Table 381: UART1 Line Status Register (LSR - address 0x4001 0014) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready. LSR[0] is set when the RBR holds an unread character and is cleared when the UART1 RBR FIFO is empty.	0
		0	The UART1 receiver FIFO is empty.	
		1	The UART1 receiver FIFO is not empty.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. An LSR read clears LSR[1]. LSR[1] is set when UART1 RSR has a new character assembled and the UART1 RBR FIFO is full. In this case, the UART1 RBR FIFO will not be overwritten and the character in the UART1 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 LSR read clears LSR[2]. Time of parity error detection is dependent on FCR[0]. **Note:** A parity error is associated with the character at the top of the UART1 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 LSR read clears LSR[3]. The time of the framing error detection is dependent on FCR0. 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 UART1 RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	BI		Break Interrupt. When RXD1 is held in the spacing state (all zeroes) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all ones). An LSR read clears this status bit. The time of break detection is dependent on FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty UART1 THR and is cleared on a THR write.	1
		0	THR contains valid data.	
		1	THR is empty.	

Table 381: UART1 Line Status Register (LSR - address 0x4001 0014) bit description

Bit	Symbol	Value	Description	Reset Value
6	TEMT		Transmitter Empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the TSR or the THR contain valid data.	1
		0	THR and/or the TSR contains valid data.	
		1	THR and the TSR are empty.	
7	RXFE		Error in RX FIFO. LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the RBR. This bit is cleared when the LSR register is read and there are no subsequent errors in the UART1 FIFO.	0
		0	RBR contains no UART1 RX errors or FCR[0]=0.	
		1	UART1 RBR contains at least one UART1 RX error.	
31:8	-		Reserved, the value read from a reserved bit is not defined.	NA

17.5.11 UART1 Modem Status Register

The U1MSR is a read-only register that provides status information on the modem input signals. U1MSR[3:0] is cleared on U1MSR read. Note that modem signals have no direct effect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 382: UART1 Modem Status Register (MSR - address 0x4001 0018) bit description

Bit	Symbol	Value	Description	Reset Value
0	DCTS		Delta CTS. Set upon state change of input CTS. Cleared on an MSR read.	0
		0	No change detected on modem input, CTS.	
		1	State change detected on modem input, CTS.	
1	DDSR		Delta DSR. Set upon state change of input DSR. Cleared on an MSR read.	0
		0	No change detected on modem input, DSR.	
		1	State change detected on modem input, DSR.	
2	TERI		Trailing Edge RI. Set upon low to high transition of input RI. Cleared on an MSR read.	0
		0	No change detected on modem input, RI.	
		1	Low-to-high transition detected on RI.	
3	DDCD		Delta DCD. Set upon state change of input DCD. Cleared on an MSR read.	0
		0	No change detected on modem input, DCD.	
		1	State change detected on modem input, DCD.	
4	CTS	-	Clear To Send State. Complement of input signal CTS. This bit is connected to MCR[1] in modem loopback mode.	0
5	DSR	-	Data Set Ready State. Complement of input signal DSR. This bit is connected to MCR[0] in modem loopback mode.	0

Table 382: UART1 Modem Status Register (MSR - address 0x4001 0018) bit description

Bit	Symbol	Value	Description	Reset Value
6	RI	-	Ring Indicator State. Complement of input RI. This bit is connected to MCR[2] in modem loopback mode.	0
7	DCD	-	Data Carrier Detect State. Complement of input DCD. This bit is connected to MCR[3] in modem loopback mode.	0
31:8	-	-	Reserved, the value read from a reserved bit is not defined.	NA

17.5.12 UART1 Scratch Pad Register

The U1SCR has no effect on the UART1 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 U1SCR has occurred.

Table 383: UART1 Scratch Pad Register (SCR - address 0x4001 0014) bit description

Bit	Symbol	Description	Reset Value
7:0	Pad	A readable, writable byte.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

17.5.13 UART1 Auto-baud Control Register

The UART1 Auto-baud Control Register (U1ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 384: Auto-baud Control Register (ACR - address 0x4001 0020) bit description

Bit	Symbol	Value	Description	Reset value
0	START		Auto-baud start bit. This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	MODE		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AUTORESTART		Auto-baud restart bit.	0
		0	No restart	
		1	Restart in case of time-out (counter restarts at next UART1 Rx falling edge)	
7:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOINTCLR		End of auto-baud interrupt clear bit (write-only).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the IIR.	
9	ABTOINTCLR		Auto-baud time-out interrupt clear bit (write-only).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the IIR.	
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

17.5.14 Auto-baud

The UART1 auto-baud function can be used to measure the incoming baud rate based on the “AT” protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers U1DLM and U1DLL accordingly.

Remark: the fractional rate divider is not connected during auto-baud operations, and therefore should not be used when the auto-baud feature is needed.

Auto-baud is started by setting the U1ACR Start bit. Auto-baud can be stopped by clearing the U1ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the U1ACR Mode bit. In mode 0 the baud rate is measured on two subsequent falling edges of the UART1 RX pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud rate is measured between the falling edge and the subsequent rising edge of the UART1 RX pin (the length of the start bit).

The U1ACR AutoRestart bit can be used to automatically restart baud rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART1 RX pin.

The auto-baud function can generate two interrupts.

- The U1IIR ABTOInt interrupt will get set if the interrupt is enabled (U1IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U1IIR ABEOInt interrupt will get set if the interrupt is enabled (U1IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U1ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART1 RX pin baud rate, but the value of the U1FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U1DLM and U1DLL registers should be done before U1ACR register write. The minimum and the maximum baud rates supported by UART1 are function of pclk, number of data bits, stop bits and parity bits.

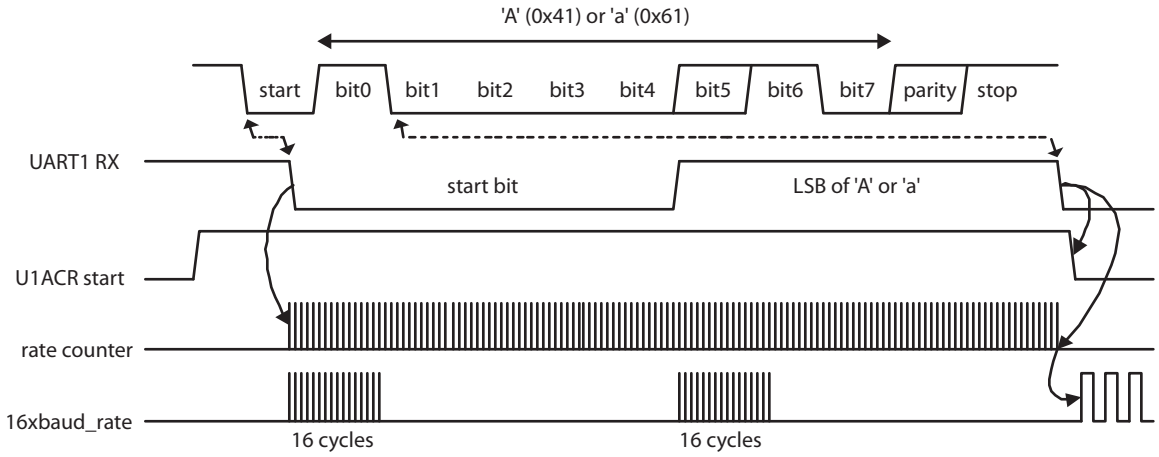
(1)

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART1_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

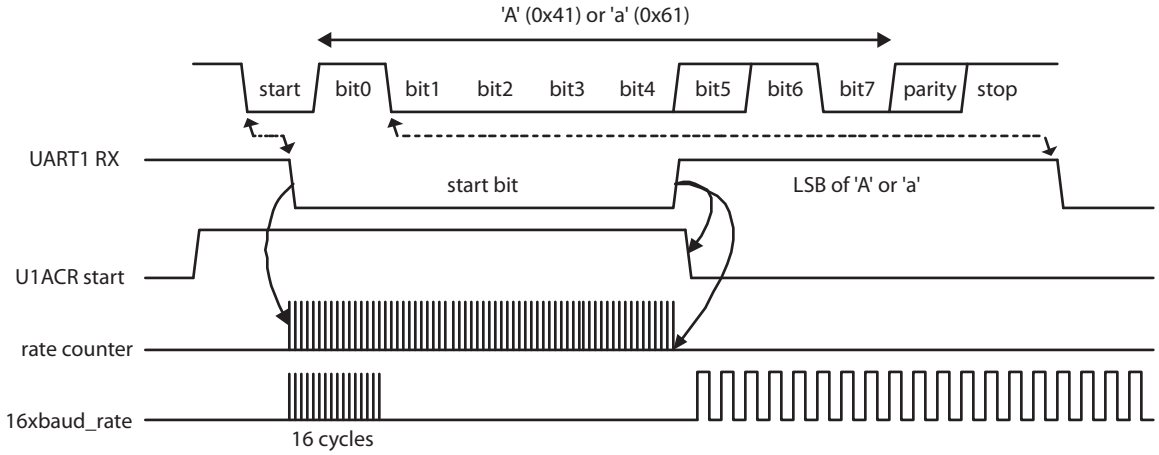
17.5.15 Auto-baud modes

When the software is expecting an “AT” command, it configures the UART1 with the expected character format and sets the U1ACR Start bit. The initial values in the divisor latches U1DLM and U1DLM don‘t care. Because of the “A” or “a” ASCII coding (”A" = 0x41, “a” = 0x61), the UART1 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U1ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U1ACR Start bit setting, the baud rate measurement counter is reset and the UART1 U1RSR is reset. The U1RSR baud rate is switch to the highest rate.
2. A falling edge on UART1 RX pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud rate pre-scaled) UART1 input clock, guaranteeing the start bit is stored in the U1RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART1 input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART1 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART1 Rx pin.
6. The rate counter is loaded into U1DLM/U1DLL and the baud rate will be switched to normal operation. After setting the U1DLM/U1DLL the end of auto-baud interrupt U1IIR ABEOInt will be set, if enabled. The U1RSR will now continue receiving the remaining bits of the “A/a” character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 73. Auto-baud a) mode 0 and b) mode 1 waveform

17.5.16 UART1 Fractional Divider Register

The UART1 Fractional Divider Register (U1FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be greater than 2.

Table 385: UART1 Fractional Divider Register (FDR - address 0x4001 0028) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud rate generation pre-scaler divisor value. If this field is 0, fractional baud rate generator will not impact the UART1 baud rate.	0
7:4	MULVAL	1	Baud rate pre-scaler multiplier value. This field must be greater or equal 1 for UART1 to operate properly, regardless of whether the fractional baud rate generator is used or not.	1
31:8	-		Reserved. Read value is undefined, only zero should be written.	0

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of UART1 disabled making sure that UART1 is fully software and hardware compatible with UARTs not equipped with this feature.

UART1 baud rate can be calculated as (n = 1):

(2)

$$UART1_{baudrate} = \frac{PCLK}{16 \times (256 \times U1DLM + U1DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where PCLK is the peripheral clock, U1DLM and U1DLL are the standard UART1 baud rate divider registers, and DIVADDVAL and MULVAL are UART1 fractional baud rate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 14$
3. $DIVADDVAL < MULVAL$

The value of the U1FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the U1FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

17.5.16.1 Baud rate calculation

UART1 can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baud rate with a relative error of less than 1.1% from the desired one.

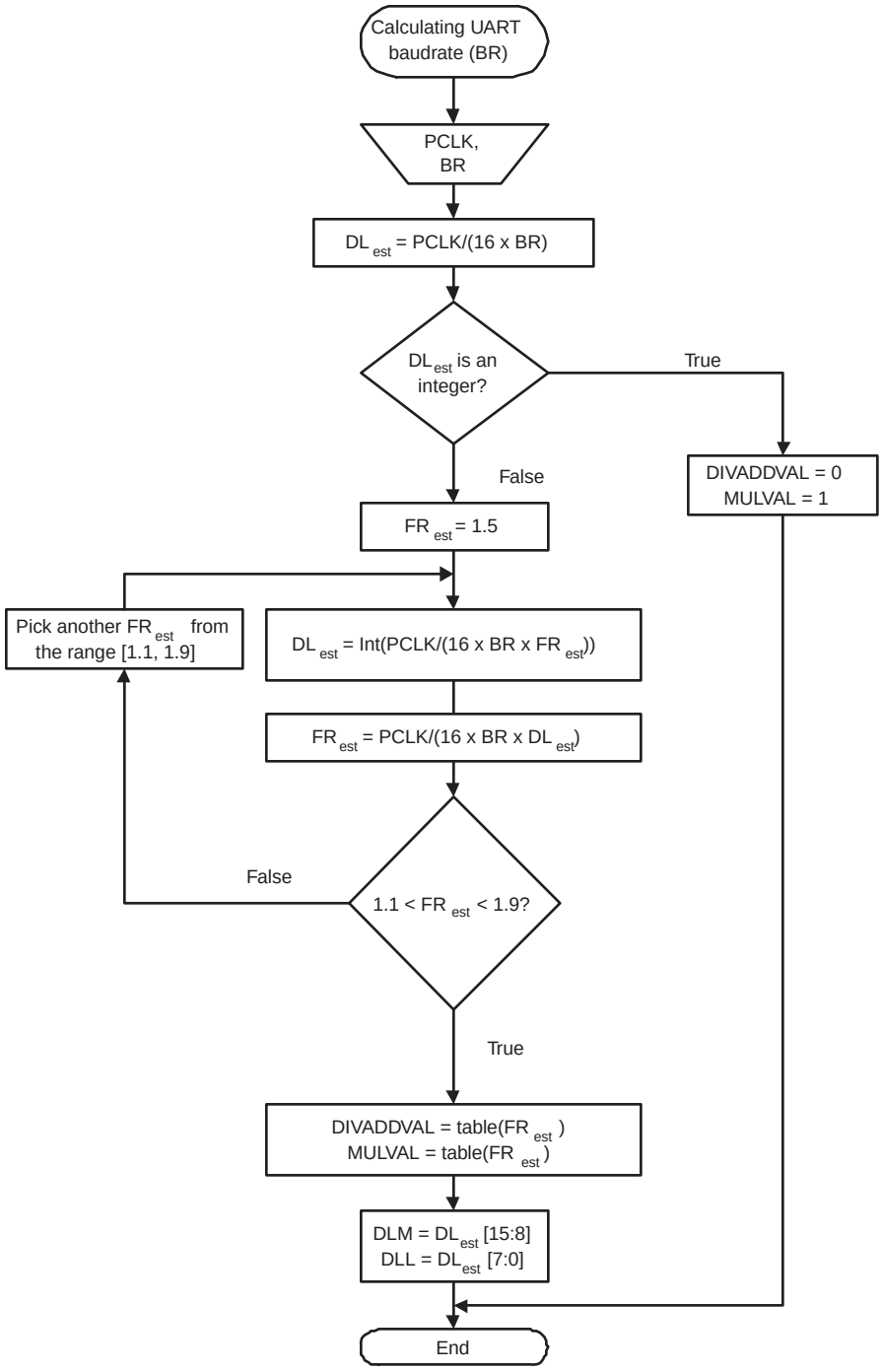


Fig 74. Algorithm for setting UART dividers

Table 386. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

17.5.16.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

17.5.16.1.2 Example 2: PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 386 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 2 the UART rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

17.5.17 UART1 Transmit Enable Register

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), U1TER enables implementation of software flow control, too. When TxEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TxEn becomes 0, UART1 transmission will stop.

Table 387 describes how to use the TxEn bit in order to achieve hardware flow control. However, it is strongly suggested to let UART1 hardware implemented auto flow control features take for this purpose, and limit the scope of TxEn to software flow control.

U1TER enables implementation of software and hardware flow control. When TXEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART1 transmission will stop.

Table 387: UART1 Transmit Enable Register (TER - address 0x4001 0030) bit description

Bit	Symbol	Description	Reset Value
6:0	-	Reserved. Read value is undefined, only zero should be written.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software can clear this bit when it detects that the a hardware-handshaking TX-permit signal (CTS) has gone false, or with software handshaking, when it receives an XOFF character (DC3). Software can set this bit again when it detects that the TX-permit signal has gone true, or when it receives an XON (DC1) character.	1
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

17.5.18 UART1 RS485 Control register

The U1RS485CTRL register controls the configuration of the UART in RS-485/EIA-485 mode.

Table 388: UART1 RS485 Control register (RS485CTRL - address 0x4001 004C) bit description

Bit	Symbol	Value	Description	Reset value
0	NMMEN		RS-485/EIA-485 Normal Multidrop Mode (NMM) mode select.	0
		0	Disabled.	
		1	Enabled. In this mode, an address is detected when a received byte causes the UART to set the parity error and generate an interrupt.	
1	RXDIS		Receive enable.	0
		0	Enabled.	
		1	Disabled.	
2	AADEN		Auto Address Detect (AAD) enable.	0
		0	Disabled.	
		1	Enabled.	
3	SEL		Direction control.	0
		0	RTS. If direction control is enabled (bit DCTRL = 1), pin $\overline{\text{RTS}}$ is used for direction control.	
		1	DTR. If direction control is enabled (bit DCTRL = 1), pin DTR is used for direction control.	
4	DCTRL		Direction control enable.	0
		0	Disable Auto Direction Control.	
		1	Enable Auto Direction Control.	
5	OINV		Polarity. This bit reverses the polarity of the direction control signal on the RTS (or DTR) pin.	0
		0	LOW. The direction control pin will be driven to logic ‘0’ when the transmitter has data to be sent. It will be driven to logic ‘1’ after the last bit of data has been transmitted.	
		1	HIGH. The direction control pin will be driven to logic ‘1’ when the transmitter has data to be sent. It will be driven to logic ‘0’ after the last bit of data has been transmitted.	
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

17.5.19 UART1 RS-485 Address Match register

The U1RS485ADRMATCH register contains the address match value for RS-485/EIA-485 mode.

Table 389. UART1 RS-485 Address Match register (RS485ADRMATCH - address 0x4001 0050) bit description

Bit	Symbol	Description	Reset value
7:0	ADRMATCH	Contains the address match value.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

17.5.20 UART1 RS-485 Delay value register

The user may program the 8-bit RS485DLY register with a delay between the last stop bit leaving the TXFIFO and the de-assertion of RTS (or DTR). This delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

Table 390. UART1 RS-485 Delay value register (RS485DLY - address 0x4001 0054) bit description

Bit	Symbol	Description	Reset value
7:0	DLY	Contains the direction control (RTS or DTR) delay value. This register works in conjunction with an 8-bit counter.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

17.5.21 RS-485/EIA-485 modes of operation

The RS-485/EIA-485 feature allows the UART to be configured as an addressable slave. The addressable slave is one of multiple slaves controlled by a single master.

The UART master transmitter will identify an address character by setting the parity bit to ‘1’. For data characters, the parity bit is set to ‘0’.

Each UART slave receiver can be assigned a unique address. The slave can be programmed to either manually or automatically reject data following an address which is not theirs.

RS-485/EIA-485 Normal Multidrop Mode (NMM)

Setting the RS485CTRL bit 0 enables this mode. In this mode, the parity bit is used for the alternative purpose of making a distinction between address and data in received data.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received data bytes will be ignored and will not be stored in the RXFIFO. When an address byte is detected (parity bit = ‘1’) it will be placed into the RXFIFO and an Rx Data Ready Interrupt will be generated. The processor can then read the address byte and decide whether or not to enable the receiver to accept the following data.

While the receiver is ENABLED (RS485CTRL bit 1 =’0’) all received bytes will be accepted and stored in the RXFIFO regardless of whether they are data or address.

RS-485/EIA-485 Auto Address Detection (AAD) mode

When both RS485CTRL register bits 0 (9-bit mode enable) and 2 (AAD mode enable) are set, the UART is in auto address detect mode.

In this mode, the receiver will compare any address byte received (parity = ‘1’) to the 8-bit value programmed into the RS485ADRMATCH register.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received byte will be discarded if it is either a data byte OR an address byte which fails to match the RS485ADRMATCH value.

When a matching address character is detected it will be pushed onto the RXFIFO along with the parity bit, and the receiver will be automatically enabled (RS485CTRL bit 1 will be cleared by hardware). The receiver will also generate n Rx Data Ready Interrupt.

While the receiver is ENABLED (RS485CTRL bit 1 = ‘0’) all bytes received will be accepted and stored in the RXFIFO until an address byte which does not match the RS485ADRMATCH value is received. When this occurs, the receiver will be automatically disabled in hardware (RS485CTRL bit 1 will be set), The received non-matching address character will not be stored in the RXFIFO.

RS-485/EIA-485 Auto Direction Control

RS485/EIA-485 Mode includes the option of allowing the transmitter to automatically control the state of either the $\overline{\text{RTS}}$ pin or the $\overline{\text{DTR}}$ pin as a direction control output signal.

Setting RS485CTRL bit 4 = ‘1’ enables this feature.

Direction control, if enabled, will use the $\overline{\text{RTS}}$ pin when RS485CTRL bit 3 = ‘0’. It will use the $\overline{\text{DTR}}$ pin when RS485CTRL bit 3 = ‘1’.

When Auto Direction Control is enabled, the selected pin will be asserted (driven low) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven high) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

The RS485CTRL bit 4 takes precedence over all other mechanisms controlling $\overline{\text{RTS}}$ (or $\overline{\text{DTR}}$) with the exception of loopback mode.

RS485/EIA-485 driver delay time

The driver delay time is the delay between the last stop bit leaving the TXFIFO and the de-assertion of RTS (or DTR). This delay time can be programmed in the 8-bit RS485DLY register. The delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

RS485/EIA-485 output inversion

The polarity of the direction control signal on the $\overline{\text{RTS}}$ (or $\overline{\text{DTR}}$) pins can be reversed by programming bit 5 in the U1RS485CTRL register. When this bit is set, the direction control pin will be driven to logic 1 when the transmitter has data waiting to be sent. The direction control pin will be driven to logic 0 after the last bit of data has been transmitted.

18.1 How to read this chapter

Most LPC408x/407x family devices include 5 UARTs. A few devices implement only 4 UARTs. Refer to a specific device data sheet for details. UARTs 0, 2, and 3 are essentially the same as UART1, but without modem/flow control signals. UART4, described in the next chapter, adds a synchronous mode and a smart card mode.

18.2 Basic configuration

The UART0/2/3 peripherals are configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bits PCUART0/2/3.

   Remark: On reset, UART0 is enabled (PCUART0 = 1), and UART2/3 are disabled (PCUART2/3 = 0).
2. Peripheral clock: These UARTs operate from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Baud rate: In register U0/2/3LCR (Table 401), set bit DLAB =1. This enables access to registers DLL (Table 395) and DLM (Table 396) for setting the baud rate. Also, if needed, set the fractional baud rate in the fractional divider register (Table 405).
4. UART FIFO: Use bit FIFO enable (bit 0) in register U0/2/3FCR (Table 400) to enable the FIFOs.
5. Pins: Select UART pins and pin modes through the relevant IOCON registers (Section 7.4.1).

   Remark: UART receive pins should not have pull-down resistors enabled.
6. Interrupts: To enable UART interrupts set bit DLAB =0 in register U0/2/3LCR (Table 401). This enables access to U0/2/3IER (Table 397). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
7. DMA: UART0/2/3 transmit and receive functions can operate with the GPDMA controller (see Table 696).

18.3 Features

- Data sizes of 5, 6, 7, and 8 bits.
- Parity generation and checking: odd, even mark, space or none.
- One or two stop bits.
- 16 byte Receive and Transmit FIFOs.
- Built-in baud rate generator, including a fractional rate divider for great versatility.
- Supports DMA for both transmit and receive.
- Auto-baud capability
- Break generation and detection.
- Multiprocessor addressing mode.
- Support for software flow control.
- RS-485/EIA-485 support.

18.4 Architecture

The architecture of the UARTs 0, 2, and 3 are shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART.

The UARTn receiver block, UnRX, monitors the serial input line, RXDn, for valid input. The UARTn RX Shift Register (UnRSR) accepts valid characters via RXDn. After a valid character is assembled in the UnRSR, it is passed to the UARTn RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UARTn transmitter block, UnTX, accepts data written by the CPU or host and buffers the data in the UARTn TX Holding Register FIFO (UnTHR). The UARTn TX Shift Register (UnTSR) reads the data stored in the UnTHR and assembles the data to transmit via the serial output pin, TXDn.

The UARTn Baud Rate Generator block, UnBRG, generates the timing enables used by the UARTn TX block. The UnBRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the UnDLL and UnDLM registers. This divided down clock is the 16x oversample clock.

The interrupt interface contains registers UnIER and UnIIR. The interrupt interface receives several one clock wide enables from the UnTX and UnRX blocks.

Status information from the UnTX and UnRX is stored in the UnLSR. Control information for the UnTX and UnRX is stored in the UnLCR.

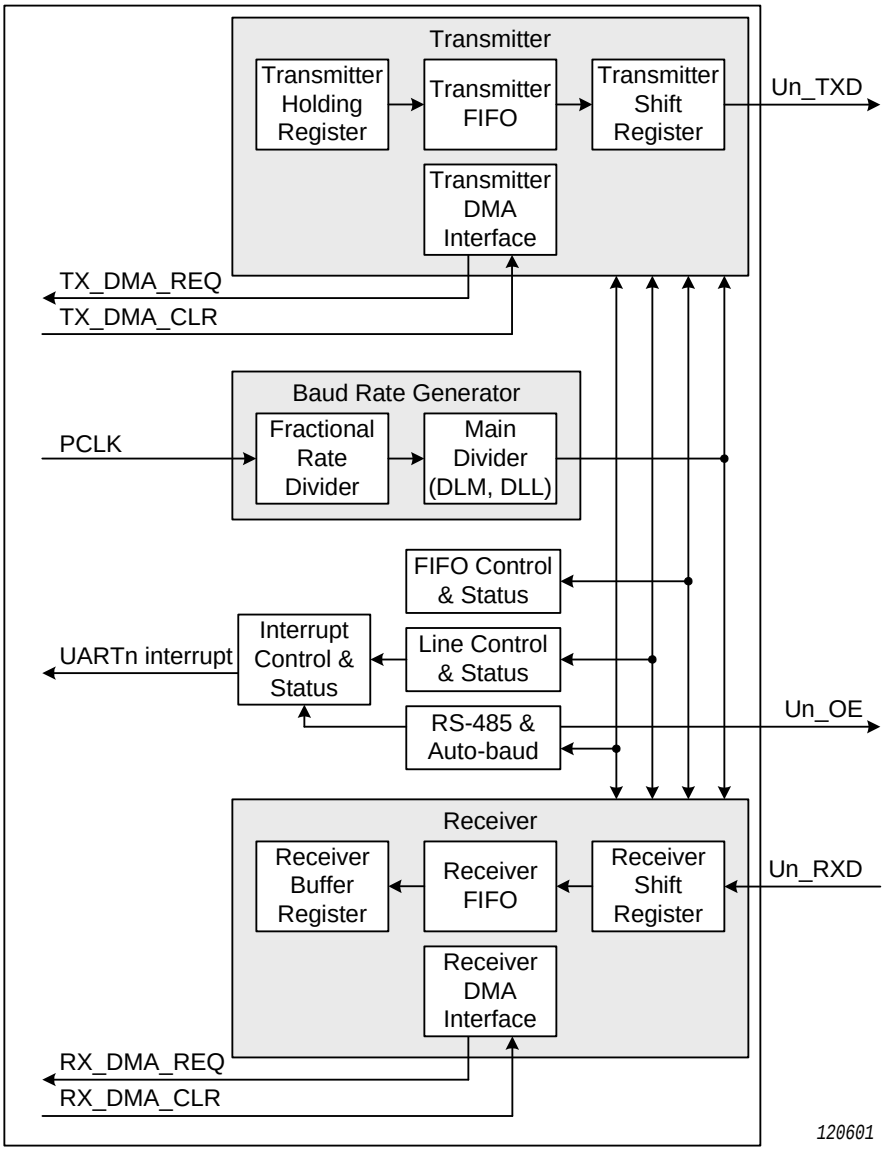


Fig 75. UART0, 2, and 3 block diagram

18.5 Pin description

Table 391: UARTn Pin description

Pin	Type	Description
U0_RXD, U2_RXD, U3_RXD	Input	**Serial Input.** Serial receive data.
U0_TXD, U2_TXD, U3_TXD	Output	**Serial Output.** Serial transmit data.
U0_OE, U2_OE, U3_OE	Output	**Output Enable.** RS-485/EIA-485 output enable.

18.6 Register description

Each UART contains registers as shown in Table 392. The Divisor Latch Access Bit (DLAB) is contained in UnLCR7 and enables access to the Divisor Latches.

Table 392. Register overview: UART0/2/3 (base address: 0x4000 C000, 0x4008 8000, 0x4009 C000)

Name	Access	Address offset	Description	Reset value[1]	Table
RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read.	NA	393
THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here (DLAB =0).	NA	394
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	395
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	396
IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts (DLAB =0).	0	397
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01	398
FCR	WO	0x008	FIFO Control Register. Controls UART FIFO usage and modes.	0	400
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0	401
-	-	0x010	Reserved.	-	-
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60	402
-	-	0x018	Reserved.	-	-
SCR	R/W	0x01C	Scratch Pad Register. 8-bit temporary storage for software.	0	403
ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0	404
FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10	405
-	-	0x02C	Reserved.	-	-
TER	R/W	0x030	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	0x80	407
-	-	0x034 to 0x048	Reserved.	-	-
RS485CTRL	R/W	0x04C	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	0	408
RS485 ADRMATCH	R/W	0x050	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	0	409
RS485DLY	R/W	0x054	RS-485/EIA-485 direction control delay.	0	410

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

18.6.1 UARTn Receiver Buffer Register

The UnRBR is the top byte of the UARTn 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 zeroes.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the UnRBR. The UnRBR 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 U0LSR register, and then to read a byte from the UnRBR.

Table 393: UARTn Receiver Buffer Register when DLAB = 0, read only (RBR - address 0x4000 C000 (UART0), 0x4009 8000 (UART2), 04009 C000 (UART3)) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UARTn Receiver Buffer Register contains the oldest received byte in the UARTn Rx FIFO.	Undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

18.6.2 UARTn Transmit Holding Register

The UnTHR is the top byte of the UARTn 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 UnLCR must be zero in order to access the UnTHR. The UnTHR is always write-only.

Table 394: UARTn Transmit Holding Register when DLAB = 0, write only (THR - address 0x4000 C000 (UART0), 0x4009 8000 (UART2), 0x4009 C000 (UART3)) bit description

Bit	Symbol	Description
7:0	THR	Writing to the UARTn Transmit Holding Register causes the data to be stored in the UARTn transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.
31:8	-	Reserved. Read value is undefined, only zero should be written.

18.6.3 UARTn Divisor Latch LSB register

The UARTn Divisor Latch is part of the UARTn Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16× the desired baud rate. The UnDLL and UnDLM registers together form a 16-bit divisor where UnDLL contains the lower 8 bits of the divisor and UnDLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed. The Divisor Latch Access Bit (DLAB) in UnLCR must be one in order to access the UARTn Divisor Latches. Details on how to select the right value for UnDLL and UnDLM can be found later in this chapter, see Section 18.6.11.

Table 395: UARTn Divisor Latch LSB register when DLAB = 1 (DLL - address 0x4000 C000 (UART0), 0x4009 8000 (UART2), 0x4009 C000 (UART3)) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.	0x01
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 396: UARTn Divisor Latch MSB register when DLAB = 1 (DLM - address 0x4000 C004 (UART0), 0x4009 8004 (UART2), 0x4009 C004 (UART3)) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UARTn Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UARTn.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

18.6.4 UARTn Interrupt Enable Register

The UnIER is used to enable the three UARTn interrupt sources.

Table 397: UARTn Interrupt Enable Register when DLAB = 0 (IER - address 0x4000 C004 (UART0), 0x4009 8004 (UART2), 0x4009 C004 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for UARTn. 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 UARTn. The status of this can be read from UnLSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXIE		RX Line Status Interrupt Enable. Enables the UARTn RX line status interrupts. The status of this interrupt can be read from UnLSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
7:3	-		Reserved. Read value is undefined, only zero should be written.	NA
8	ABEOINTEN		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	
9	ABTOINTEN		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

18.6.5 UARTn Interrupt Identification Register

The UnIIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an UnIIR access. If an interrupt occurs during an UnIIR access, the interrupt is recorded for the next UnIIR access.

Table 398: UARTn Interrupt Identification Register, read only (IIR - address 0x4000 C008 (UART0), 0x4009 8008 (UART2), 0x4009 C008 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	INTSTATUS		Interrupt status. Note that UnIIR[0] is active low. The pending interrupt can be determined by evaluating UnIIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. UnIER[3:1] identifies an interrupt corresponding to the UARTn Rx or TX FIFO. All other combinations of UnIER[3:1] not listed below are reserved (000,100,101,111).	0
		0x3	1 - Receive Line Status (RLS).	
		0x2	2a - Receive Data Available (RDA).	
		0x6	2b - Character Time-out Indicator (CTI).	
		0x1	3 - THRE Interrupt	
5:4	-		Reserved. Read value is undefined, only zero should be written.	NA
7:6	FIFOENABLE		Copies of UnFCR[0].	0
8	ABEOINT		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOINT		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

Bit UnIIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. 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 399. Given the status of UnIIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The UnIIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UARTn RLS interrupt (UnIIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UARTn Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UARTn Rx error condition that set the interrupt can be observed via UnLSR[4:1]. The interrupt is cleared upon an UnLSR read.

The UARTn RDA interrupt (UnIIR[3:1] = 010) shares the second level priority with the CTI interrupt (UnIIR[3:1] = 110). The RDA is activated when the UARTn Rx FIFO reaches the trigger level defined in UnFCR[7:6] and is reset when the UARTn 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 (UnIIR[3:1] = 110) is a second level interrupt and is set when the UARTn Rx FIFO contains at least one character and no UARTn Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UARTn Rx FIFO activity (read or write of UARTn RSR) will clear the interrupt. This interrupt is intended to flush the UARTn 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 399: UARTn Interrupt Handling

U0IIR[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]	UnLSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (UnFCR0=1)	UnRBR Read[3] or UARTn 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	UnRBR Read[3]
0010	Third	THRE	THRE[2]	UnIIR 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 18.6.8 “UARTn Line Status Register”

[3] For details see Section 18.6.1 “UARTn Receiver Buffer Register”

[4] For details see Section 18.6.5 “UARTn Interrupt Identification Register” and Section 18.6.2 “UARTn Transmit Holding Register”

The UARTn THRE interrupt (UnIIR[3:1] = 001) is a third level interrupt and is activated when the UARTn THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UARTn 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 UnTHR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to UnTHR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UARTn THR FIFO has held two or more characters at one time and currently, the UnTHR is empty. The THRE interrupt is reset when a UnTHR write occurs or a read of the UnIIR occurs and the THRE is the highest interrupt (UnIIR[3:1] = 001).

18.6.6 UARTn FIFO Control Register

The write-only UnFCR controls the operation of the UARTn Rx and TX FIFOs.

Table 400: UARTn FIFO Control Register, write only (FCR - address 0x4000 C008 (UART0), 0x4009 8008 (UART2), 0x4007 C008 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFOEN		FIFO Enable.	0
		0	UARTn FIFOs are disabled. Must not be used in the application.	
		1	Active high enable for both UARTn Rx and TX FIFOs and UnFCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the related UART FIFOs.	
1	RXFIFORES		RX FIFO Reset.	0
		0	No impact on either of UARTn FIFOs.	
		1	Writing a logic 1 to UnFCR[1] will clear all bytes in UARTn Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset.	0
		0	No impact on either of UARTn FIFOs.	
		1	Writing a logic 1 to UnFCR[2] will clear all bytes in UARTn 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 18.6.6.1.	0
5:4	-		Reserved. Read value is undefined, only zero should be written.	NA
7:6	RXTRIGLVL		RX Trigger Level. These two bits determine how many receiver UARTn 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 zero should be written.	NA

18.6.6.1 DMA Operation

The user can optionally operate the UART 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.

UART 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.

UART 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.

18.6.7 UARTn Line Control Register

The UnLCR determines the format of the data character that is to be transmitted or received.

Table 401: UARTn Line Control Register (LCR - address 0x4000 C00C (UART0), 0x4009 800C (UART2), 0x4009 C00C (UART3)) 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 UnLCR[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	BC		Break Control	0
		0	Disable break transmission.	
		1	Enable break transmission. Output pin UARTn TXD is forced to logic 0 when UnLCR[6] is active high.	
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 zero should be written.	NA

18.6.8 UARTn Line Status Register

The UnLSR is a read-only register that provides status information on the UARTn TX and RX blocks.

Table 402: UARTn Line Status Register (LSR - address 0x4000 C014 (UART0), 0x4009 8014 (UART2), 0x4009 C014 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready. UnLSR[0] is set when the UnRBR holds an unread character and is cleared when the UARTn RBR FIFO is empty.	0
		0	The UARTn receiver FIFO is empty.	
		1	The UARTn receiver FIFO is not empty.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. An UnLSR read clears UnLSR[1]. UnLSR[1] is set when UARTn RSR has a new character assembled and the UARTn RBR FIFO is full. In this case, the UARTn RBR FIFO will not be overwritten and the character in the UARTn 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 UnLSR read clears UnLSR[2]. Time of parity error detection is dependent on UnFCR[0]. **Note:** A parity error is associated with the character at the top of the UARTn 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 UnLSR read clears UnLSR[3]. The time of the framing error detection is dependent on UnFCR[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 UARTn RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	BI		Break Interrupt. When RXDn is held in the spacing state (all zeroes) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXDn goes to marking state (all ones). An UnLSR read clears this status bit. The time of break detection is dependent on UnFCR[0]. **Note:** The break interrupt is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty UARTn THR and is cleared on a UnTHR write.	1
		0	UnTHR contains valid data.	
		1	UnTHR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both UnTHR and UnTSR are empty; TEMT is cleared when either the UnTSR or the UnTHR contain valid data.	1
		0	UnTHR and/or the UnTSR contains valid data.	
		1	UnTHR and the UnTSR are empty.	

Table 402: UARTn Line Status Register (LSR - address 0x4000 C014 (UART0), 0x4009 8014 (UART2), 0x4009 C014 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
7	RXFE		Error in RX FIFO. UnLSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the UnRBR. This bit is cleared when the UnLSR register is read and there are no subsequent errors in the UARTn FIFO.	0
		0	UnRBR contains no UARTn RX errors or UnFCR[0]=0.	
		1	UARTn RBR contains at least one UARTn RX error.	
31:8	-		Reserved. The value read from a reserved bit is not defined.	NA

18.6.9 UARTn Scratch Pad Register

The UnSCR has no effect on the UARTn 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 UnSCR has occurred.

Table 403: UARTn Scratch Pad Register (SCR - address 0x4000 C01C (UART0), 0x4009 801C (UART2), 0x4009 C01C (UART3)) bit description

Bit	Symbol	Description	Reset Value
7:0	PAD	A readable, writable byte.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

18.6.10 UARTn Auto-baud Control Register

The UARTn Auto-baud Control Register (UnACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 404: UARTn Auto-baud Control Register (ACR - address 0x4000 C020 (UART0), 0x4009 8020 (UART2), 0x4009 C020 (UART3)) bit description

Bit	Symbol	Value	Description	Reset value
0	START		Start bit. This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	MODE		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AUTORESTART		Restart bit.	0
		0	No restart.	
		1	Restart in case of time-out (counter restarts at next UARTn Rx falling edge)	0
7:3	-		Reserved. Read value is undefined, only zero should be written.	NA
8	ABEOINTCLR		End of auto-baud interrupt clear bit (write-only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
		0	No impact.	
		1	Clear the corresponding interrupt in the IIR.	
9	ABTOINTCLR		Auto-baud time-out interrupt clear bit (write-only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
		0	No impact.	
		1	Clear the corresponding interrupt in the IIR.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

18.6.10.1 Auto-baud

The UARTn auto-baud function can be used to measure the incoming baud rate based on the "AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers UnDLM and UnDLL accordingly.

Remark: the fractional rate divider is not connected during auto-baud operations, and therefore should not be used when the auto-baud feature is needed.

Auto-baud is started by setting the UnACR Start bit. Auto-baud can be stopped by clearing the UnACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the UnACR Mode bit. In mode 0 the baud rate is measured on two subsequent falling edges of the UARTn Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud rate is measured between the falling edge and the subsequent rising edge of the UARTn Rx pin (the length of the start bit).

The UnACR AutoRestart bit can be used to automatically restart baud rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UARTn Rx pin.

The auto-baud function can generate two interrupts.

- The UnIIR ABTOInt interrupt will get set if the interrupt is enabled (UnIER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The UnIIR ABEOInt interrupt will get set if the interrupt is enabled (UnIER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding UnACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UARTn Rx pin baud rate, but the value of the UnFDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to UnDLM and UnDLL registers should be done before UnACR register write. The minimum and the maximum baud rates supported by UARTn are function of pclk, number of data bits, stop bits and parity bits.

(3)

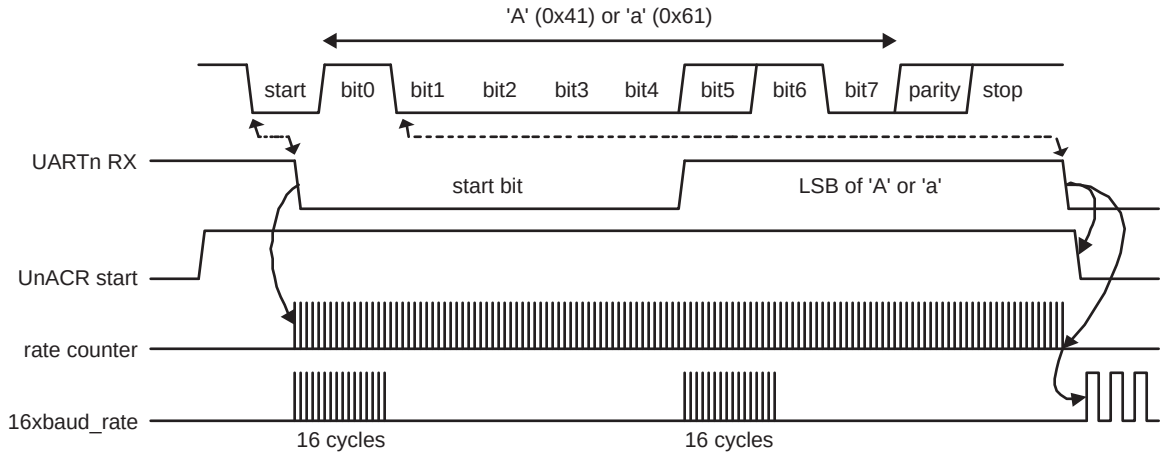
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UARTn_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

18.6.10.2 Auto-baud modes

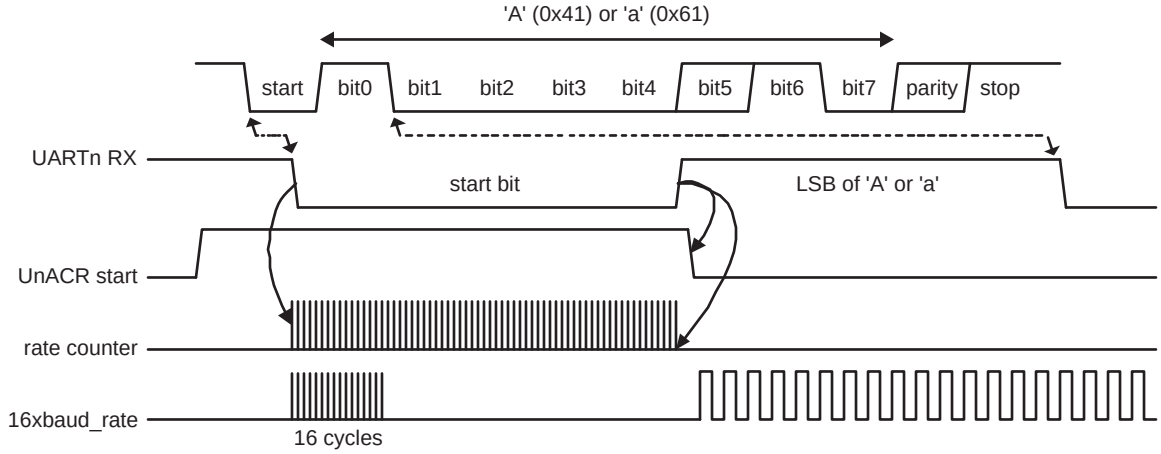
When the software is expecting an “AT” command, it configures the UARTn with the expected character format and sets the UnACR Start bit. The initial values in the divisor latches UnDLM and UnDLM don‘t care. Because of the “A” or “a” ASCII coding (”A" = 0x41, “a” = 0x61), the UARTn Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the UnACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On UnACR Start bit setting, the baud rate measurement counter is reset and the UARTn UnRSR is reset. The UnRSR baud rate is switch to the highest rate.
2. A falling edge on UARTn Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud rate pre-scaled) UARTn input clock, guaranteeing the start bit is stored in the UnRSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UARTn input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UARTn Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UARTn Rx pin.

6. The rate counter is loaded into UnDLM/UnDLL and the baud rate will be switched to normal operation. After setting the UnDLM/UnDLL the end of auto-baud interrupt UnIIR ABEOInt will be set, if enabled. The UnRSR will now continue receiving the remaining bits of the “A/a” character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 76. Auto-baud a) mode 0 and b) mode 1 waveform

18.6.11 UARTn Fractional Divider Register

The UARTn Fractional Divider Register (UnFDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be greater than 2.

Table 405: UARTn Fractional Divider Register (FDR - address 0x4000 C028 (UART0), 0x4009 8028 (UART2), 0x4009 C028 (UART3)) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud Rate generation pre-scaler divisor value. If this field is 0, fractional baud rate generator will not impact the UARTn baud rate.	0
7:4	MULVAL	1	Baud Rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud rate generator is used or not.	1
31:8	-		Reserved. Read value is undefined, only zero should be written.	0

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of the UART disabled, making sure that the UART is fully software and hardware compatible with UARTs not equipped with this feature.

The UART baud rate can be calculated as (n = 0/2/3):

(4)

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where PCLK is the peripheral clock, UnDLM and UnDLL are the standard UART baud rate divider registers, and DIVADDVAL and MULVAL are UART fractional baud rate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 14$
3. $DIVADDVAL < MULVAL$

The value of the UnFDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the UnFDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

18.6.11.1 Baud rate calculation

UARTn can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baud rate with a relative error of less than 1.1% from the desired one.

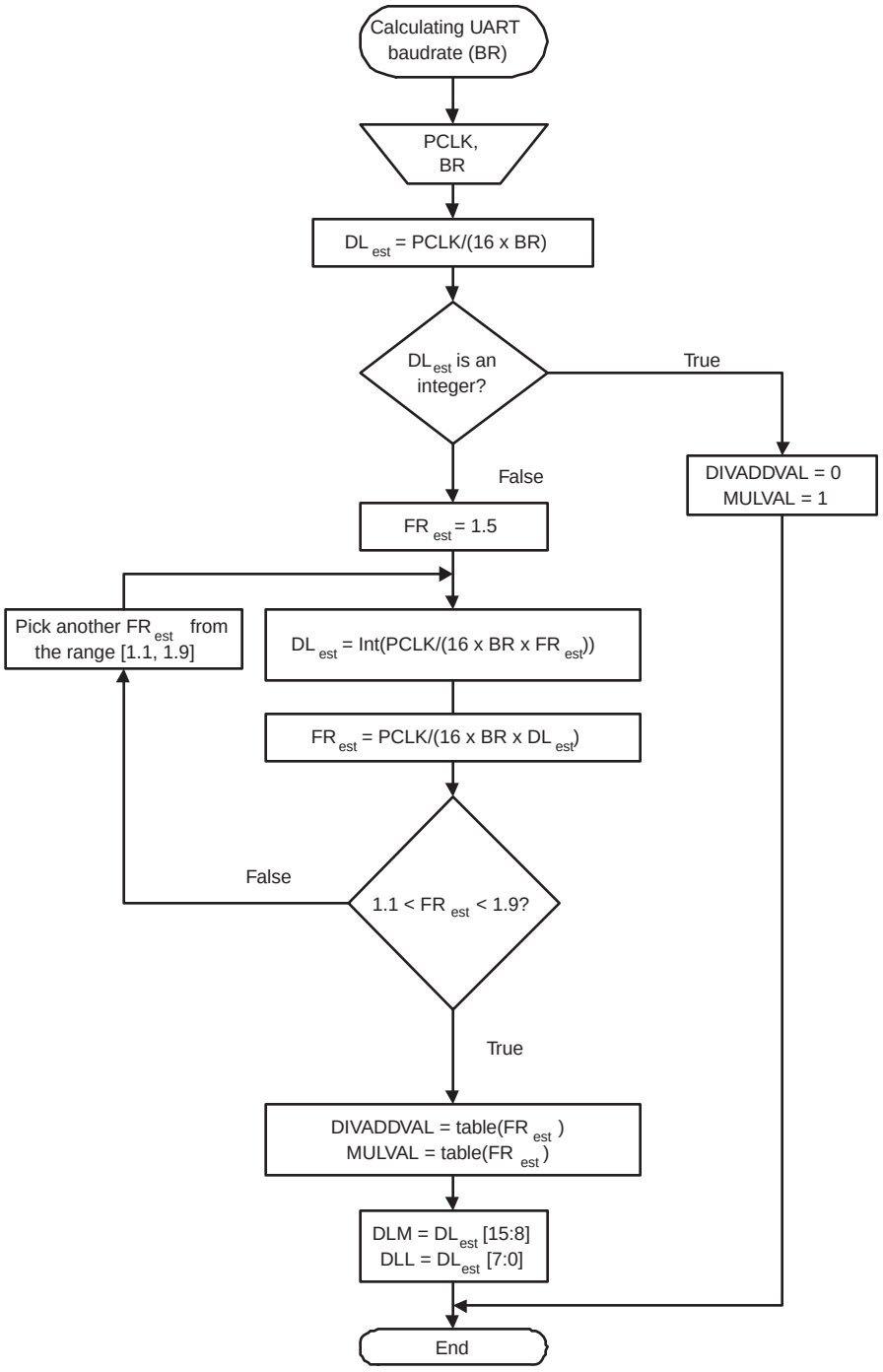


Fig 77. Algorithm for setting UART dividers

Table 406. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

18.6.11.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

18.6.11.1.2 Example 2: PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 406 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 4 the UART rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

18.6.12 UARTn Transmit Enable Register

The UnTER register enables implementation of software flow control. When TXEn=1, UARTn transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UARTn transmission will stop.

Table 407 describes how to use the TXEn bit in order to achieve software flow control.

Table 407: UARTn Transmit Enable Register (TER - address 0x4000 C030 (UART0), 0x4009 8030 (UART2), 0x4009 C030 (UART3)) bit description

Bit	Symbol	Description	Reset Value
6:0	-	Reserved. Read value is undefined, only zero should be written.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit is cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software implementing software-handshaking can clear this bit when it receives an XOFF character (DC3). Software can set this bit again when it receives an XON (DC1) character.	1
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

18.6.13 UARTn RS485 Control register

The UnRS485CTRL register controls configuration of the RS-485/EIA-485 mode.

Table 408: UARTn RS485 Control register (RS485CTRL - address 0x4000 C04C (UART0), 0x4009 804C (UART2), 0x4009 C04C (UART3)) bit description

Bit	Symbol	Value	Description	Reset value
0	NMMEN		NMM enable.	0
		0	RS-485/EIA-485 Normal Multidrop Mode (NMM) is disabled.	
		1	RS-485/EIA-485 Normal Multidrop Mode (NMM) is enabled. In this mode, an address is detected when a received byte has the parity bit = 1, generating a received data interrupt. See Section 18.6.16 “RS-485/EIA-485 modes of operation”.	
1	RXDIS		Receiver enable.	0
		0	The receiver is enabled.	
		1	The receiver is disabled.	
2	AADEN		AAD enable.	0
		0	Auto Address Detect (AAD) is disabled.	
		1	Auto Address Detect (AAD) is enabled.	
3	-		Reserved. Read value is undefined, only zero should be written.	NA
4	DCTRL		Direction control enable.	0
		0	Disable Auto Direction Control.	
		1	Enable Auto Direction Control.	
5	OINV		Direction control pin polarity. This bit reverses the polarity of the direction control signal on the Un_OE pin.	0
		0	The direction control pin will be driven to logic ‘0’ when the transmitter has data to be sent. It will be driven to logic ‘1’ after the last bit of data has been transmitted.	
		1	The direction control pin will be driven to logic ‘1’ when the transmitter has data to be sent. It will be driven to logic ‘0’ after the last bit of data has been transmitted.	
31:6	-		Reserved. Read value is undefined, only zero should be written.	NA

18.6.14 UARTn RS-485 Address Match register

The UnRS485ADRMATCH register contains the address match value for RS-485/EIA-485 mode.

Table 409. UARTn RS-485 Address Match register (RS485ADRMATCH - address 0x4000 C050 (UART0), RS485ADRMATCH - 0x4009 8050 (UART2), RS485ADRMATCH - 0x4009 C050 (UART3)) bit description

Bit	Symbol	Description	Reset value
7:0	ADRMATCH	Contains the address match value.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

18.6.15 UARTn RS-485 Delay value register

The user may program the 8-bit RS485DLY register with a delay between the last stop bit leaving the TXFIFO and the de-assertion of Un_OE. This delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

Table 410. UARTn RS-485 Delay value register (RS485DLY - address 0x4000 0054 (UART0), RS485DLY - 0x4009 8054 (UART2), RS485DLY - 0x4009 C054 (UART3)) bit description

Bit	Symbol	Description	Reset value
7:0	DLY	Contains the direction control (UnOE) delay value. This register works in conjunction with an 8-bit counter.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

18.6.16 RS-485/EIA-485 modes of operation

The RS-485/EIA-485 feature allows the UART to be configured as an addressable slave. The addressable slave is one of multiple slaves controlled by a single master.

The UART master transmitter will identify an address character by setting the parity bit to ‘1’. For data characters, the parity bit is set to ‘0’.

Each UART slave receiver can be assigned a unique address. The slave can be programmed to either manually or automatically reject data following an address which is not theirs.

RS-485/EIA-485 Normal Multidrop Mode (NMM)

Setting the RS485CTRL bit 0 enables this mode. In this mode, the parity bit is used for the alternative purpose of making a distinction between address and data in received data.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received data bytes will be ignored and will not be stored in the RXFIFO. When an address byte is detected (parity bit = ‘1’) it will be placed into the RXFIFO and an Rx Data Ready Interrupt will be generated. The processor can then read the address byte and decide whether or not to enable the receiver to accept the following data.

While the receiver is ENABLED (RS485CTRL bit 1 =’0’) all received bytes will be accepted and stored in the RXFIFO regardless of whether they are data or address.

RS-485/EIA-485 Auto Address Detection (AAD) mode

When both RS485CTRL register bits 0 (9-bit mode enable) and 2 (AAD mode enable) are set, the UART is in auto address detect mode.

In this mode, the receiver will compare any address byte received (parity = ‘1’) to the 8-bit value programmed into the RS485ADRMATCH register.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received byte will be discarded if it is either a data byte OR an address byte which fails to match the RS485ADRMATCH value.

When a matching address character is detected it will be pushed onto the RXFIFO along with the parity bit, and the receiver will be automatically enabled (RS485CTRL bit 1 will be cleared by hardware). The receiver will also generate n Rx Data Ready Interrupt.

While the receiver is ENABLED (RS485CTRL bit 1 = ‘0’) all bytes received will be accepted and stored in the RXFIFO until an address byte which does not match the RS485ADRMATCH value is received. When this occurs, the receiver will be automatically disabled in hardware (RS485CTRL bit 1 will be set), The received non-matching address character will not be stored in the RXFIFO.

RS-485/EIA-485 Auto Direction Control

RS485/EIA-485 Mode includes the option of allowing the transmitter to automatically control the state of the UnOE pin as a direction control output signal.

Setting RS485CTRL bit 4 = ‘1’ enables this feature.

When Auto Direction Control is enabled, the UnOE pin will be asserted (driven low) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven high) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

When Auto Direction Control is enabled, the selected pin will be asserted (driven low) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven high) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

RS485/EIA-485 driver delay time

The driver delay time is the delay between the last stop bit leaving the TXFIFO and the de-assertion of UnOE. This delay time can be programmed in the 8-bit RS485DLY register. The delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

RS485/EIA-485 output inversion

The polarity of the direction control signal on the UnOE pin can be reversed by programming bit 5 in the UnRS485CTRL register. When this bit is set, the direction control pin will be driven to logic 1 when the transmitter has data waiting to be sent. The direction control pin will be driven to logic 0 after the last bit of data has been transmitted.

19.1 How to read this chapter

Most LPC408x/407x family devices include 5 UARTs. A few devices do not include UART4. Refer to Section 1.4 and specific device data sheets for details. UART4 is essentially the same as UARTs 02/3, but with an added synchronous mode and smart card mode.

19.2 Basic configuration

UART4 is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCUART4.

   Remark: On reset, UART4 is disabled (PCUART4 = 0).
2. Peripheral clock: This UART operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Baud rate: In register U4LCR (Table 421), set bit DLAB =1. This enables access to registers DLL (Table 415) and DLM (Table 416) for setting the baud rate. Also, if needed, set the fractional baud rate in the fractional divider register (Table 427).
4. UART FIFO: Use bit FIFO enable (bit 0) in register U4FCR (Table 420) to enable the FIFOs.
5. Pins: Select UART pins and pin modes through the relevant IOCON registers (Section 7.4.1).

   Remark: UART receive pins should not have pull-down resistors enabled.
6. Interrupts: To enable UART interrupts set bit DLAB =0 in register U4LCR (Table 421). This enables access to U4IER (Table 417). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
7. DMA: UART4 transmit and receive functions can operate with the GPDMA controller (see Table 696).

19.3 Features

- Data sizes of 5, 6, 7, and 8 bits.
- Parity generation and checking: odd, even mark, space or none.
- One or two stop bits.
- 16 byte Receive and Transmit FIFOs.
- Built-in baud rate generator, including a fractional rate divider for great versatility.
- Supports DMA for both transmit and receive.
- Auto-baud capability
- Break generation and detection.
- Multiprocessor addressing mode.
- IrDA mode to support infrared communication.
- Support for software flow control.
- RS-485/EIA-485 9-bit mode support with output enable.
- Optional synchronous send or receive mode.
- Optional ISO 7816-3 compliant smart card interface.

19.4 Architecture

The architecture of UART4 is shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART.

The UART4 receiver block, U4RX, monitors the serial input line, RXDn, for valid input. The UART4 RX Shift Register (U4RSR) accepts valid characters via RXDn. After a valid character is assembled in U4RSR, it is passed to the UART4 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART4 transmitter block, U4TX, accepts data written by the CPU or host and buffers the data in the UART4 TX Holding Register FIFO (U4THR). The UART4 TX Shift Register (U4TSR) reads the data stored in U4THR and assembles the data to transmit via the serial output pin, TXDn.

The UART4 Baud Rate Generator block, U4BRG, generates the timing enables used by the UART4 TX block. The U4BRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the U4DLL and U4DLM registers. This divided down clock is the 16x oversample clock.

The interrupt interface contains registers U4IER and U4IIR. The interrupt interface receives several one clock wide enables from the U4TX and U4RX blocks.

Status information from the U4TX and U4RX is stored in the U4LSR. Control information for the U4TX and U4RX is stored in U4LCR.

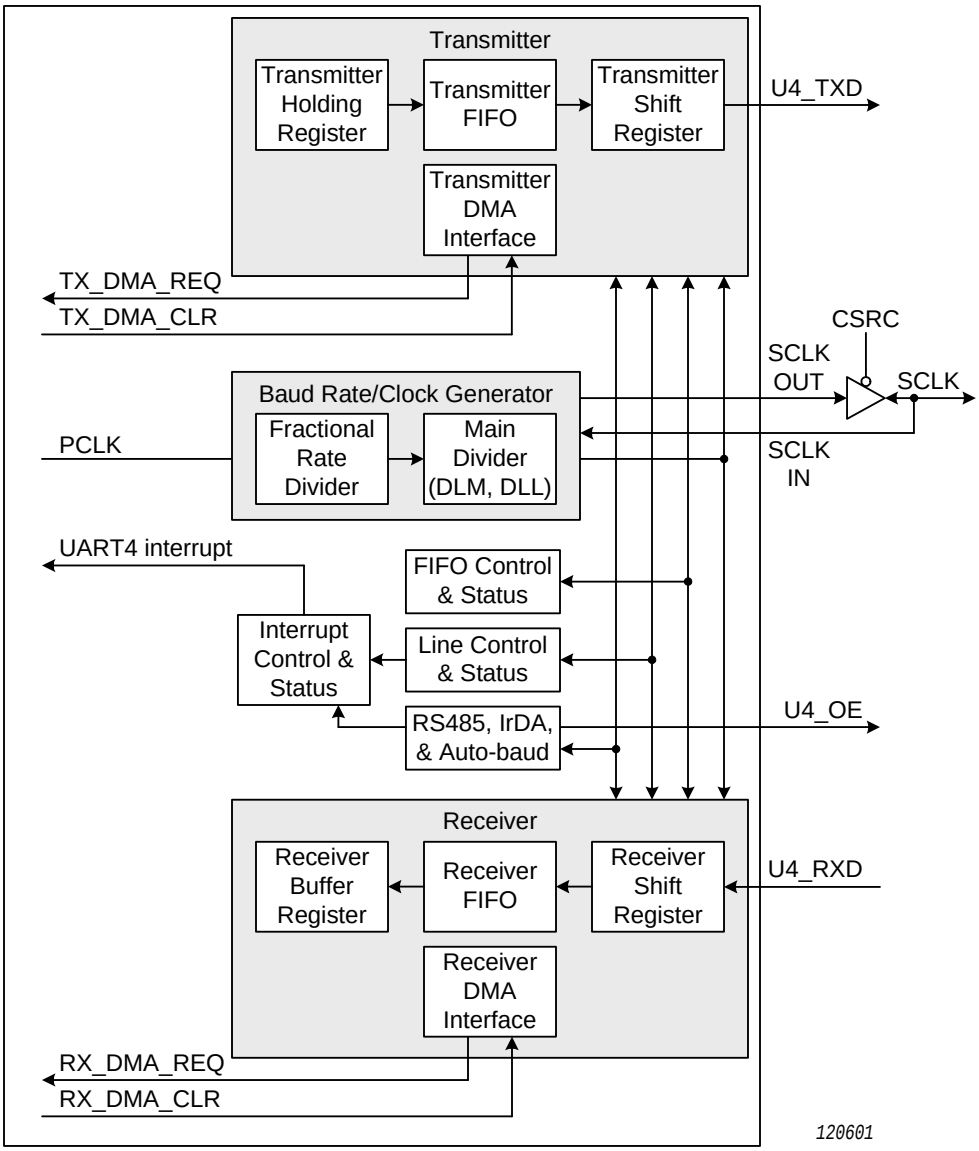


Fig 78. UART 4 block diagram

19.5 Pin description

Table 411: UART4 Pin description

Pin	Type	Description
U4_RXD	Input	**Serial Input.** Serial receive data.
U4_TXD	Output	**Serial Output.** Serial transmit data (input/output in smart card mode).
U4_OE	Output	**Output Enable.** RS-485/EIA-485 output enable.
U4_SCLK	I/O	**Serial Clock.** Clock input or output in synchronous mode and smart card mode.

19.6 Register description

The Divisor Latch Access Bit (DLAB) is contained in U4LCR7 and enables access to the Divisor Latches.

Table 412. Register overview: UART4 (base address: 0x400A 4000)

Name	Access	Address offset	Description	Reset value[1]	Table
RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read (DLAB =0).	NA	413
THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here (DLAB =0).	NA	414
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	415
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	416
IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts (DLAB =0).	0	417
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01	418
FCR	WO	0x008	FIFO Control Register. Controls UART FIFO usage and modes.	0	420
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0	421
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60	422
SCR	R/W	0x01C	Scratch Pad Register. 8-bit temporary storage for software.	0	423
ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0	424
ICR	R/W	0x024	IrDA Control Register. Enables and configures the IrDA mode.	0	425
FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10	427
OSR	R/W	0x02C	Oversampling register. Controls the degree of oversampling during each bit time.	0xF0	429
SCICTRL	R/W	0x048	Smart card Interface control register. Enables and configures the smart card Interface feature.	0	430
RS485CTRL	R/W	0x04C	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	0	431
ADRMATCH	R/W	0x050	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	0	432
RS485DLY	R/W	0x054	RS-485/EIA-485 direction control delay.	0	433
SYNCCTRL	R/W	0x058	Synchronous mode control register.	0	434

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

19.6.1 UART4 Receiver Buffer Register

The U4RBR is the top byte of the UART4 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 zeroes.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the U4RBR. The U4RBR 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 U0LSR register, and then to read a byte from the U4RBR.

Table 413: UART4 Receiver Buffer Register when DLAB = 0 (RBR - address 0x400A 4000) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UART4 Receiver Buffer Register contains the oldest received byte in the UART4 Rx FIFO.	Undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

19.6.2 UART4 Transmit Holding Register

The U4THR is the top byte of the UART4 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 U4LCR must be zero in order to access the U4THR. The U4THR is always write-only.

Table 414: UART4 Transmit Holding Register when DLAB = 0 (THR -address 0x400A 4000) bit description

Bit	Symbol	Description
7:0	THR	Writing to the UART4 Transmit Holding Register causes the data to be stored in the UART4 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.
31:8	-	Reserved. Read value is undefined, only zero should be written.

19.6.3 UART4 Divisor Latch LSB register

The UART4 Divisor Latch is part of the UART4 Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16× the desired baud rate. The U4DLL and U4DLM registers together form a 16-bit divisor where U4DLL contains the lower 8 bits of the divisor and U4DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed. The Divisor Latch Access Bit (DLAB) in U4LCR must be one in order to access the UART4 Divisor Latches. Details on how to select the right value for U4DLL and U4DLM can be found later in this chapter, see Section 19.6.12.

Table 415: UART4 Divisor Latch LSB register when DLAB = 1 (DLL - address 0x400A 4000) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UART4 Divisor Latch LSB Register, along with the U4DLM register, determines the baud rate of the UART4.	0x01
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

Table 416: UART4 Divisor Latch MSB register when DLAB = 1 (DLM - address 0x400A 4004) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UART4 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART4.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

19.6.4 UART4 Interrupt Enable Register

The U4IER is used to enable the three UART4 interrupt sources.

Table 417: UART4 Interrupt Enable Register when DLAB = 0 (IER - address 0x400A 4004) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for UARTn. 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 UARTn. The status of this can be read from UnLSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXIE		RX Line Status Interrupt Enable. Enables the UARTn RX line status interrupts. The status of this interrupt can be read from UnLSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
7:3	-		Reserved. Read value is undefined, only zero should be written.	NA
8	ABEOINTEN		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	
9	ABTOINTEN		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

19.6.5 UART4 Interrupt Identification Register

The U4IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U4IIR access. If an interrupt occurs during an U4IIR access, the interrupt is recorded for the next U4IIR access.

Table 418: UART4 Interrupt Identification Register (IIR - address 0x400A 4008) bit description

Bit	Symbol	Value	Description	Reset Value
0	INTSTATUS		Interrupt status. Note that U4IIR[0] is active low. The pending interrupt can be determined by evaluating U4IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. U4IER[3:1] identifies an interrupt corresponding to the UART4 Rx or TX FIFO. All other combinations of U4IER[3:1] not listed below are reserved (000,100,101,111).	0
		0x3	1 - Receive Line Status (RLS).	
		0x2	2a - Receive Data Available (RDA).	
		0x6	2b - Character Time-out Indicator (CTI).	
		0x1	3 - THRE Interrupt	
5:4	-		Reserved. Read value is undefined, only zero should be written.	NA
7:6	FIFOENABLE		Copies of U4FCR[0].	0
8	ABEOINT		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOINT		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

Bit U4IIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. 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 419. Given the status of U4IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U4IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART4 RLS interrupt (U4IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART4 Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART4 Rx error condition that set the interrupt can be observed via U0LSR[4:1]. The interrupt is cleared upon an U4LSR read.

The UART4 RDA interrupt (U4IIR[3:1] = 010) shares the second level priority with the CTI interrupt (U4IIR[3:1] = 110). The RDA is activated when the UART4 Rx FIFO reaches the trigger level defined in U4FCR[7:6] and is reset when the UART4 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 (U4IIR[3:1] = 110) is a second level interrupt and is set when the UART4 Rx FIFO contains at least one character and no UART4 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART4 Rx FIFO activity (read or write of UART4 RSR) will clear the interrupt. This interrupt is intended to flush the UART4 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 419: UART4 Interrupt Handling

U0IIR[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]	U4LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U4FCR0=1)	U4RBR Read[3] or UART4 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	U4RBR Read[3]
0010	Third	THRE	THRE[2]	U4IIR 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 19.6.8 “UART4 Line Status Register”

[3] For details see Section 19.6.1 “UART4 Receiver Buffer Register”

[4] For details see Section 19.6.5 “UART4 Interrupt Identification Register” and Section 19.6.2 “UART4 Transmit Holding Register”

The UART4 THRE interrupt (U4IIR[3:1] = 001) is a third level interrupt and is activated when the UART4 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART4 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 U4THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U4THR without a THRE interrupt to decode and service. A THRE interrupt is set

immediately if the UART4 THR FIFO has held two or more characters at one time and currently, the U4THR is empty. The THRE interrupt is reset when a U4THR write occurs or a read of the U4IIR occurs and the THRE is the highest interrupt (U4IIR[3:1] = 001).

19.6.6 UART4 FIFO Control Register

The write-only U4FCR controls the operation of the UART4 Rx and TX FIFOs.

Table 420: UART4 FIFO Control Register (FCR - address 0x400A 4008) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFOEN		FIFO Enable.	0
		0	UARTn FIFOs are disabled. Must not be used in the application.	
		1	Active high enable for both UARTn Rx and TX FIFOs and UnFCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the related UART FIFOs.	
1	RXFIFORES		RX FIFO Reset.	0
		0	No impact on either of UARTn FIFOs.	
		1	Writing a logic 1 to UnFCR[1] will clear all bytes in UARTn Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset.	0
		0	No impact on either of UARTn FIFOs.	
		1	Writing a logic 1 to UnFCR[2] will clear all bytes in UARTn 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 19.6.6.1.	0
5:4	-		Reserved. Read value is undefined, only zero should be written.	NA
7:6	RXTRIGLVL		RX Trigger Level. These two bits determine how many receiver UARTn 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 zero should be written.	NA

19.6.6.1 DMA Operation

The user can optionally operate the UART 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.

UART 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.

UART 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.

19.6.7 UART4 Line Control Register

The U4LCR determines the format of the data character that is to be transmitted or received.

Table 421: UART4 Line Control Register (LCR - address 0x400A 400C) 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 UnLCR[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	BC		Break Control	0
		0	Disable break transmission.	
		1	Enable break transmission. Output pin UARTn TXD is forced to logic 0 when UnLCR[6] is active high.	
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 zero should be written.	NA

19.6.8 UART4 Line Status Register

The U4LSR is a read-only register that provides status information on the UART4 TX and RX blocks.

Table 422: UART4 Line Status Register (LSR - address 0x400A 4014) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready. UnLSR[0] is set when the UnRBR holds an unread character and is cleared when the UARTn RBR FIFO is empty.	0
		0	The UARTn receiver FIFO is empty.	
		1	The UARTn receiver FIFO is not empty.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. An UnLSR read clears UnLSR[1]. UnLSR[1] is set when UARTn RSR has a new character assembled and the UARTn RBR FIFO is full. In this case, the UARTn RBR FIFO will not be overwritten and the character in the UARTn 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 UnLSR read clears UnLSR[2]. Time of parity error detection is dependent on UnFCR[0]. **Note:** A parity error is associated with the character at the top of the UARTn 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 UnLSR read clears UnLSR[3]. The time of the framing error detection is dependent on UnFCR[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 UARTn RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	BI		Break Interrupt. When RXDn is held in the spacing state (all zeroes) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXDn goes to marking state (all ones). An UnLSR read clears this status bit. The time of break detection is dependent on UnFCR[0]. **Note:** The break interrupt is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty UARTn THR and is cleared on a UnTHR write.	1
		0	UnTHR contains valid data.	
		1	UnTHR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both UnTHR and UnTSR are empty; TEMT is cleared when either the UnTSR or the UnTHR contain valid data.	1
		0	UnTHR and/or the UnTSR contains valid data.	
		1	UnTHR and the UnTSR are empty.	

Table 422: UART4 Line Status Register (LSR - address 0x400A 4014) bit description

Bit	Symbol	Value	Description	Reset Value
7	RXFE		Error in RX FIFO . UnLSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the UnRBR. This bit is cleared when the UnLSR register is read and there are no subsequent errors in the UARTn FIFO.	0
		0	UnRBR contains no UARTn RX errors or UnFCR[0]=0.	
		1	UARTn RBR contains at least one UARTn RX error.	
31:8	-		Reserved. The value read from a reserved bit is not defined.	NA

19.6.9 UART4 Scratch Pad Register

The U4SCR has no effect on the UART4 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 U4SCR has occurred.

Table 423: UART4 Scratch Pad Register (SCR - address 0x400A 401C) bit description

Bit	Symbol	Description	Reset Value
7:0	Pad	A readable, writable byte.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

19.6.10 UART4 Auto-baud Control Register

The UART4 Auto-baud Control Register (U4ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 424: UART4 Auto-baud Control Register (ACR - 0x400A 4020) bit description

Bit	Symbol	Value	Description	Reset value
0	START		Start bit. This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	MODE		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AUTORESTART		Restart bit.	0
		0	No restart.	
		1	Restart in case of time-out (counter restarts at next UARTn Rx falling edge)	0
7:3	-		Reserved. Read value is undefined, only zero should be written.	NA
8	ABEOINTCLR		End of auto-baud interrupt clear bit (write-only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
		0	No impact.	
		1	Clear the corresponding interrupt in the IIR.	
9	ABTOINTCLR		Auto-baud time-out interrupt clear bit (write-only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
		0	No impact.	
		1	Clear the corresponding interrupt in the IIR.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

19.6.10.1 Auto-baud

The UART4 auto-baud function can be used to measure the incoming baud rate based on the "AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers U4DLM and U4DLL accordingly.

Remark: the fractional rate divider is not connected during auto-baud operations, and therefore should not be used when the auto-baud feature is needed.

Auto-baud is started by setting the U4ACR Start bit. Auto-baud can be stopped by clearing the U4ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the U4ACR Mode bit. In mode 0 the baud rate is measured on two subsequent falling edges of the UART4 Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud rate is measured between the falling edge and the subsequent rising edge of the UART4 Rx pin (the length of the start bit).

The U4ACR AutoRestart bit can be used to automatically restart baud rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART4 Rx pin.

The auto-baud function can generate two interrupts.

- The U4IIR ABTOInt interrupt will get set if the interrupt is enabled (U4IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U4IIR ABEOInt interrupt will get set if the interrupt is enabled (U4IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U4ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART4 Rx pin baud rate, but the value of the U4FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U4DLM and U4DLL registers should be done before U4ACR register write. The minimum and the maximum baud rates supported by UART4 are function of pclk, number of data bits, stop bits and parity bits.

(5)

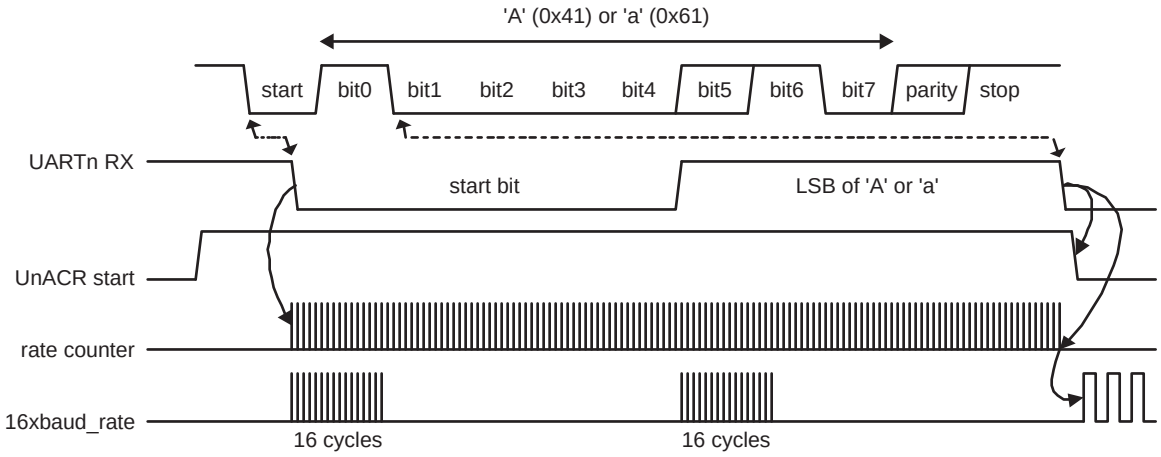
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART4_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

19.6.10.2 Auto-baud modes

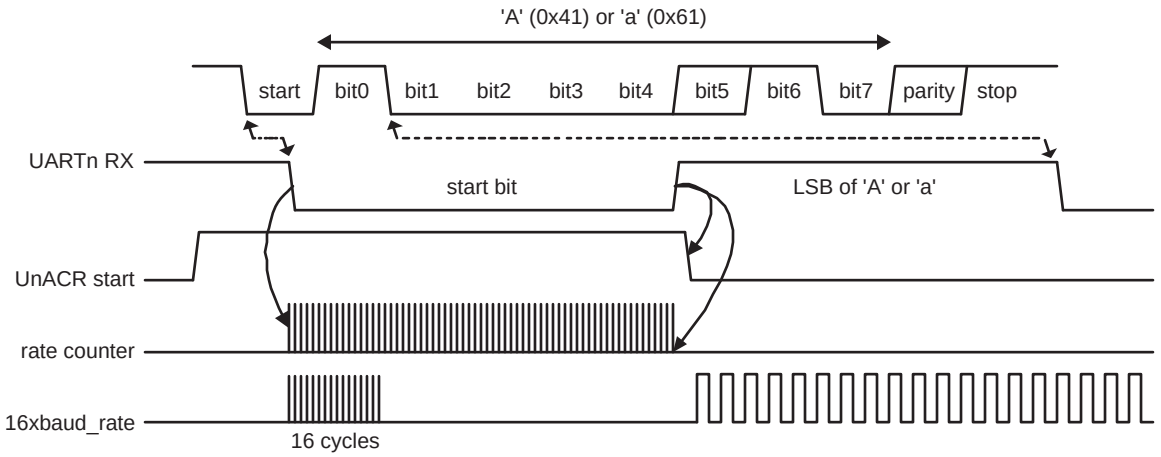
When the software is expecting an “AT” command, it configures the UART4 with the expected character format and sets the U4ACR Start bit. The initial values in the divisor latches U4DLM and U4DLM don‘t care. Because of the “A” or “a” ASCII coding (”A" = 0x41, “a” = 0x61), the UART4 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U4ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U4ACR Start bit setting, the baud rate measurement counter is reset and the UART4 U4RSR is reset. The U4RSR baud rate is switch to the highest rate.
2. A falling edge on UART4 Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud rate pre-scaled) UART4 input clock, guaranteeing the start bit is stored in the U4RSR.

4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART4 input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART4 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART4 Rx pin.
6. The rate counter is loaded into U4DLM/U4DLL and the baud rate will be switched to normal operation. After setting the U4DLM/U4DLL the end of auto-baud interrupt U4IIR ABEOInt will be set, if enabled. The U4RSR will now continue receiving the remaining bits of the “A/a” character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 79. Auto-baud a) mode 0 and b) mode 1 waveform

19.6.11 UART4 IrDA Control Register

The IrDA Control Register enables and configures the IrDA mode on each UART. The value of U4ICR should not be changed while transmitting or receiving data, or data loss or corruption may occur.

Table 425: UART4 IrDA Control Register (ICR - address 0x400A 4024) bit description

Bit	Symbol	Value	Description	Reset value
0	IRDAEN		IrDA mode	0
		0	Disabled. IrDA mode on UART4 is disabled, UART4 acts as a standard UART.	
		1	Enabled. IrDA mode on UART4 is enabled.	
1	IRDAINV		Serial input direction.	0
		0	Not inverted.	
		1	Inverted. This has no effect on the serial output.	
2	FIXPULSEEN		IrDA fixed pulse width mode.	0
		0	Disabled.	
		1	Enabled.	
5:3	PULSEDIV		Configures the pulse when FixPulseEn = 1.	0
		0x0	2xTPCLK	
		0x1	4xTPCLK	
		0x2	8xTPCLK	
		0x3	16xTPCLK	
		0x4	32xTPCLK	
		0x5	64xTPCLK	
		0x6	128xTPCLK	
		0x7	256xTPCLK	
31:6	-		Reserved. Read value is undefined, only zero should be written.	0

The PulseDiv bits in U4ICR are used to select the pulse width when the fixed pulse width mode is used in IrDA mode (IrDAEn = 1 and FixPulseEn = 1). The value of these bits should be set so that the resulting pulse width is at least 1.63 µs. Table 426 shows the possible pulse widths.

Table 426: IrDA Pulse Width

FixPulseEn	PulseDiv	IrDA Transmitter Pulse width (µs)
0	x	3 / (16 × baud rate)
1	0	$2 \times T_{PCLK}$
1	1	$4 \times T_{PCLK}$
1	2	$8 \times T_{PCLK}$
1	3	$16 \times T_{PCLK}$
1	4	$32 \times T_{PCLK}$
1	5	$64 \times T_{PCLK}$
1	6	$128 \times T_{PCLK}$
1	7	$256 \times T_{PCLK}$

19.6.12 UART4 Fractional Divider Register

The UART4 Fractional Divider Register (U4FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be greater than 2.

Table 427: UART4 Fractional Divider Register (FDR - address 0x400A 4028) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud Rate generation pre-scaler divisor value. If this field is 0, fractional baud rate generator will not impact the UART4 baud rate.	0
7:4	MULVAL	1	Baud Rate pre-scaler multiplier value. This field must be greater or equal 1 for UART4 to operate properly, regardless of whether the fractional baud rate generator is used or not.	1
31:8	-		Reserved. Read value is undefined, only zero should be written.	0

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of the UART disabled, making sure that the UART is fully software and hardware compatible with UARTs not equipped with this feature.

The UART baud rate can be calculated as:

(6)

$$UART4_{baudrate} = \frac{PCLK}{16 \times (256 \times U4DLM + U4DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where PCLK is the peripheral clock, U4DLM and U4DLL are the standard UART baud rate divider registers, and DIVADDVAL and MULVAL are UART fractional baud rate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 14$
3. $DIVADDVAL < MULVAL$

The value of the U4FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the U4FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

19.6.12.1 Baud rate calculation

UART4 can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baud rate with a relative error of less than 1.1% from the desired one.

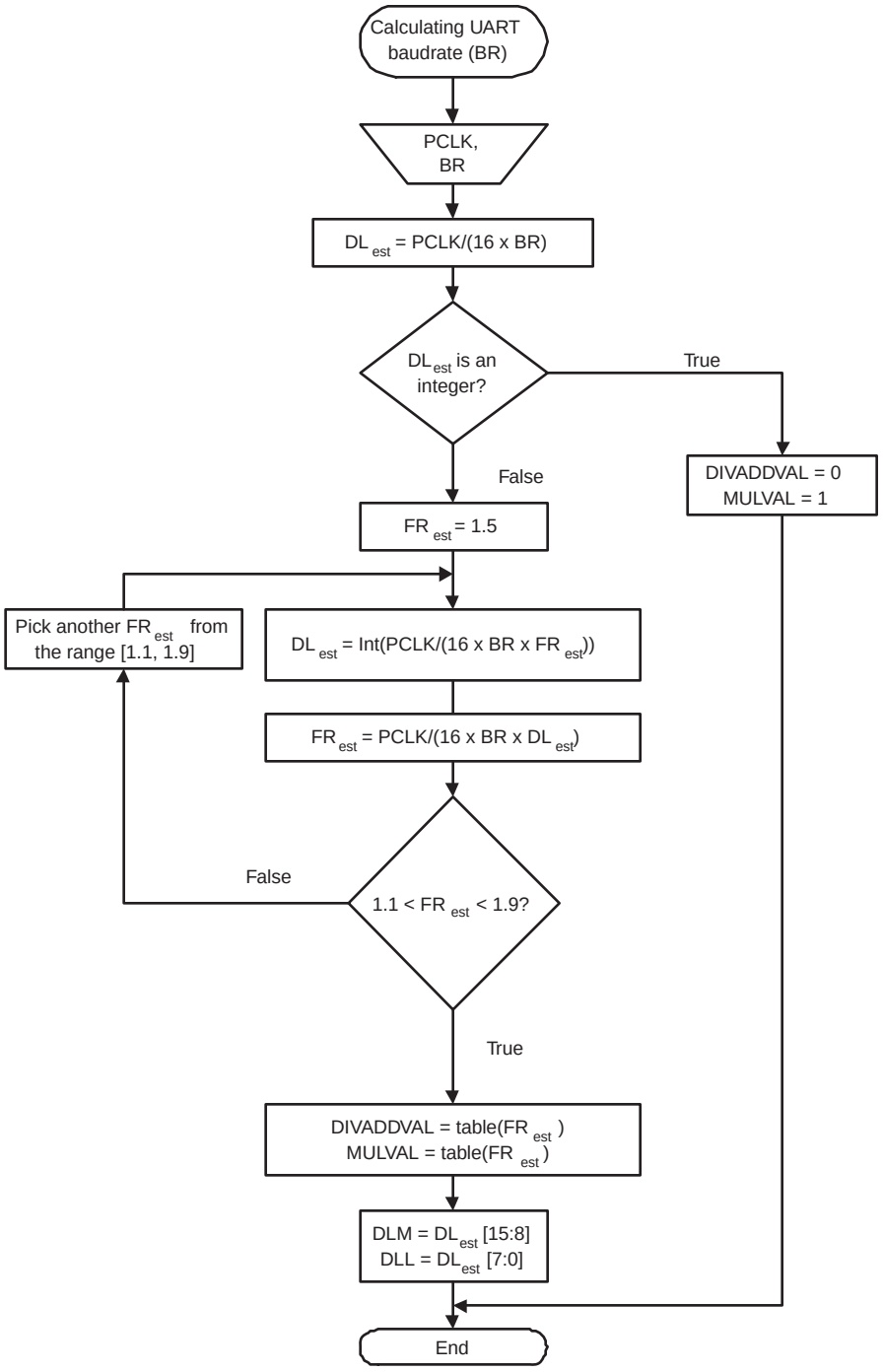


Fig 80. Algorithm for setting UART dividers

Table 428. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

19.6.12.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

19.6.12.1.2 Example 2: PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 428 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 6 the UART rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

19.6.13 UART4 Oversampling Register

In most applications, the UART 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 429. UART4 Oversampling Register (OSR - address 0x400A 402C) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. Read value is undefined, only zero 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	In smart card mode, 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 zero should be written.	NA

Example: For a baud rate of 3.25Mbps with a 24 MHz UART 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.

In smart card mode, 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, but this is not supported by the UART. When smart card mode is enabled, the initial value of OSInt and FDInt should be programmed as “00101110011” (372 minus one).

19.6.14 UART4 Smart Card Interface Control register

This register allows the UART to be used in ISO7816-3 compliant asynchronous smart card applications.

Table 430. UART4 Smart Card Interface Control register (SCICTRL - address 0x400A 4048) 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.	
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	
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.	-
15:8	GUARDTIME		Extra guard time. 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.	
31:16	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.6.14.1 Smart card Connection

When the SCIEN bit in the U4SCICTRL register is set as described above, the UART provides bidirectional serial data on the TXD pin. No RXD pin is used when SCIEN is 1. If the UART SCLK 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 in the UART). GPIO pins can be used to control the smart card reset and power pins.

19.6.14.2 Smart card Setup

The following must be set up in smart card applications:

- If necessary, reset the UART as described in Section 3.5.
- Program one IOCON register to enable a UART TXD function.
- If the smart card to be communicated with requires (or may require) a clock, program one IOCON register for the UART SCLK function. The UART will use it as an output.
- Enable the UART clock and set up UART clocking for an initial UART frequency of 3.58 MHz.
- Program the OSR (Section 19.6.13) for 372x oversampling.
- Program the LCR (Section 19.6.7) for 8-bit characters, parity enabled, even parity.

- Program the GPIO signals associated with the smart card so that (in this order):
  - Reset is low.
  - VCC is provided to the card (GPIO pins do not have the required 200 mA drive).
  - VPP (if provided to the card) is at “idle” state.
- Program SCICTRL (Section 19.6.14) to enable the smart card feature with the desired options.
- Set up one or more timer(s) to provide timing as needed for ISO 7816 startup.

Thereafter, software should monitor the UART and timer status so as to interact with the smart card as described in ISO 7816 3.2.b and subsequently.

19.6.15 UART4 RS485 Control register

The U4RS485CTRL register controls configuration of the RS-485/EIA-485 mode.

Table 431: UART4 RS485 Control register (RS485CTRL - address 0x400A 404C) bit description

Bit	Symbol	Value	Description	Reset value
0	NMMEN		NMM enable.	0
		0	RS-485/EIA-485 Normal Multidrop Mode (NMM) is disabled.	
		1	RS-485/EIA-485 Normal Multidrop Mode (NMM) is enabled. In this mode, an address is detected when a received byte causes the USART to set the parity error and generate an interrupt. See Section 19.6.18 “RS-485/EIA-485 modes of operation”.	
1	RXDIS		Receiver enable.	0
		0	Enabled.	
		1	Disabled.	
2	AADEN		AAD enable	0
		0	Disabled.	
		1	Enabled.	
3	-	-	Reserved.	-
4	DCTRL		Direction control for DIR pin.	0
		0	Disable Auto Direction Control.	
		1	Enable Auto Direction Control.	
5	OINV		Direction control pin polarity. This bit reverses the polarity of the direction control signal on the DIR pin.	0
		0	Low. The direction control pin will be driven to logic ‘0’ when the transmitter has data to be sent. It will be driven to logic ‘1’ after the last bit of data has been transmitted.	
		1	High. The direction control pin will be driven to logic ‘1’ when the transmitter has data to be sent. It will be driven to logic ‘0’ after the last bit of data has been transmitted.	
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.6.16 UART4 RS-485 Address Match register

The U4RS485ADRMATCH register contains the address match value for RS-485/EIA-485 mode.

Table 432. UART4 RS-485 Address Match register (RS485ADRMATCH - address 0x400A 4050) bit description

Bit	Symbol	Description	Reset value
7:0	ADRMATCH	Contains the address match value.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

19.6.17 UART4 RS-485 Delay value register

The user may program the 8-bit RS485DLY register with a delay between the last stop bit leaving the TXFIFO and the de-assertion of U4_OE. This delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

Table 433. UART4 RS-485 Delay value register (RS485DLY - address 0x400A 4054) bit description

Bit	Symbol	Description	Reset value
7:0	DLY	Contains the direction control (U4OE) delay value. This register works in conjunction with an 8-bit counter.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

19.6.18 RS-485/EIA-485 modes of operation

The RS-485/EIA-485 feature allows the UART to be configured as an addressable slave. The addressable slave is one of multiple slaves controlled by a single master.

The UART master transmitter will identify an address character by setting the parity bit to ‘1’. For data characters, the parity bit is set to ‘0’.

Each UART slave receiver can be assigned a unique address. The slave can be programmed to either manually or automatically reject data following an address which is not theirs.

RS-485/EIA-485 Normal Multidrop Mode (NMM)

Setting the RS485CTRL bit 0 enables this mode. In this mode, the parity bit is used for the alternative purpose of making a distinction between address and data in received data.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received data bytes will be ignored and will not be stored in the RXFIFO. When an address byte is detected (parity bit = ‘1’) it will be placed into the RXFIFO and an Rx Data Ready Interrupt will be generated. The processor can then read the address byte and decide whether or not to enable the receiver to accept the following data.

While the receiver is ENABLED (RS485CTRL bit 1 =’0’) all received bytes will be accepted and stored in the RXFIFO regardless of whether they are data or address.

RS-485/EIA-485 Auto Address Detection (AAD) mode

When both RS485CTRL register bits 0 (9-bit mode enable) and 2 (AAD mode enable) are set, the UART is in auto address detect mode.

In this mode, the receiver will compare any address byte received (parity = ‘1’) to the 8-bit value programmed into the RS485ADRMATCH register.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received byte will be discarded if it is either a data byte OR an address byte which fails to match the RS485ADRMATCH value.

When a matching address character is detected it will be pushed onto the RXFIFO along with the parity bit, and the receiver will be automatically enabled (RS485CTRL bit 1 will be cleared by hardware). The receiver will also generate n Rx Data Ready Interrupt.

While the receiver is ENABLED (RS485CTRL bit 1 = ‘0’) all bytes received will be accepted and stored in the RXFIFO until an address byte which does not match the RS485ADRMATCH value is received. When this occurs, the receiver will be automatically disabled in hardware (RS485CTRL bit 1 will be set), The received non-matching address character will not be stored in the RXFIFO.

RS-485/EIA-485 Auto Direction Control

RS485/EIA-485 Mode includes the option of allowing the transmitter to automatically control the state of the U4OE pin as a direction control output signal.

Setting RS485CTRL bit 4 = ‘1’ enables this feature.

When Auto Direction Control is enabled, the U4OE pin will be asserted (driven low) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven high) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

When Auto Direction Control is enabled, the selected pin will be asserted (driven low) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven high) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

RS485/EIA-485 driver delay time

The driver delay time is the delay between the last stop bit leaving the TXFIFO and the de-assertion of U4OE. This delay time can be programmed in the 8-bit RS485DLY register. The delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

RS485/EIA-485 output inversion

The polarity of the direction control signal on the U4OE pin can be reversed by programming bit 5 in the U4RS485CTRL register. When this bit is set, the direction control pin will be driven to logic 1 when the transmitter has data waiting to be sent. The direction control pin will be driven to logic 0 after the last bit of data has been transmitted.

19.6.19 UART4 Synchronous mode control register

SYNCCTRL register controls the synchronous mode. When this mode is in effect, the UART generates or receives a bit clock on the SCLK pin and applies it to the transmit and receive shift registers.

Table 434. UART4 Synchronous mode control register (SYNCCTRL - address 0x400A 4058) bit description

Bit	Symbol	Value	Description	Reset value
0	SYNC		Enables synchronous mode.	0
		0	Disabled	
		1	Enabled	
1	CSRC		Clock source select.	0
		0	Synchronous slave mode (SCLK in)	
		1	Synchronous master mode (SCLK out)	
2	FES		Falling edge sampling.	0
		0	RxD is sampled on the rising edge of SCLK	
		1	RxD is sampled on the falling edge of SCLK	
3	TSBYPASS		Transmit synchronization bypass in synchronous slave mode.	0
		0	The input clock is synchronized prior to being used in clock edge detection logic.	
		1	The input clock is not synchronized prior to being used in clock edge detection logic. This allows for a high er input clock rate at the expense of potential metastability.	
4	CSCEN		Continuous master clock enable (used only when CSRC is 1)	0
		0	SCLK cycles only when characters are being sent on TxD	
		1	SCLK runs continuously (characters can be received on RxD independently from transmission on TxD)	
5	SSDIS		Disable start/stop bits	0
		0	Send start and stop bits as in other modes.	
		1	Do not send start/stop bits.	
6	CCCLR		Continuous clock clear	0
		0	CSCEN is under software control.	
		1	Hardware clears CSCEN after each character is received.	
31:6	-		Reserved. The value read from a reserved bit is not defined.	NA

After reset, synchronous mode is disabled. Synchronous mode is enabled by setting the SYNC bit. When SYNC is 1, the UART operates as follows:

1. The CSRC bit controls whether the UART sends (master mode) or receives (slave mode) a serial bit clock on the SCLK pin.
2. When CSRC is 1 selecting master mode, the CSCEN bit selects whether the UART produces clocks on SCLK continuously (CSCEN=1) or only when transmit data is being sent on TxD (CSCEN=0).

3. The SSDIS bit controls whether start and stop bits are used. When SSDIS is 0, the UART sends and samples for start and stop bits as in other modes. When SSDIS is 1, the UART neither sends nor samples for start or stop bits, and each falling edge on SCLK samples a data bit on RxD into the receive shift register, as well as shifting the transmit shift register.

The rest of this section provides further details of operation when SYNC is 1.

When SSDIS is 0 (start and stop bits are used), the FES bit controls when the UART changes output data and when it samples input data. When FES = 0, output data is sent on the falling edge of SCLK and input data is sampled on the rising edge of SCLK. When FES = 1, output data is sent on the rising edge of SCLK and input data is sampled on the falling edge of SCLK.

When SSDIS is 1 (start and stop bits are not used), FES is ignored and output data is sent on the rising edge of SCLK and input data is sampled on the falling edge of SCLK.

The combination SYNC=1, CSRC=1, CSCEN=1, and SSDIS=1 (synchronous master mode with continuous clock and start and stop disabled) is a difficult operating mode. In this case, SCLK applies to both directions of data flow and there is no defined mechanism to signal the receivers when valid data is present on TxD or RxD.

Lacking such a mechanism, SSDIS=1 can be used with CSCEN=0 or CSRC=0 in a mode similar to the SPI protocol, in which characters are (at least conceptually) “exchanged” between the UART and remote device for each set of 8 clock cycles on SCLK. Such operation can be called full-duplex, but the same hardware mode can be used in a half-duplex way under control of a higher-layer protocol, in which the source of SCLK toggles it in groups of N cycles whenever data is to be sent in either direction. (N being the number of bits/character.)

When the UART4 is the clock source (CSRC=1), such half-duplex operation can lead to the requirement of writing a dummy character to the Transmitter Holding Register in order to generate 8 clocks so that a character can be received. The CCCLR bit provides a more natural way of programming half-duplex reception. When the higher-layer protocol dictates that the UART should receive a character, software should write the SYNCCTRL register with CSCEN=1 and CCCLR=1. After the UART has sent N clock cycles and thus received a character, it clears the CSCEN bit. If more characters need to be received thereafter, software can repeat setting CSCEN and CCCLR.

Aside from such half-duplex operation, the primary use of CSCEN=1 is with SSDIS=0, so that start bits indicate the transmission of each character in each direction.

20.1 Basic configuration

The CAN1/2 peripherals are configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bits PCAN1/2.

   Remark: On reset, the CAN1/2 blocks are disabled (PCAN1/2 = 0).

2. Peripheral clock: The CAN interfaces operate from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.

   Remark: If CAN baud rates above 100 kbit/s (see Table 447) are needed, do not select the IRC as the clock source (see Section 3.11).

3. Wake-up: CAN controllers are able to wake up the microcontroller from Power-down mode, see Section 3.12.8.
4. Pins: Select CAN1/2 pins through and their pin modes through the relevant IOCON registers (Section 7.4.1).
5. Interrupts: CAN interrupts are enabled using the CAN1/2IER registers (Table 446). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. CAN controller initialization: see CANMOD register (Section 20.7.1).

20.2 CAN controllers

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The CAN Controller is designed to provide a full implementation of the CAN-Protocol according to the CAN Specification Version 2.0B. Microcontrollers with this on-chip CAN controller are used to build powerful local networks by supporting distributed real-time control with a very high level of security. The applications are automotive, industrial environments, and high speed networks as well as low cost multiplex wiring. The result is a strongly reduced wiring harness and enhanced diagnostic and supervisory capabilities.

The CAN block is intended to support multiple CAN buses simultaneously, allowing the device to be used as a gateway, switch, or router among a number of CAN buses in various applications.

The CAN module consists of two elements: the controller and the Acceptance Filter. All registers and the RAM are accessed as 32-bit words.

20.3 Features

20.3.1 General CAN features

- Compatible with *CAN specification 2.0B, ISO 11898-1*.
- Multi-master architecture with non destructive bit-wise arbitration.
- Bus access priority determined by the message identifier (11-bit or 29-bit).

- Guaranteed latency time for high priority messages.
- Programmable transfer rate (up to 1 Mbit/s).
- Multicast and broadcast message facility.
- Data length from 0 up to 8 bytes.
- Powerful error handling capability.
- Non-return-to-zero (NRZ) encoding/decoding with bit stuffing.

20.3.2 CAN controller features

- 2 CAN controllers and buses.
- Supports 11-bit identifier as well as 29-bit identifier.
- Double Receive Buffer and Triple Transmit Buffer.
- Programmable Error Warning Limit and Error Counters with read/write access.
- Arbitration Lost Capture and Error Code Capture with detailed bit position.
- Single Shot Transmission (no re-transmission).
- Listen Only Mode (no acknowledge, no active error flags).
- Reception of "own" messages (Self Reception Request).

20.3.3 Acceptance filter features

- Fast hardware implemented search algorithm supporting a large number of CAN identifiers.
- Global Acceptance Filter recognizes 11-bit and 29-bit Rx Identifiers for all CAN buses.
- Allows definition of explicit and groups for 11-bit and 29-bit CAN identifiers.
- Acceptance Filter can provide FullCAN-style automatic reception for selected Standard Identifiers.

20.4 Pin description

Table 435. CAN Pin descriptions

Pin Name	Type	Description
CAN_RD1, CAN_RD2	Input	**Serial Inputs.** From CAN transceivers.
CAN_TD1, CAN_TD2	Output	**Serial Outputs.** To CAN transceivers.

20.5 CAN controller architecture

The CAN Controller is a complete serial interface with both Transmit and Receive Buffers but without Acceptance Filter. CAN Identifier filtering is done for all CAN channels in a separate block (Acceptance Filter). Except for message buffering and acceptance filtering the functionality is similar to the PeliCAN concept.

The CAN Controller Block includes interfaces to the following blocks:

- APB Interface
- Acceptance Filter

- Nested Vectored Interrupt Controller (NVIC)
- CAN Transceiver
- Common Status Registers

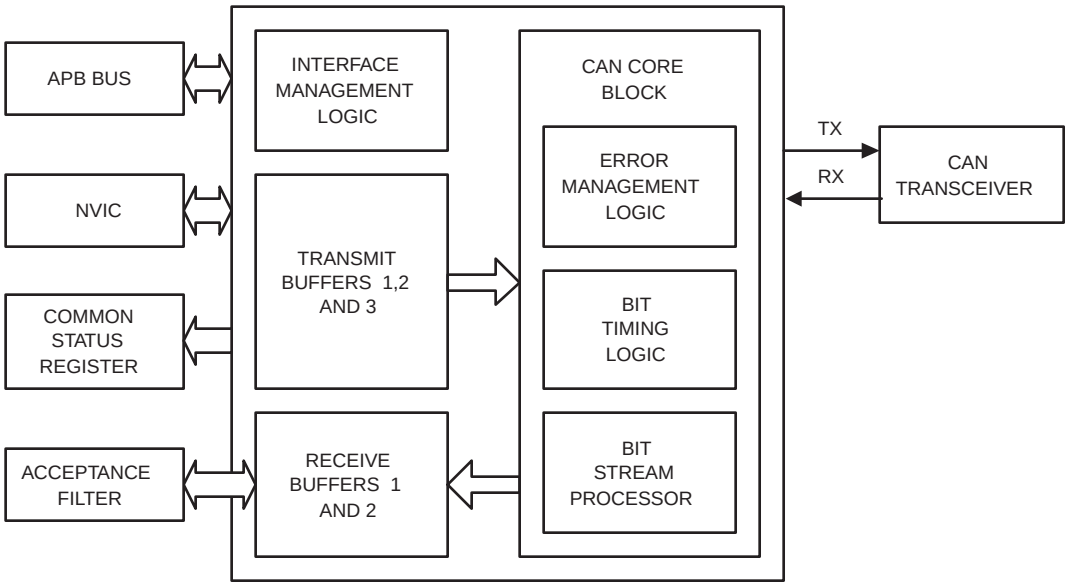


Fig 81. CAN controller block diagram

20.5.1 APB Interface Block (AIB)

The APB Interface Block provides access to all CAN Controller registers.

20.5.2 Interface Management Logic (IML)

The Interface Management Logic interprets commands from the CPU, controls internal addressing of the CAN Registers and provides interrupts and status information to the CPU.

20.5.3 Transmit Buffers (TXB)

The TXB represents a Triple Transmit Buffer, which is the interface between the Interface Management Logic (IML) and the Bit Stream Processor (BSP). Each Transmit Buffer is able to store a complete message which can be transmitted over the CAN network. This buffer is written by the CPU and read out by the BSP.

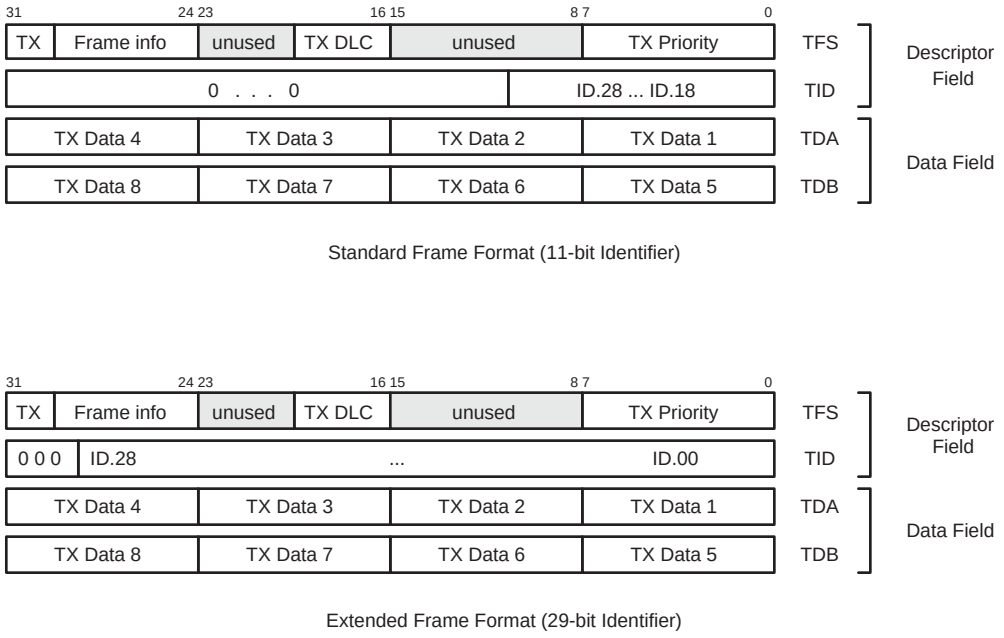


Fig 82. Transmit buffer layout for standard and extended frame format configurations

20.5.4 Receive Buffer (RXB)

The Receive Buffer (RXB) represents a CPU accessible Double Receive Buffer. It is located between the CAN Controller Core Block and APB Interface Block and stores all received messages from the CAN Bus line. With the help of this Double Receive Buffer concept the CPU is able to process one message while another message is being received.

The global layout of the Receive Buffer is very similar to the Transmit Buffer described earlier. Identifier, Frame Format, Remote Transmission Request bit and Data Length Code have the same meaning as described for the Transmit Buffer. In addition, the Receive Buffer includes an ID Index field (see Section 20.7.9.1 “ID index field”).

The received Data Length Code represents the real transmitted Data Length Code, which may be greater than 8 depending on transmitting CAN node. Nevertheless, the maximum number of received data bytes is 8. This should be taken into account by reading a message from the Receive Buffer. If there is not enough space for a new message within the Receive Buffer, the CAN Controller generates a Data Overrun condition when this message becomes valid and the acceptance test was positive. A message that is partly written into the Receive Buffer (when the Data Overrun situation occurs) is deleted. This situation is signalled to the CPU via the Status Register and the Data Overrun Interrupt, if enabled.

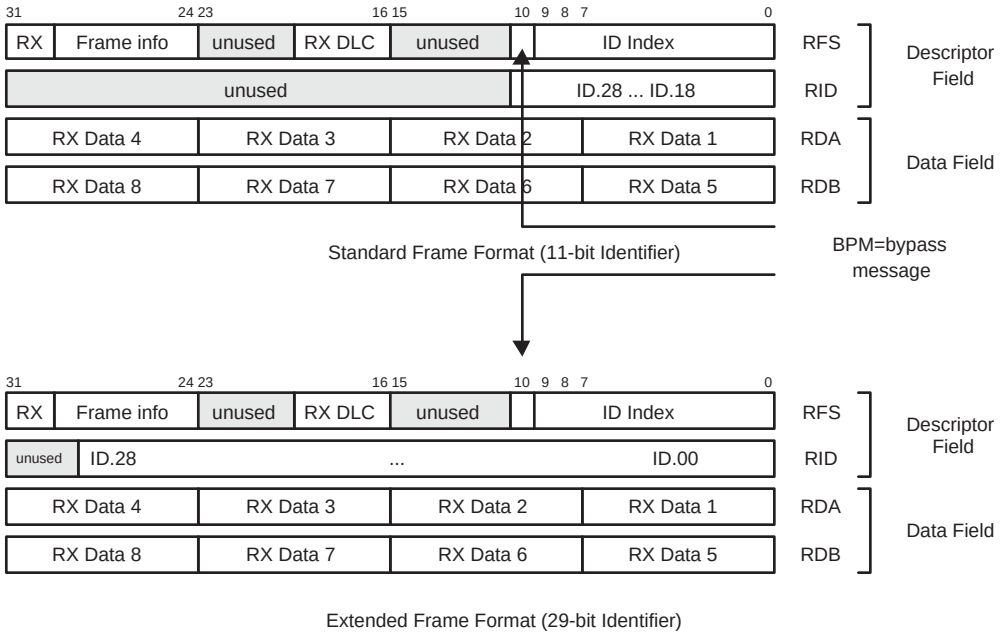


Fig 83. Receive buffer layout for standard and extended frame format configurations

20.5.5 Error Management Logic (EML)

The EML is responsible for the error confinement. It gets error announcements from the BSP and then informs the BSP and IML about error statistics.

20.5.6 Bit Timing Logic (BTL)

The Bit Timing Logic monitors the serial CAN Bus line and handles the Bus line related bit timing. It synchronizes to the bit stream on the CAN Bus on a "recessive" to "dominant" Bus line transition at the beginning of a message (hard synchronization) and re-synchronizes on further transitions during the reception of a message (soft synchronization). The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g. due to oscillator drifts) and to define the sample point and the number of samples to be taken within a bit time.

20.5.7 Bit Stream Processor (BSP)

The Bit Stream Processor is a sequencer, controlling the data stream between the Transmit Buffer, Receive Buffers and the CAN Bus. It also performs the error detection, arbitration, stuffing and error handling on the CAN Bus.

20.5.8 CAN controller self-tests

The CAN controller supports two different options for self-tests:

- Global Self-Test (setting the self reception request bit in normal Operating Mode)
- Local Self-Test (setting the self reception request bit in Self Test Mode)

Both self-tests are using the ‘Self Reception’ feature of the CAN Controller. With the Self Reception Request, the transmitted message is also received and stored in the receive buffer. Therefore the acceptance filter has to be configured accordingly. As soon as the CAN message is transmitted, a transmit and a receive interrupt are generated, if enabled.

Global self test

A Global Self-Test can for example be used to verify the chosen configuration of the CAN Controller in a given CAN system. As shown in Figure 84, at least one other CAN node, which is acknowledging each CAN message has to be connected to the CAN bus.

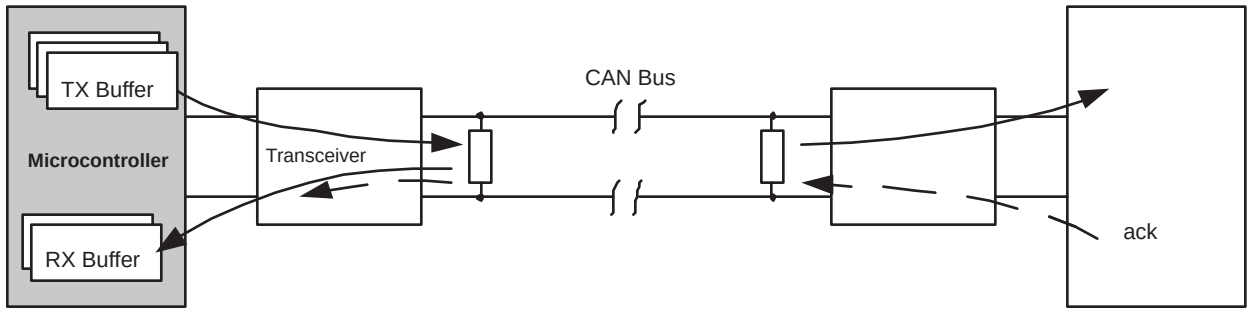


Fig 84. Global Self-Test (high-speed CAN Bus example)

Initiating a Global Self-Test is similar to a normal CAN transmission. In this case the transmission of a CAN message(s) is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer bits (STB3, STB2, STB1) in the CAN Controller Command register (CANCMR).

Local self test

The Local Self-Test perfectly fits for single node tests. In this case an acknowledge from other nodes is not needed. As shown in the Figure below, a CAN transceiver with an appropriate CAN bus termination has to be connected to the device. The CAN Controller has to be put into the 'Self Test Mode' by setting the STM bit in the CAN Controller Mode register (CANMOD). Hint: Setting the Self Test Mode bit (STM) is possible only when the CAN Controller is in Reset Mode.

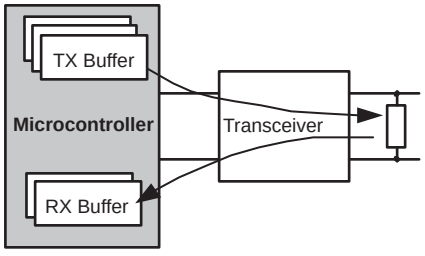


Fig 85. Local self test (high-speed CAN Bus example)

A message transmission is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer(s) (STB3, STB2, STB1).

20.6 Memory map of the CAN block

The CAN Controllers and Acceptance Filter occupy a number of APB slots, as follows:

Table 436. Memory map of the CAN block

Address Range	Used for
0x4003 8000 - 0x4003 87FF	Acceptance Filter RAM.
0x4003 C000 - 0x4003 C017	Acceptance Filter Registers.
0x4004 0000 - 0x4004 000B	Central CAN Registers.
0x4004 4000 - 0x4004 405F	CAN Controller 1 Registers.
0x4004 8000 - 0x4004 805F	CAN Controller 2 Registers.
0x400F C110 - 0x400F C114	CAN Wake and Sleep Registers.

20.7 Register description

CAN block implements the registers at several base addresses.

Table 437. Register overview: CAN acceptance filter (base address 0x4003 C000)

Name	Access	Address offset	Description	Reset Value	Table
AFMR	R/W	0x000	Acceptance Filter Register	1	465
SFF_SA	R/W	0x004	Standard Frame Individual Start Address Register	0	466
SFF_GRP_SA	R/W	0x008	Standard Frame Group Start Address Register	0	467
EFF_SA	R/W	0x00C	Extended Frame Start Address Register	0	468
EFF_GRP_SA	R/W	0x010	Extended Frame Group Start Address Register	0	469
ENDOFTABLE	R/W	0x014	End of AF Tables register	0	470
LUTERRAD	RO	0x018	LUT Error Address register	0	471
LUTERR	RO	0x01C	LUT Error Register	0	472
FCANIE	R/W	0x020	FullCAN interrupt enable register	0	473
FCANIC0	R/W	0x024	FullCAN interrupt and capture register0	0	474
FCANIC1	R/W	0x028	FullCAN interrupt and capture register1	0	475

Table 438. Register overview: central CAN (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset Value	Table
TXSR	RO	0x000	CAN Central Transmit Status Register	0x0003 0300	460
RXSR	RO	0x004	CAN Central Receive Status Register	0	461
MSR	RO	0x008	CAN Central Miscellaneous Register	0	462

Table 439. Register overview: CAN (base address 0x4004 4000 (CAN1) and 0x4004 8000 (CAN2))

Generic Name	Access	Address offset	Description	Reset value	Table
MOD	R/W	0x000	Controls the operating mode of the CAN Controller.	1	442
CMR	WO	0x004	Command bits that affect the state of the CAN Controller	0	443
GSR	RO	0x008	Global Controller Status and Error Counters. The error counters can only be written when RM in CANMOD is 1.	0x3C	444

Table 439. Register overview: CAN (base address 0x4004 4000 (CAN1) and 0x4004 8000 (CAN2))

Generic Name	Access	Address offset	Description	Reset value	Table
ICR	RO	0x00C	Interrupt status, Arbitration Lost Capture, Error Code Capture	0	445
IER	R/W	0x010	Interrupt Enable	0	446
BTR	R/W	0x014	Bus Timing. Can only be written when RM in CANMOD is 1.	0x1C0000	447
EWL	R/W	0x018	Error Warning Limit. Can only be written when RM in CANMOD is 1.	0x60	448
SR	RO	0x01C	Status Register	0x3C3C3C	449
RFS	R/W	0x020	Receive frame status. Can only be written when RM in CANMOD is 1.	0	450
RID	R/W	0x024	Received Identifier. Can only be written when RM in CANMOD is 1.	0	451
RDA	R/W	0x028	Received data bytes 1-4. Can only be written when RM in CANMOD is 1.	0	453
RDB	R/W	0x02C	Received data bytes 5-8. Can only be written when RM in CANMOD is 1.	0	454
TFI1	R/W	0x030	Transmit frame info (Tx Buffer 1)	0	455
TID1	R/W	0x034	Transmit Identifier (Tx Buffer 1)	0	456
TDA1	R/W	0x038	Transmit data bytes 1-4 (Tx Buffer 1)	0	458
TDB1	R/W	0x03C	Transmit data bytes 5-8 (Tx Buffer 1)	0	459
TFI2	R/W	0x040	Transmit frame info (Tx Buffer 2)	0	455
TID2	R/W	0x044	Transmit Identifier (Tx Buffer 2)	0	456
TDA2	R/W	0x048	Transmit data bytes 1-4 (Tx Buffer 2)	0	458
TDB2	R/W	0x04C	Transmit data bytes 5-8 (Tx Buffer 2)	0	459
TFI3	R/W	0x050	Transmit frame info (Tx Buffer 3)	0	455
TID3	R/W	0x054	Transmit Identifier (Tx Buffer 3)	0	456
TDA3	R/W	0x058	Transmit data bytes 1-4 (Tx Buffer 3)	0	458
TDB3	R/W	0x05C	Transmit data bytes 5-8 (Tx Buffer 3)	0	459

The internal registers of each CAN Controller appear to the CPU as on-chip memory mapped peripheral registers. Because the CAN Controller can operate in different modes (Operating/Reset, see also Section 20.7.1 “CAN Mode register”), one has to distinguish between different internal address definitions. Note that write access to some registers is only allowed in Reset Mode.

Table 440. CAN1 and CAN2 controller register summary

Generic Name	Operating Mode		Reset Mode	
	Read	Write	Read	Write
MOD	Mode	Mode	Mode	Mode
CMR	0x00	Command	0x00	Command
GSR	Global Status and Error Counters	-	Global Status and Error Counters	Error Counters only
ICR	Interrupt and Capture	-	Interrupt and Capture	-
IER	Interrupt Enable	Interrupt Enable	Interrupt Enable	Interrupt Enable
BTR	Bus Timing	-	Bus Timing	Bus Timing
EWL	Error Warning Limit	-	Error Warning Limit	Error Warning Limit

Table 440. CAN1 and CAN2 controller register summary

Generic Name	Operating Mode		Reset Mode	
	Read	Write	Read	Write
SR	Status	-	Status	-
RFS	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
RID	Rx Identifier	-	Rx Identifier	Rx Identifier
RDA	Rx Data	-	Rx Data	Rx Data
RDB	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
TFI1	Tx Info1	Tx Info	Tx Info	Tx Info
TID1	Tx Identifier	Tx Identifier	Tx Identifier	Tx Identifier
TDA1	Tx Data	Tx Data	Tx Data	Tx Data
TDB1	Tx Data	Tx Data	Tx Data	Tx Data

Table 441. Register overview: CAN Wake and Sleep (base address 0x400F C000)

Name	Access	Address offset	Description	Reset Value	Table
CANSLEEPCLR	R/W	0x110	Allows clearing the current CAN channel sleep state as well as reading that state.	0	36
CANWAKEFLAGS	R/W	0x114	Allows reading the wake-up state of the CAN channels.	0	37

In the following register tables, the column “Reset Value” shows how a hardware reset affects each bit or field, while the column “RM Set” indicates how each bit or field is affected if software sets the RM bit, or RM is set because of a Bus-Off condition. Note that while hardware reset sets RM, in this case the setting noted in the “Reset Value” column prevails over that shown in the “RM Set” column, in the few bits where they differ. In both columns, X indicates the bit or field is unchanged.

20.7.1 CAN Mode register

The contents of the Mode Register are used to change the behavior of the CAN Controller. Bits may be set or reset by the CPU that uses the Mode Register as a read/write memory.

The following restrictions apply to using the bits in this register:

- During a Hardware reset or when the Bus Status bit is set '1' (Bus-Off), the Reset Mode bit is set '1' (present). After the Reset Mode bit is set '0' the CAN Controller will wait for:

  - one occurrence of Bus-Free signal (11 recessive bits), if the preceding reset has been caused by a Hardware reset or a CPU-initiated reset.

  - 128 occurrences of Bus-Free, if the preceding reset has been caused by a CAN Controller initiated Bus-Off, before re-entering the Bus-On mode.
- This mode of operation forces the CAN Controller to be error passive. Message Transmission is not possible. The Listen Only Mode can be used e.g. for software driven bit rate detection and "hot plugging".
- A write access to the bits MOD.1 and MOD.2 is possible only if the Reset Mode is entered previously.

- Transmit Priority Mode is explained in more detail in Section 20.5.3 “Transmit Buffers (TXB)”.
- The CAN Controller will enter Sleep Mode, if the Sleep Mode bit is set '1' (sleep), there is no bus activity, and none of the CAN interrupts is pending. Setting of SM with at least one of the previously mentioned exceptions valid will result in a wake-up interrupt. The CAN Controller will wake up if SM is set LOW (wake-up) or there is bus activity. On wake-up, a Wake-up Interrupt is generated. A sleeping CAN Controller which wakes up due to bus activity will not be able to receive this message until it detects 11 consecutive recessive bits (Bus-Free sequence). Note that setting of SM is not possible in Reset Mode. After clearing of Reset Mode, setting of SM is possible only when Bus-Free is detected again.
- The LOM and STM bits can only be written if the RM bit is 1 prior to the write operation.

Table 442. CAN Mode register (CAN1MOD - address 0x4004 4000, CAN2MOD - address 0x4004 8000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RM		Reset Mode.	1	1
		0	Normal.The CAN Controller is in the Operating Mode, and certain registers can not be written.		
		1	Reset. CAN operation is disabled, writable registers can be written and the current transmission/reception of a message is aborted.		
1	LOM		Listen Only Mode.	0	x
		0	Normal. The CAN controller acknowledges a successfully received message on the CAN bus. The error counters are stopped at the current value.		
		1	Listen only. The controller gives no acknowledgment, even if a message is successfully received. Messages cannot be sent, and the controller operates in “error passive” mode. This mode is intended for software bit rate detection and “hot plugging”.		
2	STM		Self Test Mode.	0	x
		0	Normal. A transmitted message must be acknowledged to be considered successful.		
		1	Self test. The controller will consider a Tx message successful even if there is no acknowledgment received. In this mode a full node test is possible without any other active node on the bus using the SRR bit in CANxCMR.		
3	TPM		Transmit Priority Mode.	0	x
		0	CAN ID. The transmit priority for 3 Transmit Buffers depends on the CAN Identifier.		
		1	Local priority. The transmit priority for 3 Transmit Buffers depends on the contents of the Tx Priority register within the Transmit Buffer.		
4	SM		Sleep Mode.	0	0
		0	Wake-up. Normal operation.		
		1	Sleep. The CAN controller enters Sleep Mode if no CAN interrupt is pending and there is no bus activity. See the Sleep Mode description Section 20.8.2 on page 570.		
5	RPM		Receive Polarity Mode.	0	x
		0	Low active. RD input is active Low (dominant bit = 0).		
		1	High active. RD input is active High (dominant bit = 1) -- reverse polarity.		

Table 442. CAN Mode register (CAN1MOD - address 0x4004 4000, CAN2MOD - address 0x4004 8000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
6	-		Reserved. Read value is undefined, only zero should be written.	0	0
7	TM		Test Mode.	0	x
		0	Disabled. Normal operation.		
		1	Enabled. The TD pin will reflect the bit, detected on RD pin, with the next positive edge of the system clock.		
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA	

20.7.2 CAN Command Register

Writing to this write-only register initiates an action within the transfer layer of the CAN Controller. Reading this register yields zeroes.

At least one internal clock cycle is needed for processing between two commands.

The following restrictions apply to using the bits in this register:

- Setting the command bits TR and AT simultaneously results in transmitting a message once. No re-transmission will be performed in case of an error or arbitration lost (single shot transmission).
- Setting the command bits SRR and TR simultaneously results in sending the transmit message once using the self-reception feature. No re-transmission will be performed in case of an error or arbitration lost.
- Setting the command bits TR, AT and SRR simultaneously results in transmitting a message once as described for TR and AT. The moment the Transmit Status bit is set within the Status Register, the internal Transmission Request Bit is cleared automatically.
- Setting TR and SRR simultaneously will ignore the set SRR bit.
- If the Transmission Request or the Self-Reception Request bit was set '1' in a previous command, it cannot be cancelled by resetting the bits. The requested transmission may only be cancelled by setting the Abort Transmission bit.
- The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit a more urgent message before. A transmission already in progress is not stopped. In order to see if the original message has been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. This should be done after the Transmit Buffer Status bit has been set to '1' or a Transmit Interrupt has been generated.
- After reading the contents of the Receive Buffer, the CPU can release this memory space by setting the Release Receive Buffer bit '1'. This may result in another message becoming immediately available. If there is no other message available, the Receive Interrupt bit is reset. If the RRB command is given, it will take at least 2 internal clock cycles before a new interrupt is generated.
- This command bit is used to clear the Data Overrun condition signalled by the Data Overrun Status bit. As long as the Data Overrun Status bit is set no further Data Overrun Interrupt is generated.

- Upon Self Reception Request, a message is transmitted and simultaneously received if the Acceptance Filter is set to the corresponding identifier. A receive and a transmit interrupt will indicate correct self reception (see also Self Test Mode in Section 20.7.1 "CAN Mode register").

Table 443. CAN Command Register (CAN1CMR - address 0x4004 4004, CAN2CMR - address 0x4004 8004) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	TR		Transmission Request.	0	0
		0	Absent.No transmission request.		
		1	Present. The message, previously written to the CANxTFI, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer. If at two or all three of STB1, STB2 and STB3 bits are selected when TR=1 is written, Transmit Buffer will be selected based on the chosen priority scheme (for details see Section 20.5.3 "Transmit Buffers (TXB)")		
1	AT		Abort Transmission.	0	0
		0	No action. Do not abort the transmission.		
		1	Present. if not already in progress, a pending Transmission Request for the selected Transmit Buffer is cancelled.		
2	RRB		Release Receive Buffer.	0	0
		0	No action. Do not release the receive buffer.		
		1	Released. The information in the Receive Buffer (consisting of CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers) is released, and becomes eligible for replacement by the next received frame. If the next received frame is not available, writing this command clears the RBS bit in the Status Register(s).		
3	CDO		Clear Data Overrun.	0	0
		0	No action. Do not clear the data overrun bit.		
		1	Clear. The Data Overrun bit in Status Register(s) is cleared.		
4	SRR		Self Reception Request.	0	0
		0	Absent. No self reception request.		
		1	Present. The message, previously written to the CANxTFS, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer and received simultaneously. This differs from the TR bit above in that the receiver is not disabled during the transmission, so that it receives the message if its Identifier is recognized by the Acceptance Filter.		
5	STB1		Select Tx Buffer 1.	0	0
		0	Not selected. Tx Buffer 1 is not selected for transmission.		
		1	Selected. Tx Buffer 1 is selected for transmission.		
6	STB2		Select Tx Buffer 2.	0	0
		0	Not selected. Tx Buffer 2 is not selected for transmission.		
		1	Selected. Tx Buffer 2 is selected for transmission.		
7	STB3		Select Tx Buffer 3.	0	0
		0	Not selected. Tx Buffer 3 is not selected for transmission.		
		1	Selected. Tx Buffer 3 is selected for transmission.		
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA	

20.7.3 CAN Global Status Register

The content of the Global Status Register reflects the status of the CAN Controller. This register is read-only, except that the Error Counters can be written when the RM bit in the CANMOD register is 1. Bits not listed read as 0 and should be written as 0.

Table 444. CAN Global Status Register (CAN1GSR - address 0x4004 4008, CAN2GSR - address 0x4004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS		Receive Buffer Status. After reading all messages and releasing their memory space with the command 'Release Receive Buffer,' this bit is cleared.	0	0
		0	Empty. No message is available.		
		1	Full. At least one complete message is received by the Double Receive Buffer and available in the CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers. This bit is cleared by the Release Receive Buffer command in CANxCMR, if no subsequent received message is available.		
1	DOS		Data Overrun Status. If there is not enough space to store the message within the Receive Buffer, that message is dropped and the Data Overrun condition is signalled to the CPU in the moment this message becomes valid. If this message is not completed successfully (e.g. because of an error), no overrun condition is signalled.	0	0
		0	Absent. No data overrun has occurred since the last Clear Data Overrun command was given/written to CANxCMR (or since Reset).		
		1	Overrun. A message was lost because the preceding message to this CAN controller was not read and released quickly enough (there was not enough space for a new message in the Double Receive Buffer).		
2	TBS		Transmit Buffer Status.	1	1
		0	Locked. At least one of the Transmit Buffers is not available for the CPU, i.e. at least one previously queued message for this CAN controller has not yet been sent, and therefore software should not write to the CANxTFI, CANxTID, CANxTDA, nor CANxTDB registers of that (those) Tx buffer(s).		
		1	Released. All three Transmit Buffers are available for the CPU. No transmit message is pending for this CAN controller (in any of the 3 Tx buffers), and software may write to any of the CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS		Transmit Complete Status. The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' at least for one of the three Transmit Buffers. The Transmission Complete Status bit will remain '0' until all messages are transmitted successfully.	1	x
		0	Incomplete. At least one requested transmission has not been successfully completed yet.		
		1	Complete. All requested transmission(s) has (have) been successfully completed.		
4	RS		Receive Status. If both the Receive Status and the Transmit Status bits are '0' (idle), the CAN-Bus is idle. If both bits are set, the controller is waiting to become idle again. After hardware reset 11 consecutive recessive bits have to be detected until idle status is reached. After Bus-off this will take 128 times of 11 consecutive recessive bits.	1	0
		0	Idle. The CAN controller is idle.		
		1	Receive. The CAN controller is receiving a message.		

Table 444. CAN Global Status Register (CAN1GSR - address 0x4004 4008, CAN2GSR - address 0x4004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	TS		Transmit Status. If both the Receive Status and the Transmit Status bits are '0' (idle), the CAN-Bus is idle. If both bits are set, the controller is waiting to become idle again. After hardware reset 11 consecutive recessive bits have to be detected until idle status is reached. After Bus-off this will take 128 times of 11 consecutive recessive bits.	1	0
		0	Idle. The CAN controller is idle.		
		1	Transmit. The CAN controller is sending a message.		
6	ES		Error Status. Errors detected during reception or transmission will effect the error counters according to the CAN specification. The Error Status bit is set when at least one of the error counters has reached or exceeded the Error Warning Limit. An Error Warning Interrupt is generated, if enabled. The default value of the Error Warning Limit after hardware reset is 96 decimal, see also Section 20.7.7 “CAN Error Warning Limit register”.	0	0
		0	OK. Both error counters are below the Error Warning Limit.		
		1	Error. One or both of the Transmit and Receive Error Counters has reached the limit set in the Error Warning Limit register.		
7	BS		Bus Status. Mode bit '1' (present) and an Error Warning Interrupt is generated, if enabled. Afterwards the Transmit Error Counter is set to '127', and the Receive Error Counter is cleared. It will stay in this mode until the CPU clears the Reset Mode bit. Once this is completed the CAN Controller will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) counting down the Transmit Error Counter. After that, the Bus Status bit is cleared (Bus-On), the Error Status bit is set '0' (ok), the Error Counters are reset, and an Error Warning Interrupt is generated, if enabled. Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery.	0	0
		0	Bus-on. The CAN Controller is involved in bus activities		
		1	Bus-off. The CAN controller is currently not involved/prohibited from bus activity because the Transmit Error Counter reached its limiting value of 255.		
15:8	-		Reserved. Read value is undefined, only zero should be written.	NA	
23:16	RXERR	-	The current value of the Rx Error Counter (an 8-bit value).	0	X
31:24	TXERR	-	The current value of the Tx Error Counter (an 8-bit value).	0	X

RX error counter

The RX Error Counter Register, which is part of the Status Register, reflects the current value of the Receive Error Counter. After hardware reset this register is initialized to 0. In Operating Mode this register appears to the CPU as a read-only memory. A write access to this register is possible only in Reset Mode. If a Bus Off event occurs, the RX Error Counter is initialized to 0. As long as Bus Off is valid, writing to this register has no effect.The Rx Error Counter is determined as follows:

RX Error Counter = (CANxGSR AND 0x00FF0000) / 0x00010000

Note that a CPU-forced content change of the RX Error Counter is possible only if the Reset Mode was entered previously. An Error Status change (Status Register), an Error Warning or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

TX error counter

The TX Error Counter Register, which is part of the Status Register, reflects the current value of the Transmit Error Counter. In Operating Mode this register appears to the CPU as a read-only memory. After hardware reset this register is initialized to 0. A write access to this register is possible only in Reset Mode. If a bus-off event occurs, the TX Error Counter is initialized to 127 to count the minimum protocol-defined time (128 occurrences of the Bus-Free signal). Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery. If Bus Off is active, a write access to TXERR in the range of 0 to 254 clears the Bus Off Flag and the controller will wait for one occurrence of 11 consecutive recessive bits (bus free) after clearing of Reset Mode. The Tx error counter is determined as follows:

TX Error Counter = (CANxGSR AND 0xFF000000) / 0x01000000

Writing 255 to TXERR allows initiation of a CPU-driven Bus Off event. Note that a CPU-forced content change of the TX Error Counter is possible only if the Reset Mode was entered previously. An Error or Bus Status change (Status Register), an Error Warning, or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again. After leaving the Reset Mode, the new TX Counter content is interpreted and the Bus Off event is performed in the same way as if it was forced by a bus error event. That means, that the Reset Mode is entered again, the TX Error Counter is initialized to 127, the RX Counter is cleared, and all concerned Status and Interrupt Register bits are set. Clearing of Reset Mode now will perform the protocol defined Bus Off recovery sequence (waiting for 128 occurrences of the Bus-Free signal). If the Reset Mode is entered again before the end of Bus Off recovery (TXERR>0), Bus Off keeps active and TXERR is frozen.

20.7.4 CAN Interrupt and Capture Register

Bits in this register indicate information about events on the CAN bus. This register is read-only.

The Interrupt flags of the Interrupt and Capture Register allow the identification of an interrupt source. When one or more bits are set, a CAN interrupt will be indicated to the CPU. After this register is read from the CPU all interrupt bits are reset **except** of the Receive Interrupt bit. The Interrupt Register appears to the CPU as a read-only memory.

Bits 1 through 10 clear when they are read.

Bits 16-23 are captured when a bus error occurs. At the same time, if the BEIE bit in CANIER is 1, the BEI bit in this register is set, and a CAN interrupt can occur.

Bits 24-31 are captured when CAN arbitration is lost. At the same time, if the ALIE bit in CANIER is 1, the ALI bit in this register is set, and a CAN interrupt can occur. Once either of these bytes is captured, its value will remain the same until it is read, at which time it is released to capture a new value.

The clearing of bits 1 to 10 and the releasing of bits 16-23 and 24-31 all occur on any read from CANxICR, regardless of whether part or all of the register is read. This means that software should always read CANxICR as a word, and process and deal with all bits of the register as appropriate for the application.

Table 445. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RI		Receive Interrupt. This bit is set whenever the RBS bit in CANxSR and the RIE bit in CANxIER are both 1, indicating that a new message was received and stored in the Receive Buffer. The Receive Interrupt Bit is not cleared upon a read access to the Interrupt Register. Giving the Command “Release Receive Buffer” will clear RI temporarily. If there is another message available within the Receive Buffer after the release command, RI is set again. Otherwise RI remains cleared.	0	0
		0	Reset		
		1	Set		
1	TI1		Transmit Interrupt 1. This bit is set when the TBS1 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB1 was successfully transmitted or aborted), indicating that Transmit buffer 1 is available, and the TIE1 bit in CANxIER is 1.	0	0
		0	Reset		
		1	Set		
2	EI		Error Warning Interrupt. This bit is set on every change (set or clear) of either the Error Status or Bus Status bit in CANxSR and the EIE bit is set within the Interrupt Enable Register at the time of the change.	0	X
		0	Reset		
		1	Set		
3	DOI		Data Overrun Interrupt. This bit is set when the DOS bit in CANxSR goes from 0 to 1 and the DOIE bit in CANxIER is 1.	0	0
		0	Reset		
		1	Set		
4	WUI		Wake-Up Interrupt. This bit is set if the CAN controller is sleeping and bus activity is detected and the WUIE bit in CANxIER is 1. A Wake-Up Interrupt is also generated if the CPU tries to set the Sleep bit while the CAN controller is involved in bus activities or a CAN Interrupt is pending. The WUI flag can also get asserted when the according enable bit WUIE is not set. In this case a Wake-Up Interrupt does not get asserted.	0	0
		0	Reset		
		1	Set		
5	EPI		Error Passive Interrupt. This bit is set if the EPIE bit in CANxIER is 1, and the CAN controller switches between Error Passive and Error Active mode in either direction. This is the case when the CAN Controller has reached the Error Passive Status (at least one error counter exceeds the CAN protocol defined level of 127) or if the CAN Controller is in Error Passive Status and enters the Error Active Status again.	0	0
		0	Reset		
		1	Set		
6	ALI		Arbitration Lost Interrupt. This bit is set if the ALIE bit in CANxIER is 1, and the CAN controller loses arbitration while attempting to transmit. In this case the CAN node becomes a receiver.	0	0
		0	Reset		
		1	Set		

Table 445. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description *…continued*

Bit	Symbol	Value	Function	Reset Value	RM Set
7	BEI		Bus Error Interrupt -- this bit is set if the BEIE bit in CANxIER is 1, and the CAN controller detects an error on the bus.	0	X
		0	Reset		
		1	Set		
8	IDI		ID Ready Interrupt -- this bit is set if the IDIE bit in CANxIER is 1, and a CAN Identifier has been received (a message was successfully transmitted or aborted). This bit is set whenever a message was successfully transmitted or aborted and the IDIE bit is set in the IER register.	0	0
		0	Reset		
		1	Set		
9	TI2		Transmit Interrupt 2. This bit is set when the TBS2 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB2 was successfully transmitted or aborted), indicating that Transmit buffer 2 is available, and the TIE2 bit in CANxIER is 1.	0	0
		0	Reset		
		1	Set		
10	TI3		Transmit Interrupt 3. This bit is set when the TBS3 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB3 was successfully transmitted or aborted), indicating that Transmit buffer 3 is available, and the TIE3 bit in CANxIER is 1.	0	0
		0	Reset		
		1	Set		
15:11	-		Reserved. The value read from a reserved bit is not defined.	0	0

Table 445. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description *…continued*

Bit	Symbol	Value	Function	Reset Value	RM Set
20:16	ERRBIT4_0		Error Code Capture: when the CAN controller detects a bus error, the location of the error within the frame is captured in this field. The value reflects an internal state variable, and as a result is not very linear: 00011 = Start of Frame 00010 = ID28 ... ID21 00110 = ID20 ... ID18 00100 = SRTR Bit 00101 = IDE bit 00111 = ID17 ... 13 01111 = ID12 ... ID5 01110 = ID4 ... ID0 01100 = RTR Bit 01101 = Reserved Bit 1 01001 = Reserved Bit 0 01011 = Data Length Code 01010 = Data Field 01000 = CRC Sequence 11000 = CRC Delimiter 11001 = Acknowledge Slot 11011 = Acknowledge Delimiter 11010 = End of Frame 10010 = Intermission Whenever a bus error occurs, the corresponding bus error interrupt is forced, if enabled. At the same time, the current position of the Bit Stream Processor is captured into the Error Code Capture Register. The content within this register is fixed until the user software has read out its content once. From now on, the capture mechanism is activated again, i.e. reading the CANxICR enables another Bus Error Interrupt.	0	X
21	ERRDIR		When the CAN controller detects a bus error, the direction of the current bit is captured in this bit.	0	X
		0	Error occurred during transmitting.		
		1	Error occurred during receiving.		

Table 445. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description *…continued*

Bit	Symbol	Value	Function	Reset Value	RM Set
23:22	ERRC1:0		When the CAN controller detects a bus error, the type of error is captured in this field:	0	X
		0x0	Bit error		
		0x1	Form error		
		0x2	Stuff error		
		0x3	Other error		
31:24	ALCBIT	-	Each time arbitration is lost while trying to send on the CAN, the bit number within the frame is captured into this field. After the content of ALCBIT is read, the ALI bit is cleared and a new Arbitration Lost interrupt can occur. 00 = arbitration lost in the first bit (MS) of identifier ... 11 = arbitration lost in SRTS bit (RTR bit for standard frame messages) 12 = arbitration lost in IDE bit 13 = arbitration lost in 12th bit of identifier (extended frame only) ... 30 = arbitration lost in last bit of identifier (extended frame only) 31 = arbitration lost in RTR bit (extended frame only) On arbitration lost, the corresponding arbitration lost interrupt is forced, if enabled. At that time, the current bit position of the Bit Stream Processor is captured into the Arbitration Lost Capture Register. The content within this register is fixed until the user application has read out its contents once. From now on, the capture mechanism is activated again.	0	X

20.7.5 CAN Interrupt Enable Register

This read/write register controls whether various events on the CAN controller will result in an interrupt or not. Bits 10:0 in this register correspond 1-to-1 with bits 10:0 in the CANxICR register. If a bit in the CANxIER register is 0 the corresponding interrupt is disabled; if a bit in the CANxIER register is 1 the corresponding source is enabled to trigger an interrupt.

Table 446. CAN Interrupt Enable Register (CAN1IER - address 0x4004 4010, CAN2IER - address 0x4004 8010) bit description

Bit	Symbol	Function	Reset Value	RM Set
0	RIE	Receiver Interrupt Enable. When the Receive Buffer Status is 'full', the CAN Controller requests the respective interrupt.	0	X
1	TIE1	Transmit Interrupt Enable for Buffer1. When a message has been successfully transmitted out of TXB1 or Transmit Buffer 1 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
2	EIE	Error Warning Interrupt Enable. If the Error or Bus Status change (see Status Register), the CAN Controller requests the respective interrupt.	0	X
3	DOIE	Data Overrun Interrupt Enable. If the Data Overrun Status bit is set (see Status Register), the CAN Controller requests the respective interrupt.	0	X
4	WUIE	Wake-Up Interrupt Enable. If the sleeping CAN controller wakes up, the respective interrupt is requested.	0	X

Table 446. CAN Interrupt Enable Register (CAN1IER - address 0x4004 4010, CAN2IER - address 0x4004 8010) bit description

Bit	Symbol	Function	Reset Value	RM Set
5	EPIE	Error Passive Interrupt Enable. If the error status of the CAN Controller changes from error active to error passive or vice versa, the respective interrupt is requested.	0	X
6	ALIE	Arbitration Lost Interrupt Enable. If the CAN Controller has lost arbitration, the respective interrupt is requested.	0	X
7	BEIE	Bus Error Interrupt Enable. If a bus error has been detected, the CAN Controller requests the respective interrupt.	0	X
8	IDIE	ID Ready Interrupt Enable. When a CAN identifier has been received, the CAN Controller requests the respective interrupt.	0	X
9	TIE2	Transmit Interrupt Enable for Buffer2. When a message has been successfully transmitted out of TXB2 or Transmit Buffer 2 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
10	TIE3	Transmit Interrupt Enable for Buffer3. When a message has been successfully transmitted out of TXB3 or Transmit Buffer 3 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA	

20.7.6 CAN Bus Timing Register

This register controls how various CAN timings are derived from the APB clock. It defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). Furthermore, it defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. It can be read at any time but can only be written if the RM bit in CANmod is 1.

Table 447. CAN Bus Timing Register (CAN1BTR - address 0x4004 4014, CAN2BTR - address 0x4004 8014) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
9:0	BRP		Baud Rate Prescaler. The APB clock is divided by (this value plus one) to produce the CAN clock.	0	X
13:10	-		Reserved. Read value is undefined, only zero should be written.	NA	
15:14	SJW		The Synchronization Jump Width is (this value plus one) CAN clocks.	0	X
19:16	TESG1		The delay from the nominal Sync point to the sample point is (this value plus one) CAN clocks.	1100	X
22:20	TESG2		The delay from the sample point to the next nominal sync point is (this value plus one) CAN clocks. The nominal CAN bit time is (this value plus the value in TSEG1 plus 3) CAN clocks.	001	X
23	SAM		Sampling		
		0	The bus is sampled once (recommended for high speed buses)	0	X
		1	The bus is sampled 3 times (recommended for low to medium speed buses to filter spikes on the bus-line)		
31:24	-		Reserved. Read value is undefined, only zero should be written.	NA	

Baud rate prescaler

The period of the CAN system clock t_{SCL} is programmable and determines the individual bit timing. The CAN system clock t_{SCL} is calculated using the following equation:

(7)

$$t_{SCL} = t_{CANsuppliedCLK} \times (BRP + 1)$$

Synchronization jump width

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width t_{SJW} defines the maximum number of clock cycles a certain bit period may be shortened or lengthened by one re-synchronization:

(8)

$$t_{SJW} = t_{SCL} \times (SJW + 1)$$

Time segment 1 and time segment 2

Time segments TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

(9)

$$t_{SYNCSEG} = t_{SCL}$$

(10)

$$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$$

(11)

$$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$$

20.7.7 CAN Error Warning Limit register

This register sets a limit on Tx or Rx errors at which an interrupt can occur. It can be read at any time but can only be written if the RM bit in CANmod is 1.

Table 448. CAN Error Warning Limit register (CAN1EWL - address 0x4004 4018, CAN2EWL - address 0x4004 8018) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	EWL	During CAN operation, this value is compared to both the Tx and Rx Error Counters. If either of these counter matches this value, the Error Status (ES) bit in CANSR is set.	96_{10} = 0x60	X
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA	

Note that a content change of the Error Warning Limit Register is possible only if the Reset Mode was entered previously. An Error Status change (Status Register) and an Error Warning Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

20.7.8 CAN Status Register

This read-only register contains three status bytes in which the bits not related to transmission are identical to the corresponding bits in the Global Status Register, while those relating to transmission reflect the status of each of the 3 Tx Buffers.

The following restriction apply to using the bits in this register:

- If the CPU tries to write to this Transmit Buffer when the Transmit Buffer Status bit is '0' (locked), the written byte is not accepted and is lost without this being signalled.
- The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' for this TX buffer. The Transmission Complete Status bit remains '0' until a message is transmitted successfully.

Table 449. CAN Status Register (CAN1SR - address 0x4004 401C, CAN2SR - address 0x4004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS_1		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
1	DOS_1		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
2	TBS1_1		Transmit Buffer Status 1.	1	1
		0	Locked. Software cannot access the Tx Buffer 1 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1	Released. Software may write a message into the Transmit Buffer 1 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS1_1		Transmission Complete Status.	1	x
		0	Incomplete. The previously requested transmission for Tx Buffer 1 is not complete.		
		1	Complete. The previously requested transmission for Tx Buffer 1 has been successfully completed.		
4	RS_1		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
5	TS1_1		Transmit Status 1.	1	0
		0	Idle. There is no transmission from Tx Buffer 1.		
		1	Transmit. The CAN Controller is transmitting a message from Tx Buffer 1.		
6	ES_1		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
7	BS_1		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
8	RBS_2		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
9	DOS_2		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
10	TBS2_2		Transmit Buffer Status 2.	1	1
		0	Locked. Software cannot access the Tx Buffer 2 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1	Released. Software may write a message into the Transmit Buffer 2 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
11	TCS2_2		Transmission Complete Status.	1	x
		0	Incomplete. The previously requested transmission for Tx Buffer 2 is not complete.		
		1	Complete. The previously requested transmission for Tx Buffer 2 has been successfully completed.		
12	RS_2		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
13	TS2_2		Transmit Status 2.	1	0
		0	Idle. There is no transmission from Tx Buffer 2.		
		1	Transmit. The CAN Controller is transmitting a message from Tx Buffer 2.		
14	ES_2		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0

Table 449. CAN Status Register (CAN1SR - address 0x4004 401C, CAN2SR - address 0x4004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
15	BS_2		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
16	RBS_3		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
17	DOS_3		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
18	TBS3_3		Transmit Buffer Status 3.	1	1
		0	Locked. Software cannot access the Tx Buffer 3 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1	Released. Software may write a message into the Transmit Buffer 3 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
19	TCS3_3		Transmission Complete Status.	1	x
		0	Incomplete. The previously requested transmission for Tx Buffer 3 is not complete.		
		1	Complete. The previously requested transmission for Tx Buffer 3 has been successfully completed.		
20	RS_3		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
21	TS3_3		Transmit Status 3.	1	0
		0	Idle. There is no transmission from Tx Buffer 3.		
		1	Transmit. The CAN Controller is transmitting a message from Tx Buffer 3.		
22	ES_3		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
23	BS_3		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
31:24	-		Reserved, the value read from a reserved bit is not defined.	NA	

20.7.9 CAN Receive Frame Status register

This register defines the characteristics of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANxMOD is 1.

Table 450. CAN Receive Frame Status register (CAN1RFS - address 0x4004 4020, CAN2RFS - address 0x4004 8020) bit description

Bit	Symbol	Function	Reset Value	RM Set
9:0	IDINDEX	ID Index. If the BP bit (below) is 0, this value is the zero-based number of the Lookup Table RAM entry at which the Acceptance Filter matched the received Identifier. Disabled entries in the Standard tables are included in this numbering, but will not be matched. See Section 20.17 “Examples of acceptance filter tables and ID index values” on page 593 for examples of ID Index values.	0	X
10	BP	If this bit is 1, the current message was received in AF Bypass mode, and the ID Index field (above) is meaningless.	0	X
15:11	-	Reserved. The value read from a reserved bit is not defined.	NA	
19:16	DLC	The field contains the Data Length Code (DLC) field of the current received message. When RTR = 0, this is related to the number of data bytes available in the CANRDA and CANRDB registers as follows: 0000-0111 = 0 to 7 bytes1000-1111 = 8 bytes With RTR = 1, this value indicates the number of data bytes requested to be sent back, with the same encoding.	0	X

Table 450. CAN Receive Frame Status register (CAN1RFS - address 0x4004 4020, CAN2RFS - address 0x4004 8020) bit description

Bit	Symbol	Function	Reset Value	RM Set
29:20	-	Reserved. Read value is undefined, only zero should be written.	NA	
30	RTR	This bit contains the Remote Transmission Request bit of the current received message. 0 indicates a Data Frame, in which (if DLC is non-zero) data can be read from the CANRDA and possibly the CANRDB registers. 1 indicates a Remote frame, in which case the DLC value identifies the number of data bytes requested to be sent using the same Identifier.	0	X
31	FF	A 0 in this bit indicates that the current received message included an 11-bit Identifier, while a 1 indicates a 29-bit Identifier. This affects the contents of the CANid register described below.	0	X

20.7.9.1 ID index field

The ID Index is a 10-bit field in the Info Register that contains the table position of the ID Look-up Table if the currently received message was accepted. The software can use this index to simplify message transfers from the Receive Buffer into the Shared Message Memory. Whenever bit 10 (BP) of the ID Index in the CANRFS register is 1, the current CAN message was received in acceptance filter bypass mode.

20.7.10 CAN Receive Identifier register

This register contains the Identifier field of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANmod is 1. It has two different formats depending on the FF bit in CANRFS.

Table 451. CAN Receive Identifier register (CAN1RID - address 0x4004 4024, CAN2RID - address 0x4004 8024) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11-bit Identifier field of the current received message. In CAN 2.0A, these bits are called ID10-0, while in CAN 2.0B they’re called ID29-18.	0	X
31:11	-	Reserved. The value read from a reserved bit is not defined.	NA	

Table 452. RX Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29-bit Identifier field of the current received message. In CAN 2.0B these bits are called ID29-0.	0	X
31:29	-	Reserved. The value read from a reserved bit is not defined.	NA	

20.7.11 CAN Receive Data register A

This register contains the first 1-4 Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1.

Table 453. CAN Receive Data register A (CAN1RDA - address 0x4004 4028, CAN2RDA - address 0x4004 8028) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	DATA1	Data 1. If the DLC field in CANRFS ≥ 0001, this contains the first Data byte of the current received message.	0	X
15:8	DATA2	Data 2. If the DLC field in CANRFS ≥ 0010, this contains the first Data byte of the current received message.	0	X
23:16	DATA3	Data 3. If the DLC field in CANRFS ≥ 0011, this contains the first Data byte of the current received message.	0	X
31:24	DATA4	Data 4. If the DLC field in CANRFS ≥ 0100, this contains the first Data byte of the current received message.	0	X

20.7.12 CAN Receive Data register B

This register contains the 5th through 8th Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1.

Table 454. CAN Receive Data register B (CAN1RDB - address 0x4004 402C, CAN2RDB - address 0x4004 802C) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	DATA5	Data 5. If the DLC field in CANRFS ≥ 0101, this contains the first Data byte of the current received message.	0	X
15:8	DATA6	Data 6. If the DLC field in CANRFS ≥ 0110, this contains the first Data byte of the current received message.	0	X
23:16	DATA7	Data 7. If the DLC field in CANRFS ≥ 0111, this contains the first Data byte of the current received message.	0	X
31:24	DATA8	Data 8. If the DLC field in CANRFS ≥ 1000, this contains the first Data byte of the current received message.	0	X

20.7.13 CAN Transmit Frame Information register

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the format of the next transmit message for that Tx buffer. Bits not listed read as 0 and should be written as 0.

The values for the reserved bits of the CANxTFI register in the Transmit Buffer should be set to the values expected in the Receive Buffer for an easy comparison, when using the Self Reception facility (self test), otherwise they are not defined.

The CAN Controller consist of three Transmit Buffers. Each of them has a length of 4 words and is able to store one complete CAN message as shown in Figure 82.

The buffer layout is subdivided into Descriptor and Data Field where the first word of the Descriptor Field includes the TX Frame Info that describes the Frame Format, the Data Length and whether it is a Remote or Data Frame. In addition, a TX Priority register allows the definition of a certain priority for each transmit message. Depending on the chosen Frame Format, an 11-bit identifier for Standard Frame Format (SFF) or an 29-bit identifier for Extended Frame Format (EFF) follows. Note that unused bits in the TID field have to be defined as 0. The Data Field in TDA and TDB contains up to eight data bytes.

Table 455. CAN Transmit Frame Information register (CAN1TFI[1/2/3] - address 0x4004 40[30/40/50], CAN2TFI[1/2/3] - 0x4004 80[30/40/50]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	PRIO	If the TPM (Transmit Priority Mode) bit in the CANxMOD register is set to 1, enabled Tx Buffers contend for the right to send their messages based on this field. The buffer with the lowest TX Priority value wins the prioritization and is sent first.	0	X
15:8	-	Reserved. Read value is undefined, only zero should be written.	0	
19:16	DLC	Data Length Code. This value is sent in the DLC field of the next transmit message. In addition, if RTR = 0, this value controls the number of Data bytes sent in the next transmit message, from the CANxTDA and CANxTDB registers: 0000-0111 = 0-7 bytes 1xxx = 8 bytes	0	X
29:20	-	Reserved. Read value is undefined, only zero should be written.	0	
30	RTR	This value is sent in the RTR bit of the next transmit message. If this bit is 0, the number of data bytes called out by the DLC field are sent from the CANxTDA and CANxTDB registers. If this bit is 1, a Remote Frame is sent, containing a request for that number of bytes.	0	X
31	FF	If this bit is 0, the next transmit message will be sent with an 11-bit Identifier (standard frame format), while if it's 1, the message will be sent with a 29-bit Identifier (extended frame format).	0	X

Automatic transmit priority detection

To allow uninterrupted streams of transmit messages, the CAN Controller provides Automatic Transmit Priority Detection for all Transmit Buffers. Depending on the selected Transmit Priority Mode, internal prioritization is based on the CAN Identifier or a user defined "local priority". If more than one message is enabled for transmission (TR=1) the internal transmit message queue is organized such as that the transmit buffer with the lowest CAN Identifier (TID) or the lowest "local priority" (TX Priority) wins the prioritization and is sent first. The result of the internal scheduling process is taken into account short before a new CAN message is sent on the bus. This is also true after the occurrence of a transmission error and right before a re-transmission.

Tx DLC

The number of bytes in the Data Field of a message is coded with the Data Length Code (DLC). At the start of a Remote Frame transmission the DLC is not considered due to the RTR bit being '1 ' (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the DLC must be specified correctly to avoid bus errors, if two CAN Controllers start a Remote Frame transmission with the same identifier simultaneously. For reasons of compatibility no DLC > 8 should be used. If a value greater than 8 is selected, 8 bytes are transmitted in the data frame with the Data Length Code specified in DLC. The range of the Data Byte Count is 0 to 8 bytes and is coded as follows:

$$DataByteCount = DLC \tag{12}$$

20.7.14 CAN Transmit Identifier register

When the corresponding TBS bit in CANxSR is 1, software can write to one of these registers to define the Identifier field of the next transmit message. Bits not listed read as 0 and should be written as 0. The register assumes two different formats depending on the FF bit in CANTFI.

In Standard Frame Format messages, the CAN Identifier consists of 11 bits (ID.28 to ID.18), and in Extended Frame Format messages, the CAN identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit, and it is transmitted first on the bus during the arbitration process. The Identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority during the arbitration process.

Table 456. CAN Transfer Identifier register (CAN1TID[1/2/3] - address 0x4004 40[34/44/54], CAN2TID[1/2/3] - address 0x4004 80[34/44/54]) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11-bit Identifier to be sent in the next transmit message.	0	X
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA	

Table 457. Transfer Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29-bit Identifier to be sent in the next transmit message.	0	X
31:29	-	Reserved. Read value is undefined, only zero should be written.	NA	

20.7.15 CAN Transmit Data register A

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the first 1 - 4 data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 458. CAN Transmit Data register A (CAN1TDA[1/2/3] - address 0x4004 40[38/48/58], CAN2TDA[1/2/3] - address 0x4004 80[38/48/58]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	DATA1	Data 1. If RTR = 0 and DLC ≥ 0001 in the corresponding CANxTFI, this byte is sent as the first Data byte of the next transmit message.	0	X
15;8	DATA2	Data 2. If RTR = 0 and DLC ≥ 0010 in the corresponding CANxTFI, this byte is sent as the 2nd Data byte of the next transmit message.	0	X
23:16	DATA3	Data 3. If RTR = 0 and DLC ≥ 0011 in the corresponding CANxTFI, this byte is sent as the 3rd Data byte of the next transmit message.	0	X
31:24	DATA4	Data 4. If RTR = 0 and DLC ≥ 0100 in the corresponding CANxTFI, this byte is sent as the 4th Data byte of the next transmit message.	0	X

20.7.16 CAN Transmit Data register B

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the 5th through 8th data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 459. CAN Transmit Data register B (CAN1TDB[1/2/3] - address 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3] - address 0x4004 80[3C/4C/5C]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	DATA5	Data 5. If RTR = 0 and DLC ≥ 0101 in the corresponding CANTFI, this byte is sent as the 5th Data byte of the next transmit message.	0	X
15;8	DATA6	Data 6. If RTR = 0 and DLC ≥ 0110 in the corresponding CANTFI, this byte is sent as the 6th Data byte of the next transmit message.	0	X
23:16	DATA7	Data 7. If RTR = 0 and DLC ≥ 0111 in the corresponding CANTFI, this byte is sent as the 7th Data byte of the next transmit message.	0	X
31:24	DATA8	Data 8. If RTR = 0 and DLC ≥ 1000 in the corresponding CANTFI, this byte is sent as the 8th Data byte of the next transmit message.	0	X

20.8 CAN controller operation

20.8.1 Error handling

The CAN Controllers count and handle transmit and receive errors as specified in CAN Spec 2.0B. The Transmit and Receive Error Counters are incriminated for each detected error and are decremented when operation is error-free. If the Transmit Error counter contains 255 and another error occurs, the CAN Controller is forced into a state called Bus-Off. In this state, the following register bits are set: BS in CANxSR, BEI and EI in CANxIR if these are enabled, and RM in CANxMOD. RM resets and disables much of the CAN Controller. Also at this time the Transmit Error Counter is set to 127 and the Receive Error Counter is cleared. Software must next clear the RM bit. Thereafter the Transmit Error Counter will count down 128 occurrences of the Bus Free condition (11 consecutive recessive bits). Software can monitor this countdown by reading the Tx Error Counter. When this countdown is complete, the CAN Controller clears BS and ES in CANxSR, and sets EI in CANxSR if EIE in IER is 1.

The Tx and Rx error counters can be written if RM in CANxMOD is 1. Writing 255 to the Tx Error Counter forces the CAN Controller to Bus-Off state. If Bus-Off (BS in CANxSR) is 1, writing any value 0 through 254 to the Tx Error Counter clears Bus-Off. When software clears RM in CANxMOD thereafter, only one Bus Free condition (11 consecutive recessive bits) is needed before operation resumes.

20.8.2 Sleep mode

The CAN Controller will enter sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. Software can only set SM when RM in the CAN Mode register is 0; it can also set the WUIE bit in the CAN Interrupt Enable register to enable an interrupt on any wake-up condition.

The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1) in response to a) a dominant bit on the CAN bus, or b) software clearing SM in the CAN Mode register. A sleeping CAN Controller that wakes up in response to bus activity is not able to receive an initial message until after it detects Bus_Free (11 consecutive recessive bits). If an interrupt is pending or the CAN bus is active when software sets SM, the wake-up is immediate.

Upon wake-up, software needs to do the following things:

1. Write a 1 to the relevant bit(s) in the CANSLEEPCLR register.

2. Write a 0 to the SM bit in the CAN1MOD and/or CAN2MOD register.
3. Write a 1 to the relevant bit(s) in the CANWAKEFLAGS register. Failure to perform this step will prevent subsequent entry into Power-down mode.

If the device is in Deep Sleep or Power-down mode, CAN activity will wake up the device if the CAN activity interrupt is enabled. See Section 3.12 “Power control”.

20.8.3 Interrupts

Each CAN Controller produces 3 interrupt requests, Receive, Transmit, and “other status”. The Transmit interrupt is the OR of the Transmit interrupts from the three Tx Buffers. All of the interrupts share one NVIC channel. A separate interrupt is provided for the CAN activity interrupt.

20.8.4 Transmit priority

If the TPM bit in the CANxMOD register is 0, multiple enabled Tx Buffers contend for the right to send their messages based on the value of their CAN Identifier (TID). If TPM is 1, they contend based on the PRIO fields in bits 7:0 of their CANxTFS registers. In both cases the smallest binary value has priority. If two (or three) transmit-enabled buffers have the same smallest value, the lowest-numbered buffer sends first.

The CAN controller selects among multiple enabled Tx Buffers dynamically, just before it sends each message.

20.9 Centralized CAN registers

For easy and fast access, all CAN Controller Status bits from each CAN Controller Status register are bundled together. Each defined byte of the following registers contains one particular status bit from each of the CAN controllers, in its LS bits.

All Status registers are read-only and allow byte, half word and word access.

20.9.1 Central Transmit Status Register

Table 460. Central Transit Status Register (TXSR - address 0x4004 0000) bit description

Bit	Symbol	Description	Reset Value
0	TS1	When 1, the CAN controller 1 is sending a message (same as TS in the CAN1GSR).	0
1	TS2	When 1, the CAN controller 2 is sending a message (same as TS in the CAN2GSR)	0
7:2	-	Reserved, the value read from a reserved bit is not defined.	NA
8	TBS1	When 1, all 3 Tx Buffers of the CAN1 controller are available to the CPU (same as TBS in CAN1GSR).	1
9	TBS2	When 1, all 3 Tx Buffers of the CAN2 controller are available to the CPU (same as TBS in CAN2GSR).	1
15:10	-	Reserved, the value read from a reserved bit is not defined.	NA
16	TCS1	When 1, all requested transmissions have been completed successfully by the CAN1 controller (same as TCS in CAN1GSR).	1
17	TCS2	When 1, all requested transmissions have been completed successfully by the CAN2 controller (same as TCS in CAN2GSR).	1
31:18	-	Reserved, the value read from a reserved bit is not defined.	NA

20.9.2 Central Receive Status Register

Table 461. Central Receive Status Register (RXSR - address 0x4004 0004) bit description

Bit	Symbol	Description	Reset Value
0	RS1	When 1, CAN1 is receiving a message (same as RS in CAN1GSR).	0
1	RS2	When 1, CAN2 is receiving a message (same as RS in CAN2GSR).	0
7:2	-	Reserved, the value read from a reserved bit is not defined.	NA
8	RB1	When 1, a received message is available in the CAN1 controller (same as RBS in CAN1GSR).	0
9	RB2	When 1, a received message is available in the CAN2 controller (same as RBS in CAN2GSR).	0
15:10	-	Reserved, the value read from a reserved bit is not defined.	NA
16	DOS1	When 1, a message was lost because the preceding message to CAN1 controller was not read out quickly enough (same as DOS in CAN1GSR).	0
17	DOS2	When 1, a message was lost because the preceding message to CAN2 controller was not read out quickly enough (same as DOS in CAN2GSR).	0
31:18	-	Reserved, the value read from a reserved bit is not defined.	NA

20.9.3 Central Miscellaneous Status Register

Table 462. Central Miscellaneous Status Register (MSR - address 0x4004 0008) bit description

Bit	Symbol	Description	Reset Value
0	E1	When 1, one or both of the CAN1 Tx and Rx Error Counters has reached the limit set in the CAN1EWL register (same as ES in CAN1GSR)	0
1	E2	When 1, one or both of the CAN2 Tx and Rx Error Counters has reached the limit set in the CAN2EWL register (same as ES in CAN2GSR)	0
7:2	-	Reserved, the value read from a reserved bit is not defined.	NA
8	BS1	When 1, the CAN1 controller is currently involved in bus activities (same as BS in CAN1GSR).	0
9	BS2	When 1, the CAN2 controller is currently involved in bus activities (same as BS in CAN2GSR).	0
31:10	-	Reserved, the value read from a reserved bit is not defined.	NA

20.10 Global acceptance filter

This block provides lookup for received Identifiers (called Acceptance Filtering in CAN terminology) for all the CAN Controllers. It includes a 512 × 32 (2 kB) RAM in which software maintains one to five tables of Identifiers. This RAM can contain up to 1024 Standard Identifiers or 512 Extended Identifiers, or a mixture of both types.

20.11 Acceptance filter modes

The Acceptance Filter can be put into different modes by setting the according AccOff, AccBP, and eFCAN bits in the Acceptance Filter Mode Register (Section 20.14.1 “Acceptance Filter Mode Register”). During each mode the access to the Configuration Register and the ID Look-up table is handled differently.

Table 463. Acceptance filter modes and access control

Acceptance filter mode	Bit AccOff	Bit AccBP	Acceptance filter state	ID Look-up table RAM[1]	Acceptance filter config. registers	CAN controller message receive interrupt
Off Mode	1	0	reset & halted	r/w access from CPU	r/w access from CPU	no messages accepted
Bypass Mode	X	1	reset & halted	r/w access from CPU	r/w access from CPU	all messages accepted
Operating Mode and FullCAN Mode	0	0	running	read-only from CPU[2]	access from Acceptance filter only	hardware acceptance filtering

[1] The whole ID Look-up Table RAM is only word accessible.

[2] During the Operating Mode of the Acceptance Filter the Look-up Table can be accessed only to disable or enable Messages.

A write access to all section configuration registers is only possible during the Acceptance Filter Off and Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

20.11.1 Acceptance filter Off mode

The Acceptance Filter Off Mode is typically used during initialization. During this mode an unconditional access to all registers and to the Look-up Table RAM is possible. With the Acceptance Filter Off Mode, CAN messages are not accepted and therefore not stored in the Receive Buffers of active CAN Controllers.

20.11.2 Acceptance filter Bypass mode

The Acceptance Filter Bypass Mode can be used for example to change the acceptance filter configuration during a running system, e.g. change of identifiers in the ID-Look-up Table memory. During this re-configuration, software acceptance filtering has to be used.

It is recommended to use the ID ready Interrupt (ID Index) and the Receive Interrupt (RI). In this mode all CAN message are accepted and stored in the Receive Buffers of active CAN Controllers.

20.11.3 Acceptance filter Operating mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 0.

20.11.4 FullCAN mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 1. More details on FullCAN mode are available in Section 20.16 “FullCAN mode”.

20.12 Sections of the ID look-up table RAM

Four 12-bit section configuration registers (SFF_sa, SFF_GRP_sa, EFF_sa, EFF_GRP_sa) are used to define the boundaries of the different identifier sections in the ID-Look-up Table Memory. The fifth 12-bit section configuration register, the End of Table address register (ENDofTable) is used to define the end of all identifier sections. The End of Table address is also used to assign the start address of the section where FullCAN Message Objects, if enabled are stored.

Table 464. Section configuration register settings

ID-Look up Table Section	Register	Value	Section status
FullCAN (Standard Frame Format) Identifier Section	SFF_sa	= 0x000	disabled
		> 0x000	enabled
Explicit Standard Frame Format Identifier Section	SFF_GRP_sa	= SFF_sa	disabled
		> SFF_sa	enabled
Group of Standard Frame Format Identifier Section	EFF_sa	= SFF_GRP_sa	disabled
		> SFF_GRP_sa	enabled
Explicit Extended Frame Format Identifier Section	EFF_GRP_sa	= EFF_sa	disabled
		> EFF_sa	enabled
Group of Extended Frame Format Identifier Section	ENDofTable	= EFF_GRP_sa	disabled
		> EFF_GRP_sa	enabled

20.13 ID look-up table RAM

The Whole ID Look-up Table RAM is only word accessible. A write access is only possible during the Acceptance Filter Off or Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

If Standard (11-bit) Identifiers are used in the application, at least one of 3 tables in Acceptance Filter RAM must not be empty. If the optional “FullCAN mode” is enabled, the first table contains Standard identifiers for which reception is to be handled in this mode. The next table contains individual Standard Identifiers and the third contains ranges of Standard Identifiers, for which messages are to be received via the CAN Controllers. The tables of FullCAN and individual Standard Identifiers must be arranged in ascending numerical order, one per halfword, two per word. Since each CAN bus has its own address map, each entry also contains the number of the CAN Controller (SCC = 000 (CAN1) -or SCC = 001 (CAN2)) to which it applies.

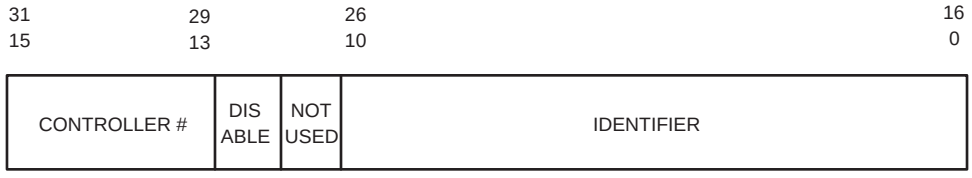


Fig 86. Entry in FullCAN and individual standard identifier tables

The table of Standard Identifier Ranges contains paired upper and lower (inclusive) bounds, one pair per word. These must also be arranged in ascending numerical order.

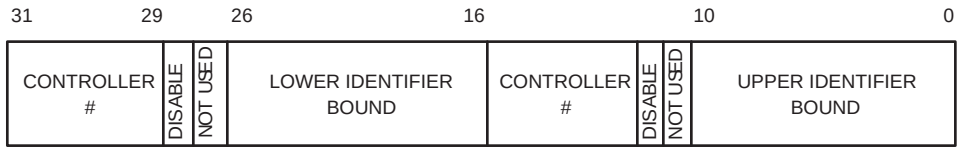


Fig 87. Entry in standard identifier range table

The disable bits in Standard entries provide a means to turn response to particular CAN Identifiers or ranges of Identifiers on and off dynamically. When the Acceptance Filter function is enabled, only the disable bits in Acceptance Filter RAM can be changed by software. Response to a range of Standard addresses can be enabled by writing 32 zero bits to its word in RAM, and turned off by writing 32 one bits (0xFFFF FFFF) to its word in RAM. Only the disable bits are actually changed. Disabled entries must maintain the ascending sequence of Identifiers.

If Extended (29-bit) Identifiers are used in the application, at least one of the other two tables in Acceptance Filter RAM must not be empty, one for individual Extended Identifiers and one for ranges of Extended Identifiers. The table of individual Extended Identifiers must be arranged in ascending numerical order.

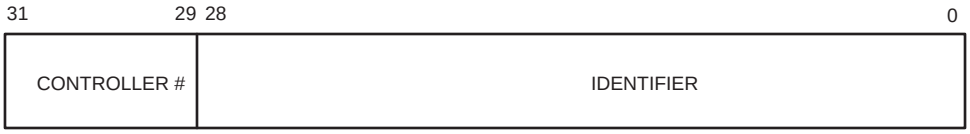


Fig 88. Entry in either extended identifier table

The table of ranges of Extended Identifiers must contain an even number of entries, of the same form as in the individual Extended Identifier table. Like the Individual Extended table, the Extended Range must be arranged in ascending numerical order. The first and second (3rd and 4th …) entries in the table are implicitly paired as an inclusive range of Extended addresses, such that any received address that falls in the inclusive range is received (accepted). Software must maintain the table to consist of such word pairs.

There is no facility to receive messages to Extended identifiers using the FullCAN method.

Five address registers point to the boundaries between the tables in Acceptance Filter RAM: FullCAN Standard addresses, Standard Individual addresses, Standard address ranges, Extended Individual addresses, and Extended address ranges. These tables must be consecutive in memory. The start of each of the latter four tables is implicitly the end of the preceding table. The end of the Extended range table is given in an End of Tables register. If the start address of a table equals the start of the next table or the End Of Tables register, that table is empty.

When the Receive side of a CAN controller has received a complete Identifier, it signals the Acceptance Filter of this fact. The Acceptance Filter responds to this signal, and reads the Controller number, the size of the Identifier, and the Identifier itself from the Controller. It then proceeds to search its RAM to determine whether the message should be received or ignored.

If FullCAN mode is enabled and the CAN controller signals that the current message contains a Standard identifier, the Acceptance Filter first searches the table of identifiers for which reception is to be done in FullCAN mode. Otherwise, or if the AF doesn't find a match in the FullCAN table, it searches its individual Identifier table for the size of Identifier signalled by the CAN controller. If it finds an equal match, the AF signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register.

If the Acceptance Filter does not find a match in the appropriate individual Identifier table, it then searches the Identifier Range table for the size of Identifier signalled by the CAN controller. If the AF finds a match to a range in the table, it similarly signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register. If the Acceptance Filter does not find a match in either the individual or Range table for the size of Identifier received, it signals the CAN controller to discard/ignore the received message.

20.14 Acceptance filter registers

20.14.1 Acceptance Filter Mode Register

The AccBP and AccOff bits of the acceptance filter mode register are used for putting the acceptance filter into the Bypass and Off mode. The eFCAN bit of the mode register can be used to activate a FullCAN mode enhancement for received 11-bit CAN ID messages.

The following restrictions apply to using the bits in this register:

- Acceptance Filter Bypass Mode (AccBP): By setting the AccBP bit in the Acceptance Filter Mode Register, the Acceptance filter is put into the Acceptance Filter Bypass mode. During bypass mode, the internal state machine of the Acceptance Filter is reset and halted. All received CAN messages are accepted, and acceptance filtering can be done by software.
- Acceptance Filter Off mode (AccOff): After power-up or hardware reset, the Acceptance filter will be in Off mode, the AccOff bit in the Acceptance filter Mode register 0 will be set to 1. The internal state machine of the acceptance filter is reset and halted. If not in Off mode, setting the AccOff bit, either by hardware or by software, will force the acceptance filter into Off mode.
- FullCAN Mode Enhancements: A FullCAN mode for received CAN messages can be enabled by setting the eFCAN bit in the acceptance filter mode register.

Table 465. Acceptance Filter Mode Register (AFMR - address 0x4003 C000) bit description

Bit	Symbol	Value	Description	Reset Value
0	ACCOFF		if AccBP is 0, the Acceptance Filter is not operational. All Rx messages on all CAN buses are ignored.	1
1	ACCBP		All Rx messages are accepted on enabled CAN controllers. Software must set this bit before modifying the contents of any of the registers described below, and before modifying the contents of Lookup Table RAM in any way other than setting or clearing Disable bits in Standard Identifier entries. When both this bit and AccOff are 0, the Acceptance filter operates to screen received CAN Identifiers.	0

Table 465. Acceptance Filter Mode Register (AFMR - address 0x4003 C000) bit description

Bit	Symbol	Value	Description	Reset Value
2	EFCAN		FullCAN mode	0
		0	Software must read all messages for all enabled IDs on all enabled CAN buses, from the receiving CAN controllers.	
		1	The Acceptance Filter itself will take care of receiving and storing messages for selected Standard ID values on selected CAN buses. See Section 20.16 “FullCAN mode” on page 582.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	NA

20.14.2 Section configuration registers

The 10-bit section configuration registers are used for the ID look-up table RAM to indicate the boundaries of the different sections for explicit and group of CAN identifiers for 11-bit CAN and 29-bit CAN identifiers, respectively. The 10-bit wide section configuration registers allow the use of a 512x32 (2 kB) look-up table RAM. The whole ID Look-up Table RAM is only word accessible. All five section configuration registers contain APB addresses for the acceptance filter RAM and do not include the APB base address. A write access to all section configuration registers is only possible during the Acceptance filter off and Bypass modes. Read access is allowed in all acceptance filter modes.

20.14.3 Standard Frame Individual Start Address register

Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

Table 466. Standard Frame Individual Start Address register (SFF_SA - address 0x4003 C004) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
10:2	SFF_SA	The start address of the table of individual Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the SFF_GRP_sa register described below. For compatibility with possible future devices, write zeroes in bits 31:11 and 1:0 of this register. If the eFCAN bit in the AFMR is 1, this value also indicates the size of the table of Standard IDs which the Acceptance Filter will search and (if found) automatically store received messages in Acceptance Filter RAM.	0
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.4 Standard Frame Group Start Address register

Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

Table 467. Standard Frame Group Start Address register (SFF_GRP_SA - address 0x4003 C008) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
11:2	SFF_GRP_SA	The start address of the table of grouped Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_sa register described below. The largest value that should be written to this register is 0x800, when only the Standard Individual table is used, and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.5 Extended Frame Start Address register

Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

Table 468. Extended Frame Start Address register (EFF_SA - address 0x4003 C00C) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
10:2	EFF_SA	The start address of the table of individual Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_GRP_sa register described below. The largest value that should be written to this register is 0x800, when both Extended Tables are empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:11 and 1:0 of this register.	0
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.6 Extended Frame Group Start Address register

Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

Table 469. Extended Frame Group Start Address register (EFF_GRP_SA - address 0x4003 C010) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
11:2	EFF_GRP_SA	The start address of the table of grouped Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the ENDofTable register described below. The largest value that should be written to this register is 0x800, when this table is empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.7 End of AF Tables register

Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

Table 470. End of AF Tables register (ENDOFTABLE - address 0x4003 C014) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
11:2	ENDOFTABLE	The address above the last active address in the last active AF table. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register. If the eFCAN bit in the AFMR is 0, the largest value that should be written to this register is 0x800, which allows the last word (address 0x7FC) in AF Lookup Table RAM to be used. If the eFCAN bit in the AFMR is 1, this value marks the start of the area of Acceptance Filter RAM, into which the Acceptance Filter will automatically receive messages for selected IDs on selected CAN buses. In this case, the maximum value that should be written to this register is 0x800 minus 6 times the value in SFF_sa. This allows 12 bytes of message storage between this address and the end of Acceptance Filter RAM, for each Standard ID that is specified between the start of Acceptance Filter RAM, and the next active AF table.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.8 Status registers

The look-up table error status registers, the error addresses, and the flag register provide information if a programming error in the look-up table RAM during the ID screening was encountered. The look-up table error address and flag register have only read access. If an error is detected, the LUTerror flag is set, and the LUTerrorAddr register provides the information under which address during an ID screening an error in the look-up table was encountered. Any read of the LUTerrorAddr Filter block can be used for a look-up table interrupt.

20.14.9 LUT Error Address register

Table 471. LUT Error Address register (LUTERRAD - address 0x4003 C018) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
10:2	LUTERRAD	It the LUT Error bit (below) is 1, this read-only field contains the address in AF Lookup Table RAM, at which the Acceptance Filter encountered an error in the content of the tables.	0
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.10 LUT Error register

Table 472. LUT Error register (LUTERR - address 0x4003 C01C) bit description

Bit	Symbol	Description	Reset Value
0	LUTERR	This read-only bit is set to 1 if the Acceptance Filter encounters an error in the content of the tables in AF RAM. It is cleared when software reads the LUTerrAd register. This condition is ORed with the other CAN interrupts from the CAN controllers, to produce the request that is connected to the NVIC.	0
31:1	-	Reserved, the value read from a reserved bit is not defined.	NA

20.14.11 Global FullCANInterrupt Enable register

A write access to the Global FullCAN Interrupt Enable register is only possible when the Acceptance Filter is in the off mode.

Table 473. Global FullCAN Enable register (FCANIE - address 0x4003 C020) bit description

Bit	Symbol	Description	Reset Value
0	FCANIE	Global FullCAN Interrupt Enable. When 1, this interrupt is enabled.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	NA

20.14.12 FullCAN Interrupt and Capture registers

For detailed description on these two registers, see Section 20.16.2 “FullCAN interrupts”.

Table 474. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0x4003 C024) bit description

Bit	Symbol	Description	Reset Value
31:0	INTPND	FullCan Interrupt Pending 0 = FullCan Interrupt Pending bit 0. 1 = FullCan Interrupt Pending bit 1. ... 31 = FullCan Interrupt Pending bit 31.	0

Table 475. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0x4003 C028) bit description

Bit	Symbol	Description	Reset Value
31:0	IntPnd32	FullCan Interrupt Pending bit 32. 0 = FullCan Interrupt Pending bit 32. 1 = FullCan Interrupt Pending bit 33. ... 31 = FullCan Interrupt Pending bit 63.	0

20.15 Configuration and search algorithm

The CAN Identifier Look-up Table Memory can contain explicit identifiers and groups of CAN identifiers for Standard and Extended CAN Frame Formats. They are organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

Every CAN identifier is linked to an ID Index number. In case of a CAN Identifier match, the matching ID Index is stored in the Identifier Index of the Frame Status Register (CANRFS) of the according CAN Controller.

20.15.1 Acceptance filter search algorithm

The identifier screening process of the acceptance filter starts in the following order:

1. FullCAN (Standard Frame Format) Identifier Section
2. Explicit Standard Frame Format Identifier Section

3. Group of Standard Frame Format Identifier Section
4. Explicit Extended Frame Format Identifier Section
5. Group of Extended Frame Format Identifier Section

Note: Only activated sections will take part in the screening process.

In cases where equal message identifiers of same frame format are defined in more than one section, the first match will end the screening process for this identifier.

For example, if the same Source CAN Channel in conjunction with the identifier is defined in the FullCAN, the Explicit Standard Frame Format and the Group of Standard Frame Format Identifier Sections, the screening will already be finished with the match in the FullCAN section.

In the example of Figure 89, Identifiers with their Source CAN Channel have been defined in the FullCAN, Explicit and Group of Standard Frame Format Identifier Sections.

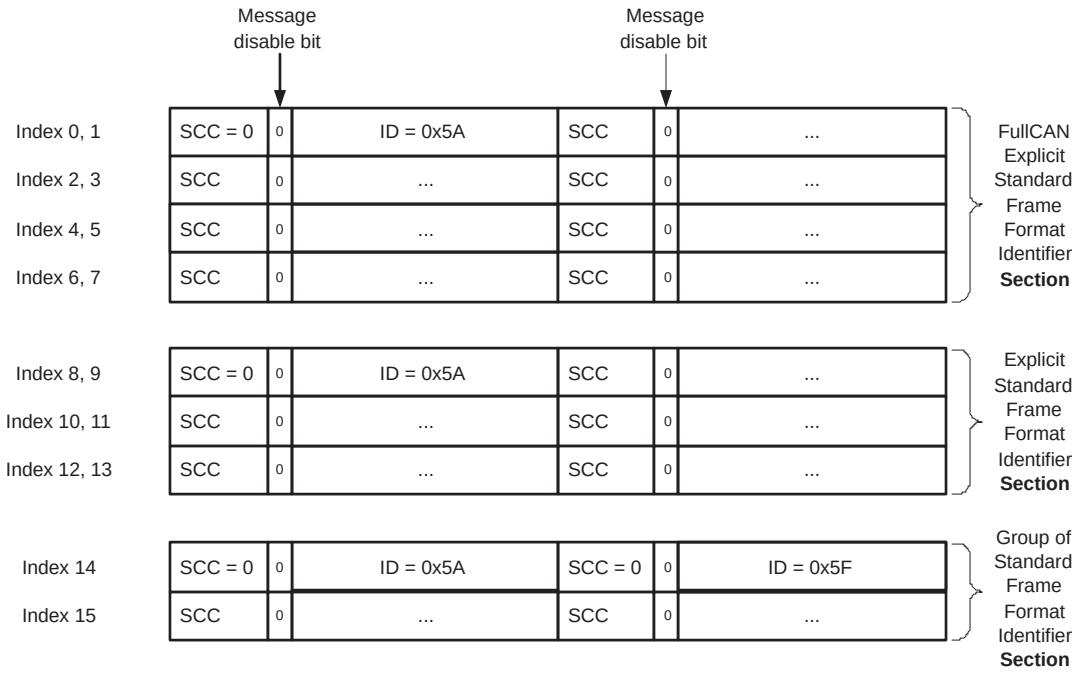


Fig 89. ID Look-up table example explaining the search algorithm

The identifier 0x5A of the CAN Controller 1 with the Source CAN Channel SCC = 0, is defined in all three sections. With this configuration incoming CAN messages on CAN Controller 1 with a 0x5A identifier will find a match in the FullCAN section.

It is possible to disable the '0x5A identifier' in the FullCAN section. With that, the screening process would be finished with the match in the Explicit Identifier Section.

The first group in the Group Identifier Section has been defined such that incoming CAN messages with identifiers of 0x5A up to 0x5F are accepted on CAN Controller 1 with the Source CAN Channel SCC = 0. As stated above, the identifier 0x5A would find a match

already in the FullCAN or in the Explicit Identifier section if enabled. The rest of the defined identifiers of this group (0x5B to 0x5F) will find a match in this Group Identifier Section.

This way the user can switch dynamically between different filter modes for same identifiers.

20.16 FullCAN mode

The FullCAN mode is based on capabilities provided by the CAN Gateway module. This block uses the Acceptance Filter to provide filtering for both CAN channels.

The concept of the CAN Gateway block is mainly based on a BasicCAN functionality. This concept fits perfectly in systems where a gateway is used to transfer messages or message data between different CAN channels. A BasicCAN device is generating a receive interrupt whenever a CAN message is accepted and received. Software has to move the received message out of the receive buffer from the according CAN controller into the user RAM.

To cover dashboard like applications where the controller typically receives data from several CAN channels for further processing, the CAN Gateway block was extended by a so-called FullCAN receive function. This additional feature uses an internal message handler to move received FullCAN messages from the receive buffer of the according CAN controller into the FullCAN message object data space of Look-up Table RAM.

When FullCAN mode is enabled, the Acceptance Filter itself takes care of receiving and storing messages for selected Standard ID values on selected CAN buses, in the style of “FullCAN” controllers.

In order to set this bit and use this mode, two other conditions must be met with respect to the contents of Acceptance Filter RAM and the pointers into it:

- The Standard Frame Individual Start Address Register (SFF_sa) must be greater than or equal to the number of IDs for which automatic receive storage is to be done, times two. SFF_sa must be rounded up to a multiple of 4 if necessary.
- The EndOfTable register must be less than or equal to 0x800 minus 6 times the SFF_sa value, to allow 12 bytes of message storage for each ID for which automatic receive storage will be done.

When these conditions are met and eFCAN is set:

- The area between the start of Acceptance Filter RAM and the SFF_sa address, is used for a table of individual Standard IDs and CAN Controller/bus identification, sorted in ascending order and in the same format as in the Individual Standard ID table (see Figure 86 “Entry in FullCAN and individual standard identifier tables” on page 574). Entries can be marked as “disabled” as in the other Standard tables. If there are an odd number of “FullCAN” ID’s, at least one entry in this table must be so marked.
- The first (SFF_sa)/2 IDindex values are assigned to these automatically-stored ID’s. That is, IDindex values stored in the Rx Frame Status Register, for IDs not handled in this way, are increased by (SFF_sa)/2 compared to the values they would have when eFCAN is 0.

- When a Standard ID is received, the Acceptance Filter searches this table before the Standard Individual and Group tables.
- When a message is received for a controller and ID in this table, the Acceptance filter reads the received message out of the CAN controller and stores it in Acceptance Filter RAM, starting at (EndOfTable) + its IDindex*12.
- The format of such messages is shown in Table 476.

20.16.1 FullCAN message layout

Table 476. Format of automatically stored Rx messages

Address	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
0	FF	RTR	0000				SEM [1:0]		0000				DLC				00000					ID.28 ... ID.18										
+4	Rx Data 4								Rx Data 3								Rx Data 2								Rx Data 1							
+8	Rx Data 8								Rx Data 7								Rx Data 6								Rx Data 5							

The FF, RTR, and DLC fields are as described in Table 450.

Since the FullCAN message object section of the Look-up table RAM can be accessed both by the Acceptance Filter and the CPU, there is a method for insuring that no CPU reads from FullCAN message object occurs while the Acceptance Filter hardware is writing to that object.

For this purpose the Acceptance Filter uses a 3-state semaphore, encoded with the two semaphore bits SEM1 and SEM0 (see Table 476 “Format of automatically stored Rx messages”) for each message object. This mechanism provides the CPU with information about the current state of the Acceptance Filter activity in the FullCAN message object section.

The semaphore operates in the following manner:

Table 477. FullCAN semaphore operation

SEM1	SEM0	activity
0	1	Acceptance Filter is updating the content
1	1	Acceptance Filter has finished updating the content
0	0	CPU is in process of reading from the Acceptance Filter

Prior to writing the first data byte into a message object, the Acceptance Filter will write the FrameInfo byte into the according buffer location with SEM[1:0] = 01.

After having written the last data byte into the message object, the Acceptance Filter will update the semaphore bits by setting SEM[1:0] = 11.

Before reading a message object, the CPU should read SEM[1:0] to determine the current state of the Acceptance Filter activity therein. If SEM[1:0] = 01, then the Acceptance Filter is currently active in this message object. If SEM[1:0] = 11, then the message object is available to be read.

Before the CPU begins reading from the message object, it should clear SEM[1:0] = 00.

When the CPU is finished reading, it can check SEM[1:0] again. At the time of this final check, if SEM[1:0] = 01 or 11, then the Acceptance Filter has updated the message object during the time when the CPU reads were taking place, and the CPU should discard the data. If, on the other hand, SEM[1:0] = 00 as expected, then valid data has been successfully read by the CPU.

Figure 90 shows how software should use the SEM field to ensure that all three words read from the message are all from the same received message.

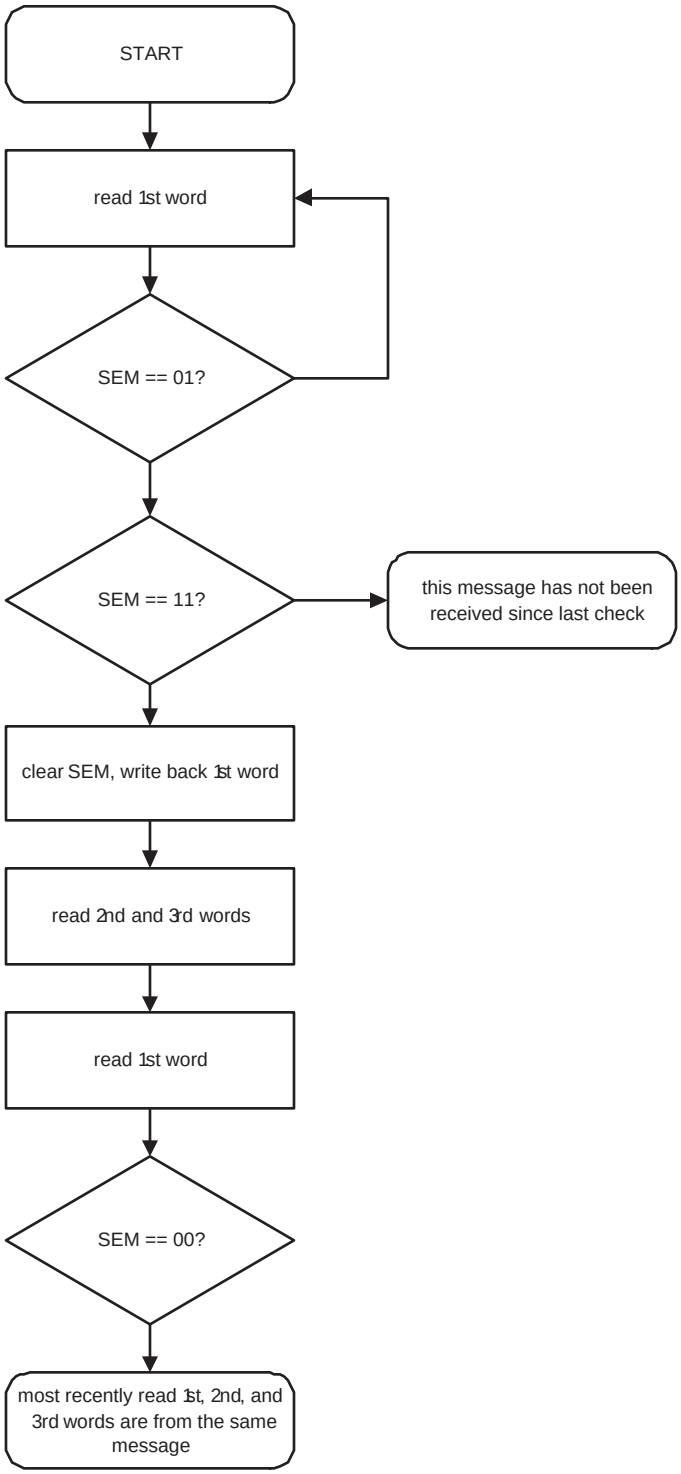


Fig 90. Semaphore procedure for reading an auto-stored message

20.16.2 FullCAN interrupts

The CAN Gateway Block contains a 2 kB ID Look-up Table RAM. With this size a maximum number of 146 FullCAN objects can be defined if the whole Look-up Table RAM is used for FullCAN objects only. Only the first 64 FullCAN objects can be configured to participate in the interrupt scheme. It is still possible to define more than 64 FullCAN objects. The only difference is, that the remaining FullCAN objects will not provide a FullCAN interrupt.

The FullCAN Interrupt Register-set contains interrupt flags (IntPndx) for (pending) FullCAN receive interrupts. As soon as a FullCAN message is received, the according interrupt bit (IntPndx) in the FCAN Interrupt Register gets asserted. In case that the Global FullCAN Interrupt Enable bit is set, the FullCAN Receive Interrupt is passed to the Vectored Interrupt Controller.

Application Software has to solve the following:

1. Index/Object number calculation based on the bit position in the FCANIC Interrupt Register for more than one pending interrupt.
2. Interrupt priority handling if more than one FullCAN receive interrupt is pending.

The software that covers the interrupt priority handling has to assign a receive interrupt priority to every FullCAN object. If more than one interrupt is pending, then the software has to decide, which received FullCAN object has to be served next.

To each FullCAN object a new FullCAN Interrupt Enable bit (FCANIntxEn) is added, so that it is possible to enable or disable FullCAN interrupts for each object individually. The new Message Lost flag (MsgLstx) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read by the CPU. The Interrupt Enable and the Message Lost bits reside in the existing Look-up Table RAM.

20.16.2.1 FullCAN message interrupt enable bit

In Figure 91 8 FullCAN Identifiers with their Source CAN Channel are defined in the FullCAN, Section. The new introduced FullCAN Message Interrupt enable bit can be used to enable for each FullCAN message an Interrupt.

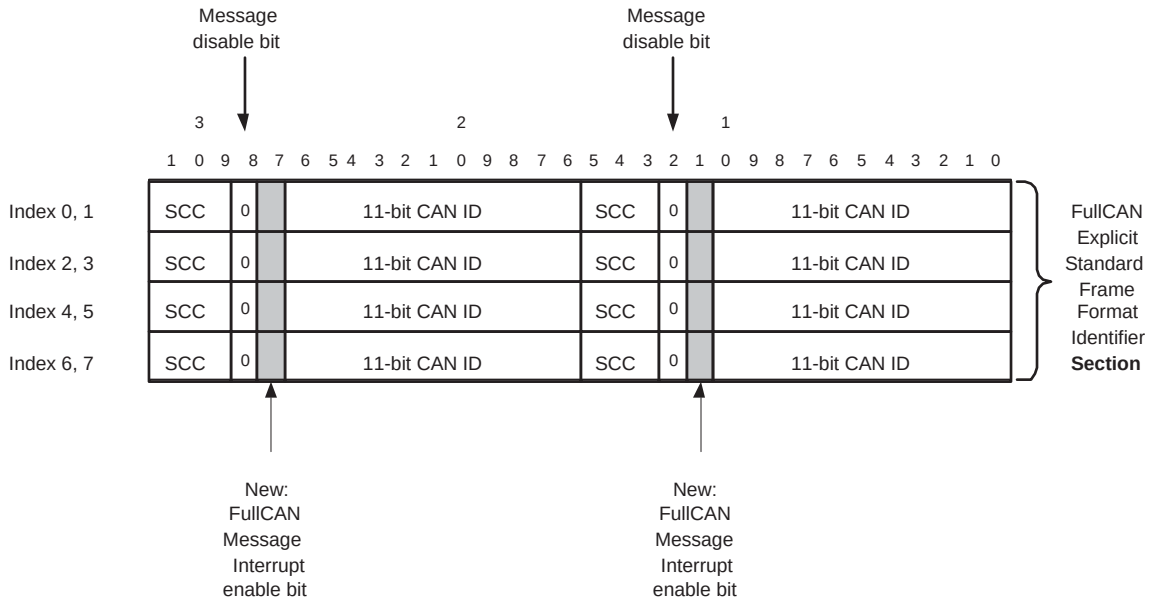


Fig 91. FullCAN section example of the ID look-up table

20.16.2.2 Message lost bit and CAN channel number

Figure 92 is the detailed layout structure of one FullCAN message stored in the FullCAN message object section of the Look-up Table.

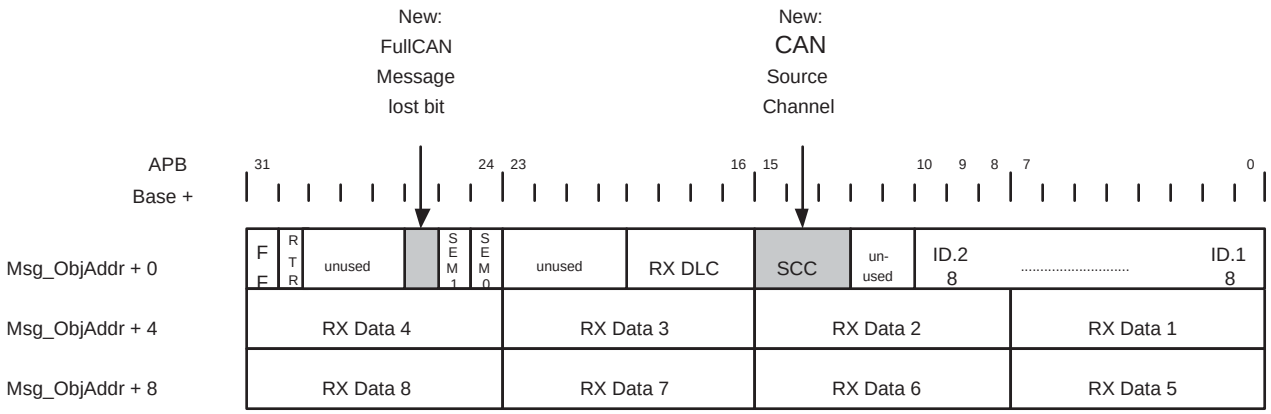


Fig 92. FullCAN message object layout

The new message lost bit (MsgLst) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read. For more information the CAN Source Channel (SCC) of the received FullCAN message is added to Message Object.

20.16.2.3 Setting the interrupt pending bits (IntPnd 63 to 0)

The interrupt pending bit (IntPndx) gets asserted in case of an accepted FullCAN message and if the interrupt of the according FullCAN Object is enabled (enable bit FCANIntxEn) is set).

During the **last write access** from the data storage of a FullCAN message object the interrupt pending bit of a FullCAN object (IntPndx) gets asserted.

20.16.2.4 Clearing the interrupt pending bits (IntPnd 63 to 0)

Each of the FullCAN Interrupt Pending requests gets cleared when the semaphore bits of a message object are cleared by Software (ARM CPU).

20.16.2.5 Setting the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets asserted in case of an accepted FullCAN message and when the FullCAN Interrupt of the same object is asserted already.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets asserted if the interrupt pending bit is set already.

20.16.2.6 Clearing the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets cleared when the FullCAN Interrupt of the same object is not asserted.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets cleared if the interrupt pending bit is not set.

20.16.3 Set and clear mechanism of the FullCAN interrupt

Special precaution is needed for the built-in set and clear mechanism of the FullCAN Interrupts. The following text illustrates how the already existing Semaphore Bits (see Section 20.16.1 “FullCAN message layout” for more details) and how the new introduced features (IntPndx, MsgLstx) will behave.

20.16.3.1 Scenario 1: Normal case, no message lost

Figure 93 below shows a typical “normal” scenario in which an accepted FullCAN message is stored in the FullCAN Message Object Section. After storage the message is read out by Software (ARM CPU).

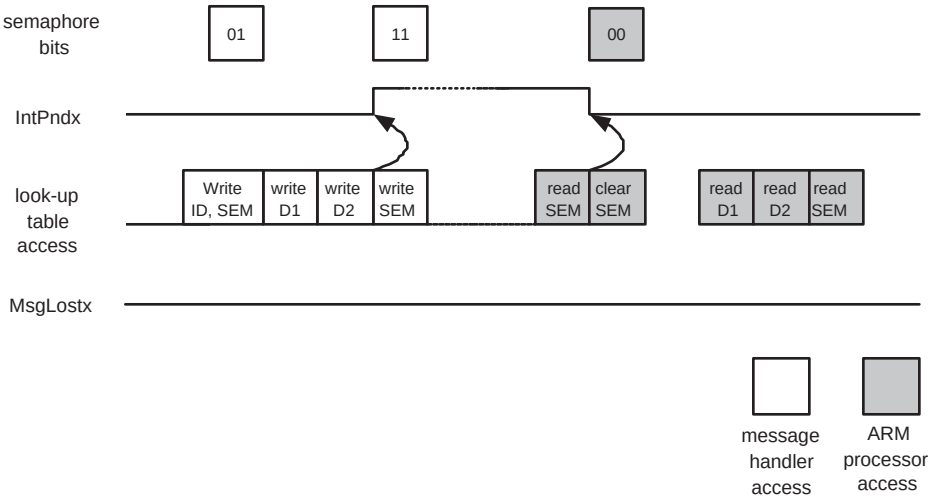


Fig 93. Normal case, no messages lost

20.16.3.2 Scenario 2: Message lost

In this scenario a first FullCAN Message is stored and read out by Software (1st Object write and read). In a second course a second message is stored (2nd Object write) but not read out before a third message gets stored (3rd Object write). Since the FullCAN Interrupt of that Object (IntPndx) is already asserted, the Message Lost Signal gets asserted.

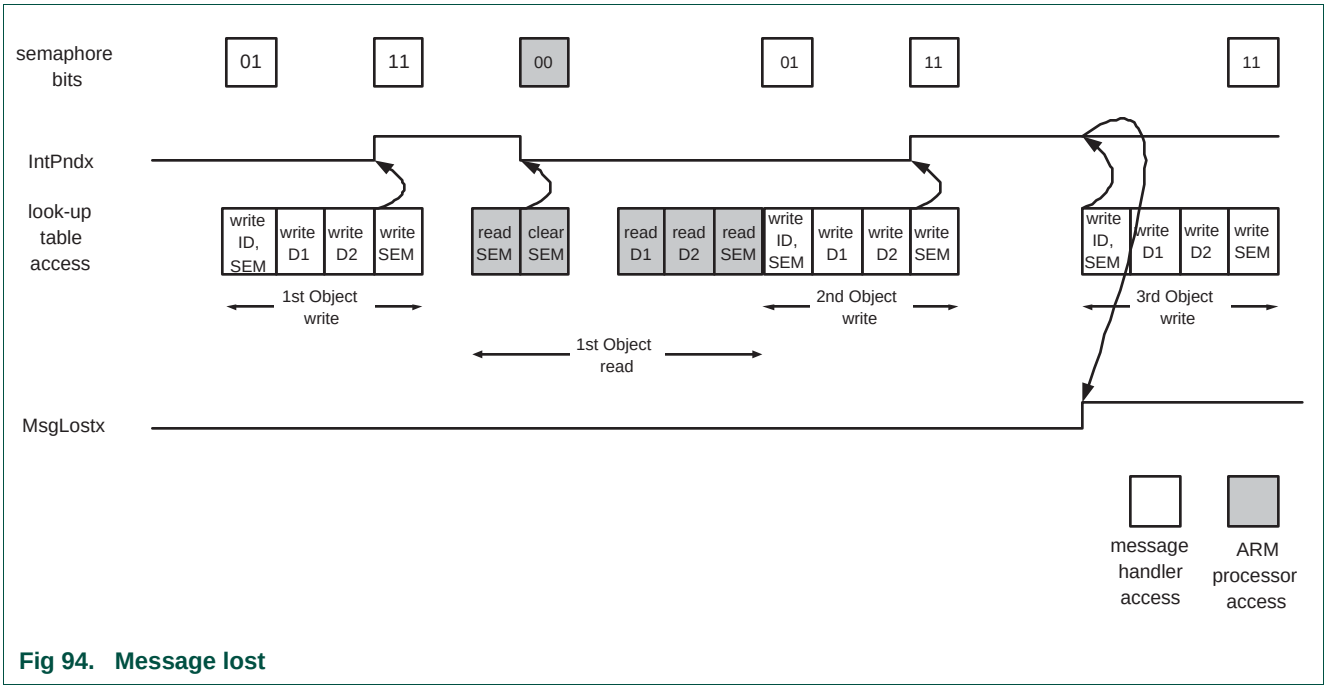


Fig 94. Message lost

20.16.3.3 Scenario 3: Message gets overwritten indicated by Semaphore bits

This scenario is a special case in which the lost message is indicated by the existing semaphore bits. The scenario is entered, if during a Software read of a message object another new message gets stored by the message handler. In this case, the FullCAN Interrupt bit gets set for a second time with the 2nd Object write.

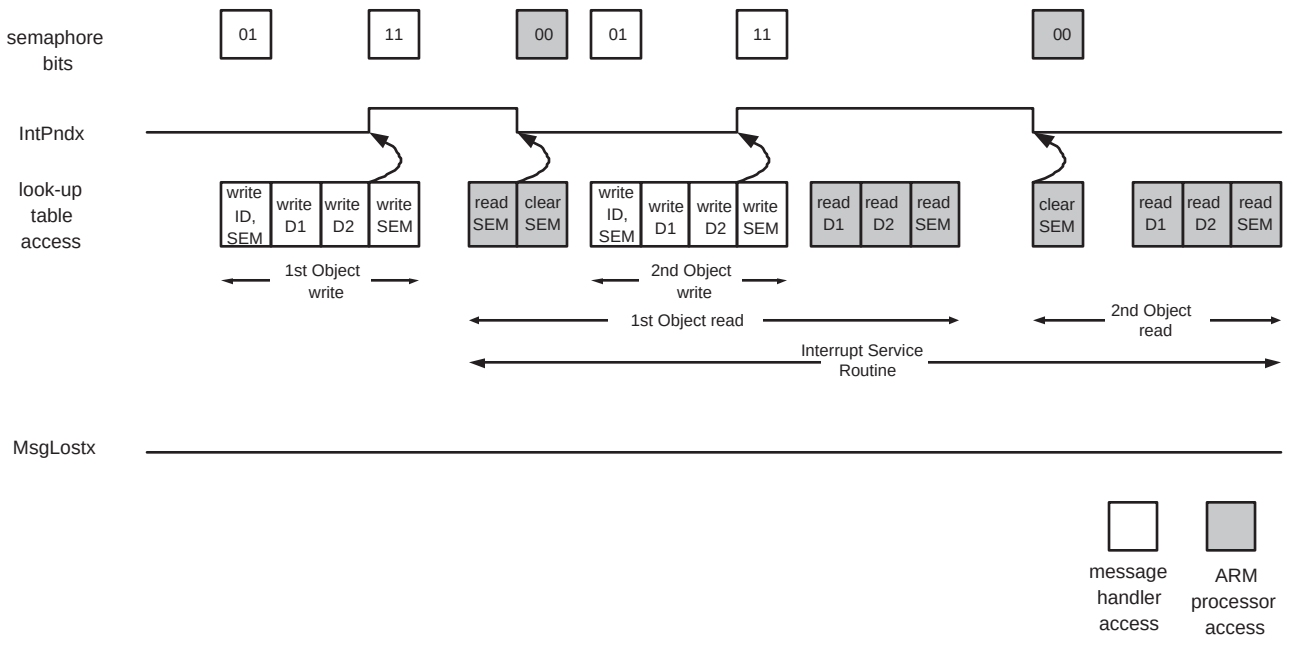


Fig 95. Message gets overwritten

20.16.3.4 Scenario 3.1: Message gets overwritten indicated by Semaphore bits and Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by the existing semaphore bits and by Message Lost.

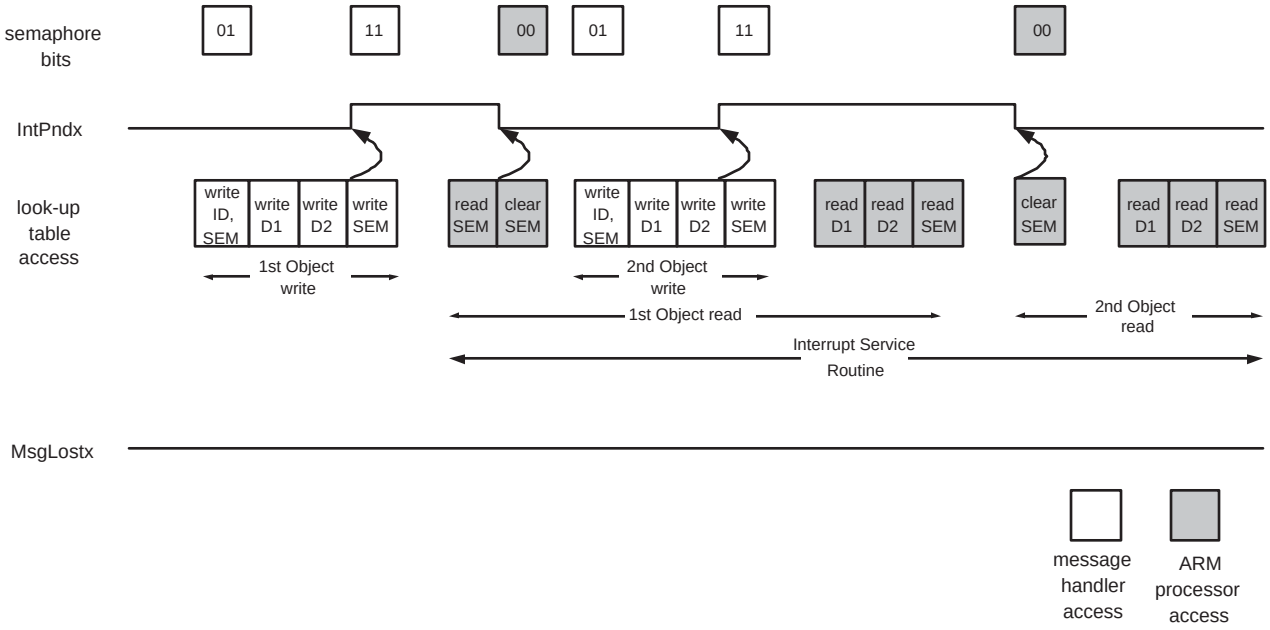


Fig 96. Message overwritten indicated by semaphore bits and message lost

20.16.3.5 Scenario 3.2: Message gets overwritten indicated by Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by Message Lost.

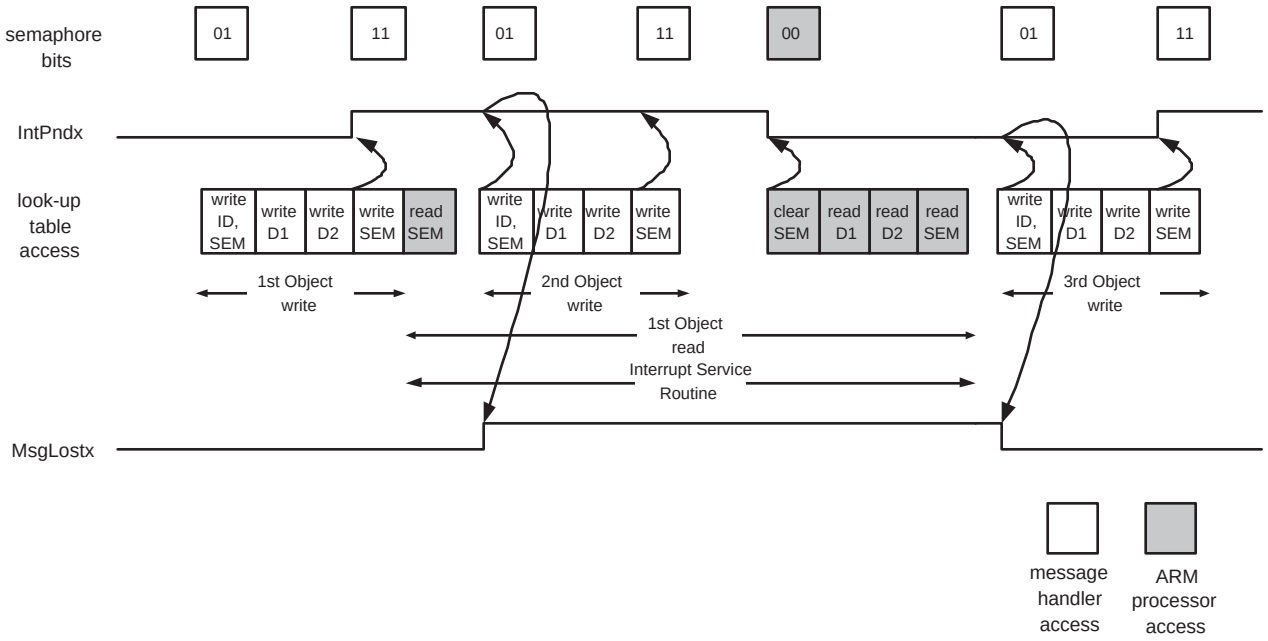


Fig 97. Message overwritten indicated by message lost

20.16.3.6 Scenario 4: Clearing Message Lost bit

This scenario is a special case in which the lost message bit of an object gets set during an overwrite of a none read message object (2nd Object write). The subsequent read out of that object by Software (1st Object read) clears the pending Interrupt. The 3rd Object write clears the Message Lost bit. Every “write ID, SEM” clears Message Lost bit if no pending Interrupt of that object is set.

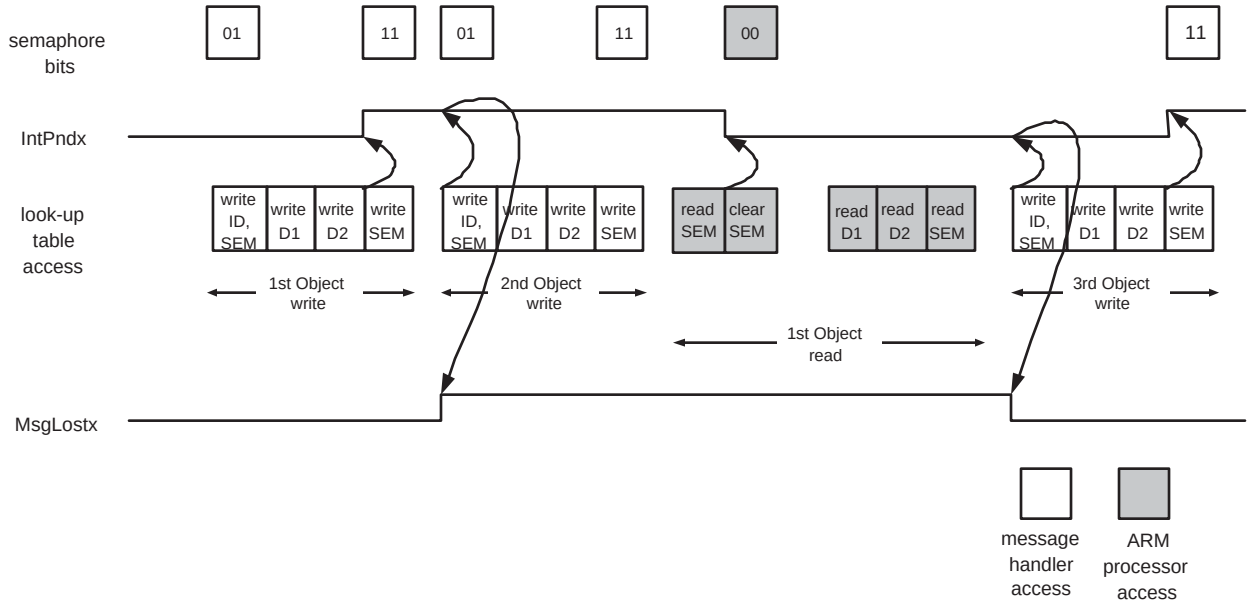


Fig 98. Clearing message lost

20.17 Examples of acceptance filter tables and ID index values

20.17.1 Example 1: only one section is used

```
SFF_sa     <    ENDofTable      OR
SFF_GRP_sa <    ENDofTable      OR
EFF_sa     <    ENDofTable      OR
EFF_GRP_sa <    ENDofTable
```

The start address of a section is lower than the end address of all programmed CAN identifiers.

20.17.2 Example 2: all sections are used

```
SFF_sa     <    SFF_GRP_sa      AND
SFF_GRP_sa <    EFF_sa          AND
EFF_sa     <    EFF_GRP_sa      AND
EFF_GRP_sa <    ENDofTable
```

In cases of a section not being used, the start address has to be set onto the value of the next section start address.

20.17.3 Example 3: more than one but not all sections are used

If the SFF group is not used, the start address of the SFF Group Section (SFF_GRP_sa register) has to be set to the same value of the next section start address, in this case the start address of the Explicit SFF Section (SFF_sa register).

In cases where explicit identifiers as well as groups of the identifiers are programmed, a CAN identifier search has to start in the explicit identifier section first. If no match is found, it continues the search in the group of identifier section. By this order it can be guaranteed that in case where an explicit identifier match is found, the succeeding software can directly proceed on this certain message whereas in case of a group of identifier match the succeeding software needs more steps to identify the message.

20.17.4 Configuration example 4

Suppose that the five Acceptance Filter address registers contain the values shown in the third column below. In this case each table contains the decimal number of words and entries shown in the next two columns, and the ID Index field of the CANRFS register can return the decimal values shown in the column ID Indexes for CAN messages whose Identifiers match the entries in that table.

Table 478. Example of Acceptance Filter Tables and ID index Values

Table	Register	Value	# Words	# Entire	ID Indexes
Standard Individual	SFF_sa	0x040	8_{10}	16_{10}	$0\text{-}15_{10}$
Standard Group	SFF_GRP_sa	0x060	4_{10}	4_{10}	$16\text{-}19_{10}$
Extended Individual	EFF_sa	0x070	8_{10}	16_{10}	$20\text{-}55_{10}$
Extended Group	EFF_GRP_sa	0x100	8_{10}	16_{10}	$56\text{-}57_{10}$
	ENDofTable	0x110			

20.17.5 Configuration example 5

Figure 99 below is a more detailed and graphic example of the address registers, table layout, and ID Index values. It shows:

- A Standard Individual table starting at the start of Acceptance Filter RAM and containing 26 Identifiers, followed by:
- A Standard Group table containing 12 ranges of Identifiers, followed by:
- An Extended Individual table containing 3 Identifiers, followed by:
- An Extended Group table containing 2 ranges of Identifiers.

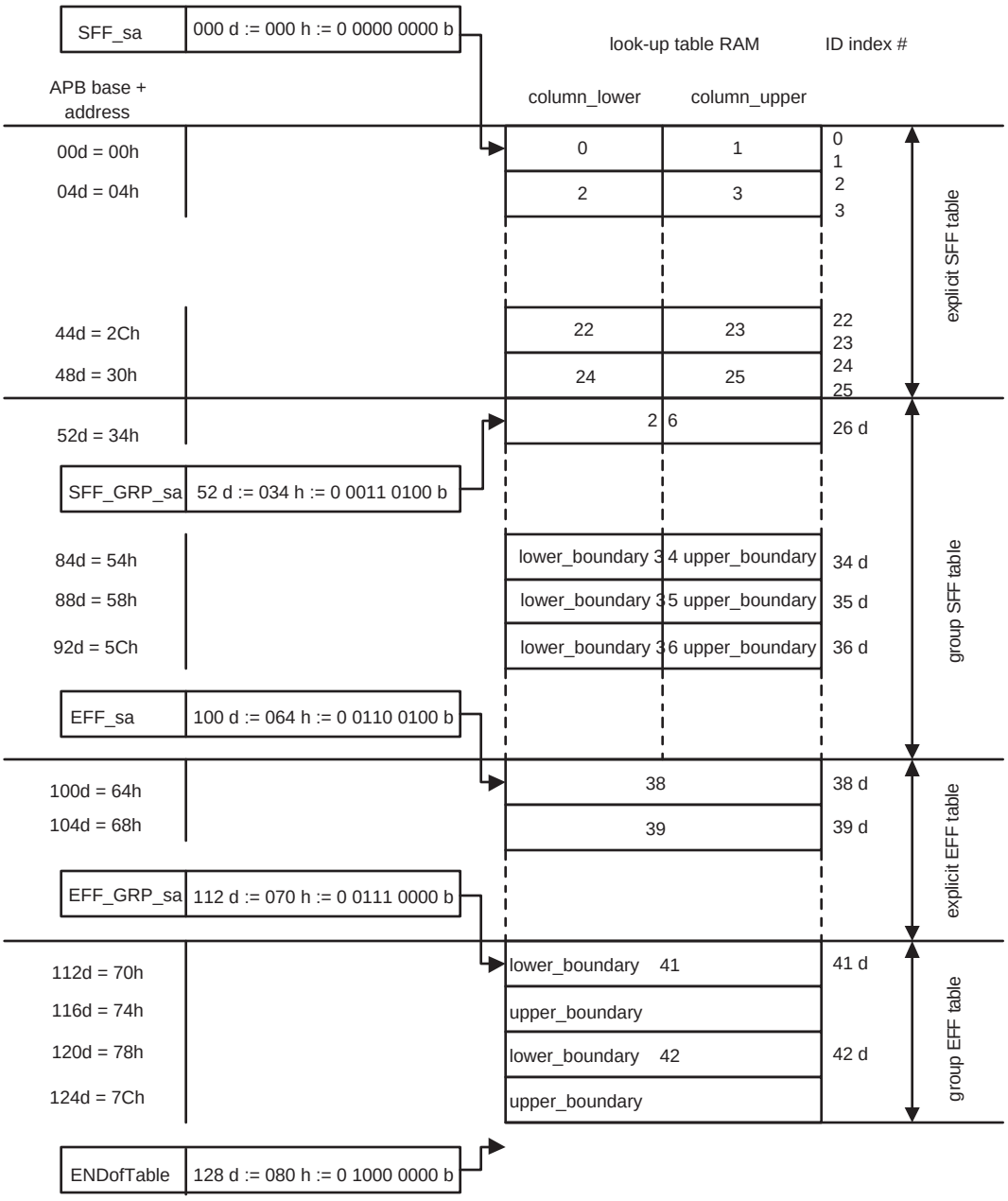


Fig 99. Detailed example of acceptance filter tables and ID index values

20.17.6 Configuration example 6

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 100.

Table 479. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	not activated
Explicit Standard Frame Format	activated
Group of Standard Frame Format	activated
Explicit Extended Frame Format	activated
Group of Extended Frame Format	activated

Explicit standard frame format identifier section (11-bit CAN ID):

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x00. The end of this section is defined in the SFF_GRP_sa register. In the Explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit.

Group of standard frame format identifier section (11-bit CAN ID):

The start address of the Group of Standard Frame Format section is defined with the SFF_GRP_sa register with the value of 0x10. The end of this section is defined with the EFF_sa register. In the Group of Standard Frame Format section two CAN Identifiers with the same Source CAN Channel (SCC) share one 32-bit word and represent a range of CAN Identifiers to be accepted. Bit 31 down to 16 represents the lower boundary and bit 15 down to 0 represents the upper boundary of the range of CAN Identifiers. All Identifiers within this range (including the boundary identifiers) will be accepted. A whole group can be disabled and not used by the acceptance filter by setting the message disable bit in the upper and lower boundary identifier. To provide memory space for four Groups of Standard Frame Format identifiers, the EFF_sa register value is set to 0x20. The identifier group with the Index 9 of this section is not used and therefore disabled.

Explicit extended frame format identifier section (29-bit CAN ID, Figure 100)

The start address of the Explicit Extended Frame Format section is defined with the EFF_sa register with the value of 0x20. The end of this section is defined with the EFF_GRP_sa register. In the explicit Extended Frame Format section only one CAN Identifier with its Source CAN Channel (SCC) is programmed per address line. To provide memory space for four Explicit Extended Frame Format identifiers, the EFF_GRP_sa register value is set to 0x30.

Group of extended frame format identifier section (29-bit CAN ID, Figure 100)

The start address of the Group of Extended Frame Format is defined with the EFF_GRP_sa register with the value of 0x30. The end of this section is defined with the End of Table address register (ENDofTable). In the Group of Extended Frame Format section the boundaries are programmed with a pair of address lines; the first is the lower boundary, the second the upper boundary. To provide memory space for two Groups of Extended Frame Format Identifiers, the ENDofTable register value is set to 0x40.

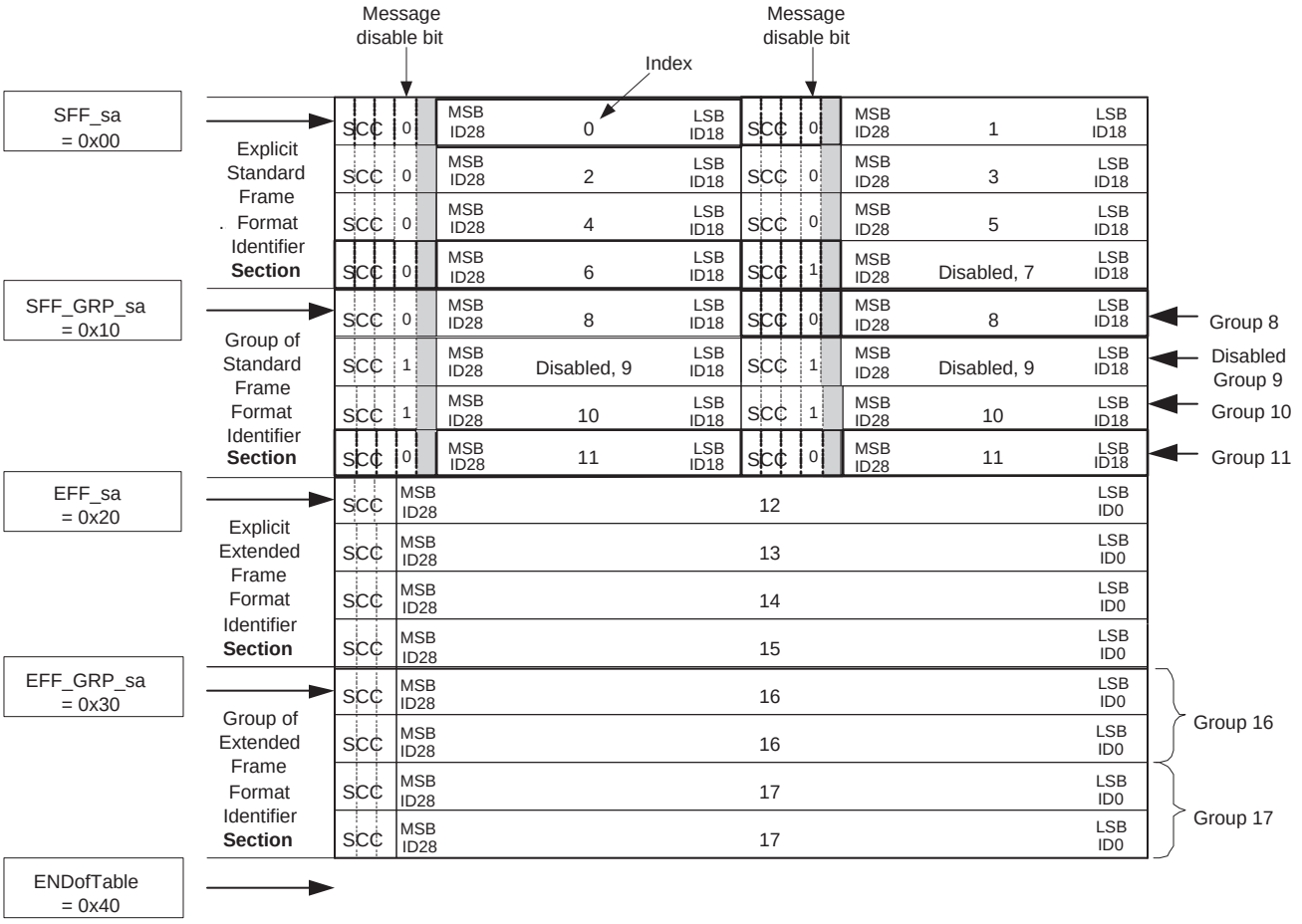


Fig 100. ID Look-up table configuration example (no FullCAN)

20.17.7 Configuration example 7

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 101.

This example uses a typical configuration in which FullCAN as well as Explicit Standard Frame Format messages are defined. As described in Section 20.15.1 “Acceptance filter search algorithm”, acceptance filtering takes place in a certain order. With the enabled FullCAN section, the identifier screening process of the acceptance filter starts always in the FullCAN section first, before it continues with the rest of enabled sections.e disabled.

Table 480. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	activated and enabled
Explicit Standard Frame Format	activated
Group of Standard Frame Format	not activated
Explicit Extended Frame Format	not activated
Group of Extended Frame Format	not activated

FullCAN explicit standard frame format identifier section (11-bit CAN ID)

The start address of the FullCAN Explicit Standard Frame Format Identifier section is (automatically) set to 0x00. The end of this section is defined in the SFF_sa register. In the FullCAN ID section only identifiers of FullCAN Object are stored for acceptance filtering. In this section two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. The FullCAN Object data for each defined identifier can be found in the FullCAN Message Object section. In case of an identifier match during the acceptance filter process, the received FullCAN message object data is moved from the Receive Buffer of the appropriate CAN Controller into the FullCAN Message Object section. To provide memory space for eight FullCAN, Explicit Standard Frame Format identifiers, the SFF_sa register value is set to 0x10. The identifier with the Index 1 of this section is not used and therefore disabled.

Explicit standard frame format identifier section (11-bit CAN ID)

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x10. The end of this section is defined in the End of Table address register (ENDofTable). In the explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channel (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. To provide memory space for eight Explicit Standard Frame Format identifiers, the ENDofTable register value is set to 0x20.

FullCAN message object data section

The start address of the FullCAN Message Object Data section is defined with the ENDofTable register. The number of enabled FullCAN identifiers is limited to the available memory space in the FullCAN Message Object Data section. Each defined FullCAN Message needs three address lines for the Message Data in the FullCAN Message Object Data section. The FullCAN Message Object section is organized in that way, that each Index number of the FullCAN Identifier section corresponds to a Message Object Number in the FullCAN Message Object section.

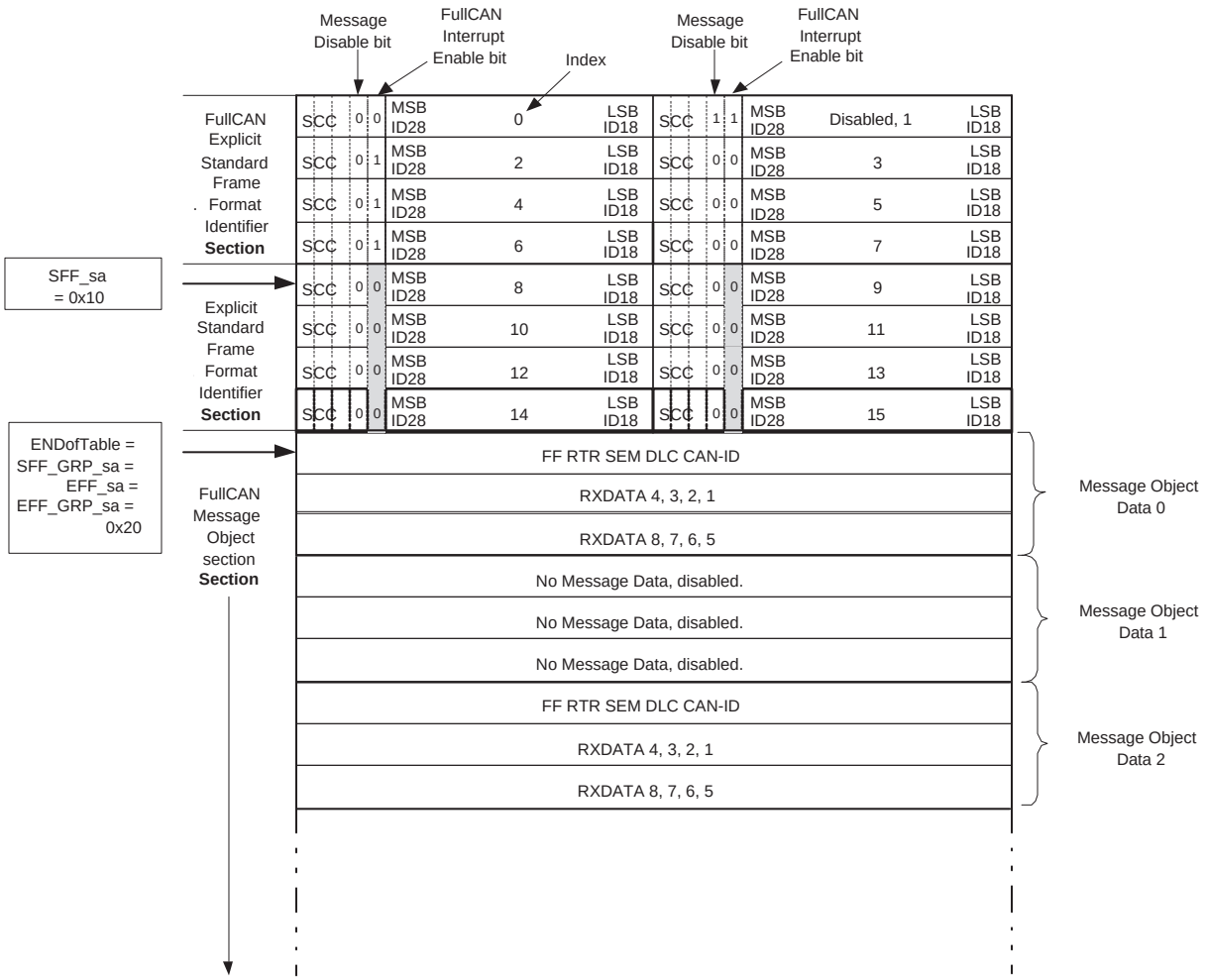


Fig 101. ID Look-up table configuration example (FullCAN activated and enabled)

20.17.8 Look-up table programming guidelines

All identifier sections of the ID Look-up Table have to be programmed in such a way, that each active section is organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

In cases, where a syntax error in the ID Look-up Table is encountered, the Look-up Table address of the incorrect line is made available in the Look-up Table Error Address Register (LUTerrAd).

The reporting process in the Look-up Table Error Address Register (LUTerrAd) is a “run-time” process. Only those address lines with syntax error are reported, which were passed through the acceptance filtering process.

The following general rules for programming the Look-up Table apply:

- Each section has to be organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) in conjunction with the CAN Identifier (there is no exception for disabled identifiers).
- The upper and lower bound in a Group of Identifiers definition has to be from the same Source CAN Channel.
- To disable a Group of Identifiers the message disable bit has to be set for both, the upper and lower bound.

21.1 Basic configuration

The three SSP interfaces, SSP0, SSP1, and SSP2 are configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCSSP0 to enable SSP0 and bit PCSSP1 to enable SSP1.

   Remark: On reset, SSP interfaces 0 and 1 are enabled (PCSSP0/1 = 1), while SSP2 is disabled (PCSSP2 = 0).
2. Peripheral clock: The SSPs operate from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5. In master mode, the clock must be scaled down (see Section 21.6.5).
3. Pins: Select the SSP pins and pin modes through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: Interrupts are enabled in the SSP0IMSC register for SSP0 and SSP1IMSC register for SSP1 Table 488. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register, see Table 51.
5. Initialization: There are two control registers for each of the SSP ports to be configured: SSP0CR0 and SSP0CR1 for SSP0, SSP1CR0 and SSP1CR1 for SSP1, SSP2CR0 and SSP2CR1 for SSP2. See Section 21.6.1 and Section 21.6.2.
6. DMA: The Rx and Tx FIFOs of the SSP interfaces can be connected to the GPDMA controller (see Section 21.6.10). For GPDMA system connections, see Table 696.

21.2 Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses.
- Synchronous Serial Communication.
- Master or slave operation.
- 8 frame FIFOs for both transmit and receive.
- 4 to 16 bit data frame.
- DMA transfers supported by GPDMA.

21.3 Description

The SSP is a Synchronous Serial Port (SSP) controller capable of operation on a SPI, 4-wire SSI, or Microwire bus. It can interact with multiple masters and slaves on the bus. Only a single master and a single slave can communicate on the bus during a given data transfer. Data transfers are in principle full duplex, with frames of 4 to 16 bits of data flowing from the master to the slave and from the slave to the master. In practice it is often the case that only one of these data flows carries meaningful data.

Three Synchronous Serial Port controllers are provided -- SSP0, SSP1, and SSP2.

21.4 Pin descriptions

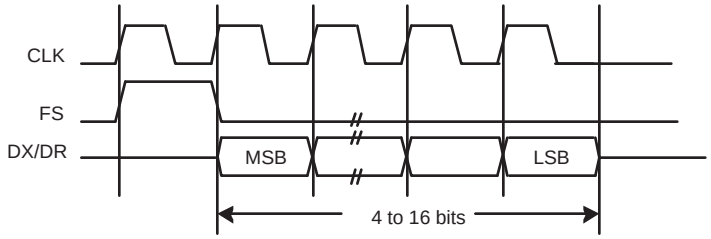
Table 481. SSP pin descriptions

Pin Name	Type	Interface pin name/function			Pin Description
		SPI	SSI	Microwire	
SCK0/1/2	I/O	SCK	CLK	SK	**Serial Clock.** SCK/CLK/SK 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, otherwise it is always active-high. SCK only switches during a data transfer. Any other time, the SSPn interface either holds it in its inactive state, or does not drive it (leaves it in high-impedance state). When this pin is an 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.
SSEL0/1/2	I/O	SSEL	FS	CS	**Frame Sync/Slave Select.** When the SSPn interface is a bus master, it drives this signal to an active state before the start of serial data, and then releases it to an inactive state after the serial data has been sent. The active state of this signal can be high or low depending upon the selected bus and mode. When the SSPn is a bus slave, this signal qualifies the presence of data from the Master, according to the protocol in use. When there is just one bus master and one bus slave, the Frame Sync or Slave Select signal from the Master can be connected directly to the slave's corresponding input. When there is more than one slave on the bus, further qualification of their Frame Select/Slave Select inputs will typically be necessary to prevent more than one slave from responding to a transfer.
MISO0/1/2	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**Master In Slave Out.** The MISO signal transfers serial data from the slave to the master. When the SSPn is a slave, serial data is output on this signal. When the SSPn is a master, it clocks in serial data from this signal. When the SSPn is a slave and is not selected by FS/SSEL, it does not drive this signal (leaves it in high-impedance state).
MOSI0/1/2	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**Master Out Slave In.** The MOSI signal transfers serial data from the master to the slave. When the SSPn is a master, it outputs serial data on this signal. When the SSPn is a slave, it clocks in serial data from this signal.

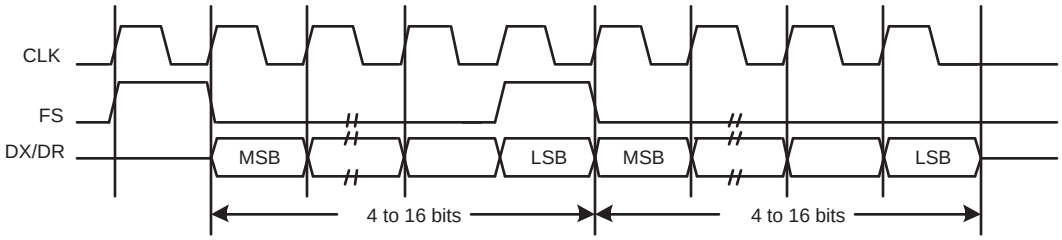
21.5 Bus description

21.5.1 Texas Instruments synchronous serial frame format

Figure 102 shows the 4-wire Texas Instruments synchronous serial frame format supported by the SSP module.



a. Single frame transfer



b. Continuous/back-to-back frames transfer

Fig 102. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer

For device configured as a master in this mode, CLK and FS are forced LOW, and the transmit data line DX is tri-stated whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, FS is pulsed HIGH for one CLK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of CLK, the MSB of the 4-bit to 16-bit data frame is shifted out on the DX pin. Likewise, the MSB of the received data is shifted onto the DR pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each CLK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of CLK after the LSB has been latched.

21.5.2 SPI frame format

The SPI interface is a four-wire interface where the SSEL signal behaves as a slave select. The main feature of the SPI format is that the inactive state and phase of the SCK signal are programmable through the CPOL and CPHA bits within the SSPCR0 control register.

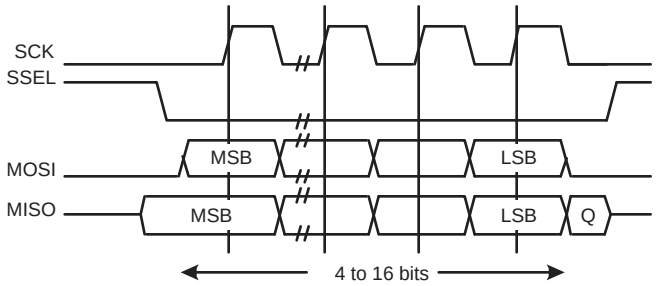
21.5.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

When the CPOL clock polarity control bit is 0, it produces a steady state low value on the SCK pin. If the CPOL clock polarity control bit is 1, a steady state high value is placed on the CLK pin when data is not being transferred.

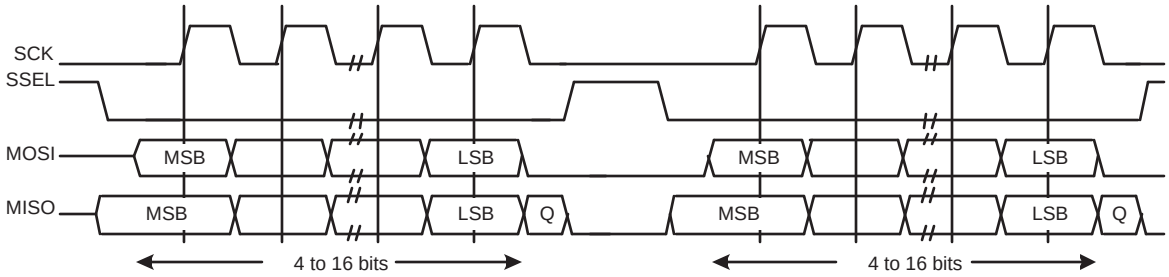
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is 0, data is captured on the first clock edge transition. If the CPHA clock phase control bit is 1, data is captured on the second clock edge transition.

21.5.2.2 SPI format with CPOL=0,CPHA=0

Single and continuous transmission signal sequences for SPI format with CPOL = 0, CPHA = 0 are shown in Figure 103.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 103. SPI frame format with CPOL=0 and CPHA=0 (a) Single and b) Continuous Transfer)

In this configuration, during idle periods:

- The CLK signal is forced LOW.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. This causes slave data to be enabled onto the MISO input line of the master. Master's MOSI is enabled.

One half SCK period later, valid master data is transferred to the MOSI pin. Now that both the master and slave data have been set, the SCK master clock pin goes HIGH after one further half SCK period.

The data is now captured on the rising and propagated on the falling edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

21.5.2.3 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 104, which covers both single and continuous transfers.

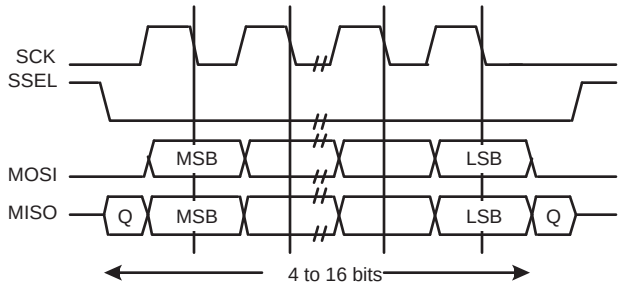


Fig 104. SPI frame format with CPOL=0 and CPHA=1

In this configuration, during idle periods:

- The CLK signal is forced LOW.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI pin is enabled. After a further one half SCK period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SCK is enabled with a rising edge transition.

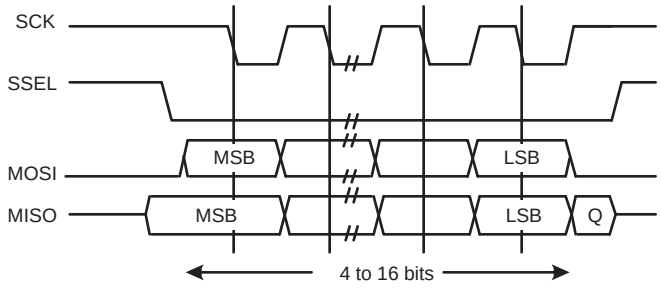
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

In the case of a single word transfer, after all bits have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

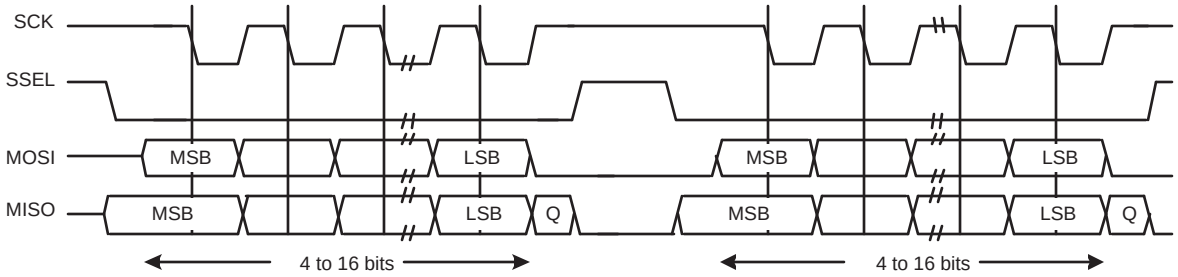
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

21.5.2.4 SPI format with CPOL = 1,CPHA = 0

Single and continuous transmission signal sequences for SPI format with CPOL=1, CPHA=0 are shown in Figure 105.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 105. SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer)

In this configuration, during idle periods:

- The CLK signal is forced HIGH.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW, which causes slave data to be immediately transferred onto the MISO line of the master. Master's MOSI pin is enabled.

One half period later, valid master data is transferred to the MOSI line. Now that both the master and slave data have been set, the SCK master clock pin becomes LOW after one further half SCK period. This means that data is captured on the falling edges and be propagated on the rising edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

21.5.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 106, which covers both single and continuous transfers.

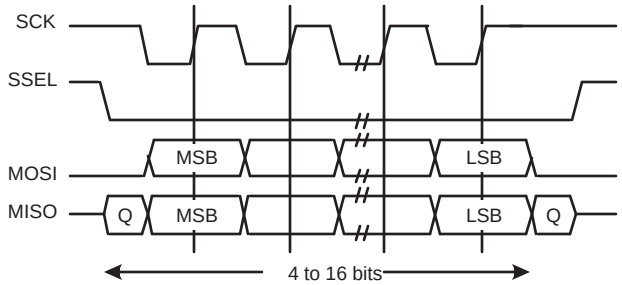


Fig 106. SPI Frame Format with CPOL = 1 and CPHA = 1

In this configuration, during idle periods:

- The CLK signal is forced HIGH.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI is enabled. After a further one half SCK period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SCK is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SCK signal.

After all bits have been transferred, in the case of a single word transmission, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured. For continuous back-to-back transmissions, the SSEL pins remains in its active LOW state, until the final bit of the last word has been captured, and then returns to its idle state as described above. In general, for continuous back-to-back transfers the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

21.5.3 National Semiconductor Microwire frame format

Figure 107 shows the Microwire frame format for a single frame. Figure 108 shows the same format when back-to-back frames are transmitted.

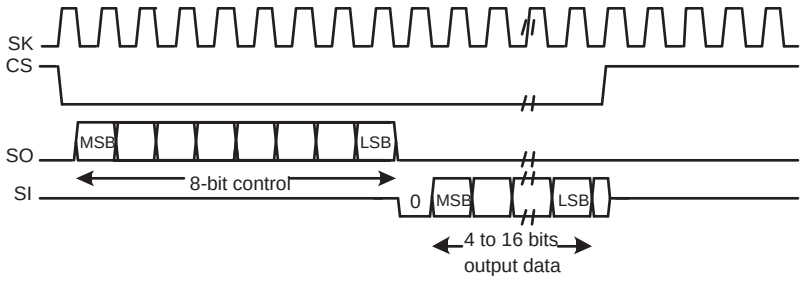


Fig 107. Microwire frame format (single transfer)

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmission, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- The SK signal is forced LOW.
- CS is forced HIGH.
- The transmit data line SO is arbitrarily forced LOW.

A transmission is triggered by writing a control byte to the transmit FIFO.The falling edge of CS causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SO pin. CS remains LOW for the duration of the frame transmission. The SI pin remains tri-stated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SK. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSP. Each bit is driven onto SI line on the falling edge of SK. The SSP in turn latches each bit on the rising edge of SK. At the end of the frame, for single transfers, the CS signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tri-state the receive line either on the falling edge of SK after the LSB has been latched by the receive shiftier, or when the CS pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the CS line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SK, after the LSB of the frame has been latched into the SSP.

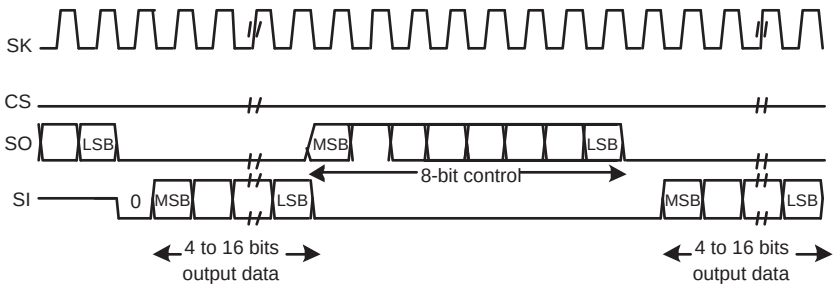


Fig 108. Microwire frame format (continuos transfers)

21.5.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode

In the Microwire mode, the SSP slave samples the first bit of receive data on the rising edge of SK after CS has gone LOW. Masters that drive a free-running SK must ensure that the CS signal has sufficient setup and hold margins with respect to the rising edge of SK.

Figure 109 illustrates these setup and hold time requirements. With respect to the SK rising edge on which the first bit of receive data is to be sampled by the SSP slave, CS must have a setup of at least two times the period of SK on which the SSP operates. With respect to the SK rising edge previous to this edge, CS must have a hold of at least one SK period.

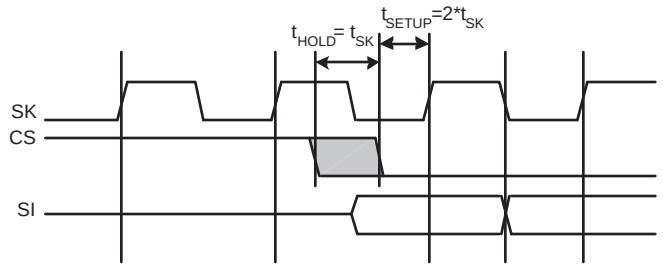


Fig 109. Microwire frame format setup and hold details

21.6 Register description

Table 482. Register overview: SSP (base address 0x4008 8000 (SSP0), 0x4003 0000 (SSP1), 0x400A C000 (SSP2))

Generic Name	Access	Address offset	Description	Reset Value[1]	Table
CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0	483
CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0	484
DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0	485
SR	RO	0x00C	Status Register		486
CPSR	R/W	0x010	Clock Prescale Register	0	487
IMSC	R/W	0x014	Interrupt Mask Set and Clear Register	0	488
RIS	R/W	0x018	Raw Interrupt Status Register		489
MIS	R/W	0x01C	Masked Interrupt Status Register	0	490
ICR	R/W	0x020	SSPICR Interrupt Clear Register	NA	491
DMACR	R/W	0x024	DMA Control Register	0	492

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

21.6.1 SSPn Control Register 0

This register controls the basic operation of the SSP controller.

Table 483: SSPn Control Register 0 (CR0 - address 0x4008 8000 (SSP0), 0x4003 0000 (SSP1) , 0x400A C000 (SSP2)) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0011	4-bit transfer	
		0100	5-bit transfer	
		0101	6-bit transfer	
		0110	7-bit transfer	
		0111	8-bit transfer	
		1000	9-bit transfer	
		1001	10-bit transfer	
		1010	11-bit transfer	
		1011	12-bit transfer	
		1100	13-bit transfer	
		1101	14-bit transfer	
		1110	15-bit transfer	
		1111	16-bit transfer	
5:4	FRF		Frame Format.	00
		00	SPI	
		01	TI	
		10	Microwire	
		11	This combination is not supported and should not be used.	
6	CPOL		Clock Out Polarity. This bit is only used in SPI mode.	0
		0	SSP controller maintains the bus clock low between frames.	
		1	SSP controller maintains the bus clock high between frames.	
7	CPHA		Clock Out Phase. This bit is only used in SPI mode.	0
		0	SSP controller captures serial data on the first clock transition of the frame, that is, the transition **away from** the inter-frame state of the clock line.	
		1	SSP controller captures serial data on the second clock transition of the frame, that is, the transition **back to** the inter-frame state of the clock line.	
15:8	SCR		Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR $\times$ [SCR+1]).	0
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

21.6.2 SSPn Control Register 1

This register controls certain aspects of the operation of the SSP controller.

Table 484: SSPn Control Register 1 (CR1 - address 0x4008 8004 (SSP0), 0x4003 0004 (SSP1), 0x400A C004 (SSP2)) bit description

Bit	Symbol	Value	Description	Reset Value
0	LBM		Loop Back Mode.	0
		0	During normal operation.	
		1	Serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively).	
1	SSE		SSP Enable.	0
		0	The SSP controller is disabled.	
		1	The SSP controller will interact with other devices on the serial bus. Software should write the appropriate control information to the other SSP registers and interrupt controller registers, before setting this bit.	
2	MS		Master/Slave Mode.This bit can only be written when the SSE bit is 0.	0
		0	The SSP controller acts as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line.	
		1	The SSP controller acts as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines.	
3	SOD		Slave Output Disable. This bit is relevant only in slave mode (MS = 1). If it is 1, this blocks this SSP controller from driving the transmit data line (MISO).	0
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

21.6.3 SSPn Data Register

Software can write data to be transmitted to this register, and read data that has been received.

Table 485: SSPn Data Register (DR - address 0x4008 8008 (SSP0), 0x4003 0008 (SSP1), 0x400A C008 (SSP2)) bit description

Bit	Symbol	Description	Reset Value
15:0	DATA	**Write:** software can write data to be sent in a future frame to this register whenever the TNF bit in the Status register is 1, indicating that the Tx FIFO is not full. If the Tx FIFO was previously empty and the SSP controller is not busy on the bus, transmission of the data will begin immediately. Otherwise the data written to this register will be sent as soon as all previous data has been sent (and received). If the data length is less than 16 bits, software must right-justify the data written to this register. **Read:** software can read data from this register whenever the RNE bit in the Status register is 1, indicating that the Rx FIFO is not empty. When software reads this register, the SSP controller returns data from the least recent frame in the Rx FIFO. If the data length is less than 16 bits, the data is right-justified in this field with higher order bits filled with 0s.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

21.6.4 SSPn Status Register

This read-only register reflects the current status of the SSP controller.

Table 486: SSPn Status Register (SR - address 0x4008 800C (SSP0), 0x4003 000C (SSP1), 0x400A C00C (SSP2)) bit description

Bit	Symbol	Description	Reset Value
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
4	BSY	Busy. This bit is 0 if the SSPn controller is idle, or 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
31:5	-	Reserved. The value read from a reserved bit is not defined.	NA

21.6.5 SSPn Clock Prescale Register

This register controls the factor by which the Prescaler divides PCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in SSPnCR0, to determine the bit clock.

Table 487: SSPn Clock Prescale Register (CPSR - address 0x4008 8010 (SSP0), 0x4003 0010 (SSP1), 0x400A C010 (SSP2)) bit description

Bit	Symbol	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which PCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

Important: the SSPnCPSR value must be properly initialized or the SSP controller will not be able to transmit data correctly.

In Slave mode, the SSP clock rate provided by the master must not exceed 1/12 of the peripheral clock selected in Section 3.3.3.5. The content of the SSPnCPSR register is not relevant.

In master mode, $CPSDVSR_{min}$ = 2 or larger (even numbers only).

21.6.6 SSPn Interrupt Mask Set/Clear Register

This register controls whether each of the four possible interrupt conditions in the SSP controller are enabled. Note that ARM uses the word “masked” in the opposite sense from classic computer terminology, in which “masked” meant “disabled”. ARM uses the word “masked” to mean “enabled”. To avoid confusion we will not use the word “masked”.

Table 488: SSPn Interrupt Mask Set/Clear register (IMSC - address 0x4008 8014 (SSP0), 0x4003 0014 (SSP1), 0x400A C014 (SSP2)) bit description

Bit	Symbol	Description	Reset Value
0	RORIM	Software should set this bit to enable interrupt when a Receive Overrun occurs, that is, when the Rx FIFO is full and another frame is completely received. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTIM	Software should set this bit to enable interrupt when a Receive Timeout condition occurs. A Receive Timeout occurs when the Rx FIFO is not empty, and has not been read for a period of 32 bit times.	0
2	RXIM	Software should set this bit to enable interrupt when the Rx FIFO is at least half full.	0
3	TXIM	Software should set this bit to enable interrupt when the Tx FIFO is at least half empty.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

21.6.7 SSPn Raw Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted, regardless of whether or not the interrupt is enabled in the SSPnIMSC.

Table 489: SSPn Raw Interrupt Status register (RIS - address 0x4008 8018 (SSP0), 0x4003 0018 (SSP1), 0x400A C018 (SSP2)) bit description

Bit	Symbol	Description	Reset Value
0	RORRIS	This bit is 1 if another frame was completely received while the RxFIFO was full. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTRIS	This bit is 1 if the Rx FIFO is not empty, and has not been read for a period of 32 bit times.	0
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
31:4	-	Reserved. The value read from a reserved bit is not defined.	NA

21.6.8 SSPn Masked Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted and enabled in the SSPnIMSC. When an SSP interrupt occurs, the interrupt service routine should read this register to determine the cause(s) of the interrupt.

Table 490: SSPn Masked Interrupt Status register (MIS - address 0x4008 801C (SSP0), 0x4003 001C (SSP1), 0x400A C01C (SSP2)) bit description

Bit	Symbol	Description	Reset Value
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0
1	RTMIS	This bit is 1 if the Rx FIFO is not empty, has not been read for a period of 32 bit times, and this interrupt is enabled.	0
2	RXMIS	This bit is 1 if the Rx FIFO is at least half full, and this interrupt is enabled.	0
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, and this interrupt is enabled.	0
31:4	-	Reserved. The value read from a reserved bit is not defined.	NA

21.6.9 SSPn Interrupt Clear Register

Software can write one or more one(s) to this write-only register, to clear the corresponding interrupt condition(s) in the SSP controller. Note that the other two interrupt conditions can be cleared by writing or reading the appropriate FIFO, or disabled by clearing the corresponding bit in SSPnIMSC.

Table 491: SSPn interrupt Clear Register (ICR - address 0x4008 8020 (SSP0), 0x4003 0020 (SSP1) , 0x400A C020 (SSP2)) bit description

Bit	Symbol	Description	Reset Value
0	RORIC	Writing a 1 to this bit clears the “frame was received when RxFIFO was full” interrupt.	NA
1	RTIC	Writing a 1 to this bit clears the "Rx FIFO was not empty and has not been read for a period of 32 bit times" interrupt.	NA
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

21.6.10 SSPn DMA Control Register

The SSPnDMACR register is the DMA control register. It is a read/write register.

Table 492: SSPn DMA Control Register (DMACR - address 0x4008 8024 (SSP0), 0x4003 0024 (SSP1), 0x400A C024 (SSP2)) bit description

Bit	Symbol	Description	Reset Value
0	RXDMAE	Receive DMA Enable. When this bit is set to one 1, DMA for the receive FIFO is enabled, otherwise receive DMA is disabled.	0
1	TXDMAE	Transmit DMA Enable. When this bit is set to one 1, DMA for the transmit FIFO is enabled, otherwise transmit DMA is disabled	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

22.1 Basic configuration

The I2C0/1/2 interfaces are configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCI2C0/1/2.

   Remark: On reset, all I2C interfaces are enabled (PCI2C0/1/2 = 1).

2. Peripheral clock: The I2C interfaces operate from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.

3. Pins: Select I2C0, I2C1, or I2C2 pins and pin modes using the relevant IOCON registers (See Section 7.4.1).

   Remark: The pins P0[27] and P0[28] are specialized open-drain I2C pins that support fully compliant fast mode (400 kHz) and standard mode (100 kHz) I2C. These pins have no on-chip pull-up devices at all and must always be pulled up externally when they are outputs (per the I2C bus specification). Pins P5[2] and P5[3] are similar, but in addition, also support Fast Mode Plus (1 MHz) I2C. Both sets of pins have somewhat different configuration options than most port pins, see Section 7.4.1 for details. For all other pins that can be used for I2C communication, see the Remark below.

   Remark: I2C pins that do not use specialized I2C pads (as identified in Table 75) can be configured to an open-drain mode via the relevant IOCON registers, and can be used with fast mode (400 kHz) and standard mode (100 kHz) I2C. These pins do not include an analog filter to suppress line glitches, but a similar function is performed by the digital filter in the I2C block itself. These pins should be configured as: no pull-up, no pull-down, open drain mode.

4. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

5. Initialization: see Section 22.9.8.1 and Section 22.10.1.

22.2 Features

- Supports 1 MHz Fast Mode Plus (some pinouts of I^2C0 only), 400 kHz fast mode, and 100 kHz standard mode.
- Standard I^2C compliant bus interfaces may be configured as Master, Slave, or Master/Slave.
- Arbitration is handled between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock allows adjustment of I^2C transfer rates.
- Data transfer is bidirectional between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization is used as a handshake mechanism to suspend and resume serial transfer.
- Optional recognition of up to 4 distinct slave addresses.
- Monitor mode allows observing all I^2C-bus traffic, regardless of slave address, without affecting the actual I^2C-bus traffic.

22.3 Applications

Interfaces to external I^2C standard parts, such as serial RAMs, LCDs, tone generators, other microcontrollers, etc. Can also be used as a diagnostic/test bus.

22.4 Description

A typical I^2C-bus configuration is shown in Figure 110. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^2C-bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte, unless the slave device is unable to accept more data.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I^2C-bus will not be released.

The I^2C interfaces are byte oriented and have four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

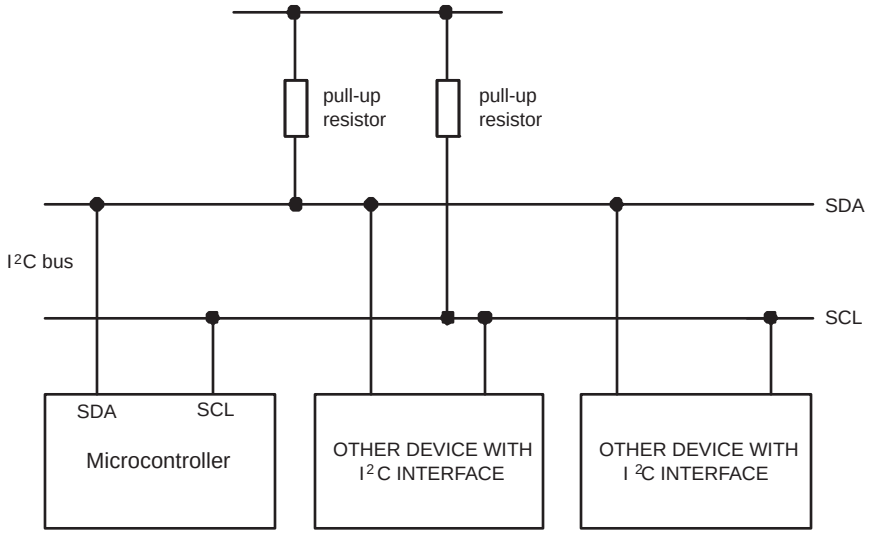


Fig 110. I²C-bus configuration

22.4.1 I²C FAST Mode Plus

Fast Mode Plus is a 1 Mbit/sec transfer rate to communicate with the I²C products which the NXP Semiconductors is now providing.

In order to use Fast Mode Plus, the I²C0 pins must be configured, then rates above 400 kHz and up to 1 MHz may be selected, see Table 508. To configure the pins for Fast Mode Plus, the I2CMODE bits in the IOCON_P5_02 and IOCON_P5_03 registers must be set to binary 10, see Section 7.4.1.

22.5 Pin description

Table 493. I^2C Pin Description

Pin	Type	Description
I2C0_SDA	Input/Output	I^2C0 Serial Data
I2C0_SCL	Input/Output	I^2C0 Serial Clock
I2C1_SDA	Input/Output	I^2C1 Serial Data
I2C1_SCL	Input/Output	I^2C1 Serial Clock
I2C2_SDA	Input/Output	I^2C2 Serial Data
I2C2_SCL	Input/Output	I^2C2 Serial Clock

The internal logic of the 3 I^2C interfaces is identical. These interfaces can be brought out to device pins in several ways, some of which have different pin I/O characteristics. I2C0 on pins P0[27] and P0[28] use specialized I^2C pads that support fully spec compliant fast mode and standard mode I^2C. I2C0 on pins P5[2] and P5[3] also use specialized I^2C pads. These pads support Fast Mode Plus in additional to the previously mentioned modes.

Any of the I^2C interfaces brought out to pins other than those just mentioned use standard I/O pins. These pins also support I^2C operation in fast mode and standard mode. The primary difference is that these pins do not include an analog spike suppression filter that exists on the specialized I^2C pads. The I^2C interfaces all include a digital filter that can serve the same purpose.

22.6 I^2C operating modes

In a given application, the I^2C block may operate as a master, a slave, or both. In the slave mode, the I^2C hardware looks for any one of its four slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I^2C block switches to the slave mode immediately and can detect any of its own configured slave addresses in the same serial transfer.

22.6.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 494. I2EN must be set to 1 to enable the I^2C function. If the AA bit is 0, the I^2C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI bit is cleared by writing 1 to the SIC bit in the I2CONCLR register. THe STA bit should be cleared after writing the slave address.

Table 494. I2C0CONSET and I2C1CONSET used to configure Master mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	0	-	-

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I^2C interface will enter master transmitter mode when software sets the STA bit. The I^2C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the I2STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 0x18, 0x20, or 0x38 for the master mode, or 0x68, 0x78, or 0xB0 if the slave mode was enabled (by setting AA to 1). The appropriate actions to be taken for each of these status codes are shown in Table 512 to Table 515.

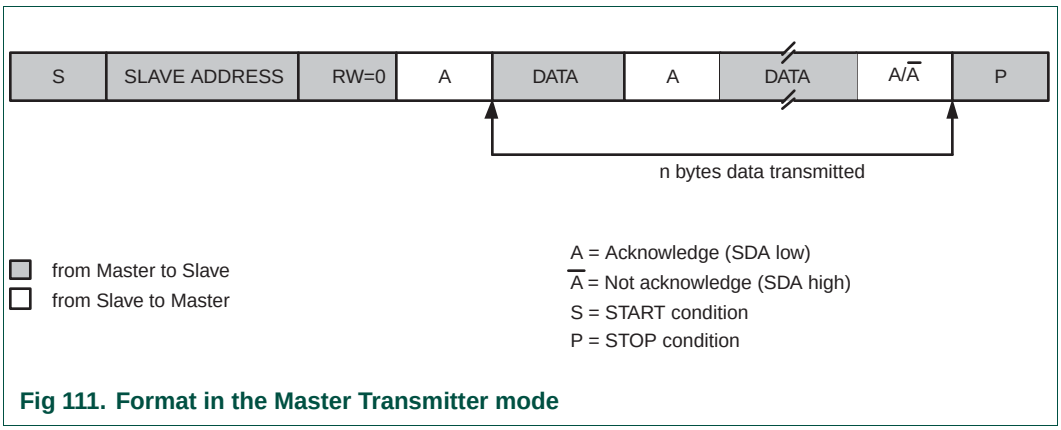


Fig 111. Format in the Master Transmitter mode

22.6.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^2C Data register (I2DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 513.

When the CPU needs to acknowledge a received byte, the AA bit needs to be set accordingly prior to clearing the SI bit and initiating the byte read. When the I^2C interface needs to not acknowledge a received byte, the AA bit needs to be cleared prior to clearing the SI bit and initiating the byte read.

Note that the last received byte is always followed by a "Not Acknowledge" from the I^2C interface so that the master can signal the slave that the reading sequence is finished and that it needs to issue a STOP or repeated START Command. Once the "Not Acknowledge has been sent and the SI bit is set, the I^2C interface can send either a STOP (STO bit is set) or a repeated START (STA bit is set). Then the SI bit is cleared to initiate the requested operation.

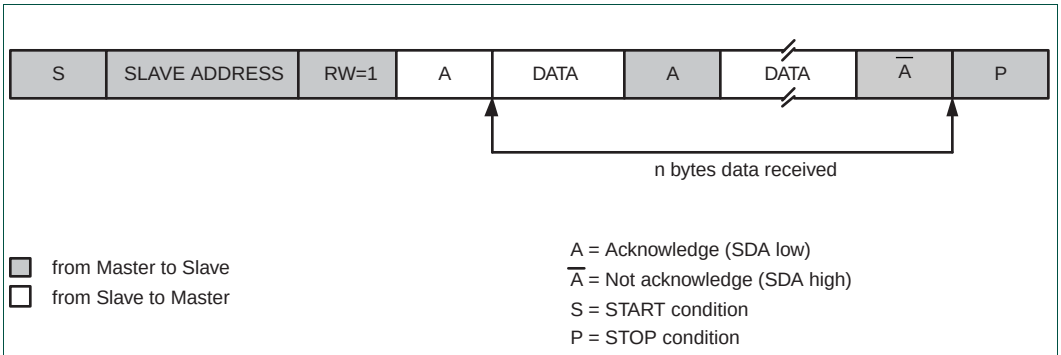


Fig 112. Format of Master Receiver mode

After a repeated START condition, I^2C may switch to the master transmitter mode.

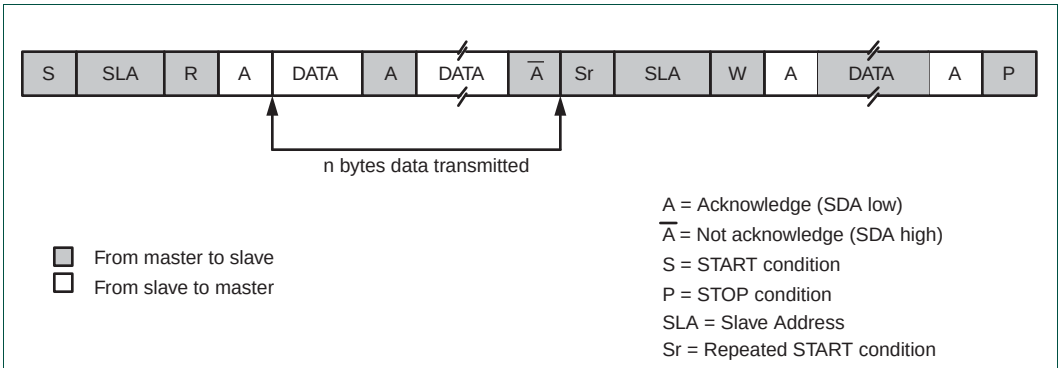


Fig 113. A Master Receiver switches to Master Transmitter after sending repeated START

22.6.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, write any of the Slave Address registers (I2ADR0-3) and Slave Mask registers (I2MASK0-3) and write the I^2C Control Set register (I2CONSET) as shown in Table 495.

Table 495. I2C0CONSET and I2C1CONSET used to configure Slave mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

I2EN must be set to 1 to enable the I^2C function. AA bit must be set to 1 to acknowledge any of its own slave addresses or the General Call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I^2C interface waits until it is addressed by its any of its own slave addresses or General Call address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status register (I2STAT). Refer to Table 514 for the status codes and actions.

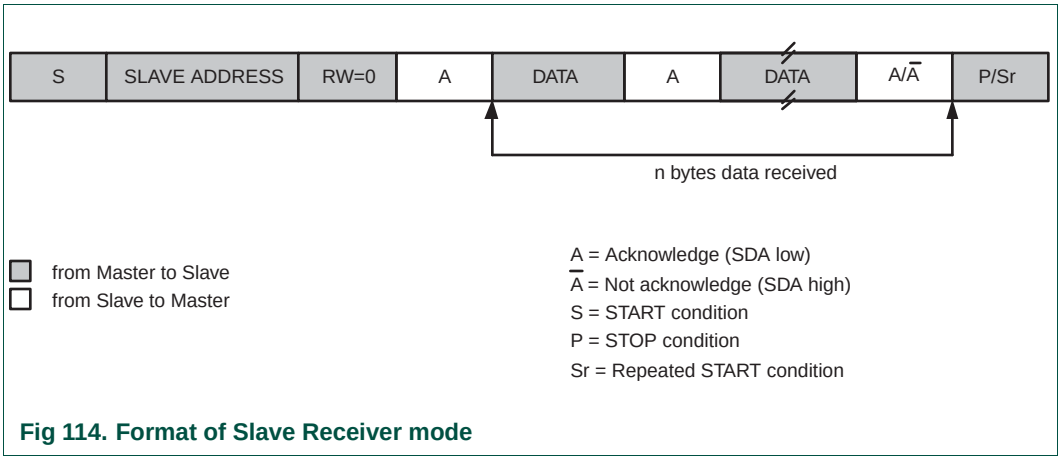


Fig 114. Format of Slave Receiver mode

22.6.4 Slave Transmitter mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for any of its own slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^2C interface switches to the slave mode immediately and can detect any of its own slave addresses in the same serial transfer.

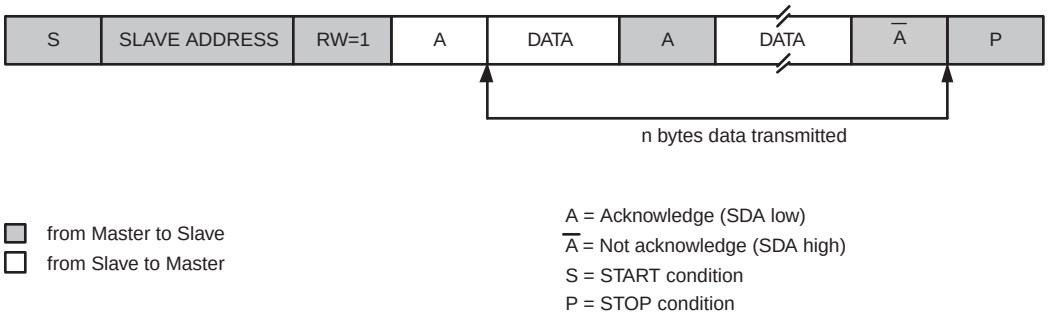


Fig 115. Format of Slave Transmitter mode

22.7 I2C implementation and operation

Figure 116 shows how the on-chip I2C-bus interface is implemented, and the following text describes the individual blocks.

22.7.1 Input filters and output stages

Input signals are synchronized with the internal clock, and spikes shorter than three clocks are filtered out.

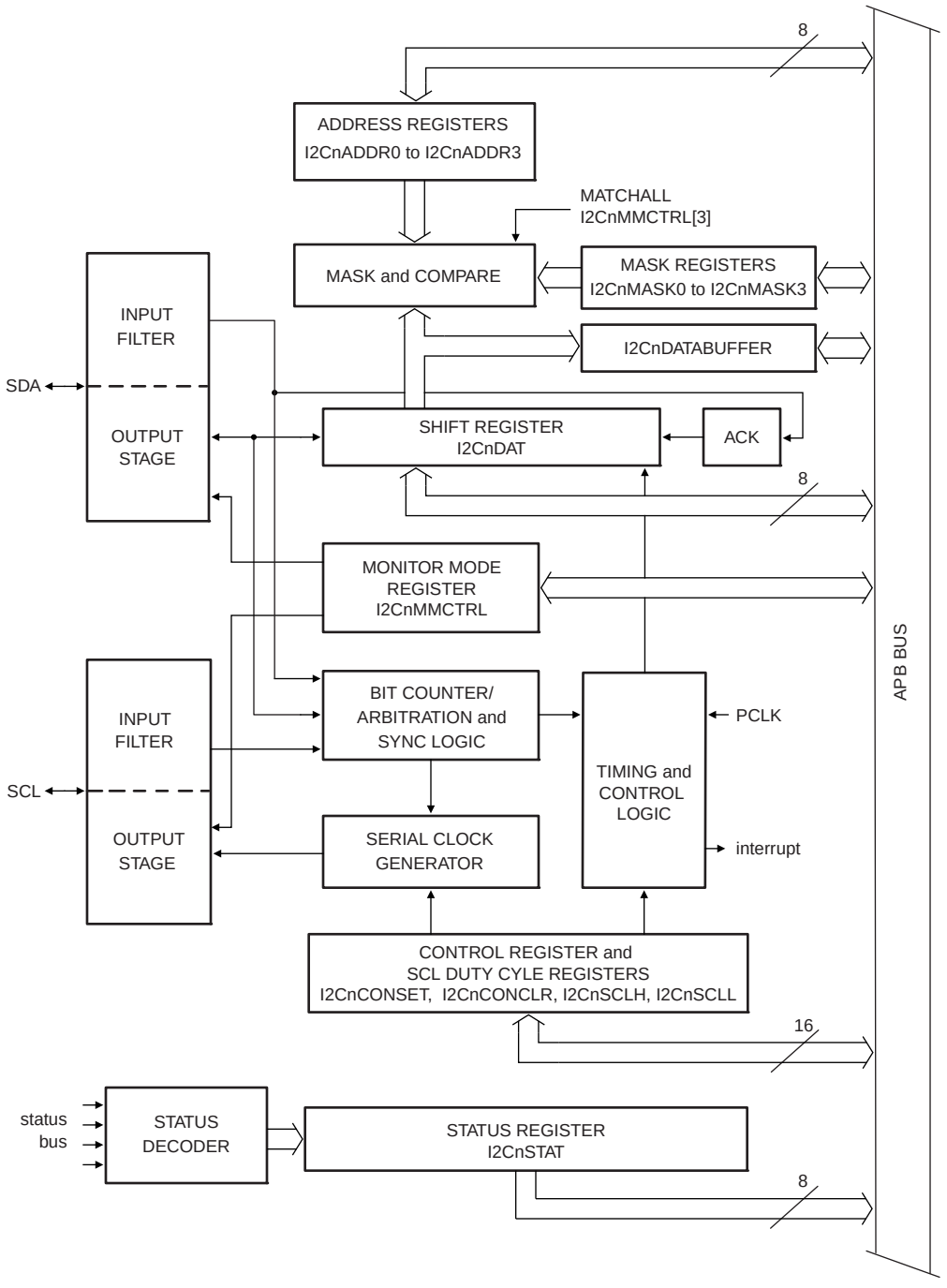


Fig 116. I²C serial interface block diagram

22.7.2 Address Registers, I2ADR0 to I2ADR3

These registers may be loaded with the 7-bit slave address (7 most significant bits) to which the I^2C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable General Call address (0x00) recognition. When multiple slave addresses are enabled, the actual address received may be read from the I2DAT register at the state where the “own slave address” has just been received.

Remark: in the remainder of this chapter, when the phrase “own slave address” is used, it refers to any of the four configured slave addresses after address masking.

22.7.3 Address mask registers, I2MASK0 to I2MASK3

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADRn register associated with that mask register. In other words, bits in an I2ADRn register which are masked are not taken into account in determining an address match.

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine which received address actually caused the match.

22.7.4 Comparator

The comparator compares the received 7-bit slave address with any of the four configured slave addresses in I2ADR0 through I2ADR3 after masking. It also compares the first received 8-bit byte with the General Call address (0x00). If an a match is found, the appropriate status bits are set and an interrupt is requested.

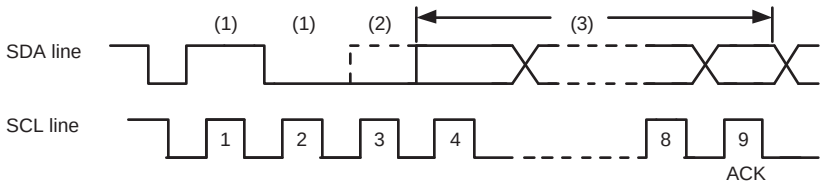
22.7.5 Shift register, I2DAT

This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in I2DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in I2DAT.

22.7.6 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^2C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^2C block immediately changes from master transmitter to slave receiver. The I^2C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^2C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal low. Since this can occur only at the end of a serial byte, the I^2C block generates no further clock pulses. Figure 117 shows the arbitration procedure.



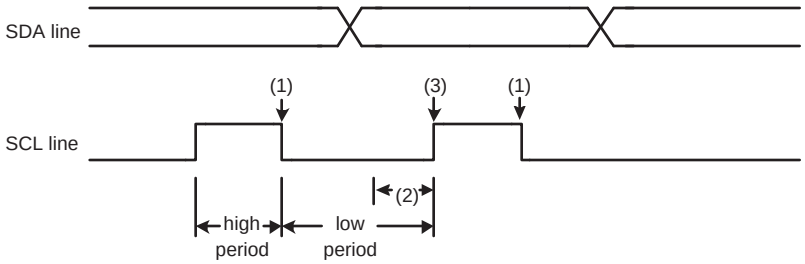
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I2C master by pulling the SDA line low. Arbitration is lost, and this I2C enters Slave Receiver mode.

(3) This I2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 117. Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 118 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I2C has timed a complete low time and released SCL. The I2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 118. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

22.7.7 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I2C block is in the master transmitter or master receiver mode. It is switched off when the I2C block is in a slave mode. The I2C output clock frequency and duty cycle is programmable

via the I2C Clock Control Registers. See the description of the I2CSCLL and I2CSCLH registers for details. The output clock pulses have a duty cycle as programmed unless the bus is synchronizing with other SCL clock sources as described above.

22.7.8 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for I2DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I2C-bus status.

22.7.9 Control register, I2CONSET and I2CONCLR

The I2C control register contains bits used to control the following I2C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I2C control register may be read as I2CONSET. Writing to I2CONSET will set bits in the I2C control register that correspond to ones in the value written. Conversely, writing to I2CONCLR will clear bits in the I2C control register that correspond to ones in the value written.

22.7.10 Status decoder and status register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I2C-bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I2C block are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

22.8 Register description

Table 496. Register overview: I2C-bus interface (base address 0x4001 C000 (I2C0), 0x4005 C000 (I2C1), 0x400A 0000 (I2C2))

Name	Access	Address offset	Description	Reset value[1]	Table
CONSET	R/W	0x000	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I^2C control register is set. Writing a zero has no effect on the corresponding bit in the I^2C control register.	0x00	497
STAT	RO	0x004	**I2C Status Register.** During I^2C operation, this register provides detailed status codes that allow software to determine the next action needed.	0xF8	499
DAT	R/W	0x008	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	0x00	500
ADR0	R/W	0x00C	**I2C Slave Address Register 0.** Contains the 7-bit slave address for operation of the I^2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	503
SCLH	R/W	0x010	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I^2C clock.	0x04	506
SCLL	R/W	0x014	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I^2C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I^2C master and certain times used in slave mode.	0x04	507
CONCLR	WO	0x018	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I^2C control register is cleared. Writing a zero has no effect on the corresponding bit in the I^2C control register.	NA	498
MMCTRL	R/W	0x01C	**Monitor mode control register.**	0x00	501
ADR1	R/W	0x020	**I2C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I^2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	504
ADR2	R/W	0x024	**I2C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I^2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	504
ADR3	R/W	0x028	**I2C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I^2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00	504
DATA_ BUFFER	RO	0x02C	**Data buffer register.** The contents of the 8 MSBs of the I2DAT shift register will be transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus.	0x00	502
MASK0	R/W	0x030	**I2C Slave address mask register 0**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	505

Table 496. Register overview: I2C-bus interface (base address 0x4001 C000 (I2C0), 0x4005 C000 (I2C1), 0x400A 0000 (I2C2))

Name	Access	Address offset	Description	Reset value[1]	Table
MASK1	R/W	0x034	**I2C Slave address mask register 1**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	505
MASK2	R/W	0x038	**I2C Slave address mask register 2**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	505
MASK3	R/W	0x03C	**I2C Slave address mask register 3**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00	505

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

22.8.1 I²C Control Set register

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I²C interface. Writing a one to a bit of this register causes the corresponding bit in the I²C control register to be set. Writing a zero has no effect. Reading this register provides the current values of the control and flag bits.

Table 497. I²C Control Set register (CONSET - addresses 0x4001 C000 (I2C0), 0x4005 C000 (I2C1) , 0x400A 0000 (I2C2)) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	NA
2	AA	Assert acknowledge flag.	0
3	SI	I²C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I²C interface enable.	0
31:7	-	Reserved. Read value is undefined, only zero should be written.	NA

I2EN I²C Interface Enable. When I2EN is 1, the I²C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I²C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I²C block is in the “not addressed” slave state, and the STO bit is forced to “0”.

I2EN should not be used to temporarily release the I²C-bus since, when I2EN is reset, the I²C-bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I²C interface to enter master mode and transmit a START condition or transmit a repeated START condition if it is already in master mode.

When STA is 1 and the I²C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I²C interface is

already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I^2C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the I2CONCLR register. When STA is 0, no START condition or repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I^2C-bus if it the interface is in master mode, and transmits a START condition thereafter. If the I^2C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

STO is the STOP flag. Setting this bit causes the I^2C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I^2C-bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I^2C Interrupt Flag. This bit is set when the I^2C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is HIGH, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in I2CONCLR register. The SI bit should be cleared only after the required bit(s) has (have) been set and the value in I2DAT has been loaded or read.

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A matching address defined by registers I2ADR0 through I2ADR3, masked by I2MASK0 though I2MASK3, has been received.
2. The General Call address has been received while the General Call bit (GC) in I2ADR is set.
3. A data byte has been received while the I^2C is in the master receiver mode.
4. A data byte has been received while the I^2C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (HIGH level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I^2C is in the master receiver mode.
2. A data byte has been received while the I^2C is in the addressed slave receiver mode.

22.8.2 I²C Control Clear register

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I²C interface. Writing a one to a bit of this register causes the corresponding bit in the I²C control register to be cleared. Writing a zero has no effect. I2CONCLR is a write-only register. The value of the related bits can be read from the I2CONSET register.

Table 498. I²C Control Clear register (CONCLR - addresses 0x4001 C018 (I2C0), 0x4005 C018 (I2C1), 0x400A 0018 (I2C2)) bit description

Bit	Symbol	Description
1:0	-	Reserved. Read value is undefined, only zero should be written.
2	AAC	Assert acknowledge Clear bit.
3	SIC	I²C interrupt Clear bit.
4	-	Reserved. Read value is undefined, only zero should be written.
5	STAC	START flag Clear bit.
6	I2ENC	I²C interface Disable bit.
31:7	-	Reserved. Read value is undefined, only zero should be written.

AAC is the Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.

SIC is the I²C Interrupt Clear bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.

STAC is the START flag Clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.

I**2ENC** is the I²C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.

22.8.3 I²C Status register

Each I²C Status register reflects the condition of the corresponding I²C interface. The I²C Status register is read-only.

Table 499. I²C Status register (STAT - addresses 0x4001 C004 (I2C0), 0x4005 C004 (I2C1), 0x400A 0004 (I2C2)) bit description

Bit	Symbol	Description	Reset value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I²C interface.	0x1F
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I²C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 512 to Table 515.

22.8.4 I²C Data register

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

Table 500. I²C Data register (DAT- addresses 0x4001 C008 (I2C0), 0x4005 C008 (I2C1), 0x400A 0008 (2C2)) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds data values that have been received or are to be transmitted.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

22.8.5 I²C Monitor mode control register

This register controls the Monitor mode which allows the I²C module to monitor traffic on the I²C-bus without actually participating in traffic or interfering with the I²C-bus.

Table 501. I²C Monitor mode control register (MMCTRL - addresses 0x4001 C01C (I2C0), 0x4005 C01C (I2C1), 0x400A 001C (I2C2)) bit description

Bit	Symbol	Value	Description	Reset value
0	MM_ENA		Monitor mode enable.	0
		0	Monitor mode disabled.	
		1	The I²C module will enter monitor mode. In this mode the SDA output will be put in high impedance mode. This prevents the I²C module from outputting data of any kind (including ACK) onto the I²C data bus. Depending on the state of the ENA_SCL bit, the output may be also forced high, preventing the module from having control over the I²C clock line.	
1	ENA_SCL		SCL output enable.	0
		0	When this bit is cleared to '0', the SCL output will be forced high when the module is in monitor mode. As described above, this will prevent the module from having any control over the I²C clock line.	
		1	When this bit is set, the I²C module may exercise the same control over the clock line that it would in normal operation. This means that, acting as a slave peripheral, the I²C module can "stretch" the clock line (hold it low) until it has had time to respond to an I²C interrupt.[1]	
2	MATCH_ALL		Select interrupt register match.	0
		0	When this bit is cleared, an interrupt will only be generated when a match occurs to one of the (up-to) four address registers, I2ADR0 through I2ADR3. That is, the module will respond as a normal slave as far as address-recognition is concerned.	
		1	When this bit is set to '1' and the I²C is in monitor mode, an interrupt will be generated on ANY address received. This will enable the part to monitor all traffic on the bus.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	NA

[1] When the ENA_SCL bit is cleared and the I²C no longer has the ability to stretch the clock, interrupt response time becomes important. To give the part more time to respond to an I²C interrupt under these conditions, an I2DATA_BUFFER register is used (Section 22.8.6) to hold received data for a full 9-bit word transmission time.

Remark: The ENA_SCL and MATCH_ALL bits have no effect if the MM_ENA is ‘0’ (i.e. if the module is NOT in monitor mode).

22.8.5.1 Interrupt in Monitor mode

All interrupts will occur as normal when the module is in monitor mode. This means that the first interrupt will occur when an address-match is detected (any address received if the MATCH_ALL bit is set, otherwise an address matching one of the four address registers).

Subsequent to an address-match detection, interrupts will be generated after each data byte is received for a slave-write transfer, or after each byte that the module believes it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

22.8.5.2 Loss of arbitration in Monitor mode

In monitor mode, the I^2C module will not be able to respond to a request for information by the bus master or issue an ACK. Some other slave on the bus will respond instead.

Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected.

22.8.6 I^2C Data buffer register

In monitor mode, the I^2C module may lose the ability to stretch the clock if the ENA_SCL bit is not set. This means that the processor will have a limited amount of time to read the contents of the data received on the bus. If the processor reads the I2DAT shift register, as it ordinarily would, it could have only one bit-time to respond to the interrupt before the received data is overwritten by new data.

To give the processor more time to respond, a new 8-bit, read-only I2DATA_BUFFER register has been added. The contents of the 8 MSBs of the I2DAT shift register are transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus. This means that the processor will have 9 bit transmission times to respond to the interrupt and read the data before it is overwritten.

The processor will still have the ability to read I2DAT directly, as usual, and the behavior of I2DAT will not be altered in any way.

Although the I2DATA_BUFFER register is primarily intended for use in monitor mode with the ENA_SCL bit = ‘0’, it is available for reading at any time under any mode of operation.

Table 502. I^2C Data buffer register (DATA_BUFFER - addresses 0x4001 C02C (I2C0), 0x4005 C02C (I2C1), 0x400A 002C (I2C2)) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds contents of the 8 MSBs of the I2DAT shift register.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

22.8.7 I²C Slave Address registers

These registers are readable and writable and are only used when an I²C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

If these registers contain 0x00, the I²C will not acknowledge any address on the bus. All four registers will be cleared to this disabled state on reset.

Table 503. I²C Slave Address register 0 (ADR0 - address 0x4001 C00C (I2C0), 0x4005 C00C (I2C1), 0x400A 000C (I2C2)) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I²C device address for slave mode.	0x00
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

Table 504. I²C Slave Address registers (ADR[1:3] - address 0x4001 C020 (ADR1) to 0x4001 C028 (ADR3) (I2C0), 0x4005 C020 (ADR1) to 0x4005 C028 (ADR3) (I2C1), 0x400A 0020 (ADR1) to 0x400A 0028 (ADR3) (I2C2)) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I²C device address for slave mode.	0x00
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

22.8.8 I²C Mask registers

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADRn register associated with that mask register. In other words, bits in an I2ADRn register which are masked are not taken into account in determining an address match.

The mask register has no effect on comparison to the General Call address (“0000000”).

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine which received address actually caused the match.

Table 505. I²C Mask registers (MASK[0:3] - address 0x4001 C030 (MASK0) to 0x4001 C03C (MASK3) (I2C0), 0x4005 C030 (MASK0) to 0x4005 C03C (MASK3) (I2C1), 0x400A 0030 (MASK0) to 0x400A 003C (MASK3) (I2C1)) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. User software should not write ones to reserved bits. This bit reads always back as 0.	0
7:1	MASK	Mask bits.	0x00
31:8	-	Reserved. The value read from a reserved bit is not defined.	-

22.8.9 I²C SCL HIGH duty cycle register

Table 506. I²C SCL HIGH Duty Cycle register (SCLH - address 0x4001 C010 (I2C0), 0x4005 C010 (I2C1), 0x400A 0010(I2C2)) bit description

Bit	Symbol	Description	Reset value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

22.8.10 I²C SCL Low duty cycle register

Table 507. I²C SCL Low duty cycle register (SCLL - address 0x4001 C014 (I2C0), 0x4005 C014 (I2C1), 0x400A 0014 (I2C2)) bit description

Bit	Symbol	Description	Reset value
15:0	SCLL	Count for SCL low time period selection.	0x0004
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

22.8.11 Selecting the appropriate I²C data rate and duty cycle

Software must set values for the registers I2CSCLH and I2CSCLL to select the appropriate data rate and duty cycle. I2CSCLH defines the number of PCLK cycles for the SCL HIGH time, I2CSCLL defines the number of PCLK cycles for the SCL low time. The frequency is determined by the following formula (PCLK is the frequency of the peripheral bus APB):

$$I^2C_{bitfrequency} = \frac{PCLKI2C}{I2CSCLH + I2CSCLL} \quad (13)$$

The values for I2CSCLL and I2CSCLH must ensure that the data rate is in the appropriate I²C data rate range. Each register value must be greater than or equal to 4. Table 508 gives some examples of I²C-bus rates based on PCLK frequency and I2CSCLL and I2CSCLH values.

Table 508. Example I²C clock rates[1]

I²C Rate	I2CSCLL + I2CSCLH values at PCLK (MHz)													
	6	8	10	12	16	20	30	40	50	60	70	80	90	100
100 kHz (Standard)	60	80	100	120	160	200	300	400	500	600	700	800	900	1000
400 kHz (Fast Mode)	15	20	25	30	40	50	75	100	125	150	175	200	225	250
1 MHz (Fast Mode Plus)	-	8	10	12	16	20	30	40	50	60	70	80	90	100

[1] The slew rate of the rising edge influences the bit rate due to clock stretching.

I2CSCLL and I2CSCLH values should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I²C-bus specification defines the SCL low time and high time at different values for Fast Mode and Fast Mode Plus I²C.

22.9 Details of I²C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figure 119, Figure 120, Figure 121, Figure 122, and Figure 123. Table 509 lists abbreviations used in these figures when describing the I²C operating modes.

Table 509. Abbreviations used to describe an I²C operation

Abbreviation	Explanation
S	START condition
SLA	7-bit slave address
R	Read bit (HIGH level at SDA)
W	Write bit (LOW level at SDA)
A	Acknowledge bit (LOW level at SDA)
$\overline{A}$	Not acknowledge bit (HIGH level at SDA)
Data	8-bit data byte
P	STOP condition
Sr	Repeated START condition

In Figure 119 to Figure 123, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the I2STAT register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in I2STAT is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in tables from Table 512 to Table 516.

22.9.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 119). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 510. I2CONSET used to initialize Master Transmitter mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	x	-	-

The I^2C rate must also be configured in the I2CSCLL and I2CSCLH registers. I2EN must be set to logic 1 to enable the I^2C block. If the AA bit is reset, the I^2C block will not acknowledge its own slave address or the General Call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I^2C interface cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I^2C logic will now test the I^2C-bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (I2STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads I2DAT with the slave address and the data direction bit (SLA+W). The SI bit in I2CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 512. After a repeated START condition (state 0x10). The I^2C block may switch to the master receiver mode by loading I2DAT with SLA+R).

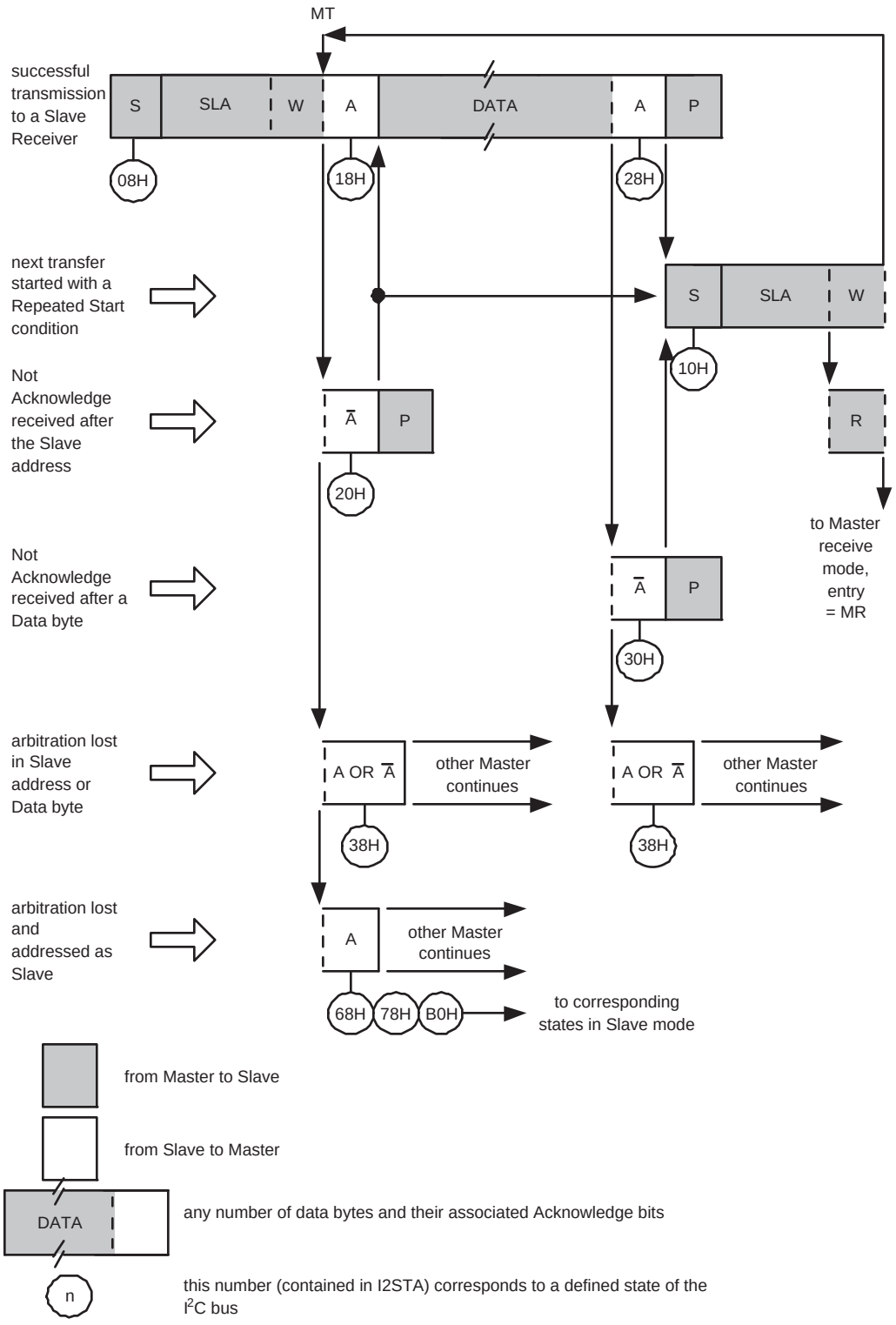


Fig 119. Format and states in the Master Transmitter mode

22.9.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 120). The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load I2DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in I2CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. These are 0x40, 0x48, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 513. After a repeated START condition (state 0x10), the I^2C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

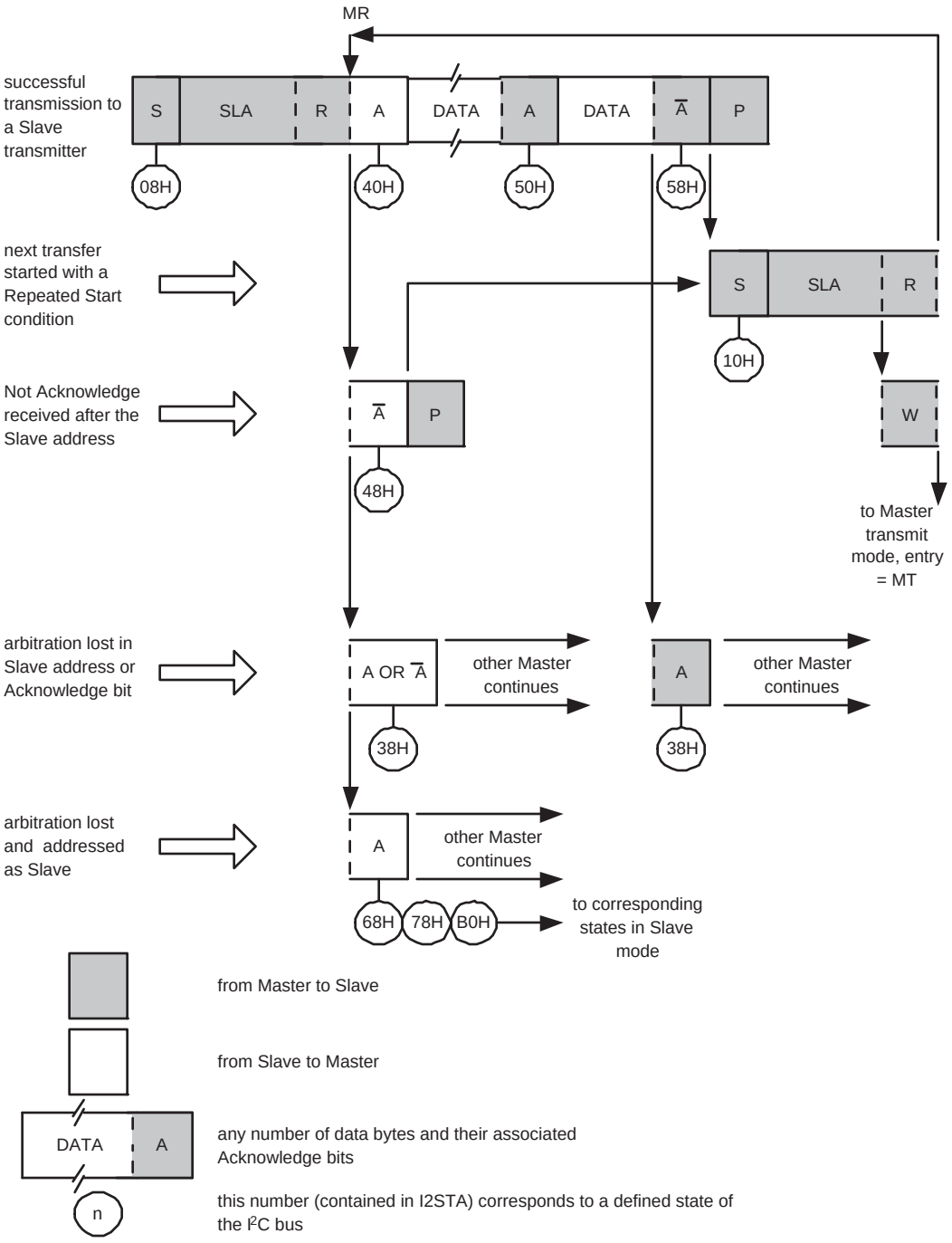


Fig 120. Format and states in the Master Receiver mode

22.9.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 121). To initiate the slave receiver mode, I2CON register, the I2ADR registers, and the I2MASK registers must be configured.

The values on the four I2ADR registers combined with the values on the four I2MASK registers determines which address or addresses the I^2C block will respond to when slave functions are enabled. See sections 22.7.2, 22.7.3, 22.8.7, and 22.8.8 for details.

Table 511. I2CONSET used to initialize Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

The I^2C-bus rate settings do not affect the I^2C block in the slave mode. I2EN must be set to logic 1 to enable the I^2C block. The AA bit must be set to enable the I^2C block to acknowledge its own slave address or the General Call address. STA, STO, and SI must be reset.

When the I2ADR, I2MASK, and I2CON registers have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^2C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 514. The slave receiver mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^2C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

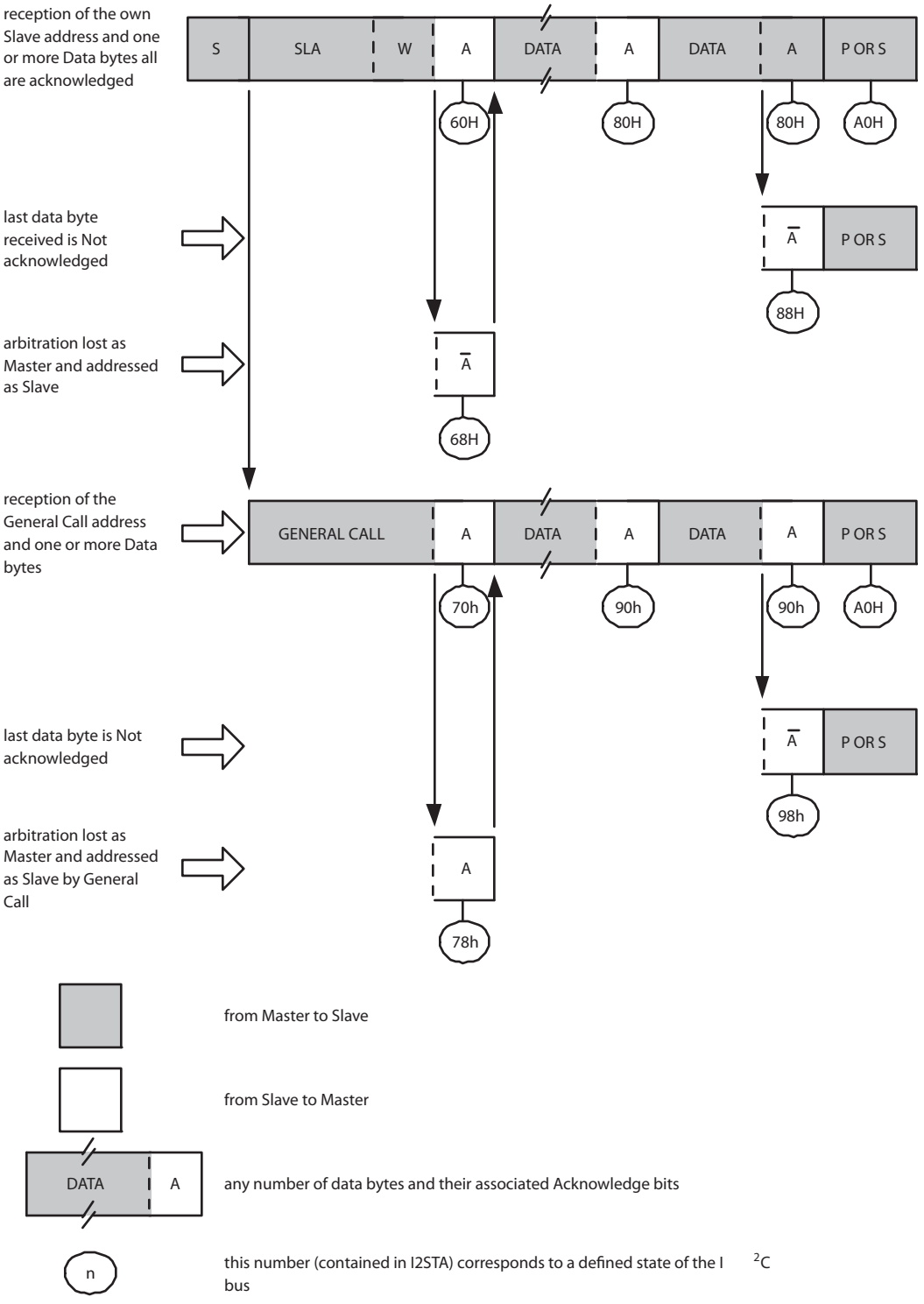


Fig 121. Format and states in the Slave Receiver mode

22.9.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 122). Data transfer is initialized as in the slave receiver mode. When I2ADR and I2CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “1” (R) for the I^2C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 515. The slave transmitter mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I^2C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^2C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

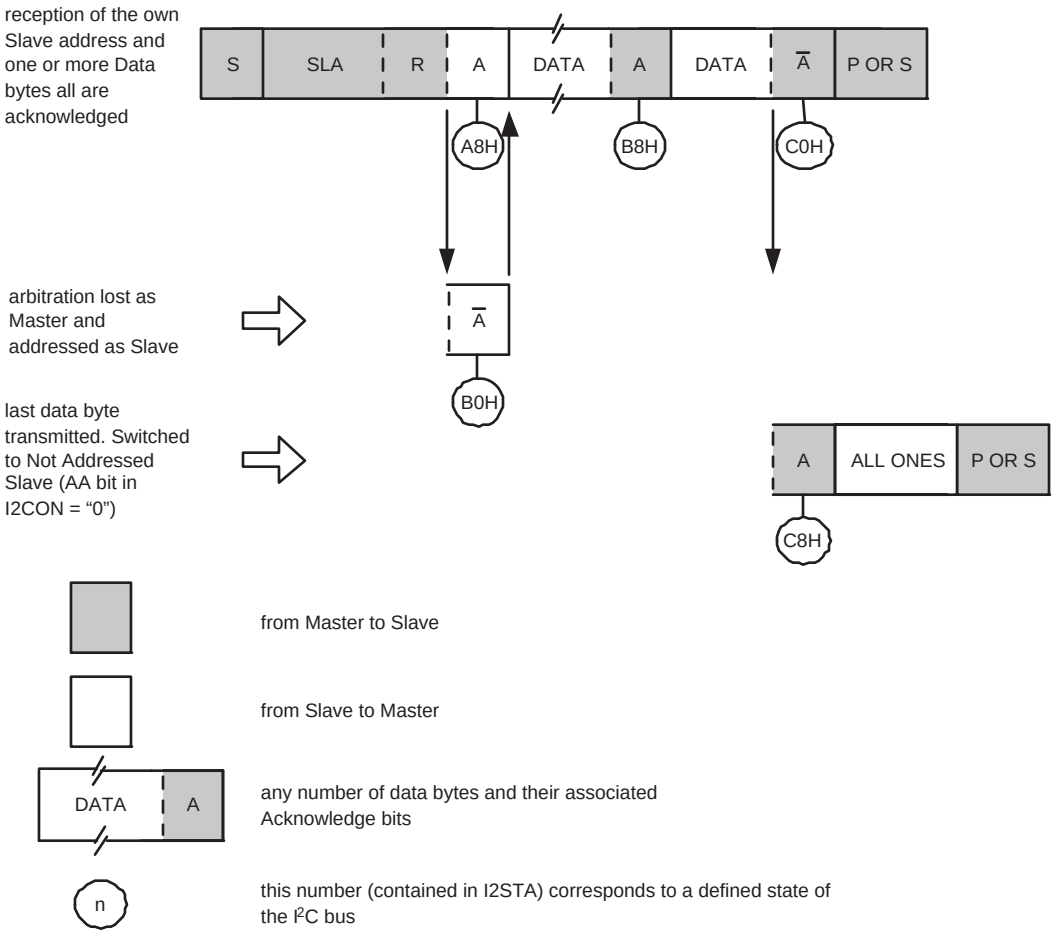


Fig 122. Format and states in the Slave Transmitter mode

22.9.5 Detailed state tables

The following tables show detailed state information for the four I^2C operating modes.

Table 512. Master Transmitter mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W; clear STA	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R; Clear STA	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in I2DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No I2DAT action or	0	0	0	X	I^2C-bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

Table 513. Master Receiver mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No I2DAT action or	0	0	0	X	I^2C-bus will be released; the I^2C block will enter a slave mode.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No I2DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No I2DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No I2DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 514. Slave Receiver mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General Call address (0x00) has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General Call address has been received, ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLA address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Table 514. Slave Receiver mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave Receiver or Slave Transmitter.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

Table 515. Slave Transmitter mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received, ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xC0	Data byte in I2DAT has been transmitted; NOT ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No I2DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in I2DAT has been transmitted (AA = 0); ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No I2DAT action	1	0	0	01	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

22.9.6 Miscellaneous states

There are two I2STAT codes that do not correspond to a defined I^2C hardware state (see Table 516). These are discussed below.

22.9.6.1 I2STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I^2C block is not involved in a serial transfer.

22.9.6.2 I2STAT = 0x00

This status code indicates that a bus error has occurred during an I^2C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I^2C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I^2C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in I2CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 516. Miscellaneous States

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xF8	No relevant state information available; SI = 0.	No I2DAT action	No I2CON action				Wait or proceed current transfer.
0x00	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 0x00 can also occur when interference causes the I^2C block to enter an undefined state.	No I2DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the I^2C block is switched to the not addressed SLV mode. STO is reset.

22.9.7 Some special cases

The I²C hardware has facilities to handle the following special cases that may occur during a serial transfer:

22.9.7.1 Simultaneous repeated START conditions from two masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 123). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I²C hardware detects a repeated START condition on the I²C-bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, the I²C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

22.9.7.2 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 117). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 119 and Figure 120).

If the STA flag in I2CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 0x08) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

22.9.7.3 Forced access to the I²C-bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I²C-bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I²C-bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I²C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware Figure 124.

22.9.7.4 I²C-bus obstructed by a LOW level on SCL or SDA

An I²C-bus hang-up can occur if either the SDA or SCL line is held LOW by any device on the bus. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the problem must be resolved by the device that is pulling the SCL bus line LOW.

Typically, the SDA line may be obstructed by another device on the bus that has become out of synchronization with the current bus master by either missing a clock, or by sensing a noise pulse as a clock. In this case, the problem can be solved by transmitting additional clock pulses on the SCL line Figure 125. The I²C interface does not include a dedicated time-out timer to detect an obstructed bus, but this can be implemented using another timer in the system. When detected, software can force clocks (up to 9 may be required)

on SCL until SDA is released by the offending device. At that point, the slave may still be out of synchronization, so a START should be generated to insure that all I^2C peripherals are synchronized.

22.9.7.5 Bus error

A bus error occurs when a START or STOP condition is detected at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I^2C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^2C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 516.

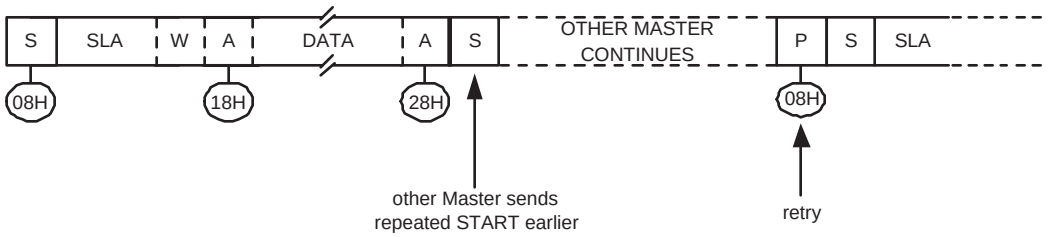


Fig 123. Simultaneous repeated START conditions from two masters

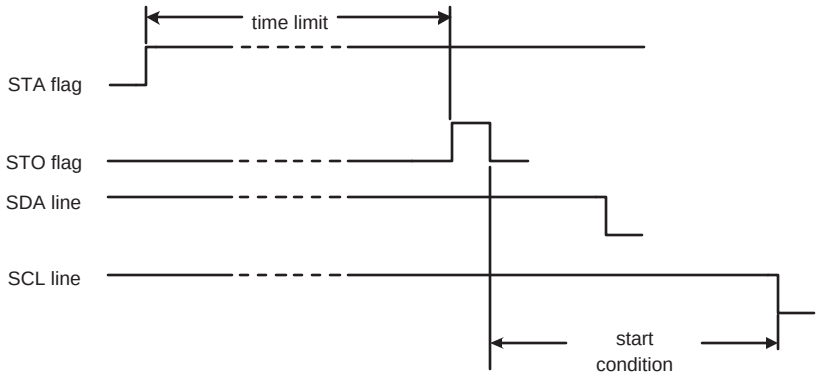
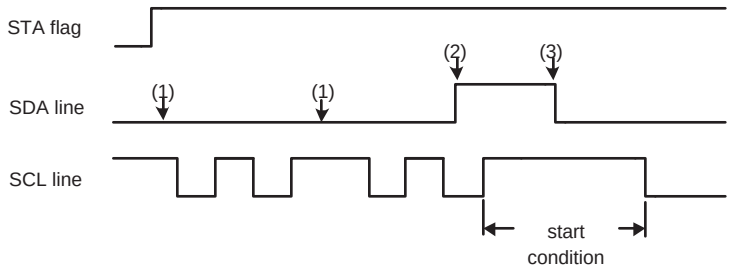


Fig 124. Forced access to a busy I^2C-bus



(1) Unsuccessful attempt to send a START condition.

(2) SDA line is released.

(3) Successful attempt to send a START condition. State 08H is entered.

Fig 125. Recovering from a bus obstruction caused by a LOW level on SDA

22.9.8 I²C state service routines

This section provides examples of operations that must be performed by various I²C state service routines. This includes:

- Initialization of the I²C block after a Reset.
- I²C Interrupt Service
- The 26 state service routines providing support for all four I²C operating modes.

22.9.8.1 Initialization

In the initialization example, the I²C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- The I2ADR registers and I2MASK registers are loaded with values to configure the part's own slave address(es) and the General Call bit (GC)
- The I²C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by loading the I2CSCLH and I2CSCLL registers. The master routines must be started in the main program.

The I²C hardware now begins checking the I²C-bus for its own slave address and General Call. If the General Call or the own slave address is detected, an interrupt is requested and I2STAT is loaded with the appropriate state information.

22.9.8.2 I²C interrupt service

When the I²C interrupt is entered, I2STAT contains a status code which identifies one of the 26 state services to be executed.

22.9.8.3 The state service routines

Each state routine is part of the I²C interrupt routine and handles one of the 26 states.

22.9.8.4 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I²C state codes. If one or more of the four I²C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of timeout during I²C operations, in order to trap an inoperative bus or a lost service routine.

22.10 Software example

22.10.1 Initialization routine

Example to initialize I^2C Interface as a Slave and/or Master.

1. Load the I2ADR registers and I2MASK registers with values to configure the own Slave Address, enable General Call recognition if needed.
2. Enable I^2C interrupt.
3. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to I2CONSET.

22.10.2 Start Master Transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

22.10.3 Start Master Receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

22.10.4 I^2C interrupt routine

Determine the I^2C state and which state routine will be used to handle it.

1. Read the I^2C status from I2STA.
2. Use the status value to branch to one of 26 possible state routines.

22.10.5 Non mode specific states

22.10.5.1 State: 0x00

Bus Error. Enter not addressed Slave mode and release bus.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.5.2 Master States

State 0x08 and State 0x10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

22.10.5.3 State: 0x08

A START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

22.10.5.4 State: 0x10

A repeated START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

22.10.6 Master Transmitter states

22.10.6.1 State: 0x18

Previous state was State 0x08 or State 0x10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted.

1. Load I2DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

22.10.6.2 State: 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.6.3 State: 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a STOP condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit
5. Load I2DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to I2CONSET to set the AA bit.
7. Write 0x08 to I2CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

22.10.6.4 State: 0x30

Data has been transmitted, NOT ACK received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.6.5 State: 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new START condition will be transmitted when the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.7 Master Receiver states

22.10.7.1 State: 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.7.2 State: 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.7.3 State: 0x50

Data has been received, ACK has been returned. Data will be read from I2DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from I2DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

22.10.7.4 State: 0x58

Data has been received, NOT ACK has been returned. Data will be read from I2DAT. A STOP condition will be transmitted.

1. Read data byte from I2DAT into Master Receive buffer.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit

22.10.8 Slave Receiver states

22.10.8.1 State: 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

22.10.8.2 State: 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

22.10.8.3 State: 0x70

General Call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

22.10.8.4 State: 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General Call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

22.10.8.5 State: 0x80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

22.10.8.6 State: 0x88

Previously addressed with own Slave Address. Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.8.7 State: 0x90

Previously addressed with General Call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
3. Exit

22.10.8.8 State: 0x98

Previously addressed with General Call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.8.9 State: 0xA0

A STOP condition or repeated START has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

22.10.9 Slave Transmitter states

22.10.9.1 State: 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

22.10.9.2 State: 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to I2CONSET to set the STA and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

22.10.9.3 State: 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

22.10.9.4 State: 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit.

22.10.9.5 State: 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

23.1 Basic configuration

The I2S interface is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCI2S.

   Remark: On reset, the I2S interface is disabled (PCI2S = 0).
2. Peripheral clock: The functional portion of the I2S interface operates from the CPU clock (CCLK), rather than PCLK. The bus interface operates from the common PCLK for APB peripherals. See Section 3.3.3.5.
3. Pins: Select I2S pins and their modes in the relevant IOCON registers (see Section 7.4.1).
4. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
5. DMA: The I2S interface supports two DMA requests, see Table 524 and Table 525, and Table 696.

23.2 Features

The I2S bus provides a standard communication interface for digital audio applications. The I2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select signal. The basic I2S connection has one master, which is always the master, and one slave. The I2S interface provides a separate transmit and receive channel, each of which can operate as either a master or a slave, and an optional oversample master clock output (MCLK).

- The I2S input can operate in both master and slave mode.
- The I2S output can operate in both master and slave mode, independent of the input.
- Capable of handling 8-bit, 16-bit, and 32-bit word sizes.
- Mono and stereo audio data supported.
- Versatile clocking includes independent transmit and receive fractional rate generators, and an ability to use a single clock input or output for a 4-wire mode.
- Sampling frequencies (fs) supported include standard 16 to 96 kHz ranges (16, 22.05, 32, 44.1, 48, or 96 kHz) for audio applications, and above, depending on the clock frequency.
- Separate Master Clock outputs for both transmit and receive channels support a clock up to 512 times the I2S sampling frequency.
- Word Select period in master mode is separately configurable for I2S input and output.
- Two 8 word (32 byte) FIFO data buffers, one set each for transmit and receive.
- Generates interrupt requests when buffer levels cross a programmable boundary.
- Two DMA requests, controlled by programmable buffer levels. These are connected to the General Purpose DMA block.
- Controls include reset, stop and mute options separately for I2S input and I2S output.
- Optional MCLK (oversample) output.

23.3 Description

The I2S performs serial data out via the transmit channel and serial data in via the receive channel. These support the NXP Inter IC Audio format for 8-bit, 16-bit and 32-bit audio data, both for stereo and mono modes. Configuration, data access and control is performed by a APB register set. Data streams are buffered by FIFOs with a depth of 8 words.

The I2S receive and transmit stage can operate independently in either slave or master mode. Within the I2S module the difference between these modes lies in the word select (WS) signal which determines the timing of data transmissions. Data words start on the next falling edge of the transmitting clock after a WS change. In stereo mode when WS is low left data is transmitted and right data when WS is high. In mono mode the same data is transmitted twice, once when WS is low and again when WS is high.

- In master mode, word select is generated internally with a 9-bit counter. The half period count value of this counter can be set in the control register.
- In slave mode, word select is input from the relevant bus pin.
- When an I2S bus is active, the word select, receive clock and transmit clock signals are sent continuously by the bus master, while data is sent continuously by the transmitter.
- Disabling the I2S can be done with the stop or mute control bits separately for the transmit and receive.
- The stop bit will disable accesses by the transmit channel or the receive channel to the FIFOs and will place the transmit channel in mute mode.
- The mute control bit will place the transmit channel in mute mode. In mute mode, the transmit channel FIFO operates normally, but the output is discarded and replaced by zeroes. This bit does not affect the receive channel, data reception can occur normally.

23.4 Pin descriptions

Table 517. Pin descriptions

Pin Name	Type	Description
I2S_RX_CLK	Input/ Output	Receive Clock. A clock signal used to synchronize the transfer of data on the receive channel. It is driven by the master and received by the slave. Corresponds to the signal SCK in the I2S bus specification. When this pin is an 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.
I2S_RX_WS	Input/ Output	Receive Word Select. Selects the channel from which data is to be received. It is driven by the master and received by the slave. Corresponds to the signal WS in the I2S bus specification. WS = 0 indicates that data is being received by channel 1 (left channel). WS = 1 indicates that data is being received by channel 2 (right channel).
I2S_RX_SDA	Input/ Output	Receive Data. Serial data, received MSB first. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the I2S bus specification.
I2S_RX_MCLK	Output	Optional master clock output for the I2S receive function.
I2S_TX_CLK	Input/ Output	Transmit Clock. A clock signal used to synchronize the transfer of data on the transmit channel. It is driven by the master and received by the slave. Corresponds to the signal SCK in the I2S bus specification. When this pin is an 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.
I2S_TX_WS	Input/ Output	Transmit Word Select. Selects the channel to which data is being sent. It is driven by the master and received by the slave. Corresponds to the signal WS in the I2S bus specification. WS = 0 indicates that data is being sent to channel 1 (left channel). WS = 1 indicates that data is being sent to channel 2 (right channel).
I2S_TX_SDA	Input/ Output	Transmit Data. Serial data, sent MSB first. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the I2S bus specification.
I2S_TX_MCLK	Output	Optional master clock output for the I2S transmit function.

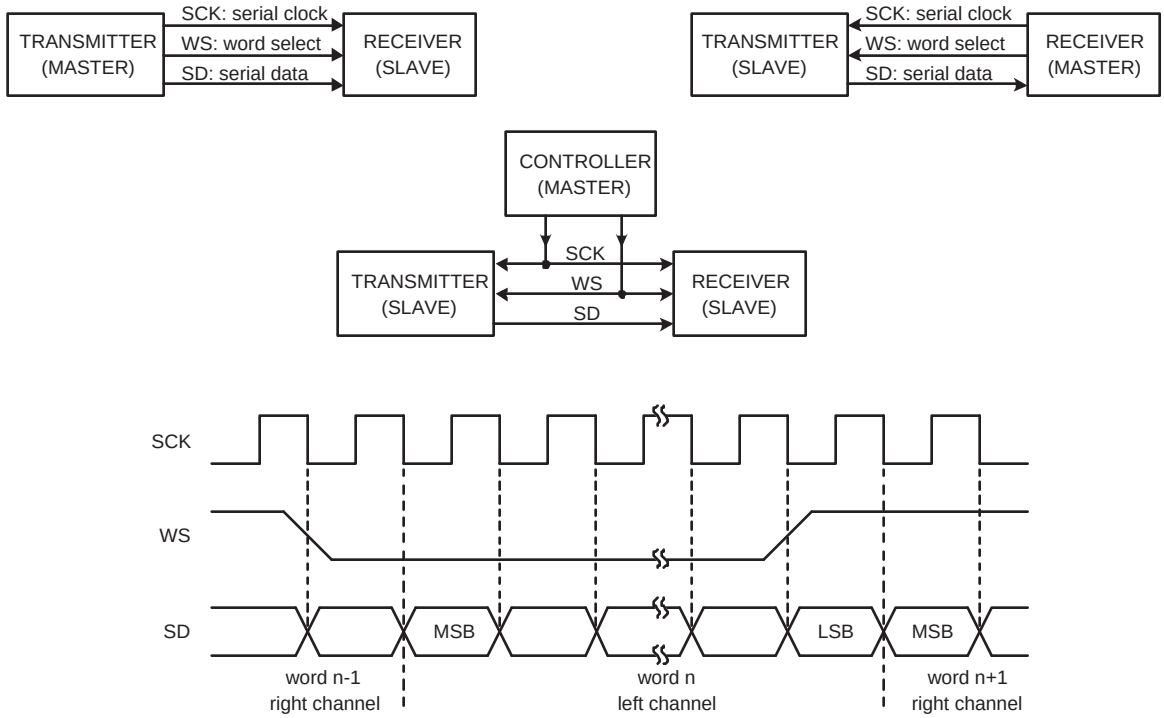


Fig 126. Simple I²S configurations and bus timing

23.5 Register description

Table 518. Register overview: I2S (base address 0x400A 8000)

Name	Access	Address offset	Description	Reset Value[1]	Table
DAO	R/W	0x000	Digital Audio Output Register. Contains control bits for the I2S transmit channel.	0x87E1	519
DAI	R/W	0x004	Digital Audio Input Register. Contains control bits for the I2S receive channel.	0x07E1	520
TXFIFO	WO	0x008	Transmit FIFO. Access register for the 8 × 32-bit transmitter FIFO.	0	521
RXFIFO	RO	0x00C	Receive FIFO. Access register for the 8 × 32-bit receiver FIFO.	0	522
STATE	RO	0x010	Status Feedback Register. Contains status information about the I2S interface.	0x7	523
DMA1	R/W	0x014	DMA Configuration Register 1. Contains control information for DMA request 1.	0	524
DMA2	R/W	0x018	DMA Configuration Register 2. Contains control information for DMA request 2.	0	525
IRQ	R/W	0x01C	Interrupt Request Control Register. Contains bits that control how the I2S interrupt request is generated.	0	526
TXRATE	R/W	0x020	Transmit reference clock divider. This register determines the I2S TX_REF rate by specifying the value to divide CCLK by in order to produce TX_REF.	0	527
RXRATE	R/W	0x024	Receive reference clock divider. This register determines the I2S RX_REF rate by specifying the value to divide CCLK by in order to produce RX_REF.	0	528
TXBITRATE	R/W	0x028	Transmit bit rate divider. This register determines the I2S transmit bit rate by specifying the value to divide TX_REF by in order to produce the transmit bit clock.	0	529
RXBITRATE	R/W	0x02C	Receive bit rate divider. This register determines the I2S receive bit rate by specifying the value to divide RX_REF by in order to produce the receive bit clock.	0	530
TXMODE	R/W	0x030	Transmit mode control.	0	531
RXMODE	R/W	0x034	Receive mode control.	0	532

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

23.5.1 Digital Audio Output register

The I2SDAO register controls the operation of the I2S transmit channel. The function of bits in DAO are shown in Table 519.

Table 519: Digital Audio Output register (DAO - address 0x400A 8000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WORDWIDTH		Selects the number of bytes in data as follows:	01
		0x0	8-bit data	
		0x1	16-bit data	
		0x2	Reserved, do not use this setting	
		0x3	32-bit data	
2	MONO		When 1, data is of monaural format. When 0, the data is in stereo format.	0
3	STOP		When 1, disables accesses on FIFOs, places the transmit channel in mute mode.	0
4	RESET		When 1, asynchronously resets the transmit channel and FIFO.	0
5	WS_SEL		When 0, the interface is in master mode. When 1, the interface is in slave mode. See Section 23.7 for a summary of useful combinations for this bit with I2STXMODE.	1
14:6	WS_HALFPERIOD		Word select half period minus 1, i.e. WS 64clk period -> ws_halfperiod = 31.	0x1F
15	MUTE		When 1, the transmit channel sends only zeroes.	1
31:16	-		Reserved. Read value is undefined, only zero should be written.	NA

23.5.2 Digital Audio Input register

The I2SDAI register controls the operation of the I2S receive channel. The function of bits in DAI are shown in Table 520.

Table 520: Digital Audio Input register (DAI - address 0x400A 8004) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WORDWIDTH		Selects the number of bytes in data as follows:	01
		0x0	8-bit data	
		0x1	16-bit data	
		0x2	Reserved, do not use this setting	
		0x3	32-bit data	
2	MONO		When 1, data is of monaural format. When 0, the data is in stereo format.	0
3	STOP		When 1, disables accesses on FIFOs, places the transmit channel in mute mode.	0
4	RESET		When 1, asynchronously reset the transmit channel and FIFO.	0
5	WS_SEL		When 0, the interface is in master mode. When 1, the interface is in slave mode. See Section 23.7 for a summary of useful combinations for this bit with I2SRXMODE.	1
14:6	WS_HALFPERIOD		Word select half period minus 1, i.e. WS 64clk period -> ws_halfperiod = 31.	0x1F
31:15	-		Reserved. Read value is undefined, only zero should be written.	NA

23.5.3 Transmit FIFO register

The I2STXFIFO register provides access to the transmit FIFO. The function of bits in I2STXFIFO are shown in Table 521.

Table 521: Transmit FIFO register (TXFIFO - address 0x400A 8008) bit description

Bit	Symbol	Description	Reset Value
31:0	I2STXFIFO	8 × 32-bit transmit FIFO.	Level = 0

23.5.4 Receive FIFO register

The I2SRXFIFO register provides access to the receive FIFO. The function of bits in I2SRXFIFO are shown in Table 522.

Table 522: Receive FIFO register (RXFIFO - address 0x400A 800C) bit description

Bit	Symbol	Description	Reset Value
31:0	I2SRXFIFO	8 × 32-bit transmit FIFO.	level = 0

23.5.5 Status Feedback register

The I2SSTATE register provides status information about the I²S interface. The meaning of bits in I2SSTATE are shown in Table 523.

Table 523: Status Feedback register (STATE - address 0x400A 8010) bit description

Bit	Symbol	Description	Reset Value
0	IRQ	This bit reflects the presence of Receive Interrupt or Transmit Interrupt. This is determined by comparing the current FIFO levels to the rx_depth_irq and tx_depth_irq fields in the I2SIRQ register.	1
1	DMAREQ1	This bit reflects the presence of Receive or Transmit DMA Request 1. This is determined by comparing the current FIFO levels to the rx_depth_dma1 and tx_depth_dma1 fields in the I2SDMA1 register.	1
2	DMAREQ2	This bit reflects the presence of Receive or Transmit DMA Request 2. This is determined by comparing the current FIFO levels to the rx_depth_dma2 and tx_depth_dma2 fields in the I2SDMA2 register.	1
7:3	-	Unused.	0
11:8	RX_LEVEL	Reflects the current level of the Receive FIFO.	0
15:12	-	Reserved. Read value is undefined, only zero should be written.	NA
19:16	TX_LEVEL	Reflects the current level of the Transmit FIFO.	0
31:20	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.6 DMA Configuration Register 1

The I2SDMA1 register controls the operation of DMA request 1. The function of bits in I2SDMA1 are shown in Table 524. Refer to the General Purpose DMA Controller chapter for details of DMA operation.

Table 524: DMA Configuration register 1 (DMA1 - address 0x400A 8014) bit description

Bit	Symbol	Description	Reset Value
0	RX_DMA1_ENABLE	When 1, enables DMA1 for I2S receive.	0
1	TX_DMA1_ENABLE	When 1, enables DMA1 for I2S transmit.	0
7:2	-	Reserved. Read value is undefined, only zero should be written.	0
11:8	RX_DEPTH_DMA1	Set the FIFO level that triggers a receive DMA request on DMA1.	0
15:12	-	Reserved. Read value is undefined, only zero should be written.	NA
19:16	TX_DEPTH_DMA1	Set the FIFO level that triggers a transmit DMA request on DMA1.	0
31:20	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.7 DMA Configuration Register 2

The I2SDMA2 register controls the operation of DMA request 2. The function of bits in I2SDMA2 are shown in Table 519.

Table 525: DMA Configuration register 2 (DMA2 - address 0x400A 8018) bit description

Bit	Symbol	Description	Reset Value
0	RX_DMA2_ENABLE	When 1, enables DMA1 for I2S receive.	0
1	TX_DMA2_ENABLE	When 1, enables DMA1 for I2S transmit.	0
7:2	-	Unused.	0
11:8	RX_DEPTH_DMA2	Set the FIFO level that triggers a receive DMA request on DMA2.	0
15:12	-	Reserved. Read value is undefined, only zero should be written.	NA
19:16	TX_DEPTH_DMA2	Set the FIFO level that triggers a transmit DMA request on DMA2.	0
31:20	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.8 Interrupt Request Control register

The I2SIRQ register controls the operation of the I2S interrupt request. The function of bits in I2SIRQ are shown in Table 519.

Table 526: Interrupt Request Control register (IRQ - address 0x400A 801C) bit description

Bit	Symbol	Description	Reset Value
0	RX_IRQ_ENABLE	When 1, enables I2S receive interrupt.	0
1	TX_IRQ_ENABLE	When 1, enables I2S transmit interrupt.	0
7:2	-	Unused.	0
11:8	RX_DEPTH_IRQ	Set the FIFO level on which to create an irq request.	0
15:12	-	Reserved. Read value is undefined, only zero should be written.	NA
19:16	TX_DEPTH_IRQ	Set the FIFO level on which to create an irq request.	0
31:20	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.9 Transmit Clock Rate register

The TX_REF rate for the I2S transmitter is determined by the values in the I2STXRATE register. The required I2STXRATE setting depends on the desired audio sample rate desired, the format (stereo/mono) used, and the data size. When the MCLK output is enabled for the transmit function, it is TX_REF that is sent to the I2S_TX_MCLK pin.

The TX_REF rate is generated using a fractional rate generator, dividing down the frequency of CCLK. Values of the numerator (X) and the denominator (Y) must be chosen to produce a frequency twice that desired for TX_REF, which must be an integer multiple of the transmitter bit clock rate. Fractional rate generators have some aspects that the user should be aware of when choosing settings. These are discussed in Section 23.5.9.1. The equation for the fractional rate generator is:

I2S TX_REF = CCLK * (X/Y) /2

Note: If the value of X or Y is 0, the clock divider is bypassed. Also, the value of Y must be greater than or equal to X.

Table 527: Transmit Clock Rate register (TXRATE - address 0x400A 8020) bit description

Bit	Symbol	Description	Reset Value
7:0	Y_DIVIDER	I2S transmit TX_REF rate denominator. This value is used to divide CCLK to produce TX_REF. Eight bits of fractional divide supports a wide range of possibilities. A value of 0 causes the clock divider to be bypassed.	0
15:8	X_DIVIDER	I2S transmit TX_REF rate numerator. This value is used to multiply CCLK by to produce the TX_REF. A value of 0 causes the clock divider to be bypassed. Eight bits of fractional divide supports a wide range of possibilities. Note: the resulting ratio X/Y is divided by 2.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.9.1 Notes on fractional rate generators

The nature of a fractional rate generator is that there will be some output jitter with some divide settings. This is because the fractional rate generator is a fully digital function, so output clock transitions are synchronous with the source clock, whereas a theoretical perfect fractional rate may have edges that are not related to the source clock. So, output jitter will not be greater than plus or minus one source clock between consecutive clock edges.

For example, if X = 0x07 and Y = 0x11, the fractional rate generator will output 7 clocks for every 17 (11 hex) input clocks, distributed as evenly as it can. In this example, there is no way to distribute the output clocks in a perfectly even fashion, so some clocks will be longer than others. The output is divided by 2 in order to square it up, which also helps with the jitter. The frequency averages out to exactly (7/17) / 2, but some clocks will be a slightly different length than their neighbors. It is possible to avoid jitter entirely by choosing fractions such that X divides evenly into Y, such as 2/4, 2/6, 3/9, 1/N, etc.

23.5.10 Receive Clock Rate register

The RX_REF rate for the I2S receiver is determined by the values in the I2SRXRATE register. The required I2SRXRATE setting depends on the CPU clock rate (CCLK) and the desired RX_REF rate (such as 256 fs). When the MCLK output is enabled for the receive function, it is RX_REF that is sent to the I2S_RX_MCLK pin.

The RX_REF rate is generated using a fractional rate generator, dividing down the frequency of CCLK. Values of the numerator (X) and the denominator (Y) must be chosen to produce a frequency twice that desired for the RX_REF, which must be an integer multiple of the receiver bit clock rate. Fractional rate generators have some aspects that the user should be aware of when choosing settings. These are discussed in Section 23.5.9.1. The equation for the fractional rate generator is:

I2S RX_REF = CCLK * (X/Y) /2

Note: If the value of X or Y is 0, the clock divider is bypassed. Also, the value of Y must be greater than or equal to X.

Table 528: Receive Clock Rate register (RXRATE - address 0x400A 8024) bit description

Bit	Symbol	Description	Reset Value
7:0	Y_DIVIDER	I2S receive RX_REF rate denominator. This value is used to divide CCLK to produce RX_REF. Eight bits of fractional divide supports a wide range of possibilities. A value of 0 causes the clock divider to be bypassed.	0
15:8	X_DIVIDER	I2S receive RX_REF rate numerator. This value is used to multiply CCLK by to produce RX_REF. A value of 0 causes the clock divider to be bypassed. Eight bits of fractional divide supports a wide range of possibilities. Note: the resulting ratio X/Y is divided by 2.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.11 Transmit Clock Bit Rate register

The bit rate for the I2S transmitter is determined by the value of the I2STXBITRATE register. The value depends on the audio sample rate desired, and the data size and format (stereo/mono) used. For example, a 48 kHz sample rate for 16-bit stereo data requires a bit rate of 48,000×16×2 = 1.536 MHz.

Table 529: Transmit Clock Bit Rate register (TXBITRATE - address 0x400A 8028) bit description

Bit	Symbol	Description	Reset Value
5:0	TX_BITRATE	I2S transmit bit rate. This value plus one is used to divide TX_REF to produce the transmit bit clock.	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.12 Receive Clock Bit Rate register

The bit rate for the I2S receiver is determined by the value of the I2SRXBITRATE register. The value depends on the audio sample rate, as well as the data size and format used. The calculation is the same as for I2SRXBITRATE.

Table 530: Receive Clock Rate Bit register (RXBITRATE - address 0x400A 802C) bit description

Bit	Symbol	Description	Reset Value
5:0	RX_BITRATE	I2S receive bit rate. This value plus one is used to divide RX_REF to produce the receive bit clock.	0
31:6	-	Reserved. Read value is undefined, only zero should be written.	NA

23.5.13 Transmit Mode Control register

The Transmit Mode Control register contains additional controls for transmit clock source, enabling the 4-pin mode, how TX_REF is used, and whether the MCLK output is enabled. See Section 23.7 for a summary of useful mode combinations.

Table 531: Transmit Mode Control register (TXMODE - 0x400A 8030) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	TXCLKSEL		Clock source selection for the transmit bit clock divider.	0
		0x0	Select the TX fractional rate divider clock output as the source	
		0x1	Reserved	
		0x2	Select the RX_REF signal as the TX_REF clock source	
		0x3	Reserved	
2	TX4PIN		Transmit 4-pin mode selection. When 1, enables 4-pin mode.	0
3	TXMCENA		Enable for the TX_MCLK output.	0
		0	Output of TX_MCLK to a pin is disabled.	
		1	Output of TX_MCLK to a pin is enabled.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

23.5.14 Receive Mode Control register

The Receive Mode Control register contains additional controls for receive clock source, enabling the 4-pin mode, and how RX_REF is used. See Section 23.7 for a summary of useful mode combinations.

Table 532: Receive Mode Control register (RXMODE - 0x400A 8034) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	RXCLKSEL		Clock source selection for the receive bit clock divider.	0
		0x0	Select the RX fractional rate divider clock output as the source	
		0x1	Reserved	
		0x2	Select the TX_REF signal as the RX_REF clock source	
		0x3	Reserved	
2	RX4PIN		Receive 4-pin mode selection. When 1, enables 4-pin mode.	0
3	RXMCENA		Enable for the RX_MCLK output.	0
		0	Output of RX_MCLK to a pin is disabled.	
		1	Output of RX_MCLK to a pin is enabled.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

23.6 I2S transmit and receive interfaces

The I2S interface can transmit and receive 8-bit, 16-bit or 32-bit stereo or mono audio information. Some details of I2S implementation are:

- When the FIFO is empty, the transmit channel will repeat transmitting the same data until new data is written to the FIFO.
- When mute is true, the data value 0 is transmitted.
- When mono is false, two successive data words are respectively left and right data.
- Data word length is determined by the wordwidth value in the configuration register. There is a separate wordwidth value for the receive channel and the transmit channel.
  - 0: word is considered to contain four 8-bit data words.
  - 1: word is considered to contain two 16-bit data words.
  - 3: word is considered to contain one 32-bit data word.
- When the transmit FIFO contains insufficient data the transmit channel will repeat transmitting the last data until new data is available. This can occur when the microprocessor or the DMA at some time is unable to provide new data fast enough. Because of this delay in new data there is a need to fill the gap, which is accomplished by continuing to transmit the last sample. The data is not muted as this would produce an noticeable and undesirable effect in the sound.
- The transmit channel and the receive channel only handle 32-bit aligned words, data chunks must be clipped or extended to a multiple of 32 bits.

When switching between data width or modes the I2S must be reset via the reset bit in the control register in order to ensure correct synchronization. It is advisable to set the stop bit also until sufficient data has been written in the transmit FIFO. Note that when stopped data output is muted.

All data accesses to FIFOs are 32 bits. Figure 140 shows the possible data sequences.

A data sample in the FIFO consists of:

- 1×32 bits in 8-bit or 16-bit stereo modes.
- 1×32 bits in mono modes.
- 2×32 bits, first left data, second right data, in 32-bit stereo modes.

Data is read from the transmit FIFO after the falling edge of WS, it will be transferred to the transmit clock domain after the rising edge of WS. On the next falling edge of WS the left data will be loaded in the shift register and transmitted and on the following rising edge of WS the right data is loaded and transmitted.

The receive channel will start receiving data after a change of WS. When word select becomes low it expects this data to be left data, when WS is high received data is expected to be right data. Reception will stop when the bit counter has reached the limit set by wordwidth. On the next change of WS the received data will be stored in the appropriate hold register. When complete data is available it will be written into the receive FIFO.

23.7 I2S operating modes

The clocking and WS usage of the I2S interface is configurable. In addition to master and slave modes, which are independently configurable for the transmitter and the receiver, several different clock sources are possible, including variations that share the clock and/or WS between the transmitter and receiver. This last option allows using I2S with fewer pins, typically four.

Many configurations are possible that are not considered useful, the following tables and figures give details of the configurations that are most likely to be useful.

23.7.1 I²S transmit modes

Table 533: I²S transmit modes

I2SDAO[5]	I2STXMODE[3:0]	Description
0	0 0 0 0	Typical transmitter master mode. See Figure 127. The I²S transmit function operates as a master. The transmit clock source is the fractional rate divider. The WS used is the internally generated TX_WS. TX_MCLK is not output on the I2S_TX_MCLK pin.
0	0 0 1 0	Transmitter master mode sharing the receiver reference clock. See Figure 128. The I²S transmit function operates as a master. The transmit clock source is RX_REF. The WS used is the internally generated TX_WS. TX_MCLK is not output on the I2S_TX_MCLK pin.
0	0 1 0 0	4-wire transmitter master mode sharing the receiver bit clock and WS. See Figure 129. The I²S transmit function operates as a master. The transmit clock source is the RX bit clock. The WS used is the internally generated RX_WS. TX_MCLK is not output on the I2S_TX_MCLK pin.
0	1 0 0 0	Transmitter master mode with TC_MCLK output. See Figure 127. The I²S transmit function operates as a master. The transmit clock source is the fractional rate divider. The WS used is the internally generated TX_WS. TX_MCLK is output on the I2S_TX_MCLK pin.
1	0 0 0 0	Typical transmitter slave mode. See Figure 130. The I²S transmit function operates as a slave. The transmit clock source is the TX_CLK pin. The WS used is the TX_WS pin.
1	0 0 1 0	Transmitter slave mode sharing the receiver reference clock. See Figure 131. The I²S transmit function operates as a slave. The transmit clock source is RX_REF. The WS used is the TX_WS pin.
1	0 1 0 0	4-wire transmitter slave mode sharing the receiver bit clock and WS. See Figure 132. The I²S transmit function operates as a slave. The transmit clock source is the RX bit clock. The WS used is RX_WS ref.

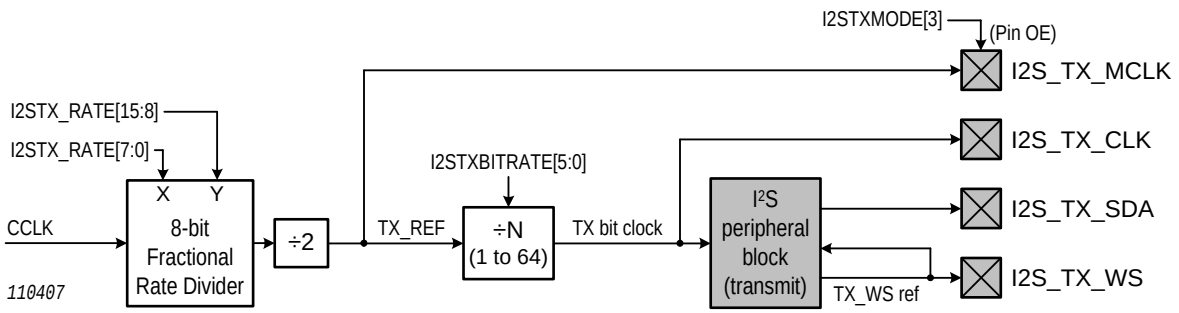


Fig 127. Typical transmitter master mode, with or without MCLK output

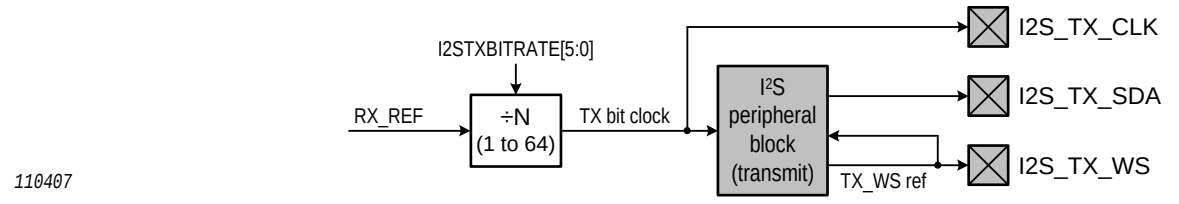


Fig 128. Transmitter master mode sharing the receiver reference clock

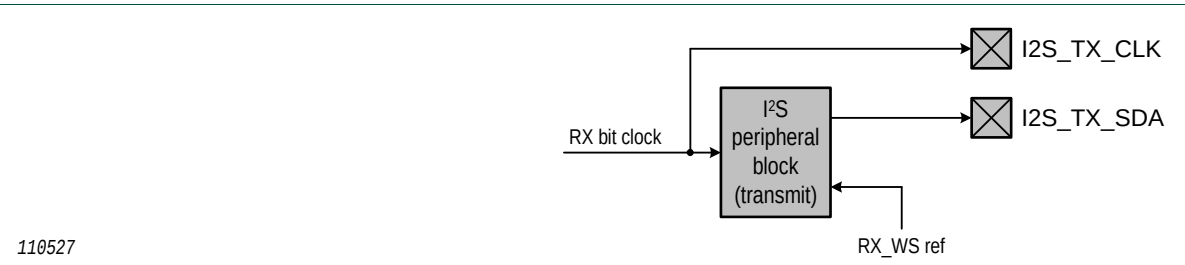


Fig 129. 4-wire transmitter master mode sharing the receiver bit clock and WS

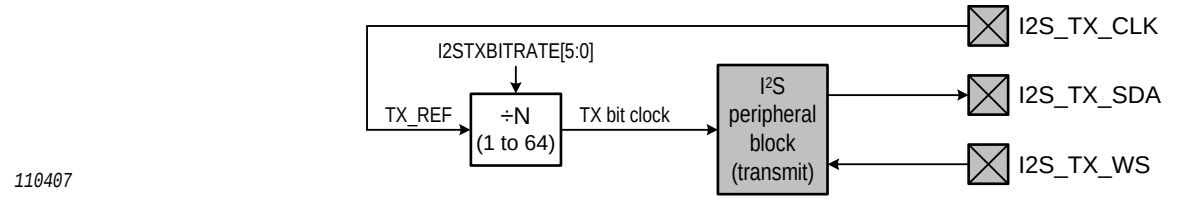


Fig 130. Typical transmitter slave mode

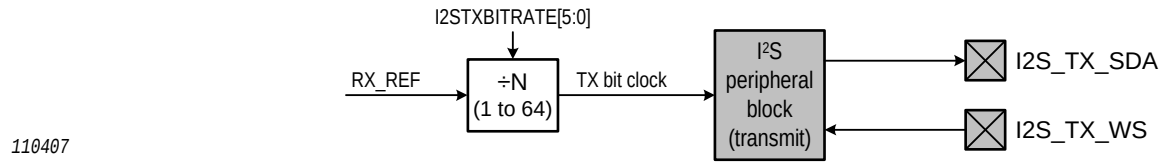


Fig 131. Transmitter slave mode sharing the receiver reference clock

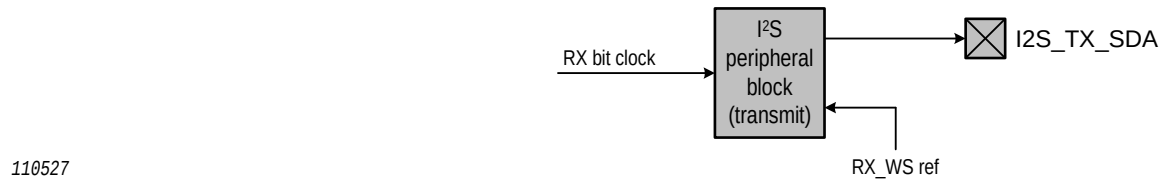


Fig 132. 4-wire transmitter slave mode sharing the receiver bit clock and WS

23.7.2 I²S receive modes

Table 534: I²S receive modes

I2SDAI[5]	I2SRXMODE[3:0]	Description
0	0 0 0 0	Typical receiver master mode. See Figure 133. The I²S receive function operates as a master. The receive clock source is the fractional rate divider. The WS used is the internally generated RX_WS. RX_MCLK is not output on the I2S_RX_MCLK pin.
0	0 0 1 0	Receiver master mode sharing the transmitter reference clock. See Figure 134. The I²S receive function operates as a master. The receive clock source is TX_REF. The WS used is the internally generated RX_WS. RX_MCLK is not output on the I2S_RX_MCLK pin.
0	0 1 0 0	4-wire receiver master mode sharing the transmitter bit clock and WS. See Figure 135. The I²S receive function operates as a master. The receive clock source is the TX bit clock. The WS used is the internally generated TX_WS. RX_MCLK is not output on the I2S_RX_MCLK pin.
0	1 0 0 0	Receiver master mode with RX_MCLK output. See Figure 133. The I²S receive function operates as a master. The receive clock source is the fractional rate divider. The WS used is the internally generated RX_WS. RX_MCLK is output on the I2S_RX_MCLK pin.
1	0 0 0 0	Typical receiver slave mode. See Figure 136. The I²S receive function operates as a slave. The receive clock source is the RX_CLK pin. The WS used is the RX_WS pin.
1	0 0 1 0	Receiver slave mode sharing the transmitter reference clock. See Figure 137. The I²S receive function operates as a slave. The receive clock source is TX_REF. The WS used is the RX_WS pin.
1	0 1 0 0	This is a 4-wire receiver slave mode sharing the transmitter bit clock and WS. See Figure 138. The I²S receive function operates as a slave. The receive clock source is the TX bit clock. The WS used is TX_WS ref.

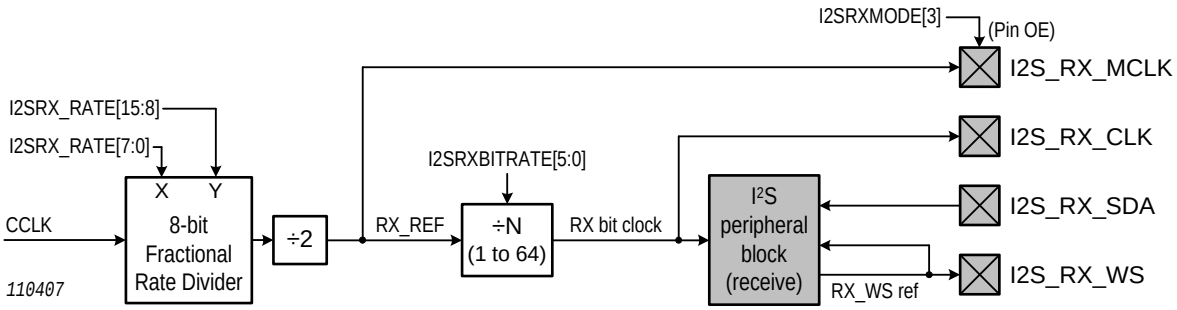


Fig 133. Typical receiver master mode, with or without MCLK output

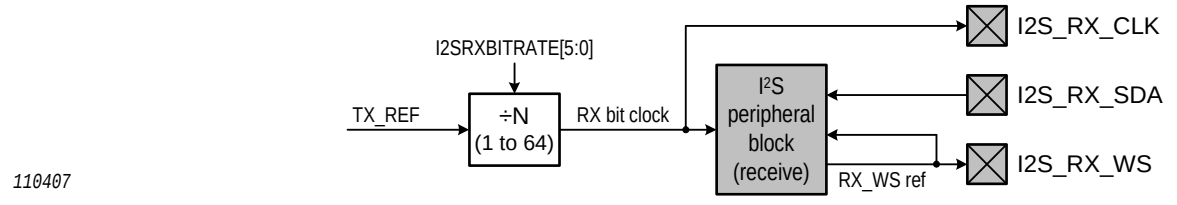


Fig 134. Receiver master mode sharing the transmitter reference clock

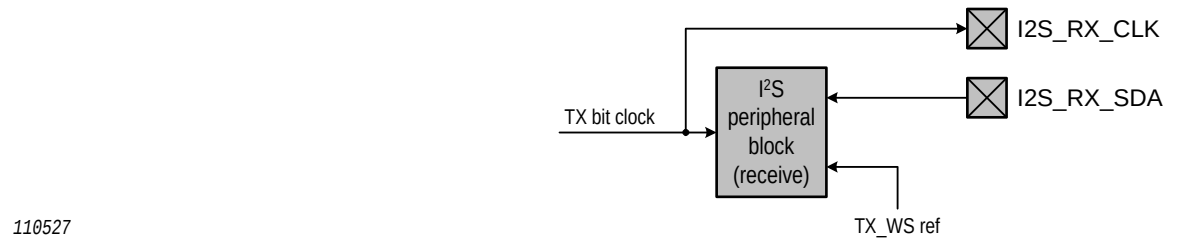


Fig 135. 4-wire receiver master mode sharing the transmitter bit clock and WS

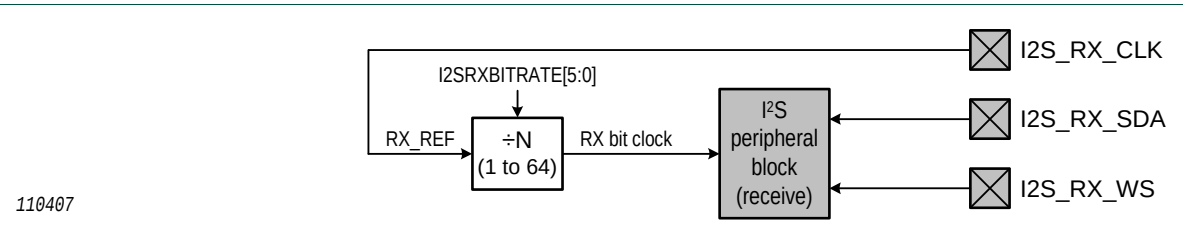


Fig 136. Typical receiver slave mode

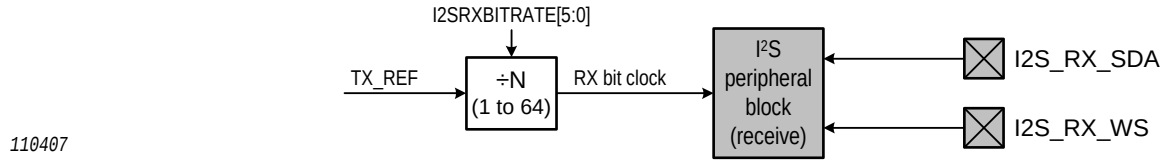


Fig 137. Receiver slave mode sharing the transmitter reference clock

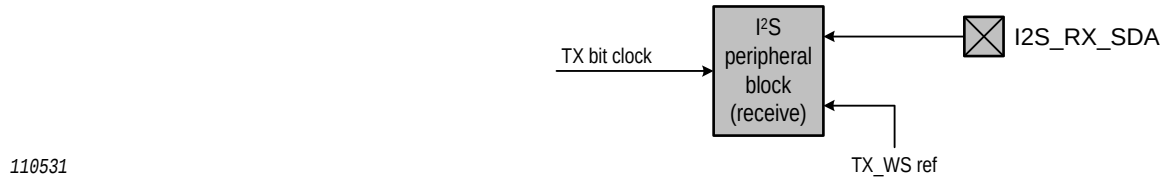


Fig 138. 4-wire receiver slave mode sharing the transmitter bit clock and WS

23.7.2.1 Overall clocking and pin connections

Figure 139 shows all of the clocking connections and pin connections for the I²S block.

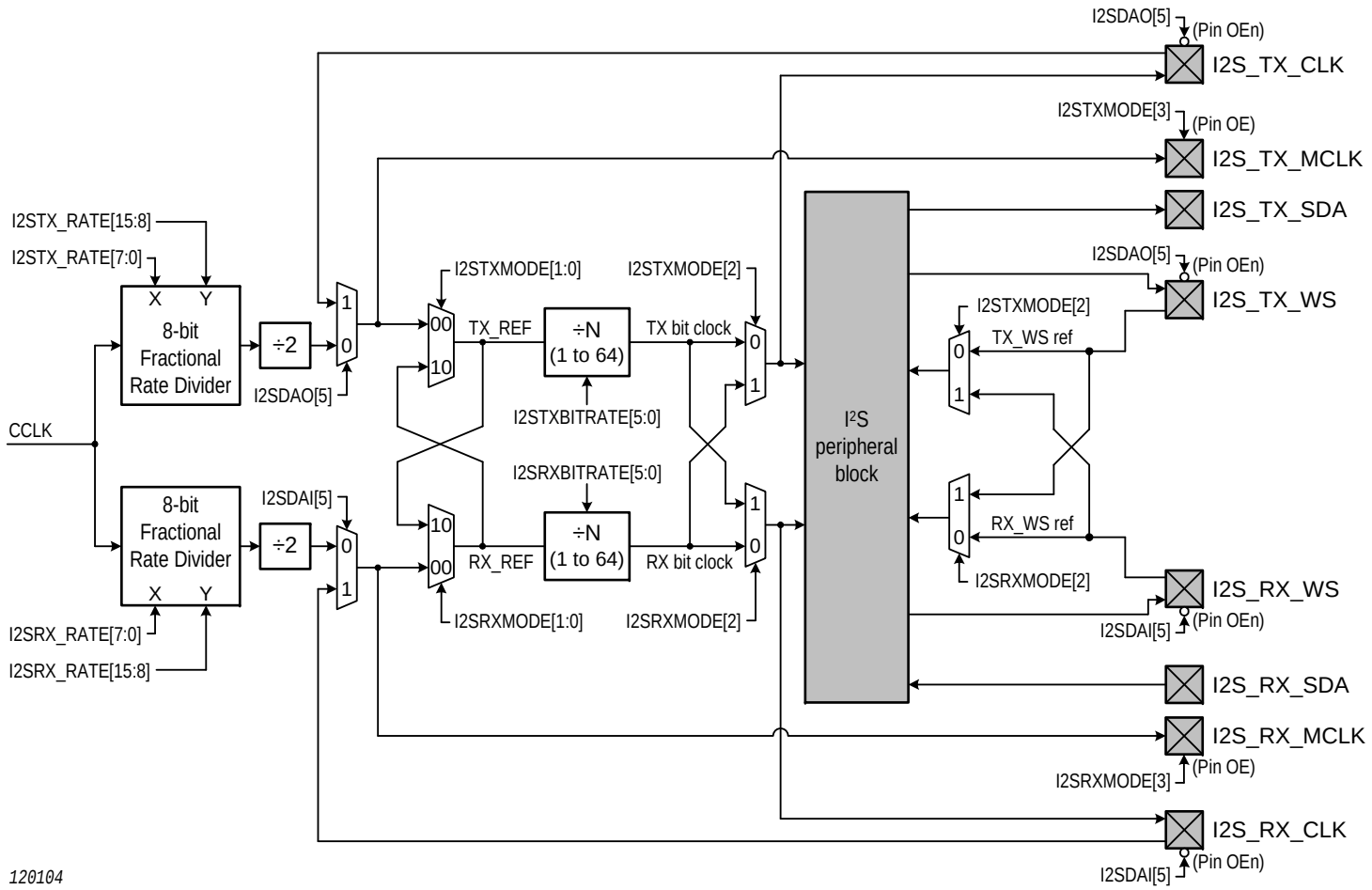


Fig 139. I²S clocking and pin connections

23.8 FIFO controller

Handling of data for transmission and reception is performed via the FIFO controller which can generate two DMA requests and an interrupt request. The controller consists of a set of comparators which compare FIFO levels with depth settings contained in registers. The current status of the level comparators can be seen in the APB status register.

How the FIFO is used in different modes and with different data widths is shown in Figure 140.

Table 535. Conditions for FIFO level comparison

Level Comparison	Condition
dmareq_tx_1	tx_depth_dma1 >= tx_level
dmareq_rx_1	rx_depth_dma1 <= rx_level
dmareq_tx_2	tx_depth_dma2 >= tx_level
dmareq_rx_2	rx_depth_dma2 <= rx_level
irq_tx	tx_depth_irq >= tx_level
irq_rx	rx_depth_irq <= rx_level

System signaling occurs when a level detection is true and enabled.

Table 536. DMA and interrupt request generation

System Signaling	Condition
irq	(irq_rx & rx_irq_enable) | (irq_tx & tx_irq_enable)
dmareq[0]	(dmareq_tx_1 & tx_dma1_enable) | (dmareq_rx_1 & rx_dma1_enable)
dmareq[1]	(dmareq_tx_2 & tx_dma2_enable) | (dmareq_rx_2 & rx_dma2_enable)

Table 537. Status feedback in the I2SSTATE register

Status Feedback	Status
irq	irq_rx | irq_tx
dmareq1	(dmareq_tx_1 | dmareq_rx_1)
dmareq2	(dmareq_rx_2 | dmareq_tx_2)

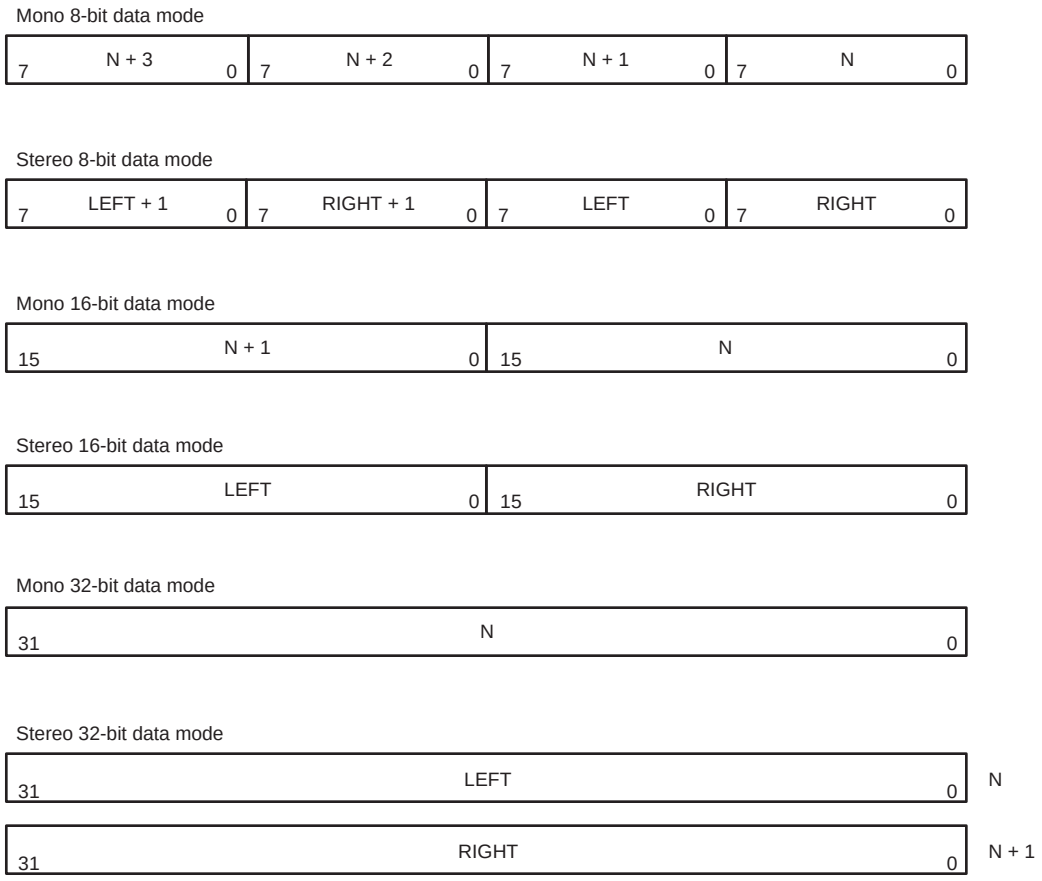


Fig 140. FIFO contents for various I²S modes

24.1 Basic configuration

The Timer 0, 1, 2, and 3 peripherals are configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bits PCTIM0/1/2/3.

   Remark: On reset, Timer0/1 are enabled (PCTIM0/1 = 1), and Timer2/3 are disabled (PCTIM2/3 = 0).
2. Peripheral clock: The timers operate from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Select timer pins and pin modes through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: See register T0/1/2/3MCR (Table 545) and T0/1/2/3CCR (Table 547) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
5. DMA: Up to two match conditions can be used to generate timed DMA requests, see Table 696.

24.2 Features

Remark: The four Timer/Counters are identical except for the peripheral base address. A minimum of two Capture inputs and two Match outputs are pinned out for all four timers, with a choice of multiple pins for each. Timer 2 brings out all four Match outputs.

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Counter or Timer operation
- Up to two 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to four external outputs corresponding to match registers, with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.

24.3 Applications

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

24.4 Description

The Timer/Counter is designed to count cycles of the peripheral clock (PCLK) or an externally-supplied clock, and can optionally generate interrupts or perform other actions at specified timer values, based on four match registers. It also includes four capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

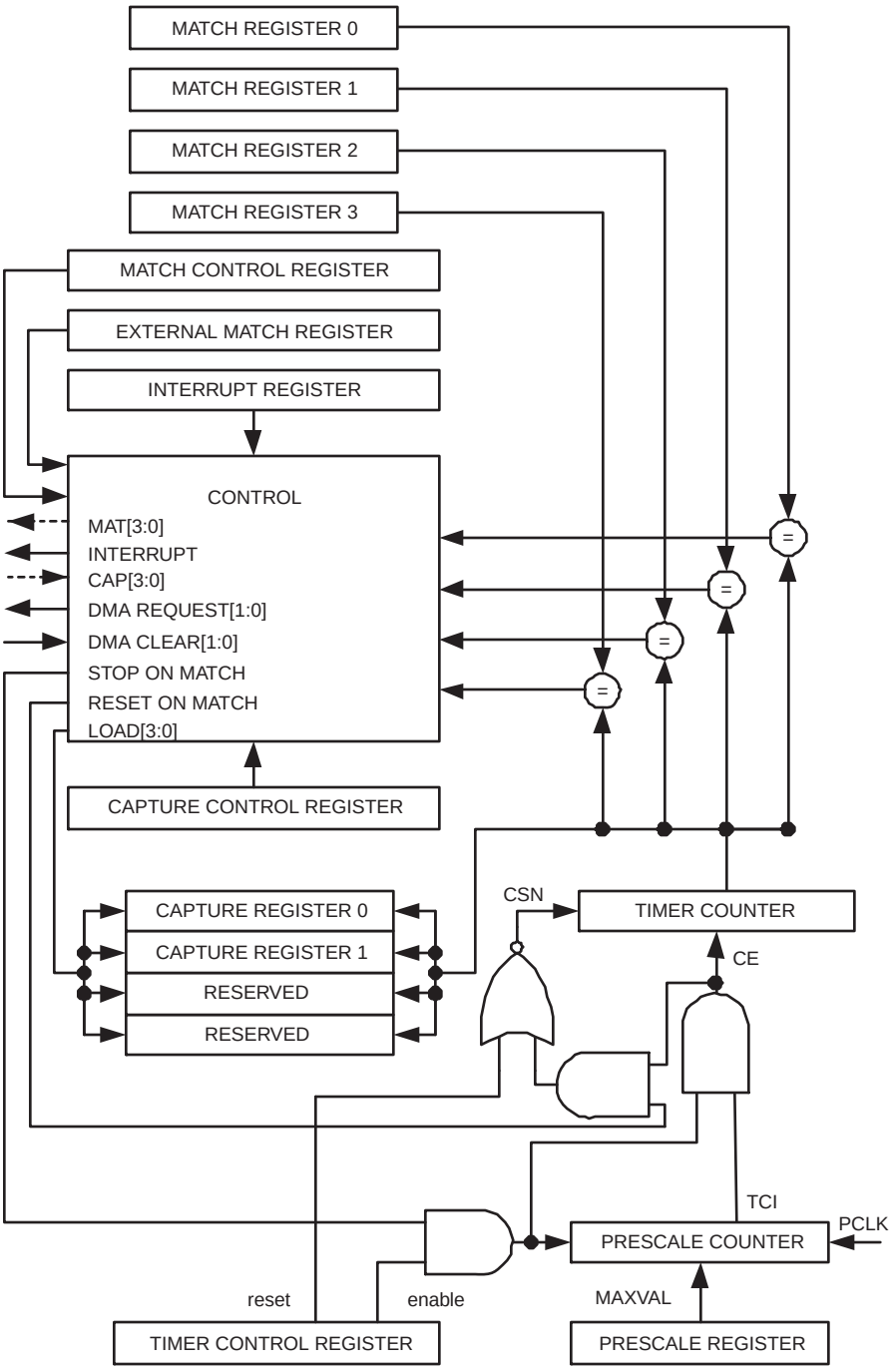


Fig 141. Timer block diagram

24.5 Pin description

Table 538 gives a brief summary of each of the Timer/Counter related pins.

Table 538. Timer/Counter pin description

Pin	Type	Description
T0_CAP1:0 T1_CAP1:0 T2_CAP1:0 T3_CAP1:0	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. When more than one pin is selected for a Capture input on a single TIMER0/1 channel, the pin with the lowest Port number is used Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 24.6.11.
T0_MAT1:0 T1_MAT1:0 T2_MAT3:0 T3_MAT1:0	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.

24.5.1 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 Section 7.4.1. When more than one pin is selected for a MAT output, all such pins are driven identically. When more than one pin is selected for a CAP input, the pin with the lowest Port number is used. Note that match conditions may be used internally without the use of a device pin.

24.6 Register description

Each Timer/Counter contains the registers shown in Table 539 ("Reset Value" refers to the data stored in used bits only; it does not include reserved bits content). More detailed descriptions follow.

Table 539. Register overview: Timer0/1/2/3 (register base addresses 0x4000 4000 (TIMER0), 0x4000 8000 (TIMER1), 0x4009 0000 (TIMER2), 0x4009 4000 (TIMER3))

Name	Access	Address offset	Description	Reset value[1]	Section
IR	R/W	0x000	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	Table 540
TCR	R/W	0x004	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	Table 541
TC	R/W	0x008	Timer Counter. The 32 bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0	Table 542
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.	0	Table 543
PC	R/W	0x010	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	Table 544
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0	Table 545
MR0	R/W	0x018	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	Table 546
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0	Table 546
MR2	R/W	0x020	Match Register 2. See MR0 description.	0	Table 546
MR3	R/W	0x024	Match Register 3. See MR0 description.	0	Table 546
CCR	R/W	0x028	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	Table 547
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 input.	0	Table 548
CR1	RO	0x030	Capture Register 1. See CR0 description.	0	Table 548
EMR	R/W	0x03C	External Match Register. The EMR controls the external match pins.	0	Table 549
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	Table 550

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

24.6.1 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.

Table 540. Interrupt Register (IR - addresses 0x4000 4000 (TIMER0), 0x4000 8000 (TIMER1), 0x4009 0000 (TIMER2), 0x4009 4000 (TIMER3)) 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
31:6	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 541. Timer Control Register (TCR - addresses 0x4000 4004 (TIMER0), 0x4000 8004 (TIMER1), 0x4009 0004 (TIMER2), 0x4009 4004 (TIMER3)) bit description

Bit	Symbol	Description	Reset Value
0	CEN	When 1, the Timer Counter and Prescale Counter are enabled for counting. When 0, the counters are disabled.	0
1	CRST	When 1, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	NA

24.6.3 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 542. Timer counter registers (TC - addresses 0x400 4008 (TIMER0), 0x4000 8008 (TIMER1), 0x4009 0008 (TIMER2), 0x4009 4008 (TIMER3)) bit description

Bit	Symbol	Description	Reset value
31:0	TC	Timer counter value.	0

24.6.4 Prescale register

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

Table 543. Timer prescale registers (PR - addresses 0x4000 400C (TIMER0), 0x4000 800C (TIMER1), 0x4009 000C (TIMER2), 0x4009 400C (TIMER3)) bit description

Bit	Symbol	Description	Reset value
31:0	PM	Prescale counter maximum value.	0

24.6.5 Prescale Counter register

The 32-bit Prescale Counter controls division of PCLK 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 PCLK. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK. This causes the Timer Counter to increment on every PCLK when PR = 0, every 2 pclks when PR = 1, etc.

Table 544. Timer prescale counter registers (PC - addresses 0x4000 4010 (TIMER0), 0x4000 8010 (TIMER1), 0x4009 0010 (TIMER2), 0x4009 4010 (TIMER3)) bit description

Bit	Symbol	Description	Reset value
31:0	PC	Prescale counter value.	0

24.6.6 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. The function of each of the bits is shown in Table 545.

Table 545. Match Control Register (MCR - addresses 0x4000 4014 (TIMER0), 0x4000 8014 (TIMER1), 0x4009 0014 (TIMER2), 0x4009 4014 (TIMER3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	MR0I		Interrupt on MR0	0
		1	Interrupt is generated when MR0 matches the value in the TC.	
		0	Interrupt is disabled	
1	MR0R		Reset on MR0	0
		1	TC will be reset if MR0 matches it.	
		0	Feature disabled.	
2	MR0S	1	Stop on MR0	0
		1	TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	
		0	Feature disabled.	
3	MR1I		Interrupt on MR1	0
		1	Interrupt is generated when MR1 matches the value in the TC.	
		0	Interrupt is disabled.	
4	MR1R		Reset on MR1	0
		1	TC will be reset if MR1 matches it.	
		0	Feature disabled.	
5	MR1S		Stop on MR1	0
		1	TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	
		0	Feature disabled.	
6	MR2I		Interrupt on MR2	0
		1	Interrupt is generated when MR2 matches the value in the TC.	
		0	Interrupt is disabled	
7	MR2R		Reset on MR2	0
		1	TC will be reset if MR2 matches it.	
		0	Feature disabled.	
8	MR2S		Stop on MR2.	0
		1	TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC	
		0	Feature disabled.	
9	MR3I		Interrupt on MR3	0
		1	Interrupt is generated when MR3 matches the value in the TC.	
		0	This interrupt is disabled	
10	MR3R		Reset on MR3	0
		1	TC will be reset if MR3 matches it.	
		0	Feature disabled.	
11	MR3S		Stop on MR3	0
		1	TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	
		0	Feature disabled.	
31:12	-		Reserved. Read value is undefined, only zero should be written.	NA

24.6.7 Match Registers (MR0 to MR3)

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.

Table 546. Timer match registers (MR[0:3], addresses 0x4000 4018 (MR0) to 0x4000 4024 (MR3) (TIMER0), 0x4000 8018 (MR0) to 0x4000 8024 (MR3) (TIMER1), 0x4009 0018 (MR0) to 0x4009 0024 (MR3) (TIMER2), 0x4009 4018 (MR0) to 0x4009 4024 (MR3)(TIMER3)) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

24.6.8 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 547. Capture Control Register (CCR - addresses 0x4000 4028 (TIMER0), 0x4000 8020 (TIMER1), 0x4009 0028 (TIMER2), 0x4009 4028 (TIMER3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	CAP0RE		Capture on CAPn.0 rising edge	0
		1	A sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	
		0	This feature is disabled.	
1	CAP0FE		Capture on CAPn.0 falling edge	0
		1	A sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	
		0	This feature is disabled.	
2	CAP0I		Interrupt on CAPn.0 event	0
		1	A CR0 load due to a CAPn.0 event will generate an interrupt.	
		0	This feature is disabled.	
3	CAP1RE		Capture on CAPn.1 rising edge	0
		1	A sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	
		0	This feature is disabled.	
4	CAP1FE		Capture on CAPn.1 falling edge	0
		1	A sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	
		0	This feature is disabled.	

Table 547. Capture Control Register (CCR - addresses 0x4000 4028 (TIMER0), 0x4000 8020 (TIMER1), 0x4009 0028 (TIMER2), 0x4009 4028 (TIMER3)) bit description

Bit	Symbol	Value	Description	Reset Value
5	CAP1I		Interrupt on CAPn.1 event	0
		1	A CR1 load due to a CAPn.1 event will generate an interrupt.	
		0	This feature is disabled.	
31:6	-		Reserved. Read value is undefined, only zero should be written.	NA

24.6.9 Capture Registers

Each Capture register is associated with a device pin and may be loaded with the Timer Counter value when a specified event occurs on that pin. 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 pin, the falling edge, or on both edges.

Table 548. Timer capture registers (CR[0:1], address 0x4000 402C (CR0) to 0x4000 4030 (CR1) (TIMER0), 0x4000 802C (CR0) to 0x4000 0030 (CR1) (TIMER1), 0x4009 002C (CR0) to 0x4009 0030 (CR1) (TIMER2), 0x4009 402C (CR0) to 0x4000 4030 (CR1) (TIMER3)) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

24.6.10 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 24.6.12.

Table 549. Timer external match registers (EMR - addresses 0x4000 403C (TIMER0), 0x4000 803C (TIMER1), 0x4009 403C (TIMER2), 0x400C 403C (TIMER3)) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0		External Match 0. When a match occurs between the TC and MR0, this bit can either toggle, go low, go high, or do nothing, depending on bits 5:4 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
1	EM1		External Match 1. When a match occurs between the TC and MR1, this bit can either toggle, go low, go high, or do nothing, depending on bits 7:6 of this register. This bit can be driven onto a MATn.1 pin, in a positive-logic manner (0 = low, 1 = high).	0
2	EM2		External Match 2. When a match occurs between the TC and MR2, this bit can either toggle, go low, go high, or do nothing, depending on bits 9:8 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
3	EM3		External Match 3. When a match occurs between the TC and MR3, this bit can either toggle, go low, go high, or do nothing, depending on bits 11:10 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (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 the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).	
		0x3	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 the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).	
		0x3	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 the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).	
		0x3	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 the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
31:12	-		Reserved. Read value is undefined, only zero should be written.	NA

24.6.11 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 PCLK 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.

Note that two successive samples of the pin are used to identify an edge on the CAP selected input. Therefore, the duration of the high and low levels on the same CAP input must be greater than 1 PCLK, and the frequency of the CAP input must be less than one quarter of the PCLK rate.

Table 550. Count Control Register (CTCR - addresses 0x4000 4070 (TIMER0), 0x4000 8070 (TIMER1), 0x4009 0070 (TIMER2), 0x4009 4070 (TIMER3)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CTMODE		Counter/Timer Mode This field selects which rising PCLK 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: every rising PCLK edge	
		0x1	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode: 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 TnCTCR, the 3 bits for that input in the Capture Control Register (TnCCR) 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	CAPn.0 for TIMERn	
		0x1	CAPn.1 for TIMERn	
		0x2	Reserved	
		0x3	Reserved	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

24.6.12 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 GPDMA must be configured and the relevant timer DMA request selected as a DMA source via the DMAREQSEL register, see Section 3.3.7.7.

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 24.6.1. A DMA request will be cleared automatically when it is acted upon by the GPDMA 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 GPDMA block unless the timer is correctly configured to generate valid DMA requests.

24.7 Example timer operation

Figure 142 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 143 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.

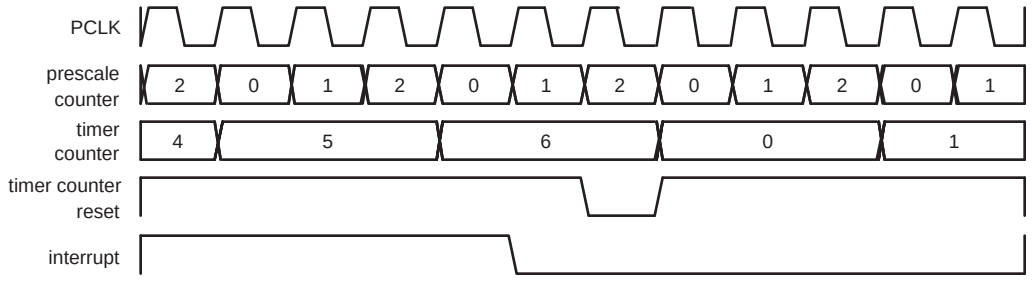


Fig 142. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

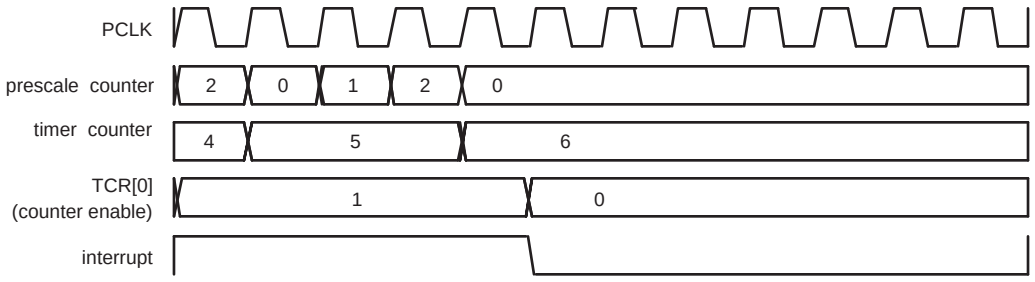


Fig 143. A timer Cycle in Which PR=2, MRx=6, and both interrupt and stop on match are enabled

25.1 Basic configuration

The System Tick Timer is configured using the following registers:

1. Clock Source: Select either the internal CCLK or external STCLK (pin P3[26]) clock as the source in the STCTRL register.
2. Pins: If STCLK (pin P3[26]) was selected as clock source enable the STCLK pin function in the relevant IOCON register (Section 7.4.1).
3. Interrupt: The System Tick Timer Interrupt is enabled in the NVIC using the appropriate Interrupt Set Enable register. The Systick interrupt is hard-wired within the Cortex-M4 as exception 15. See the ARM Cortex-M4 User Guide referred to in Section 40.1 for details of the System Tick Timer.

25.2 Features

- Times intervals of 10 milliseconds
- Dedicated exception vector
- Can be clocked internally by the CPU clock or by a clock input from a pin (STCLK)

25.3 Description

The System Tick Timer is an integral part of the Cortex-M4. The System Tick Timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the System Tick Timer is a part of the Cortex-M4, it facilitates porting of software by providing a standard timer that is available on Cortex-M4 based devices.

See the ARM Cortex-M4 User Guide referred to in Section 40.1 for details of System Tick Timer operation.

25.4 Operation

The System Tick 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 System Tick Timer may be clocked either from the CPU clock or from the external pin STCLK. The STCLK function shares pin P3[26] with other functions, and must be selected for use as the System Tick Timer clock. In order to generate recurring interrupts at a specific interval, the STRELOAD register must be initialized with the correct value for the desired interval. A default value is provided in the STCALIB register and may be changed by software. The default value gives a 10 millisecond interrupt rate if the CPU clock is set to 100 MHz.

The block diagram of the System Tick Timer is shown below in the Figure 144.

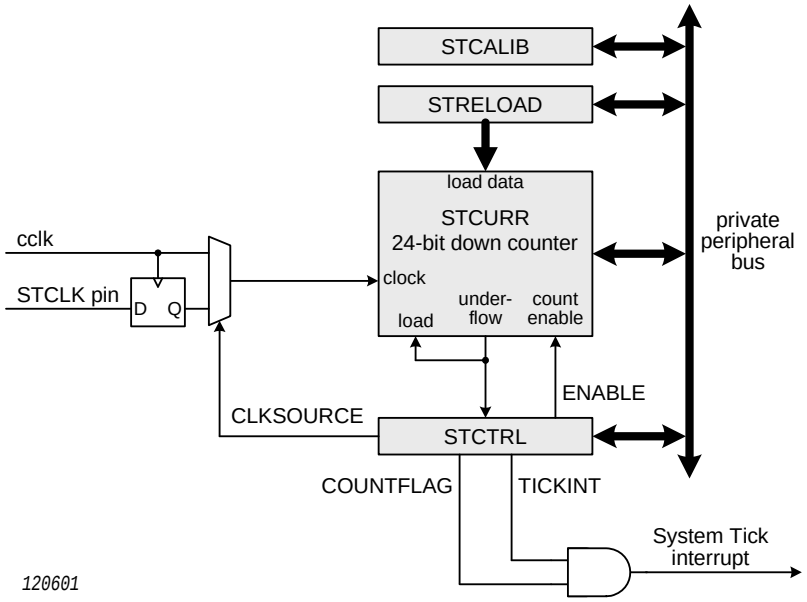


Fig 144. System Tick Timer block diagram

25.5 Register description

Table 551. System Tick Timer register map

Name	Description	Access	Reset value[1]	Address	Table
STCTRL	System Timer Control and status register	R/W	0x4	0xE000 E010	552
STRELOAD	System Timer Reload value register	R/W	0	0xE000 E014	553
STCURR	System Timer Current value register	R/W	0	0xE000 E018	554
STCALIB	System Timer Calibration value register	R/W	0x000F 423F	0xE000 E01C	555

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

25.5.1 System Timer Control and status register

The STCTRL register contains control information for the System Tick Timer, and provides a status flag.

Table 552. System Timer Control and status register (STCTRL - 0xE000 E010) 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 CPU clock is selected. When 0, the external clock pin (STCLK) is selected. If the STCLK pin is selected, each level on the pin must be at least 1 PCLK in duration in order to be sampled. The maximum frequency must therefore be less than PCLK/2.	1
15:3	-	Reserved. Read value is undefined, only zero should be written.	NA
16	COUNTFLAG	System Tick counter flag. This flag is set when the System Tick counter counts down to 0, and is cleared by reading this register.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

25.5.2 System Timer Reload value register

The STRELOAD register is set to the value that will be loaded into the System Tick Timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The STCALIB register may be read and used as the value for STRELOAD if the CPU or external clock is running at the frequency intended for use with the STCALIB value.

Table 553. System Timer Reload value register (STRELOAD - 0xE000 E014) 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

25.5.3 System Timer Current value register

The STCURR register returns the current count from the System Tick counter when it is read by software.

Table 554. System Timer Current value register (STCURR - 0xE000 E018) 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 STCTRL.	0
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

25.5.4 System Timer Calibration value register

The STCALIB register contains a value that is initialized by the Boot Code to a factory programmed value that is appropriate for generating an interrupt every 10 milliseconds if the System Tick Timer is clocked at a frequency of 100 MHz. This is the intended use of the System Tick Timer by ARM. It can be used to generate interrupts at other frequencies by selecting the correct reload value.

Table 555. System Timer Calibration value register (STCALIB - 0xE000 E01C) bit description

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value to get a 10 millisecond System Tick underflow rate when running at 100 MHz. This value initialized at reset with a factory supplied value selected for the LPC408x/407x. The provided values of TENMS, SKEW, and NOREF are applicable only when using a CPU clock or external STCLK source of 100 MHz.	0x0F 423F
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 initialized at reset with a factory supplied value selected for the LPC408x/407x. See the description of TENMS above. 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 initialized at reset with a factory supplied value selected for the LPC408x/407x. See the description of TENMS above. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0

25.6 Example timer calculations

The following examples illustrate selecting System Tick Timer values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the System Tick Timer is intended to be used.

Example 1)

This example is for the System Tick Timer running from the CPU clock (cclk), which is 100 MHz.

STCTRL = 7. This enables the timer and its interrupt, and selects cclk as the clock source.

RELOAD = (cclk / 100) - 1 = 1,000,000 - 1 = 999,999 = 0xF423F

In this case, there is no rounding error, so the result is as accurate as cclk.

Example 2)

This example is for the System Tick Timer running from the CPU clock (cclk), which is 80 MHz.

STCTRL = 7. This enables the timer and its interrupt, and selects cclk as the clock source.

RELOAD = (cclk / 100) - 1 = 800,000 - 1 = 799,999 = 0xC34FF

In this case, there is no rounding error, so the result is as accurate as cclk.

Example 3)

This example is for the CPU clock (cclk) is taken from the Internal RC Oscillator (IRC), factory trimmed to 4 MHz.

STCTRL = 7. This enables the timer and its interrupt, and selects cclk as the clock source.

RELOAD = (F_{IRC} / 100) - 1 = 40,000 - 1 = 39,999 = 0x9C3F

In this case, there is no rounding error, so the result is as accurate as the IRC.

Example 4)

This example is for the System Tick Timer running from an external clock source (the STCLK pin), which in this case happens to be 32.768 kHz.

STCTRL = 3. This enables the timer and its interrupt, and selects the STCLK pin as the clock source. STCLK must be selected as the function of the relevant pin. See Section 7.4.1.

RELOAD = (cclk / 100) - 1 = 327.6 - 1 = 327 (rounded up) = 0x0147

In this case, there is rounding error, so the interrupt rate will drift slightly relative to the input frequency.

26.1 Basic configuration

The PWM is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCPWM1.

   Remark: On reset, PWM1 is enabled (PCPWM1 = 1) and PWM0 is disabled (PCPWM1 = 0).
2. Peripheral clock: The PWMs operate from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Select PWM pins and pin modes for port pins with PWM functions through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: See registers PWMMCR (Table 564) and PWMCCR (Table 567) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

26.2 Features

- The two PWMs have the same operational features. The PWMs may be operated in a synchronized fashion by setting them both up to run at the same rate, then enabling both simultaneously. PWM0 acts as the Master and PWM1 as the slave for this use.
- Counter or Timer operation (may use the peripheral clock or one of the capture inputs as the clock source).
- Seven match registers allow up to 6 single edge controlled or 3 double edge controlled PWM outputs, or a mix of both types. The match registers also allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Supports single edge controlled and/or double edge controlled PWM outputs. Single edge controlled PWM outputs all go high at the beginning of each cycle unless the output is a constant low. Double edge controlled PWM outputs can have either edge occur at any position within a cycle. This allows for both positive going and negative going pulses.
- Pulse period and width can be any number of timer counts. This allows complete flexibility in the trade-off between resolution and repetition rate. All PWM outputs will occur at the same repetition rate.
- Double edge controlled PWM outputs can be programmed to be either positive going or negative going pulses.
- Match register updates are synchronized with pulse outputs to prevent generation of erroneous pulses. Software must "release" new match values before they can become effective.
- May be used as a standard timer if the PWM mode is not enabled.
- A 32 bit Timer/Counter with a programmable 32 bit Prescaler.
- A transition on a capture input signal can trigger a snapshot of the 32-bit timer value. A capture event may also optionally generate an interrupt.

26.3 Description

The PWM function is based on the standard Timer block and inherits all of its features, although many timer functions are not brought out to package pins. The Timer is designed to count cycles of the peripheral clock (PCLK) or a capture input and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. It also includes capture inputs to save the timer value when an input signal transitions, and optionally generate an interrupt when those events occur. The PWM function is in addition to these features, and is based on match register events.

The ability to separately control rising and falling edge locations allows the PWM to be used for more applications. For instance, multi-phase motor control typically requires three non-overlapping PWM outputs with individual control of all three pulse widths and positions.

Two match registers can be used to provide a single edge controlled PWM output. One match register (MR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position. Additional single edge controlled PWM outputs require only one match register each, since the repetition rate is the same for all PWM outputs. Multiple single edge controlled PWM outputs will all have a rising edge at the beginning of each PWM cycle, when an MR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the MR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions. Additional double edge controlled PWM outputs require only two match registers each, since the repetition rate is the same for all PWM outputs.

With double edge controlled PWM outputs, specific match registers control the rising and falling edge of the output. This allows both positive going PWM pulses (when the rising edge occurs prior to the falling edge), and negative going PWM pulses (when the falling edge occurs prior to the rising edge).

Figure 145 shows the block diagram of the PWM. The portions that have been added to the standard timer block are on the right hand side and at the top of the diagram. At the lower left of the diagram may be found the Master Enable output from the Timer Control register that allows the Master PWM (PWM0) to enable both itself and the Slave PWM (PWM1) at the same time, if desired. The Master Enable output from PWM0 is connected to the external enable input of both PWM blocks.

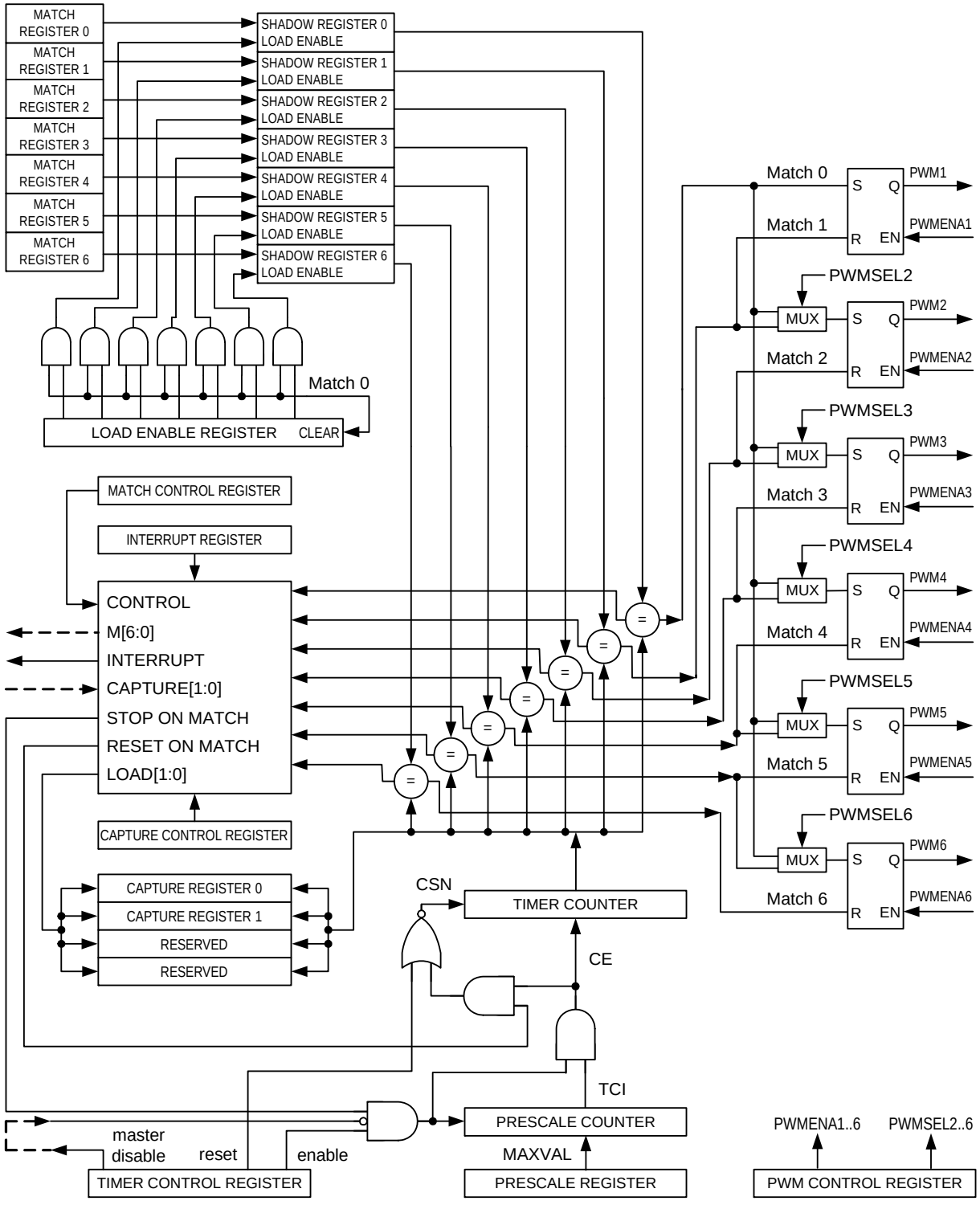
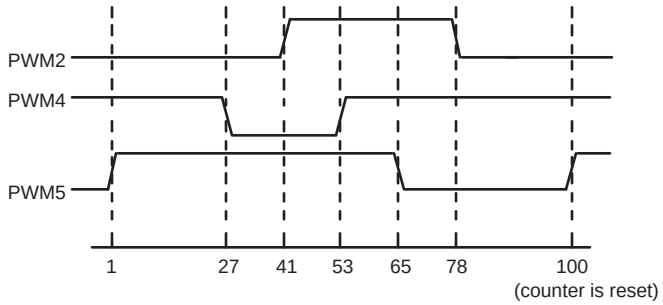


Fig 145. PWM block diagram

26.4 Sample waveform with rules for single and double edge control

A sample of how PWM values relate to waveform outputs is shown in Figure 146. PWM output logic is shown in Figure 145 that allows selection of either single or double edge controlled PWM outputs via the muxes controlled by the PWMSELn bits. The match register selections for various PWM outputs is shown in Table 556. This implementation supports up to N-1 single edge PWM outputs or (N-1)/2 double edge PWM outputs, where N is the number of match registers and outputs that are implemented. PWM types can be mixed if desired.



The waveforms below show a single PWM cycle and demonstrate PWM outputs under the following conditions:

The timer is configured for PWM mode (counter resets to 1).

Match 0 is configured to reset the timer/counter when a match event occurs.

Control bits PWMSEL2 and PWMSEL4 are set.

The Match register values are as follows:

MR0 = 100 (PWM rate)

MR1 = 41, MR2 = 78 (PWM[2] output)

MR3 = 53, MR4 = 27 (PWM[4] output)

MR5 = 65 (PWM[5] output)

Fig 146. Sample PWM waveforms

Table 556. Set and reset inputs for PWM Flip-Flops

PWM Channel	Single Edge PWM (PWMSELn = 0)		Double Edge PWM (PWMSELn = 1)	
	Set by	Reset by	Set by	Reset by
1	Match 0	Match 1	Match 0[1]	Match 1[1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2[2]	Match 3[2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4[2]	Match 5[2]
6	Match 0	Match 6	Match 5	Match 6

[1] Identical to single edge mode in this case since Match 0 is the neighboring match register. Essentially, PWM1 cannot be a double edged output.

[2] It is generally not advantageous to use PWM channels 3 and 5 for double edge PWM outputs because it would reduce the number of double edge PWM outputs that are possible. Using PWM[2], PWM[4], and PWM[6] for double edge PWM outputs provides the most pairings.

26.4.1 Rules for Single Edge Controlled PWM Outputs

1. All single edge controlled PWM outputs go high at the beginning of a PWM cycle unless their match value is equal to 0.
2. Each PWM output will go low when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM rate), the PWM output remains continuously high.

26.4.2 Rules for Double Edge Controlled PWM Outputs

Five rules are used to determine the next value of a PWM output when a new cycle is about to begin:

1. The match values for the **next** PWM cycle are used at the end of a PWM cycle (a time point which is coincident with the beginning of the next PWM cycle), except as noted in rule 3.
2. A match value equal to 0 or the current PWM rate (the same as the Match channel 0 value) have the same effect, except as noted in rule 3. For example, a request for a falling edge at the beginning of the PWM cycle has the same effect as a request for a falling edge at the end of a PWM cycle.
3. When match values are changing, if one of the "old" match values is equal to the PWM rate, it is used again once if the neither of the new match values are equal to 0 or the PWM rate, and there was no old match value equal to 0.
4. If both a set and a clear of a PWM output are requested at the same time, clear takes precedence. This can occur when the set and clear match values are the same as in, or when the set or clear value equals 0 and the other value equals the PWM rate.
5. If a match value is out of range (i.e. greater than the PWM rate value), no match event occurs and that match channel has no effect on the output. This means that the PWM output will remain always in one state, allowing always low, always high, or "no change" outputs.

26.5 Pin description

Table 557 gives a brief summary of each of PWM related pins.

Table 557. Pin summary

Pin	Type	Description
PWM0[6:1]	Output	Outputs from PWM0 channels 6 to 1.
PWM0_CAP0	Input	Capture input for PWM0. A transition on the capture pin can be configured to load the corresponding Capture register with the value of the Timer Counter and optionally generate an interrupt. 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.
PWM1[6:1]	Output	Outputs from PWM1 channels 6 to 1.
PWM1_CAP1:0	Input	Capture inputs for PWM1. A transition on the capture pin can be configured to load the corresponding Capture register with the value of the Timer Counter and optionally generate an interrupt.

26.6 Register description

Table 558. Register overview: PWM (base addresses 0x4001 4000 (PWM0) and 0x4001 8000 (PWM1))

Name	Access	Address offset	Description	Reset Value[1]	Section
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts, or read to identify which PWM interrupt sources are pending.	0	26.6.1
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions.	0	26.6.2
TC	R/W	0x008	Timer Counter. The 32 bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0	26.6.3
PR	R/W	0x00C	Prescale Register. Determines how often the PWM counter is incremented.	0	26.6.4
PC	R/W	0x010	Prescale Counter. Prescaler for the main PWM counter.	0	26.6.5
MCR	R/W	0x014	Match Control Register. The MCR is used to control whether an interrupt is generated and if the PWM counter is reset when a Match occurs.	0	26.6.6
MR0	R/W	0x018	Match Register 0. Match registers are continuously compared to the PWM counter in order to control PWM output edges.	0	26.6.7
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0	26.6.7
MR2	R/W	0x020	Match Register 2. See MR0 description.	0	26.6.7
MR3	R/W	0x024	Match Register 3. See MR0 description.	0	26.6.7
CCR	R/W	0x028	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 for a capture event.	0	26.6.8
CR0	RO	0x02C	Capture Register 0. CR0 of PWMn is loaded with the value of the TC when there is an event on the PWMn_CAP0 input.	0	26.6.9
CR1	RO	0x030	Capture Register 1. See CR0 description.	0	26.6.9
MR4	R/W	0x040	Match Register 4. See MR0 description.	0	26.6.7
MR5	R/W	0x044	Match Register 5. See MR0 description.	0	26.6.7
MR6	R/W	0x048	Match Register 6. See MR0 description.	0	26.6.7
PCR	R/W	0x04C	PWM Control Register. Enables PWM outputs and selects either single edge or double edge controlled PWM outputs.	0	26.6.10
LER	R/W	0x050	Load Enable Register. Enables use of updated PWM match values.	0	26.6.11
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0	26.6.12

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

26.6.1 PWM Interrupt Register

The PWM Interrupt register consists of eleven bits (Table 559), seven for the match interrupts and four reserved. If an interrupt is generated then the corresponding bit in the PWMIR 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.

Table 559: PWM Interrupt Register (IR - address 0x4001 4000 (PWM0) and 0x4001 8000 (PWM1)) bit description

Bit	Symbol	Description	Reset Value
0	PWMMR0INT	Interrupt flag for PWM match channel 0.	0
1	PWMMR1INT	Interrupt flag for PWM match channel 1.	0
2	PWMMR2INT	Interrupt flag for PWM match channel 2.	0
3	PWMMR3INT	Interrupt flag for PWM match channel 3.	0
4	PWMCAP0 INT	Interrupt flag for capture input 0	0
5	PWMCAP1INT	Interrupt flag for capture input 1 (available in PWM1IR only; this bit is reserved in PWM0IR).	0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	PWMMR4INT	Interrupt flag for PWM match channel 4.	0
9	PWMMR5INT	Interrupt flag for PWM match channel 5.	0
10	PWMMR6INT	Interrupt flag for PWM match channel 6.	0
31:11	-	Reserved. Read value is undefined, only zero should be written.	NA

26.6.2 PWM Timer Control Register

The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter. The function of each of the bits is shown in Table 560.

Table 560: PWM Timer Control Register (TCR - address 0x4001 4004 (PWM0) and 0x4001 8004 (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	CE		Counter Enable	0
		1	The PWM Timer Counter and PWM Prescale Counter are enabled for counting.	
		0	The counters are disabled.	
1	CR		Counter Reset	0
		1	The PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until this bit is returned to zero.	
		0	Clear reset.	
2	-		Reserved. Read value is undefined, only zero should be written.	NA
3	PWMEN		PWM Enable	0
		1	PWM mode is enabled (counter resets to 1). PWM mode causes the shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match Register 0 - MR0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	
		0	Timer mode is enabled (counter resets to 0).	
4	MDIS		Master Disable (PWM0 only). The two PWMs may be synchronized using the Master Disable control bit. The Master disable bit of the Master PWM (PWM0 module) controls a secondary enable input to both PWMs, as shown in Figure 145. This bit has no function in the Slave PWM (PWM1).	0
		1	Master use. PWM0 is the master, and both PWMs are enabled for counting.	
		0	Individual use. The PWMs are used independently, and the individual Counter Enable bits are used to control the PWMs.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	NA

26.6.3 PWM Timer Counter

The 32-bit PWM Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the PWMTC 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 561. PWM Timer counter registers (TC - addresses 0x4001 4008 (PWM0), 0x4001 8008 (PWM1)) bit description

Bit	Symbol	Description	Reset value
31:0	TC	Timer counter value.	0

26.6.4 PWM Prescale Register

The 32-bit PWM Prescale Register specifies the maximum value for the PWM Prescale Counter.

Table 562. PWM prescale registers (PR - addresses 0x4001 400C (PWM0), 0x4001 800C (PWM1)) bit description

Bit	Symbol	Description	Reset value
31:0	PM	Prescale counter maximum value.	0

26.6.5 PWM Prescale Counter Register

The 32-bit PWM Prescale Counter controls division of PCLK by some constant value before it is applied to the PWM Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The PWM Prescale Counter is incremented on every PCLK. When it reaches the value stored in the PWM Prescale Register, the PWM Timer Counter is incremented and the PWM Prescale Counter is reset on the next PCLK. This causes the PWM TC to increment on every PCLK when PWMPR = 0, every 2 PCLKs when PWMPR = 1, etc.

Table 563. PWM prescale counter registers (PC - addresses 0x4001 4010 (PWM0), 0x4001 8010 (PWM1)) bit description

Bit	Symbol	Description	Reset value
31:0	PC	Prescale counter value.	0

26.6.6 PWM Match Control Register

The PWM Match Control registers are used to control what operations are performed when one of the PWM Match registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 564.

Table 564. Match Control Register (MCR - address 0x4001 4014 (PWM0) and 0x4001 8014 (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	PWMMR0I		Interrupt PWM0	0
		0	Disabled.	
		1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.	
1	PWMMR0R		Reset PWM0	0
		0	Disabled.	
		1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.	
2	PWMMR0S		Stop PWM0	0
		0	Disabled	
		1	Stop on PWMMR0: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR0 matches the PWMTC.	
3	PWMMR1I		Interrupt PWM1	0
		0	Disabled.	
		1	Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC.	

Table 564. Match Control Register (MCR - address 0x4001 4014 (PWM0) and 0x4001 8014 (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
4	PWMMR1R		Reset PWM1	0
		0	Disabled.	
		1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.	
5	PWMMR1S		Stop PWM1	0
		0	Disabled	
		1	Stop on PWMMR1: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR1 matches the PWMTC.	
6	PWMMR2I		Interrupt PWM0	0
		0	Disabled.	
		1	Interrupt on PWMMR2: an interrupt is generated when PWMMR2 matches the value in the PWMTC.	
7	PWMMR2R		Reset PWM0	0
		0	Disabled.	
		1	Reset on PWMMR2: the PWMTC will be reset if PWMMR2 matches it.	
8	PWMMR2S		Stop PWM0	0
		0	Disabled	
		1	Stop on PWMMR2: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR0 matches the PWMTC.	
9	PWMMR3I		Interrupt PWM3	0
		0	Disabled.	
		1	Interrupt on PWMMR3: an interrupt is generated when PWMMR3 matches the value in the PWMTC.	
10	PWMMR3R		Reset PWM3	0
		0	Disabled.	
		1	Reset on PWMMR3: the PWMTC will be reset if PWMMR3 matches it.	
11	PWMMR3S		Stop PWM0	0
		0	Disabled	
		1	Stop on PWMMR3: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR0 matches the PWMTC.	
12	PWMMR4I		Interrupt PWM4	0
		0	Disabled.	
		1	Interrupt on PWMMR4: an interrupt is generated when PWMMR4 matches the value in the PWMTC.	
13	PWMMR4R		Reset PWM4	0
		0	Disabled.	
		1	Reset on PWMMR4: the PWMTC will be reset if PWMMR4 matches it.	
14	PWMMR4S		Stop PWM4	0
		0	Disabled	
		1	Stop on PWMMR4: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR4 matches the PWMTC.	

Table 564. Match Control Register (MCR - address 0x4001 4014 (PWM0) and 0x4001 8014 (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
15	PWMMR5I		Interrupt PWM5	0
		0	Disabled.	
		1	Interrupt on PWMMR5: an interrupt is generated when PWMMR5 matches the value in the PWMTC.	
16	PWMMR5R		Reset PWM5	0
		0	Disabled.	
		1	Reset on PWMMR5: the PWMTC will be reset if PWMMR5 matches it.	
17	PWMMR5S		Stop PWM5	0
		0	Disabled	
		1	Stop on PWMMR5: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR5 matches the PWMTC.	
18	PWMMR6I		Interrupt PWM6	0
		0	Disabled.	
		1	Interrupt on PWMMR6: an interrupt is generated when PWMMR6 matches the value in the PWMTC.	
19	PWMMR6R		Reset PWM6	0
		0	Disabled.	
		1	Reset on PWMMR6: the PWMTC will be reset if PWMMR6 matches it.	
20	PWMMR6S		Stop PWM6	0
		0	Disabled	
		1	Stop on PWMMR6: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR6 matches the PWMTC.	
31:21	-		Reserved. Read value is undefined, only zero should be written.	NA

26.6.7 PWM Match Registers

The 32-bit PWM Match register values are continuously compared to the PWM Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the PWM Timer Counter, or stop the timer. Actions are controlled by the settings in the PWMMCR register.

Table 565. PWM match registers (MR[0:3], addresses 0x4001 4018 (MR0) to 0x4001 4024 (MR3) (PWM0), 0x4001 8018 (MR0) to 0x4001 5024 (MR3) (PWM1)) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

Table 566. PWM match registers (MR[4:6], addresses 0x4001 4040 (MR4) to 0x4001 4048 (MR6) (PWM0), 0x4001 8040 (MR4) to 0x4001 5048 (MR6) (PWM1)) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

26.6.8 PWM Capture Control Register

The Capture Control register is used to control whether any of the Capture registers is loaded with the value in the Timer Counter when a capture event occurs on PWM0_CAP0 or PWM1_CAP1:0, 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 descriptions below, “n” represents the Timer number, 0 or 1.

Note: If Counter mode is selected for a particular PWM_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 two PWM_CAP inputs.

Table 567: PWM Capture Control Register (CCR - address 0x4001 4028 (PWM0) and 0x4001 8028 (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
0	CAP0_R		Capture on PWMn_CAP0 rising edge	0
		0	Disabled. This feature is disabled.	
		1	Rising edge. A synchronously sampled rising edge on PWMn_CAP0 will cause CR0 to be loaded with the contents of the TC.	
1	CAP0_F		Capture on PWMn_CAP0 falling edge	0
		0	Disabled. This feature is disabled.	
		1	Falling edge. A synchronously sampled falling edge on PWMn_CAP0 will cause CR0 to be loaded with the contents of TC.	
2	CAP0_I		Interrupt on PWMn_CAP0 event	0
		0	Disabled. This feature is disabled.	
		1	Interrupt. A CR0 load due to a PWMn_CAP0 event will generate an interrupt.	
3	CAP1_R		Capture on PWMn_CAP1 rising edge. Reserved for PWM0.	0
		0	Disabled. This feature is disabled.	
		1	Rising edge. A synchronously sampled rising edge on PWMn_CAP1 will cause CR1 to be loaded with the contents of the TC.	
4	CAP1_F		Capture on PWMn_CAP1 falling edge. Reserved for PWM0.	0
		0	Disabled. This feature is disabled.	
		1	Falling edge. A synchronously sampled falling edge on PWMn_CAP1 will cause CR1 to be loaded with the contents of TC.	
5	CAP1_I		Interrupt on PWMn_CAP1 event. Reserved for PWM0.	0
		0	Disabled. This feature is disabled.	
		1	Interrupt. A CR1 load due to a PWMn_CAP1 event will generate an interrupt.	
31:6	-		Reserved. Read value is undefined, only zero should be written.	NA

26.6.9 PWM Capture Registers

Each 32-bit Capture Register is associated with a device pin and may be loaded with the PWM Timer Counter value when a specified event occurs on that pin. The settings in the PWM Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

Table 568. PWM capture registers (CR[0:1], address 0x4001 402C (CR0) to 0x4001 4038 (CR3) (PWM0), 0x4001 802C (CR0) to 0x4001 8038 (CR3) (PWM1)) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

26.6.10 PWM Control Registers

The PWM Control registers are used to enable and select the type of each PWM channel. The function of each of the bits are shown in Table 569.

Table 569: PWM Control Registers (PCR - address 0x4001 404C (PWM0) and 0x4001 804C (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	-		Reserved.	-
2	PWMSEL2		PWM[2] output single/double edge mode control.	0
		0	Single edge controlled mode is selected.	
		1	Double edge controlled mode is selected.	
3	PWMSEL3		PWM[3] output edge control.	0
		0	Single edge controlled mode is selected.	
		1	Double edge controlled mode is selected.	
4	PWMSEL4		PWM[4] output edge control.	0
		0	Single edge controlled mode is selected.	
		1	Double edge controlled mode is selected.	
5	PWMSEL5		PWM[5] output edge control.	0
		0	Single edge controlled mode is selected.	
		1	Double edge controlled mode is selected.	
6	PWMSEL6		PWM[6] output edge control.	0
		0	Single edge controlled mode is selected.	
		1	Double edge controlled mode is selected.	
8:7	-		Reserved. Read value is undefined, only zero should be written.	-
9	PWMENA1		PWM[1] output enable control.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	
10	PWMENA2		PWM[2] output enable control.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	
11	PWMENA3		PWM[3] output enable control.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	
12	PWMENA4		PWM[4] output enable control.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	

Table 569: PWM Control Registers (PCR - address 0x4001 404C (PWM0) and 0x4001 804C (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
13	PWMENA5		PWM[5] output enable control.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	
14	PWMENA6		PWM[6] output enable control. See PWMENA1 for details.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	
31:15	-		Unused, always zero.	NA

26.6.11 PWM Latch Enable Register

The PWM Latch Enable registers are used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to the location of a PWM Match register while the Timer is in PWM mode, the value is actually held in a shadow register and not used immediately.

When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the actual Match registers if the corresponding bit in the Latch Enable register has been set. At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared. Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.

For example, if PWM is configured for double edge operation and is currently running, a typical sequence of events for changing the timing would be:

- Write a new value to the PWM Match1 register.
- Write a new value to the PWM Match2 register.
- Write to the PWMLER, setting bits 1 and 2 at the same time.
- The altered values will become effective at the next reset of the timer (when a PWM Match 0 event occurs).

The order of writing the two PWM Match registers is not important, since neither value will be used until after the write to PWMLER. This insures that both values go into effect at the same time, if that is required. A single value may be altered in the same way if needed.

Table 570: PWM Latch Enable Register (LER - address 0x4001 4050 (PWM0) and 0x4001 8050 (PWM1)) bit description

Bit	Symbol	Description	Reset Value
0	MAT0LATCHEN	Enable PWM Match 0 Latch. PWM MR0 register update control. Writing a one to this bit allows the last value written to the PWM Match Register 0 to be become effective when the timer is next reset by a PWM Match event. See Section 26.6.6.	0
1	MAT1LATCHEN	Enable PWM Match 1 Latch. PWM MR1 register update control. See bit 0 for details.	0
2	MAT2LATCHEN	Enable PWM Match 2 Latch. PWM MR2 register update control. See bit 0 for details.	0

Table 570: PWM Latch Enable Register (LER - address 0x4001 4050 (PWM0) and 0x4001 8050 (PWM1)) bit description

Bit	Symbol	Description	Reset Value
3	MAT3LATCHEN	Enable PWM Match 3 Latch. PWM MR3 register update control. See bit 0 for details.	0
4	MAT4LATCHEN	Enable PWM Match 4 Latch. PWM MR4 register update control. See bit 0 for details.	0
5	MAT5LATCHEN	Enable PWM Match 5 Latch. PWM MR5 register update control. See bit 0 for details.	0
6	MAT6LATCHEN	Enable PWM Match 6 Latch. PWM MR6 register update control. See bit 0 for details.	0
31:7	-	Reserved. Read value is undefined, only zero should be written.	NA

26.6.12 PWM 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 edges for counting. The function of each of the bits is shown in Table 571.

Remark: the input frequency of PWM_CAP must not exceed PCLK/4. When the PWM clock is supplied via the PWM_CAP pin, at no time can a high or low level of the signal on this pin last less than 2 PCLKs.

Table 571: PWM Count control Register (CTCR - address 0x4001 4070 (PWM0) and 0x4001 8070 (PWM1)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	MOD		Counter/Timer Mode	0
		0x0	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale register.	
		0x1	Rising edge counter Mode: the TC is incremented on rising edges of the PWM_CAP input selected by bits 3:2.	
		0x2	Falling edge counter Mode: the TC is incremented on falling edges of the PWM_CAP input selected by bits 3:2.	
		0x3	Dual edge counter Mode: the TC is incremented on both edges of the PWM_CAP input selected by bits 3:2.	
3:2	CIS		Count Input Select. When bits 1:0 are not 00, these bits select which PWM_CAP pin carries the signal used to increment the TC. Other combinations are reserved.	0
		0x0	For PWM0: 00 = PWM0_CAP0 (Other combinations are reserved) For PWM1: 00 = PWM1_CAP0, 01 = PWM1_CAP1 (Other combinations are reserved)	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

27.1 Basic configuration

The Motor Control PWM is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCMCPWM.

   Remark: On reset, the MCPWM is disabled (PCMCPWM = 0).
2. Peripheral clock: The MCPWM operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Select MCPWM pins through and pin modes for port pins with MCPWM functions through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: See Section 27.9.8. The MCPWM interrupt is enabled in the NVIC using the appropriate Interrupt Set Enable register.

27.2 Introduction

The Motor Control PWM (MCPWM) is optimized for three-phase AC and DC motor control applications, but can be used in many other applications that need timing, counting, capture, and comparison.

27.3 Description

The MCPWM contains three independent channels, each including:

- a 32-bit Timer/Counter (TC)
- a 32-bit Limit register (LIM)
- a 32-bit Match register (MAT)
- a 10-bit dead-time register (DT) and an associated 10-bit dead-time counter
- a 32-bit capture register (CAP)
- two modulated outputs (MC_A and MC_B) with opposite polarities
- a period interrupt, a pulse-width interrupt, and a capture interrupt

Input pins MC_FB0-2 can trigger TC capture or increment a channel's TC. A global Abort input can force all of the channels into "A passive" state and cause an interrupt.

27.4 Pin description

Table 572 lists the MCPWM pins.

Table 572. Pin summary

Pin	Type	Description
MC_0A, MC_0B	O	Outputs A and B for channel 0
MC_1A, MC_1B	O	Outputs A and B for channel 1
MC_2A, MC_2B	O	Outputs A and B for channel 2
$\overline{\text{MC\_ABORT}}$	I	Low-active Fast Abort
MC_FB0, MC_FB1, MC_FB2	I	Inputs for channels 0, 1, 2. Each level on this pin must be at least 1 PCLK in duration in order to be sampled.

27.5 Block diagram

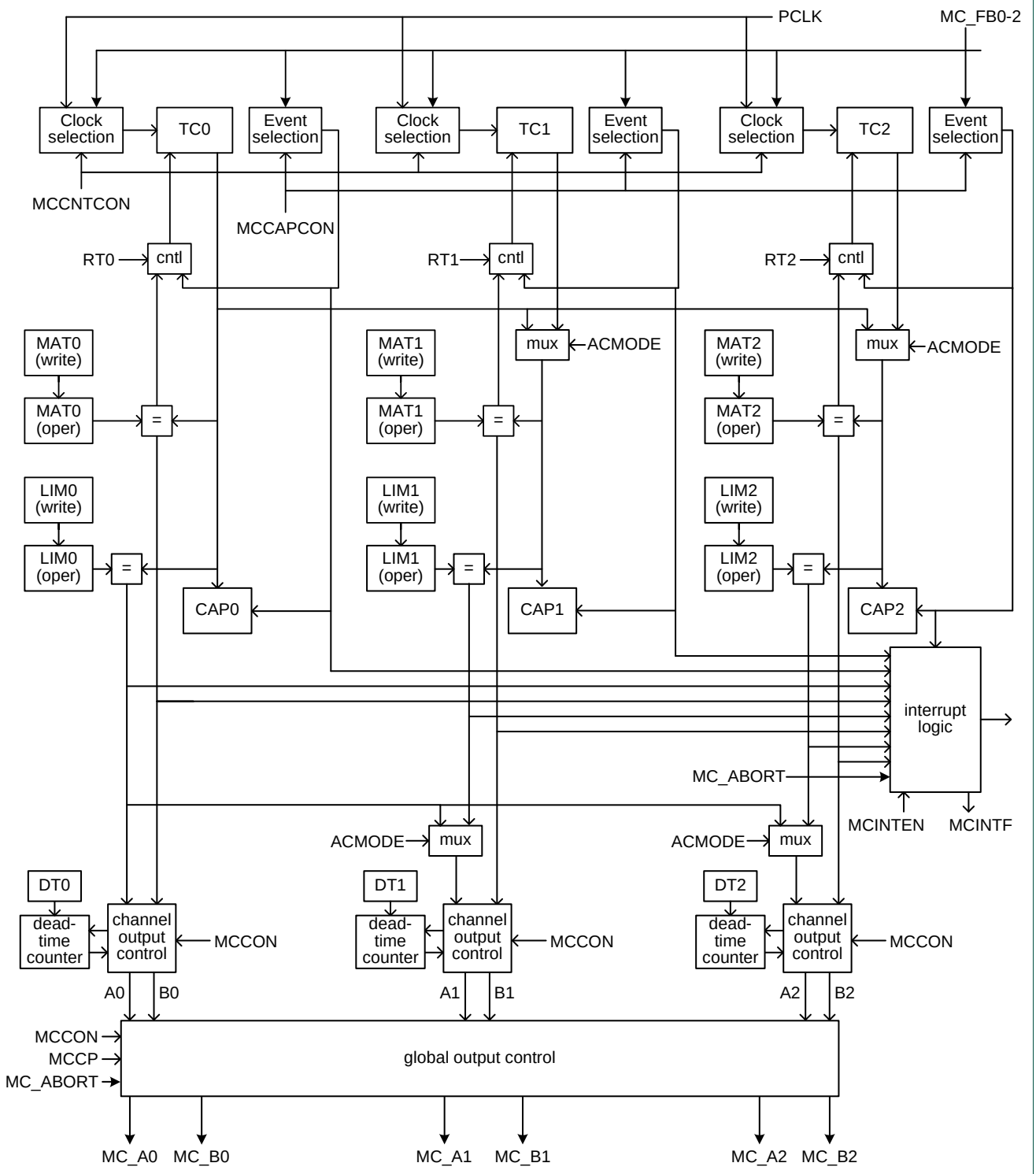


Fig 147. MCPWM Block Diagram

27.6 Configuring other modules for MCPWM use

Configure the following registers in other modules before using the Motor Control PWM:

1. Power: in the PCONP register (Section 3.3.2.2), set bit PCMCPWM.
   Remark: On reset the MCPWM is disabled (PCMCPWM = 0).
2. Peripheral clock: the MCPWM operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: select MCPWM functions and pin modes for these pins through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: See Section 27.8.9 for motor control PWM related interrupts. Interrupts can be enabled in the NVIC using the appropriate Interrupt Set Enable register.

27.7 General operation

Section 27.9 includes detailed descriptions of the various modes of MCPWM operation, but a quick preview here will provide background for the register descriptions below.

The MCPWM includes 3 channels, each of which controls a pair of outputs that in turn can control something off-chip, like one set of coils in a motor. Each channel includes a Timer/Counter (TC) register that is incremented by a processor clock (timer mode) or by an input pin (counter mode).

Each channel has a Limit register that is compared to the TC value, and when a match occurs the TC is “recycled” in one of two ways. In “edge-aligned mode” the TC is reset to 0, while in “centered mode” a match switches the TC into a state in which it decrements on each processor clock or input pin transition until it reaches 0, at which time it starts counting up again.

Each channel also includes a Match register that holds a smaller value than the Limit register. In edge-aligned mode the channel’s outputs are switched whenever the TC matches either the Match or Limit register, while in center-aligned mode they are switched only when it matches the Match register.

So the Limit register controls the period of the outputs, while the Match register controls how much of each period the outputs spend in each state. Having a small value in the Limit register minimizes “ripple” if the output is integrated into a voltage, and allows the MCPWM to control devices that operate at high speed.

The “downside” of small values in the Limit register is that they reduce the resolution of the duty cycle controlled by the Match register. If you have 8 in the Limit register, the Match register can only select the duty cycle among 0%, 12.5%, 25%, …, 87.5%, or 100%. In general, the resolution of each step in the Match value is 1 divided by the Limit value.

This trade-off between resolution and period/frequency is inherent in the design of pulse width modulators.

27.8 Register description

"Control" registers and "interrupt" registers have separate read, set, and clear addresses. Reading such a register's read address (e.g. MCCON) yields the state of the register bits. Writing ones to the set address (e.g. MCCON_SET) sets register bits, and writing ones to the clear address (e.g. MCCON_CLR) clears register bits.

The Capture registers (MCCAP) are read-only, and the write-only MCCAP_CLR address can be used to clear one or more of them. All the other MCPWM registers (MCTIM, MCPER, MCPW, MCDEADTIME, and MCCP) are normal read-write registers.

Table 573. Register overview: Motor Control Pulse Width Modulator (MCPWM) (base address 0x400B 8000)

Name	Access	Address offset	Description	Reset value	Reference
CON	RO	0x000	PWM Control read address	0	Table 574
CON_SET	WO	0x004	PWM Control set address	-	Table 575
CON_CLR	WO	0x008	PWM Control clear address	-	Table 576
CAPCON	RO	0x00C	Capture Control read address	0	Table 577
CAPCON_SET	WO	0x010	Capture Control set address	-	Table 578
CAPCON_CLR	WO	0x014	Event Control clear address	-	Table 579
TC0	R/W	0x018	Timer Counter register, channel 0	0	Table 580
TC1	R/W	0x01C	Timer Counter register, channel 1	0	Table 580
TC2	R/W	0x020	Timer Counter register, channel 2	0	Table 580
LIM0	R/W	0x024	Limit register, channel 0	0xFFFF FFFF	Table 581
LIM1	R/W	0x028	Limit register, channel 1	0xFFFF FFFF	Table 581
LIM2	R/W	0x02C	Limit register, channel 2	0xFFFF FFFF	Table 581
MAT0	R/W	0x030	Match register, channel 0	0xFFFF FFFF	Table 582
MAT1	R/W	0x034	Match register, channel 1	0xFFFF FFFF	Table 582
MAT2	R/W	0x038	Match register, channel 2	0xFFFF FFFF	Table 582
DT	R/W	0x03C	Dead time register	0x3FFF FFFF	Table 583
MCCP	R/W	0x040	Communication Pattern register	0	Table 584
CAP0	RO	0x044	Capture register, channel 0	0	Table 585
CAP1	RO	0x048	Capture register, channel 1	0	Table 585
CAP2	RO	0x04C	Capture register, channel 2	0	Table 585
INTEN	RO	0x050	Interrupt Enable read address	0	Table 587
INTEN_SET	WO	0x054	Interrupt Enable set address	-	Table 588
INTEN_CLR	WO	0x058	Interrupt Enable clear address	-	Table 589
CNTCON	RO	0x05C	Count Control read address	0	Table 590
CNTCON_SET	WO	0x060	Count Control set address	-	Table 591
CNTCON_CLR	WO	0x064	Count Control clear address	-	Table 592
INTF	RO	0x068	Interrupt flags read address	0	Table 593
INTF_SET	WO	0x06C	Interrupt flags set address	-	Table 594
INTF_CLR	WO	0x070	Interrupt flags clear address	-	Table 595
CAP_CLR	WO	0x074	Capture clear address	-	Table 596

27.8.1 MCPWM Control register

27.8.1.1 MCPWM Control read address

The CON register controls the operation of all channels of the PWM. This address is read-only, but the underlying register can be modified by writing to addresses CON_SET and CON_CLR.

Table 574. MCPWM Control read address (CON - 0x400B 8000) bit description

Bit	Symbol	Value	Description	Reset value
0	RUN0		Stops/starts timer channel 0.	0
		0	Stop.	
		1	Run.	
1	CENTER0		Edge/center aligned operation for channel 0.	0
		0	Edge-aligned.	
		1	Center-aligned.	
2	POLA0		Selects polarity of the MCOA0 and MCOB0 pins.	0
		0	Passive state is LOW, active state is HIGH.	
		1	Passive state is HIGH, active state is LOW.	
3	DTE0		Controls the dead-time feature for channel 0.	0
		0	Dead-time disabled.	
		1	Dead-time enabled.	
4	DISUP0		Enable/disable updates of functional registers for channel 0 (see Section 27.9.2).	0
		0	Functional registers are updated from the write registers at the end of each PWM cycle.	
		1	Functional registers remain the same as long as the timer is running.	
7:5	-	-	Reserved.	
8	RUN1		Stops/starts timer channel 1.	0
		0	Stop.	
		1	Run.	
9	CENTER1		Edge/center aligned operation for channel 1.	0
		0	Edge-aligned.	
		1	Center-aligned.	
10	POLA1		Selects polarity of the MCOA1 and MCOB1 pins.	0
		0	Passive state is LOW, active state is HIGH.	
		1	Passive state is HIGH, active state is LOW.	
11	DTE1		Controls the dead-time feature for channel 1.	0
		0	Dead-time disabled.	
		1	Dead-time enabled.	
12	DISUP1		Enable/disable updates of functional registers for channel 1 (see Section 27.9.2).	0
		0	Functional registers are updated from the write registers at the end of each PWM cycle.	
		1	Functional registers remain the same as long as the timer is running.	
15:13	-	-	Reserved.	0

Table 574. MCPWM Control read address (CON - 0x400B 8000) bit description

Bit	Symbol	Value	Description	Reset value
16	RUN2		Stops/starts timer channel 2.	0
		0	Stop.	
		1	Run.	
17	CENTER2		Edge/center aligned operation for channel 2.	0
		0	Edge-aligned.	
		1	Center-aligned.	
18	POLA2		Selects polarity of the MCOA2 and MCOB2 pins.	0
		0	Passive state is LOW, active state is HIGH.	
		1	Passive state is HIGH, active state is LOW.	
19	DTE2		Controls the dead-time feature for channel 1.	0
		0	Dead-time disabled.	
		1	Dead-time enabled.	
20	DISUP2		Enable/disable updates of functional registers for channel 2 (see Section 27.9.2).	0
		0	Functional registers are updated from the write registers at the end of each PWM cycle.	
		1	Functional registers remain the same as long as the timer is running.	
28:21	-	-	Reserved.	
29	INVBDC		Controls the polarity of the MCOB outputs for all 3 channels. This bit is typically set to 1 only in 3-phase DC mode.	
		0	The MCOB outputs have opposite polarity from the MCOA outputs (aside from dead time).	
		1	The MCOB outputs have the same basic polarity as the MCOA outputs. (see Section 27.9.6)	
30	ACMODE		3-phase AC mode select (see Section 27.9.7).	0
		0	3-phase AC-mode off: Each PWM channel uses its own timer-counter and period register.	
		1	3-phase AC-mode on: All PWM channels use the timer-counter and period register of channel 0.	
31	DCMODE		3-phase DC mode select (see Section 27.9.6).	0
		0	3-phase DC mode off: PWM channels are independent (unless bit ACMODE = 1)	
		1	3-phase DC mode on: The internal MCOA0 output is routed through the CP register (i.e. a mask) register to all six PWM outputs.	

27.8.1.2 MCPWM Control set address

Writing ones to this write-only address sets the corresponding bits in MCCON.

Table 575. MCPWM Control set address (CON_SET - 0x400B 8004) bit description

Bit	Symbol	Description	Reset value
0	RUN0_SET	Writing a one sets the corresponding bit in the CON register.	-
1	CENTER0_SET	Writing a one sets the corresponding bit in the CON register.	-
2	POLA0_SET	Writing a one sets the corresponding bit in the CON register.	-
3	DTE0_SET	Writing a one sets the corresponding bit in the CON register.	-

Table 575. MCPWM Control set address (CON_SET - 0x400B 8004) bit description

Bit	Symbol	Description	Reset value
4	DISUP0_SET	Writing a one sets the corresponding bit in the CON register.	-
7:5	-	Writing a one sets the corresponding bit in the CON register.	-
8	RUN1_SET	Writing a one sets the corresponding bit in the CON register.	-
9	CENTER1_SET	Writing a one sets the corresponding bit in the CON register.	-
10	POLA1_SET	Writing a one sets the corresponding bit in the CON register.	-
11	DTE1_SET	Writing a one sets the corresponding bit in the CON register.	-
12	DISUP1_SET	Writing a one sets the corresponding bit in the CON register.	-
15:13	-	Writing a one sets the corresponding bit in the CON register.	-
16	RUN2_SET	Writing a one sets the corresponding bit in the CON register.	-
17	CENTER2_SET	Writing a one sets the corresponding bit in the CON register.	-
18	POLA2_SET	Writing a one sets the corresponding bit in the CON register.	-
19	DTE2_SET	Writing a one sets the corresponding bit in the CON register.	-
20	DISUP2_SET	Writing a one sets the corresponding bit in the CON register.	-
28:21	-	Writing a one sets the corresponding bit in the CON register.	-
29	INVBDC_SET	Writing a one sets the corresponding bit in the CON register.	-
30	ACMODE_SET	Writing a one sets the corresponding bit in the CON register.	-
31	DCMODE_SET	Writing a one sets the corresponding bit in the CON register.	-

27.8.1.3 MCPWM Control clear address

Writing ones to this write-only address clears the corresponding bits in CON.

Table 576. MCPWM Control clear address (CON_CLR - 0x400B 8008) bit description

Bit	Symbol	Description	Reset value
0	RUN0_CLR	Writing a one clears the corresponding bit in the CON register.	-
1	CENTER0_CLR	Writing a one clears the corresponding bit in the CON register.	-
2	POLA0_CLR	Writing a one clears the corresponding bit in the CON register.	-
3	DTE0_CLR	Writing a one clears the corresponding bit in the CON register.	-
4	DISUP0_CLR	Writing a one clears the corresponding bit in the CON register.	-
7:5	-	Writing a one clears the corresponding bit in the CON register.	-
8	RUN1_CLR	Writing a one clears the corresponding bit in the CON register.	-
9	CENTER1_CLR	Writing a one clears the corresponding bit in the CON register.	-
10	POLA1_CLR	Writing a one clears the corresponding bit in the CON register.	-
11	DTE1_CLR	Writing a one clears the corresponding bit in the CON register.	-
12	DISUP1_CLR	Writing a one clears the corresponding bit in the CON register.	-
15:13	-	Writing a one clears the corresponding bit in the CON register.	-
16	RUN2_CLR	Writing a one clears the corresponding bit in the CON register.	-
17	CENTER2_CLR	Writing a one clears the corresponding bit in the CON register.	-
18	POLA2_CLR	Writing a one clears the corresponding bit in the CON register.	-
19	DTE2_CLR	Writing a one clears the corresponding bit in the CON register.	-

Table 576. MCPWM Control clear address (CON_CLR - 0x400B 8008) bit description

Bit	Symbol	Description	Reset value
20	DISUP2_CLR	Writing a one clears the corresponding bit in the CON register.	-
28:21	-	Writing a one clears the corresponding bit in the CON register.	-
29	INVBDC_CLR	Writing a one clears the corresponding bit in the CON register.	-
30	ACMOD_CLR	Writing a one clears the corresponding bit in the CON register.	-
31	DCMODE_CLR	Writing a one clears the corresponding bit in the CON register.	

27.8.2 PWM Capture Control register

27.8.2.1 MCPWM Capture Control read address

The MCCAPCON register controls detection of events on the MCI0-2 inputs for all MCPWM channels. Any of the three MCI inputs can be used to trigger a capture event on any or all of the three channels. This address is read-only, but the underlying register can be modified by writing to addresses CAPCON_SET and CAPCON_CLR.

Table 577. MCPWM Capture Control read address (CAPCON - 0x400B 800C) bit description

Bit	Symbol	Description	Reset value
0	CAP0MCI0_RE	A 1 in this bit enables a channel 0 capture event on a rising edge on MCI0.	0
1	CAP0MCI0_FE	A 1 in this bit enables a channel 0 capture event on a falling edge on MCI0.	0
2	CAP0MCI1_RE	A 1 in this bit enables a channel 0 capture event on a rising edge on MCI1.	0
3	CAP0MCI1_FE	A 1 in this bit enables a channel 0 capture event on a falling edge on MCI1.	0
4	CAP0MCI2_RE	A 1 in this bit enables a channel 0 capture event on a rising edge on MCI2.	0
5	CAP0MCI2_FE	A 1 in this bit enables a channel 0 capture event on a falling edge on MCI2.	0
6	CAP1MCI0_RE	A 1 in this bit enables a channel 1 capture event on a rising edge on MCI0.	0
7	CAP1MCI0_FE	A 1 in this bit enables a channel 1 capture event on a falling edge on MCI0.	0
8	CAP1MCI1_RE	A 1 in this bit enables a channel 1 capture event on a rising edge on MCI1.	0
9	CAP1MCI1_FE	A 1 in this bit enables a channel 1 capture event on a falling edge on MCI1.	0
10	CAP1MCI2_RE	A 1 in this bit enables a channel 1 capture event on a rising edge on MCI2.	0
11	CAP1MCI2_FE	A 1 in this bit enables a channel 1 capture event on a falling edge on MCI2.	0
12	CAP2MCI0_RE	A 1 in this bit enables a channel 2 capture event on a rising edge on MCI0.	0
13	CAP2MCI0_FE	A 1 in this bit enables a channel 2 capture event on a falling edge on MCI0.	0
14	CAP2MCI1_RE	A 1 in this bit enables a channel 2 capture event on a rising edge on MCI1.	0
15	CAP2MCI1_FE	A 1 in this bit enables a channel 2 capture event on a falling edge on MCI1.	0
16	CAP2MCI2_RE	A 1 in this bit enables a channel 2 capture event on a rising edge on MCI2.	0
17	CAP2MCI2_FE	A 1 in this bit enables a channel 2 capture event on a falling edge on MCI2.	0
18	RT0	If this bit is 1, TC0 is reset by a channel 0 capture event.	0
19	RT1	If this bit is 1, TC1 is reset by a channel 1 capture event.	0
20	RT2	If this bit is 1, TC2 is reset by a channel 2 capture event.	0
21	HNFCAP0	Hardware noise filter: if this bit is 1, channel 0 capture events are delayed as described in Section 27.9.4.	0

Table 577. MCPWM Capture Control read address (CAPCON - 0x400B 800C) bit description

Bit	Symbol	Description	Reset value
22	HNFCAP1	Hardware noise filter: if this bit is 1, channel 1 capture events are delayed as described in Section 27.9.4.	0
23	HNFCAP2	Hardware noise filter: if this bit is 1, channel 2 capture events are delayed as described in Section 27.9.4.	0
31:24	-	Reserved.	-

27.8.2.2 MCPWM Capture Control set address

Writing ones to this write-only address sets the corresponding bits in CAPCON.

Table 578. MCPWM Capture Control set address (CAPCON_SET - 0x400B 8010) bit description

Bit	Symbol	Description	Reset value
0	CAP0MCI0_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
1	CAP0MCI0_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
2	CAP0MCI1_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
3	CAP0MCI1_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
4	CAP0MCI2_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
5	CAP0MCI2_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
6	CAP1MCI0_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
7	CAP1MCI0_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
8	CAP1MCI1_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
9	CAP1MCI1_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
10	CAP1MCI2_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
11	CAP1MCI2_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
12	CAP2MCI0_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
13	CAP2MCI0_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
14	CAP2MCI1_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
15	CAP2MCI1_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
16	CAP2MCI2_RE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
17	CAP2MCI2_FE_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
18	RT0_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
19	RT1_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
20	RT2_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
21	HNFCAP0_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
22	HNFCAP1_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
23	HNFCAP2_SET	Writing a one sets the corresponding bits in the CAPCON register.	-
31:24	-	Reserved.	-

27.8.2.3 MCPWM Capture control clear address

Writing ones to this write-only address clears the corresponding bits in MCCAPCON.

Table 579. MCPWM Capture control clear register (CAPCON_CLR - address 0x400B 8014) bit description

Bit	Symbol	Description	Reset value
0	CAP0MCI0_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
1	CAP0MCI0_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
2	CAP0MCI1_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
3	CAP0MCI1_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
4	CAP0MCI2_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
5	CAP0MCI2_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
6	CAP1MCI0_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
7	CAP1MCI0_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
8	CAP1MCI1_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
9	CAP1MCI1_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
10	CAP1MCI2_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
11	CAP1MCI2_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
12	CAP2MCI0_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
13	CAP2MCI0_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
14	CAP2MCI1_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
15	CAP2MCI1_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
16	CAP2MCI2_RE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
17	CAP2MCI2_FE_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
18	RT0_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
19	RT1_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
20	RT2_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
21	HNFCAP0_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
22	HNFCAP1_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
23	HNFCAP2_CLR	Writing a one clears the corresponding bits in the CAPCON register.	-
31:24	-	Reserved.	-

27.8.3 MCPWM Timer/Counter 0-2 registers

These registers hold the current values of the 32-bit counter/timers for channels 0-2. Each value is incremented on every PCLK, or by edges on the MCI0-2 pins, as selected by CNTCON. The timer/counter counts up from 0 until it reaches the value in its corresponding PER register (or is stopped by writing to CON_CLR).

A TC register can be read at any time. In order to write to the TC register, its channel must be stopped. If not, the write will not take place, no exception is generated.

Table 580. MCPWM Timer/Counter 0 to 2 registers (TC[0:2] - 0x400B 8018 (TC0), 0x400B 801C (TC1), 0x400B 8020) (TC2)bit description

Bit	Symbol	Description	Reset value
31:0	MCTC	Timer/Counter value.	0

27.8.4 MCPWM Limit 0-2 registers

These registers hold the limiting values for timer/counters 0-2. When a timer/counter reaches its corresponding limiting value: 1) in edge-aligned mode, it is reset and starts over at 0; 2) in center-aligned mode, it begins counting down until it reaches 0, at which time it begins counting up again.

If the channel's CENTER bit in CON is 0 selecting edge-aligned mode, the match between TC and LIM switches the channel's A output from "active" to "passive" state. If the channel's CENTER and DTE bits in CON are both 0, the match simultaneously switches the channel's B output from "passive" to "active" state.

If the channel's CENTER bit is 0 but the DTE bit is 1, the match triggers the channel's deadtime counter to begin counting -- when the deadtime counter expires, the channel's B output switches from "passive" to "active" state.

In center-aligned mode, matches between a channel's TC and LIM registers have no effect on its A and B outputs.

Writing to either a Limit or a Match (27.8.5) register loads a "write" register, and if the channel is stopped it also loads an "operating" register that is compared to the TC. If the channel is running and its "disable update" bit in CON is 0, the operating registers are loaded from the write registers: 1) in edge-aligned mode, when the TC matches the operating Limit register; 2) in center-aligned mode, when the TC counts back down to 0. If the channel is running and the "disable update" bit is 1, the operating registers are not loaded from the write registers until software stops the channel.

Reading an LIM address always returns the operating value.

Table 581. MCPWM Limit 0 to 2 registers (LIM[0:2] - 0x400B 8024 (LIM0), 0x400B 8028 (LIM1), 0x400B 802C (LIM2)) bit description

Bit	Symbol	Description	Reset value
31:0	MCLIM	Limit value.	0xFFFF FFFF

Remark: In timer mode, the period of a channel's modulated MCO outputs is determined by its Limit register, and the pulse width at the start of the period is determined by its Match register. If it suits your way of thinking, consider the Limit register to be the "Period register" and the Match register to be the "Pulse Width register".

27.8.5 MCPWM Match 0-2 registers

These registers also have "write" and "operating" versions as described above for the Limit registers, and the operating registers are also compared to the channels' TCs. See 27.8.4 above for details of reading and writing both Limit and Match registers.

The Match and Limit registers control the MCO0-2 outputs. If a Match register is to have any effect on its channel's operation, it must contain a smaller value than the corresponding Limit register.

Table 582. MCPWM Match 0 to 2 registers (MAT[0:2] - addresses 0x400B 8030 (MAT0), 0x400B 8034 (MAT1), 0x400B 8038 (MAT2)) bit description

Bit	Symbol	Description	Reset value
31:0	MCMAT	Match value.	0xFFFF FFFF

27.8.5.1 Match register in Edge-Aligned mode

If the channel's CENTER bit in CON is 0 selecting edge-aligned mode, a match between TC and MAT switches the channel's B output from "active" to "passive" state. If the channel's CENTER and DTE bits in CON are both 0, the match simultaneously switches the channel's A output from "passive" to "active" state.

If the channel's CENTER bit is 0 but the DTE bit is 1, the match triggers the channel's deadtime counter to begin counting -- when the deadtime counter expires, the channel's A output switches from "passive" to "active" state.

27.8.5.2 Match register in Center-Aligned mode

If the channel's CENTER bit in CON is 1 selecting center-aligned mode, a match between TC and MAT while the TC is incrementing switches the channel's B output from "active" to "passive" state, and a match while the TC is decrementing switches the A output from "active" to "passive". If the channel's CENTER bit in CON is 1 but the DTE bit is 0, a match simultaneously switches the channel's other output in the opposite direction.

If the channel's CENTER and DTE bits are both 1, a match between TC and MAT triggers the channel's deadtime counter to begin counting -- when the deadtime counter expires, the channel's B output switches from "passive" to "active" if the TC was counting up at the time of the match, and the channel's A output switches from "passive" to "active" if the TC was counting down at the time of the match.

27.8.5.3 0 and 100% duty cycle

To lock a channel's MCO outputs at the state "B active, A passive", write its Match register with a higher value than you write to its Limit register. The match never occurs.

To lock a channel's MCO outputs at the opposite state, "A active, B passive", simply write 0 to its Match register.

27.8.6 MCPWM Dead-time register

This register holds the dead-time values for the three channels. If a channel's DTE bit in CON is 1 to enable its dead-time counter, the counter counts down from this value whenever one its channel's outputs changes from "active" to "passive" state. When the dead-time counter reaches 0, the channel changes its other output from "passive" to "active" state.

The motivation for the dead-time feature is that power transistors, like those driven by the A and B outputs in a motor-control application, take longer to fully turn off than they take to start to turn on. If the A and B transistors are ever turned on at the same time, a wasteful and damaging current will flow between the power rails through the transistors. In such applications, the dead-time register should be programmed with the number of PCLK periods that is greater than or equal to the transistors' maximum turn-off time minus their minimum turn-on time.

Table 583. MCPWM Dead-time register (DT - address 0x400B 803C) bit description

Bit	Symbol	Description	Reset value
9:0	DT0	Dead time for channel 0.[1]	0x3FF
19:10	DT1	Dead time for channel 1.[2]	0x3FF
29:20	DT2	Dead time for channel 2.[2]	0x3FF
31:30	-	reserved	

[1] If ACMODE is 1 selecting AC-mode, this field controls the dead time for all three channels.

[2] If ACMODE is 0.

27.8.7 MCPWM Communication Pattern register

This register is used in DC mode only. The internal MCOA0 signal is routed to any or all of the six output pins under the control of the bits in this register. Like the Match and Limit registers, this register has “write” and “operational” versions. See 27.8.4 and 27.9.2 for more about this subject.

Table 584. MCPWM Communication Pattern register (CP - address 0x400B 8040) bit description

Bit	Symbol	Value	Description	Reset value
0	CCPA0		Communication pattern output A, channel 0.	0
		0	MCOA0 passive.	
		1	internal MCOA0.	
1	CCPB0		Communication pattern output B, channel 0.	0
		0	MCOB0 passive.	
		1	MCOB0 tracks internal MCOA0.	
2	CCPA1		Communication pattern output A, channel 1.	0
		0	MCOA1 passive.	
		1	MCOA1 tracks internal MCOA0.	
3	CCPB1		Communication pattern output B, channel 1.	0
		0	MCOB1 passive.	
		1	MCOB1 tracks internal MCOA0.	
4	CCPA2		Communication pattern output A, channel 2.	0
		0	MCOA2 passive.	
		1	MCOA2 tracks internal MCOA0.	
5	CCPB2		Communication pattern output B, channel 2.	0
		0	MCOB2 passive.	
		1	MCOB2 tracks internal MCOA0.	
31:6	-		Reserved.	

27.8.8 MCPWM Capture read addresses

The CAPCON register (Table 577) allows software to select any edges on any of the MCI0-2 inputs as a capture event for each channel. When a channel’s capture event occurs, the current TC value for that channel is stored in its read-only Capture register. These addresses are read-only, but the underlying registers can be cleared by writing to the CAP_CLR address

Table 585. MCPWM Capture read addresses (CAP[0:2] - 0x400B 8044 (CAP0), 0x400B 8048 (CAP1), 0x400B 804C (CAP2)) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Current TC value at a capture event.	0x0000 0000

27.8.9 MCPWM Interrupt registers

The Motor Control PWM module includes the following interrupt sources:

Table 586. Motor Control PWM interrupts

Symbol	Description
ILIM0/1/2	Limit interrupts for channels 0, 1, 2.
IMAT0/1/2	Match interrupts for channels 0, 1, 2.
ICAP0/1/2	Capture interrupts for channels 0, 1, 2.
ABORT	Fast abort interrupt

8.9.1 MCPWM Interrupt Enable read address

The INTEN register controls which of the MCPWM interrupts are enabled. This address is read-only, but the underlying register can be modified by writing to addresses INTEN_SET and INTEN_CLR.

Table 587. MCPWM Interrupt Enable read address (INTEN - 0x400B 8050) bit description

Bit	Symbol	Value	Description	Reset value
0	ILIM0		Limit interrupt for channel 0.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
1	IMAT0		Match interrupt for channel 0.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
2	ICAP0		Capture interrupt for channel 0.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
3	-		Reserved.	-
4	ILIM1		Limit interrupt for channel 1.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
5	IMAT1		Match interrupt for channel 1.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
6	ICAP1		Capture interrupt for channel 1.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
7	-		Reserved.	-

Table 587. MCPWM Interrupt Enable read address (INTEN - 0x400B 8050) bit description

Bit	Symbol	Value	Description	Reset value
8	ILIM2		Limit interrupt for channel 2.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
9	IMAT2		Match interrupt for channel 2.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
10	ICAP2		Capture interrupt for channel 2.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
14:11	-		Reserved.	-
15	ABORT		Fast abort interrupt.	0
		0	Interrupt disabled.	
		1	Interrupt enabled.	
31:16	-		Reserved.	-

27.8.9.2 MCPWM Interrupt Enable set address

Writing ones to this write-only address sets the corresponding bits in INTEN, thus enabling interrupts.

Table 588. MCPWM interrupt enable set register (INTEN_SET - address 0x400B 8054) bit description

Bit	Symbol	Description	Reset value
0	ILIM0_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
1	IMAT0_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
2	ICAP0_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
3	-	Reserved.	-
4	ILIM1_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
5	IMAT1_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
6	ICAP1_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
7	-	Reserved.	-
9	ILIM2_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
10	IMAT2_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
11	ICAP2_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
14:12	-	Reserved.	-
15	ABORT_SET	Writing a one sets the corresponding bit in INTEN, thus enabling the interrupt.	-
31:16	-	Reserved.	-

27.8.9.3 MCPWM Interrupt Enable clear address

Writing ones to this write-only address clears the corresponding bits in INTEN, thus disabling interrupts.

Table 589. PWM interrupt enable clear register (INTEN_CLR - address 0x400B 8058) bit description

Bit	Symbol	Description	Reset value
0	ILIM0_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
1	IMAT0_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
2	ICAP0_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
3	-	Reserved.	-
4	ILIM1_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
5	IMAT1_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
6	ICAP1_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
7	-	Reserved.	-
8	ILIM2_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
9	IMAT2_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
10	ICAP2_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
14:11	-	Reserved.	-
15	ABORT_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
31:16	-	Reserved.	-

27.8.10 MCPWM Count Control register

27.8.10.1 MCPWM Count Control read address

The CNTCON register controls whether the MCPWM channels are in timer or counter mode, and in counter mode whether the counter advances on rising and/or falling edges on any or all of the three MCI inputs. If timer mode is selected, the counter advances based on the PCLK clock.

This address is read-only. To set or clear the register bits, write ones to the CNTCON_SET or CNTCON_CLR address.

Table 590. MCPWM Count Control read address (CNTCON - 0x400B 805C) bit description

Bit	Symbol	Value	Description	Reset value
0	TC0MCI0_RE		Counter 0 rising edge mode, channel 0.	0
		0	A rising edge on MCI0 does not affect counter 0.	
		1	If MODE0 is 1, counter 0 advances on a rising edge on MCI0.	
1	TC0MCI0_FE		Counter 0 falling edge mode, channel 0.	0
		0	A falling edge on MCI0 does not affect counter 0.	
		1	If MODE0 is 1, counter 0 advances on a falling edge on MCI0.	
2	TC0MCI1_RE		Counter 0 rising edge mode, channel 1.	0
		0	A rising edge on MCI1 does not affect counter 0.	
		1	If MODE0 is 1, counter 0 advances on a rising edge on MCI1.	
3	TC0MCI1_FE		Counter 0 falling edge mode, channel 1.	0
		0	A falling edge on MCI1 does not affect counter 0.	
		1	If MODE0 is 1, counter 0 advances on a falling edge on MCI1.	

Table 590. MCPWM Count Control read address (CNTCON - 0x400B 805C) bit description

Bit	Symbol	Value	Description	Reset value
4	TC0MCI2_RE		Counter 0 rising edge mode, channel 2.	0
		0	A rising edge on MCI0 does not affect counter 0.	
		1	If MODE0 is 1, counter 0 advances on a rising edge on MCI2.	
5	TC0MCI2_FE		Counter 0 falling edge mode, channel 2.	0
		0	A falling edge on MCI0 does not affect counter 0.	
		1	If MODE0 is 1, counter 0 advances on a falling edge on MCI2.	
6	TC1MCI0_RE		Counter 1 rising edge mode, channel 0.	0
		0	A rising edge on MCI0 does not affect counter 1.	
		1	If MODE1 is 1, counter 1 advances on a rising edge on MCI0.	
7	TC1MCI0_FE		Counter 1 falling edge mode, channel 0.	0
		0	A falling edge on MCI0 does not affect counter 1.	
		1	If MODE1 is 1, counter 1 advances on a falling edge on MCI0.	
8	TC1MCI1_RE		Counter 1 rising edge mode, channel 1.	0
		0	A rising edge on MCI1 does not affect counter 1.	
		1	If MODE1 is 1, counter 1 advances on a rising edge on MCI1.	
9	TC1MCI1_FE		Counter 1 falling edge mode, channel 1.	0
		0	A falling edge on MCI0 does not affect counter 1.	
		1	If MODE1 is 1, counter 1 advances on a falling edge on MCI1.	
10	TC1MCI2_RE		Counter 1 rising edge mode, channel 2.	0
		0	A rising edge on MCI2 does not affect counter 1.	
		1	If MODE1 is 1, counter 1 advances on a rising edge on MCI2.	
11	TC1MCI2_FE		Counter 1 falling edge mode, channel 2.	0
		0	A falling edge on MCI2 does not affect counter 1.	
		1	If MODE1 is 1, counter 1 advances on a falling edge on MCI2.	
12	TC2MCI0_RE		Counter 2 rising edge mode, channel 0.	0
		0	A rising edge on MCI0 does not affect counter 2.	
		1	If MODE2 is 1, counter 2 advances on a rising edge on MCI0.	
13	TC2MCI0_FE		Counter 2 falling edge mode, channel 0.	0
		0	A falling edge on MCI0 does not affect counter 2.	
		1	If MODE2 is 1, counter 2 advances on a falling edge on MCI0.	
14	TC2MCI1_RE		Counter 2 rising edge mode, channel 1.	0
		0	A rising edge on MCI1 does not affect counter 2.	
		1	If MODE2 is 1, counter 2 advances on a rising edge on MCI1.	
15	TC2MCI1_FE		Counter 2 falling edge mode, channel 1.	0
		0	A falling edge on MCI1 does not affect counter 2.	
		1	If MODE2 is 1, counter 2 advances on a falling edge on MCI1.	
16	TC2MCI2_RE		Counter 2 rising edge mode, channel 2.	0
		0	A rising edge on MCI2 does not affect counter 2.	
		1	If MODE2 is 1, counter 2 advances on a rising edge on MCI2.	

Table 590. MCPWM Count Control read address (CNTCON - 0x400B 805C) bit description

Bit	Symbol	Value	Description	Reset value
17	TC2MCI2_FE		Counter 2 falling edge mode, channel 2.	0
		0	A falling edge on MCI2 does not affect counter 2.	
		1	If MODE2 is 1, counter 2 advances on a falling edge on MCI2.	
28:18	-	-	Reserved.	-
29	CNTR0		Channel 0 counter/timer mode.	0
		0	Channel 0 is in timer mode.	
		1	Channel 0 is in counter mode.	
30	CNTR1		Channel 1 counter/timer mode.	0
		0	Channel 1 is in timer mode.	
		1	Channel 1 is in counter mode.	
31	CNTR2		Channel 2 counter/timer mode.	0
		0	Channel 2 is in timer mode.	
		1	Channel 2 is in counter mode.	

27.8.10.2 MCPWM Count Control set address

Writing ones to this write-only address sets the corresponding bits in CNTCON.

Table 591. MCPWM Count Control set address (CNTCON_SET - 0x400B 8060) bit description

Bit	Symbol	Description	Reset value
0	TC0MCI0_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
1	TC0MCI0_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
2	TC0MCI1_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
3	TC0MCI1_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
4	TC0MCI2_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
5	TC0MCI2_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
6	TC1MCI0_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
7	TC1MCI0_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
8	TC1MCI1_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
9	TC1MCI1_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
10	TC1MCI2_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
11	TC1MCI2_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
12	TC2MCI0_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
13	TC2MCI0_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
14	TC2MCI1_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
15	TC2MCI1_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
16	TC2MCI2_RE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
17	TC2MCI2_FE_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
28:18	-	Reserved.	

Table 591. MCPWM Count Control set address (CNTCON_SET - 0x400B 8060) bit description

Bit	Symbol	Description	Reset value
29	CNTR0_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
30	CNTR1_SET	Writing a one sets the corresponding bit in the CNTCON register.	-
31	CNTR2_SET	Writing a one sets the corresponding bit in the CNTCON register.	-

27.8.10.3 MCPWM Count Control clear address

Writing ones to this write-only address clears the corresponding bits in CNTCON.

Table 592. MCPWM Count Control clear address (CNTCON_CLR - 0x400B 8064) bit description

Bit	Symbol	Description	Reset value
0	TC0MCI0_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
1	TC0MCI0_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
2	TC0MCI1_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
3	TC0MCI1_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
4	TC0MCI2_RE	Writing a one clears the corresponding bit in the CNTCON register.	-
5	TC0MCI2_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
6	TC1MCI0_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
7	TC1MCI0_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
8	TC1MCI1_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
9	TC1MCI1_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
10	TC1MCI2_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
11	TC1MCI2_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
12	TC2MCI0_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
13	TC2MCI0_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
14	TC2MCI1_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
15	TC2MCI1_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
16	TC2MCI2_RE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
17	TC2MCI2_FE_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
28:18	-	Reserved.	
29	CNTR0_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
30	CNTR1_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-
31	CNTR2_CLR	Writing a one clears the corresponding bit in the CNTCON register.	-

27.8.11 MCPWM Interrupt flag registers

27.8.11.1 MCPWM Interrupt Flags read address

The INTF register includes all MCPWM interrupt flags, which are set when the corresponding hardware event occurs, or when ones are written to the INTF_SET address. When corresponding bits in this register and INTEN are both 1, the MCPWM asserts its interrupt request to the Interrupt Controller module. This address is read-only, but the bits in the underlying register can be modified by writing ones to addresses INTF_SET and INTF_CLR.

Table 593. MCPWM Interrupt flags read address (INTF - 0x400B 8068) bit description

Bit	Symbol	Value	Description	Reset value
0	ILIM0_F		Limit interrupt flag for channel 0.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
1	IMAT0_F		Match interrupt flag for channel 0.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
2	ICAP0_F		Capture interrupt flag for channel 0.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
3	-		Reserved.	-
4	ILIM1_F		Limit interrupt flag for channel 1.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
5	IMAT1_F		Match interrupt flag for channel 1.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
6	ICAP1_F		Capture interrupt flag for channel 1.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
7	-		Reserved.	-
8	ILIM2_F		Limit interrupt flag for channel 2.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
9	IMAT2_F		Match interrupt flag for channel 2.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
10	ICAP2_F		Capture interrupt flag for channel 2.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
14:11	-		Reserved.	-

Table 593. MCPWM Interrupt flags read address (INTF - 0x400B 8068) bit description

Bit	Symbol	Value	Description	Reset value
15	ABORT_F		Fast abort interrupt flag.	0
		0	This interrupt source is not contributing to the MCPWM interrupt request.	
		1	If the corresponding bit in INTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
31:16	-		Reserved.	-

27.8.11.2 MCPWM Interrupt Flags set address

Writing ones to this write-only address sets the corresponding bits in INTF, thus possibly simulating hardware interrupts.

Table 594. MCPWM Interrupt Flags set address (INTF_SET - 0x400B 806C) bit description

Bit	Symbol	Description	Reset value
0	ILIM0_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
1	IMAT0_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
2	ICAP0_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
3	-	Reserved.	-
4	ILIM1_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
5	IMAT1_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
6	ICAP1_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
7	-	Reserved.	-
8	ILIM2_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
9	IMAT2_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
10	ICAP2_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
14:11	-	Reserved.	-
15	ABORT_F_SET	Writing a one sets the corresponding bit in the INTF register, thus possibly simulating hardware interrupt.	-
31:16	-	Reserved.	-

27.8.11.3 MCPWM Interrupt Flags clear address

Writing ones to this write-only address sets the corresponding bits in INTF, thus clearing the corresponding interrupt requests. This is typically done in interrupt service routines.

Table 595. MCPWM Interrupt Flags clear address (INTF_CLR - 0x400B 8070) bit description

Bit	Symbol	Description	Reset value
0	ILIM0_F_CLR	Writing a one clears the corresponding bit in the INTF register, thus clearing the corresponding interrupt request.	-
1	IMAT0_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
2	ICAP0_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
3	-	Reserved.	-
4	ILIM1_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
5	IMAT1_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
6	ICAP1_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
7	-	Reserved.	-
8	ILIM2_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
9	IMAT2_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
10	ICAP2_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
14:11	-	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
15	ABORT_F_CLR	Writing a one clears the corresponding bit in INTEN, thus disabling the interrupt.	-
31:16	-	Reserved.	-

27.8.12 MCPWM Capture clear address

Writing ones to this write-only address clears the selected CAP registers.

Table 596. MCPWM Capture clear address (CAP_CLR - 0x400B 8074) bit description

Bit	Symbol	Description
0	CAP_CLR0	Writing a 1 to this bit clears the CAP0 register.
1	CAP_CLR1	Writing a 1 to this bit clears the CAP1 register.
2	CAP_CLR2	Writing a 1 to this bit clears the CAP2 register.
31:3	-	Reserved

27.9 PWM operation

27.9.1 Pulse-width modulation

Each channel of the MCPWM has two outputs, A and B, that can drive a pair of transistors to switch a controlled point between two power rails. Most of the time the two outputs have opposite polarity, but a dead-time feature can be enabled (on a per-channel basis) to delay both signals' transitions from "passive" to "active" state so that the transistors are never both turned on simultaneously. In a more general view, the states of each output pair can be thought of "high", "low", and "floating" or "up", "down", and "center-off".

Each channel's mapping from "active" and "passive" to "high" and "low" is programmable. After Reset, the three A outputs are passive/low, and the B outputs are active/high.

The MCPWM can perform edge-aligned and center-aligned pulse-width modulation.

Note: *In timer mode, the period of a channel's modulated MC_A and MC_B outputs is determined by its Limit register, and the pulse width at the start of the period is determined by its Match register. If it suits your way of thinking, consider the Limit register to be the "Period register" and the Match register to be the "Pulse Width register".*

Edge-aligned PWM without dead-time

In this mode the timer TC counts up from 0 to the value in the LIM register. As shown in Figure 148, the output pin state is "A passive" until the TC matches the Match register, at which point it changes to "A active". When the TC matches the Limit register, the output pin state changes back to "A passive", and the TC is reset and starts counting up again.

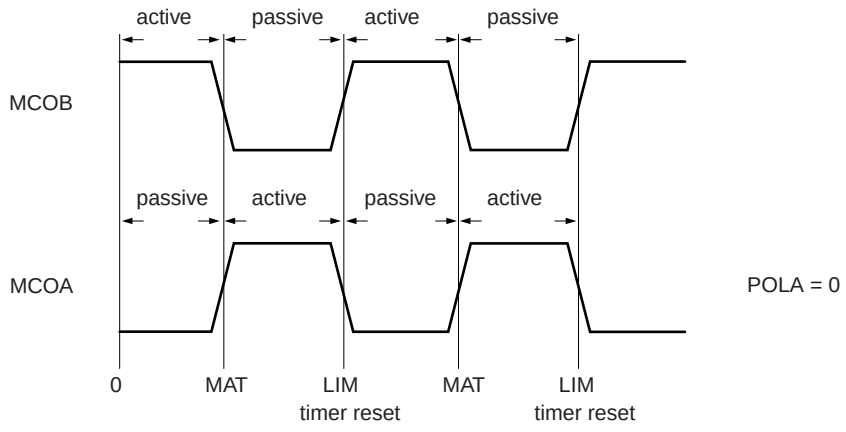


Fig 148. Edge-aligned PWM waveform without dead time, POLA = 0

Center-aligned PWM without dead-time

In this mode the timer TC counts up from 0 to the value in the LIM register, then counts back down to 0 and repeats. As shown in Figure 149, while the timer counts up, the output pin state is "A passive" until the TC matches the Match register, at which point it changes to "A active". When the TC matches the Limit register it starts counting down. When the TC matches the Match register on the way down, the output pin state changes back to "A passive".

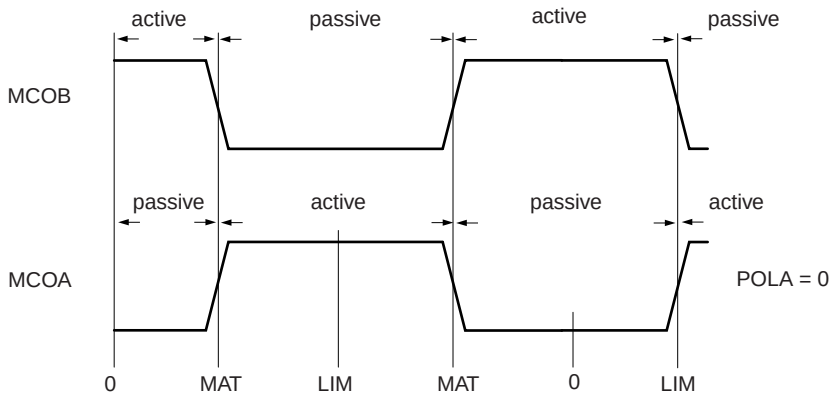


Fig 149. Center-aligned PWM waveform without dead time, POLA = 0

Dead-time counter

When the a channel's DTE bit is set in MCCON, the dead-time counter delays the passive-to-active transitions of both output pins. The dead-time counter starts counting down, from the channel's DT value (in the MCDT register) to 0, whenever the channel's A or B output changes from active to passive. The transition of the other output from passive to active is delayed until the dead-time counter reaches 0. During the dead time, the MC_A and MC_B output levels are both passive. Figure 150 shows operation in edge aligned mode with dead time, and Figure 151 shows center-aligned operation with dead time.

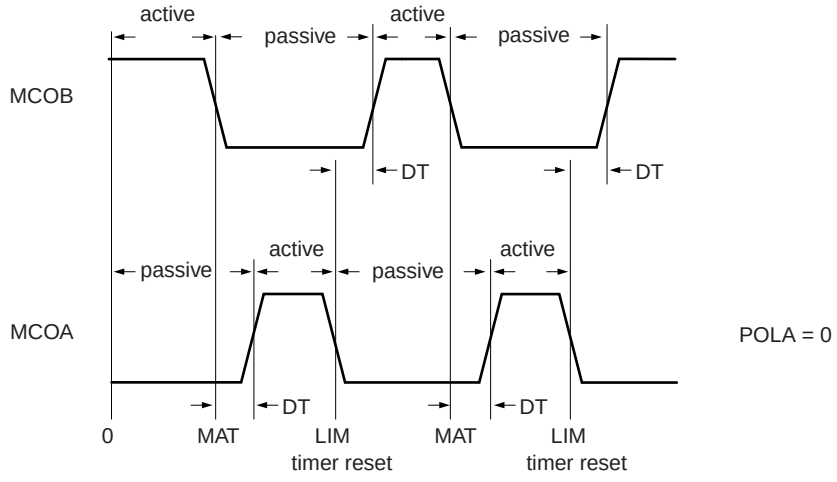


Fig 150. Edge-aligned PWM waveform with dead time, POLA = 0

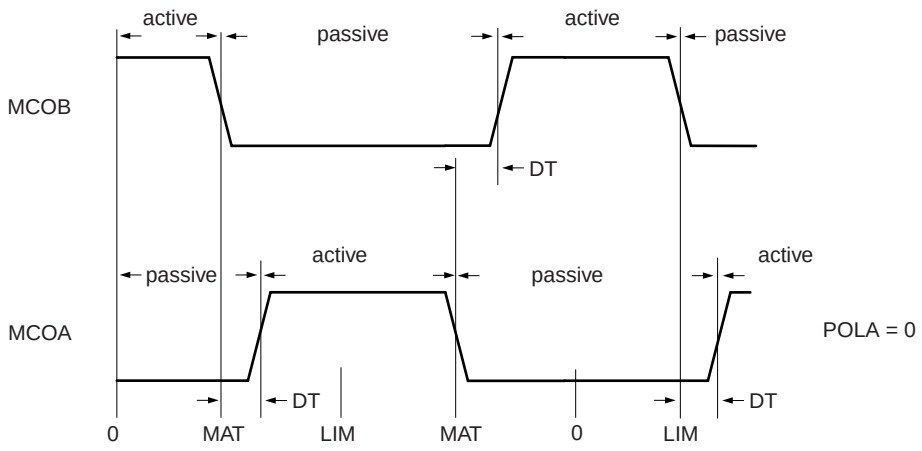


Fig 151. Center-aligned waveform with dead time, POLA = 0

27.9.2 Shadow registers and simultaneous updates

The Limit, Match, and Commutation Pattern registers (MCLIM, MCMAT, and MCCP) are implemented as register pairs, each consisting of a write register and an operational register. Software writes into the write registers. The operational registers control the actual operation of each channel and are loaded with the current value in the write registers when the TC starts counting up from 0.

Updating of the functional registers can be disabled by setting a channel's DISUP bit in the MCCON register. If the DISUP bits are set, the functional registers are not updated until software stops the channel.

If a channel is not running when software writes to its LIM or MAT register, the functional register is updated immediately.

Software can write to a TC register only when its channel is stopped.

27.9.3 Fast Abort (ABORT)

The MCPWM has an external input $\overline{\text{MC\_ABORT}}$. When this input goes low, all six output pins assume their "A passive" states, and the Abort interrupt is generated if enabled. The outputs remain locked in "A passive" state until the ABORT interrupt flag is cleared or the Abort interrupt is disabled. The ABORT flag may not be cleared before the $\overline{\text{MC\_ABORT}}$ input goes high.

In order to clear an ABORT flag, a 1 must be written to bit 15 of the MCINTF_CLR register. This will remove the interrupt request. The interrupt can also be disabled by writing a 1 to bit 15 of the MCINTEN_CLR register.

27.9.4 Capture events

Each PWM channel can take a snapshot of its TC when an input signal transitions. Any channel may use any combination of rising and/or falling edges on any or all of the MC_FB0-2 inputs as a capture event, under control of the MCCAPCON register. Rising or falling edges on the inputs are detected synchronously with respect to PCLK.

If a channel's HNF bit in the MCCAPCON register is set to enable "noise filtering", a selected edge on an MC_FB pin starts the dead-time counter for that channel, and the capture event actions described below are delayed until the dead-time counter reaches 0. This function is targeted specifically for performing three-phase brushless DC motor control with Hall sensors.

A capture event on a channel (possibly delayed by HNF) causes the following:

- The current value of the TC is stored in the Capture register (CAP).
- If the channel's capture event interrupt is enabled (see Table 587), the capture event interrupt flag is set.
- If the channel's RT bit is set in the MCCAPCON register, enabling reset on a capture event, the input event has the same effect as matching the channel's TC to its LIM register. This includes resetting the TC and switching the output pins in edge-aligned mode as described in 27.8.4 and 27.9.1.

27.9.5 External event counting (Counter mode)

If a channel's MODE bit is 1 in MCCNTCON, its TC is incremented by rising and/or falling edges (synchronously detected) on the MC_FB0-2 inputs, rather than by PCLK. The PWM functions and capture functions are unaffected.

27.9.6 Three-phase DC mode

The three-phase DC mode is selected by setting the DCMODE bit in the MCCON register.

In this mode, the internal MC_0A signal can be routed to any or all of the output pins. Each output pin is masked by a bit in the current Commutation Pattern register MCCP. If a bit in the MCCP register is 0, its output pin has the logic level for the passive state of output MC_0A. The polarity of the off state is determined by the POLA0 bit.

All output pins that have 1 bits in the MCCP register are controlled by the internal MC_0A signal.

The three MC_B output pins are inverted when the INVBDC bit is 1 in the MCCON register. This feature accommodates bridge-drivers that have active-low inputs for the low-side switches.

The MCCP register is implemented as a shadow register pair, so that changes to the active commutation pattern occur at the beginning of a new PWM cycle. See 27.8.4 and 27.9.2 for more about writing and reading such registers.

Figure 152 shows sample waveforms of the output pins in three-phase DC mode. Bits 1 and 3 in the MCCP register (corresponding to outputs MC_1B and MC_0B) are set to 0 so that these outputs are masked and in the off state. Their logic level is determined by the POLA0 bit (here, POLA0 = 0 so the passive state is logic LOW). The INVBDC bit is set to 0 (logic level not inverted) so that the B output have the same polarity as the A outputs. Note that this mode differs from other modes in that the MC_B outputs are **not** the opposite of the MC_A outputs.

In the situation shown in Figure 152, bits 0, 2, 4, and 5 in the MCCP register are set to 1. That means that MC_1A and both output pins for channel 2 follow the MC_0A signal.

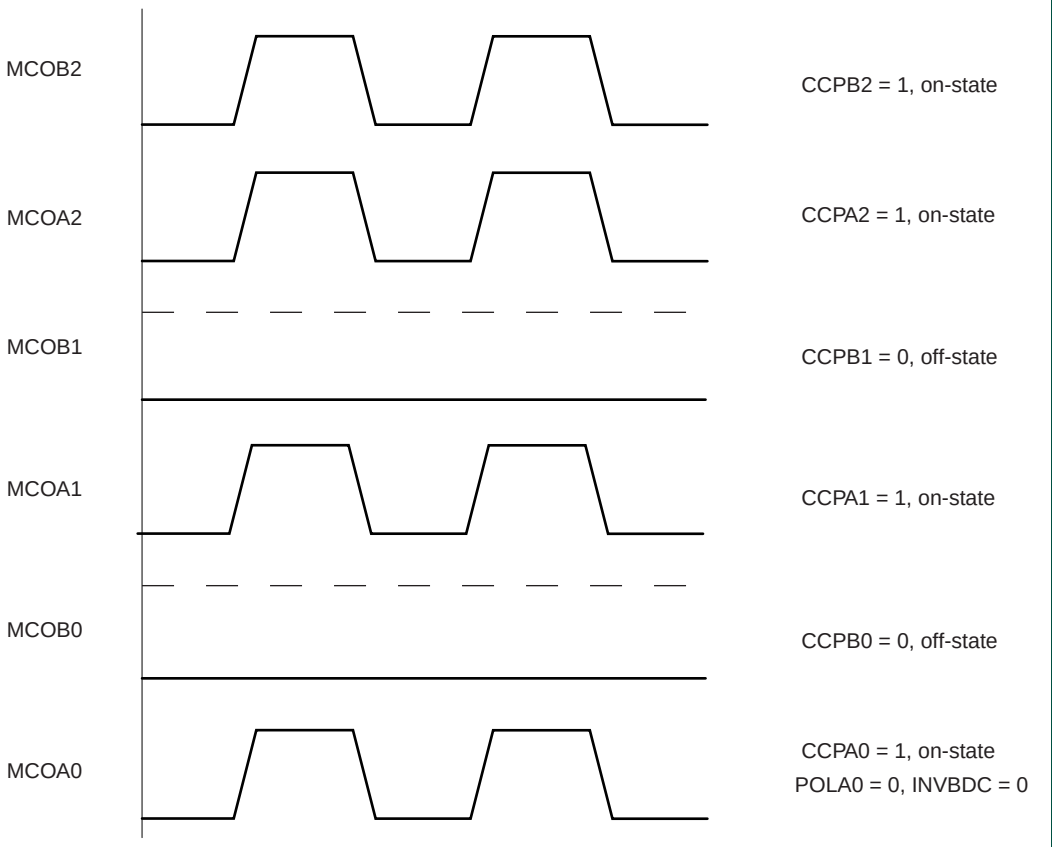


Fig 152. Three-phase DC mode sample waveforms

27.9.7 Three phase AC mode

The three-phase AC-mode is selected by setting the ACMODE bit in the MCCON register.

In this mode, the value of channel 0's TC is routed to all channels for comparison with their MAT registers. (The LIM1-2 registers are not used.)

Each channel controls its output pins by comparing its MAT value to TC0.

Figure 153 shows sample waveforms for the six output pins in three-phase AC mode. The POLA bits are set to 0 for all three channels, so that for all output pins the active levels are high and the passive levels are low. Each channel has a different MAT value which is compared to the MCTC0 value. In this mode the period value is identical for all three channels and is determined by MCLIM0. The dead-time mode is disabled.

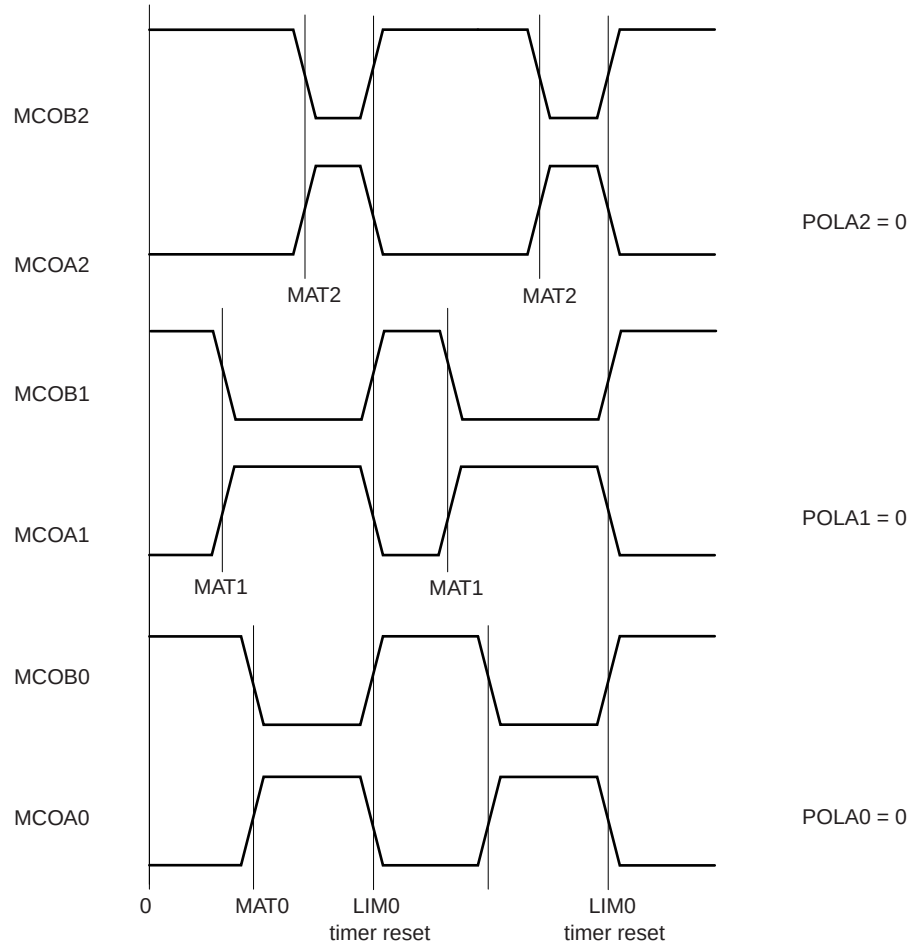


Fig 153. Three-phase AC mode sample waveforms, edge aligned PWM mode

27.9.8 Interrupts

The MCPWM includes 10 possible interrupt sources:

- When any channel's TC matches its Match register.
- When any channel's TC matches its Limit register.
- When any channel captures the value of its TC into its Capture register, because a selected edge occurs on any of MC_FB0-2.
- When all three channels' outputs are forced to "A passive" state because the MC_ABORT pin goes low.

Section 27.8.9 "MCPWM Interrupt registers" explains how to enable these interrupts, and Section 27.8.2 "PWM Capture Control register" describes how to map edges on the MC_FB0-2 inputs to "capture events" on the three channels.

28.1 How to read this chapter

The QEI is available on most LPC408x/407x devices, see Section 1.4 for details.

28.2 Basic configuration

The QEI is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCQEI.

   Remark: On reset, the QEI is disabled (PCQEI = 0).
2. Peripheral clock: The QEI operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Select QEI pins through and pin modes for port pins with QEI functions through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: See Section 28.7.4. The QEI interrupt is enabled in the NVIC using the appropriate Interrupt Set Enable register.

28.3 Features

This Quadrature Encoder Interface (QEI) has the following features:

- tracks encoder position.
- increments/ decrements depending on direction.
- programmable for 2X or 4X position counting.
- velocity capture using built-in timer.
- velocity compare function with less than interrupt.
- uses 32-bit registers for position and velocity.
- three position compare registers with interrupts.
- index counter for revolution counting.
- index compare register with interrupts.
- can combine index and position interrupts to produce an interrupt for whole and partial revolution displacement.
- digital filter with programmable delays for encoder input signals.
- can accept decoded signal inputs (clock and direction).

28.4 Introduction

A quadrature encoder, also known as a 2-channel incremental encoder, converts angular displacement into two pulse signals. By monitoring both the number of pulses and the relative phase of the two signals, position, direction of rotation, and velocity can be tracked. In addition, a third channel, or index signal, can be used to reset the position counter. This quadrature encoder interface module decodes the digital pulses from a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture the velocity of the encoder wheel.

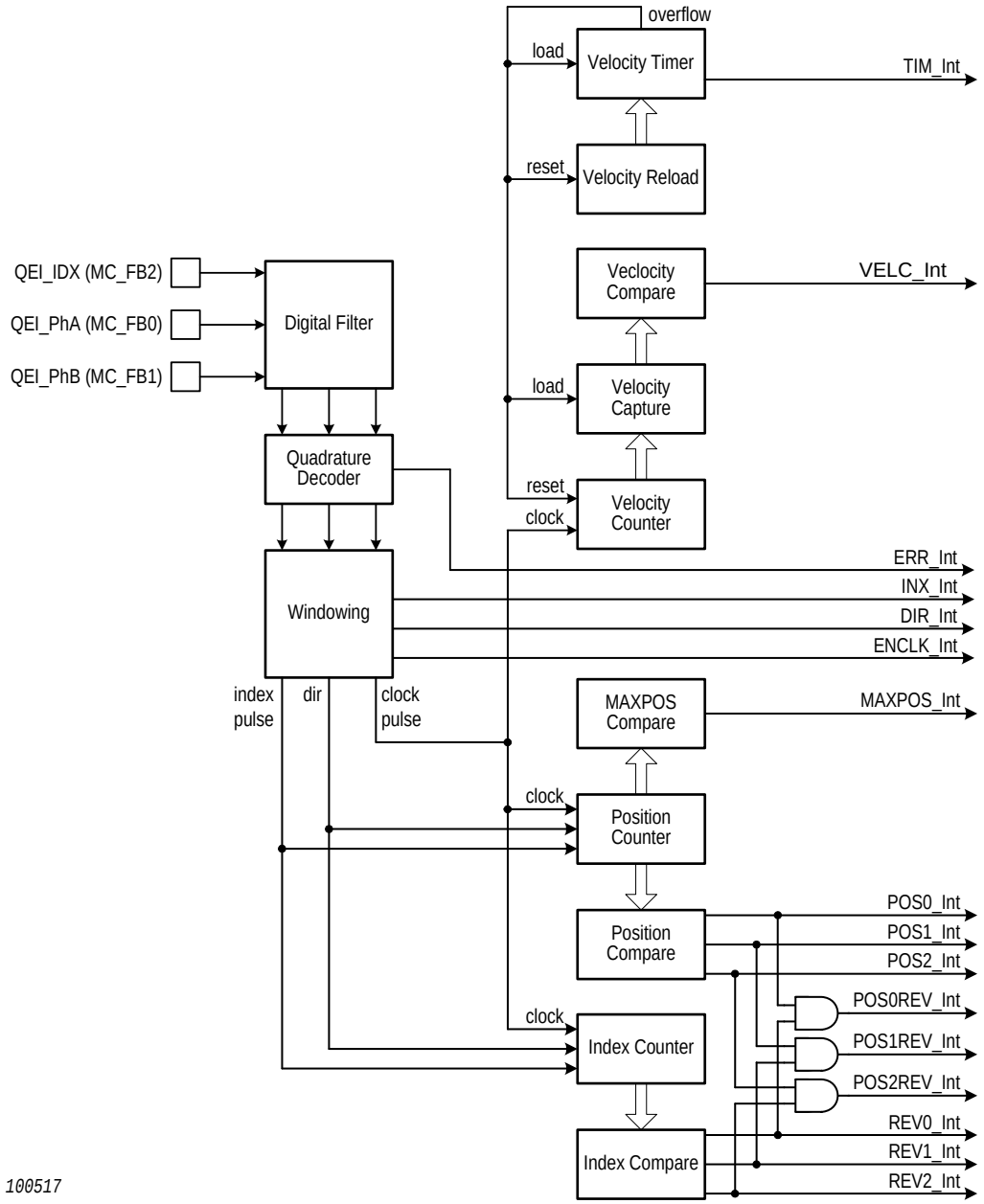


Fig 154.Encoder interface block diagram

28.5 Functional description

The QEI module interprets the two-bit gray code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture the velocity of the encoder wheel.

28.5.1 Input signals

The QEI module supports two modes of signal operation: quadrature phase mode and clock/direction mode. In quadrature phase mode, the encoder produces two clocks that are 90 degrees out of phase; the edge relationship is used to determine the direction of rotation. In clock/direction mode, the encoder produces a clock signal to indicate steps and a direction signal to indicate the direction of rotation.).

This mode is determined by the SigMode bit of the QEI Configuration register (QEICONF) register (See Table 603). When the SigMode bit = 1, the quadrature decoder is bypassed and the PhA pin functions as the direction signal and PhB pin functions as the clock signal for the counters, etc. When the SigMode bit = 0, the PhA pin and PhB pins are decoded by the quadrature decoder. In this mode the quadrature decoder produces the direction and clock signals for the counters, etc. In both modes the direction signal is subject to the effects of the direction invert (DIRINV) bit.

28.5.1.1 Quadrature input signals

When edges on PhA lead edges on PhB, the position counter is incremented. When edges on PhB lead edges on PhA, the position counter is decremented. When a rising and falling edge pair is seen on one of the phases without any edges on the other, the direction of rotation has changed.

Table 597. Encoder states

Phase A	Phase B	state
1	0	1
1	1	2
0	1	3
0	0	4

Table 598. Encoder state transitions[1]

from state	to state	Direction
1	2	positive
2	3	
3	4	
4	1	
4	3	negative
3	2	
2	1	
1	4	

[1] All other state transitions are illegal and should set the ERR bit.

Interchanging of the PhA and PhB input signals are compensated by complementing the DIR bit. When set = 1, the direction inversion bit (DIRINV) complements the DIR bit.

Table 599. Encoder direction

DIR bit	DIRINV bit	direction
0	0	forward
1	0	reverse
0	1	reverse
1	1	forward

Figure 155 shows how quadrature encoder signals equate to direction and count.

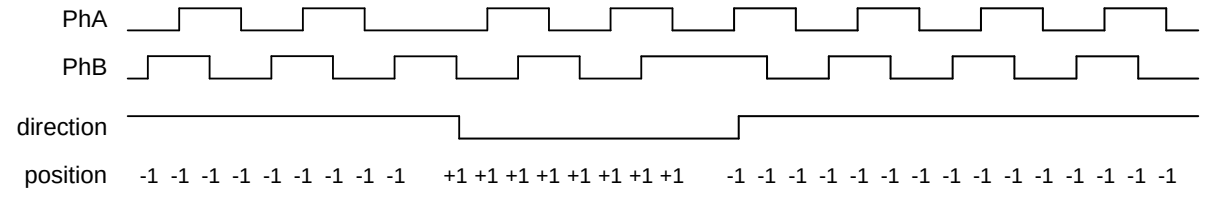


Fig 155.Quadrature Encoder Basic Operation

28.5.1.2 Digital input filtering

All three encoder inputs (PhA, PhB, and index) require digital filtering. The number of sample clocks is user programmable from 1 to 4,294,967,295 (0xFFFF FFFF). In order for a transition to be accepted, the input signal must remain in new state for the programmed number of sample clocks.

28.5.2 Position capture

The capture mode for the position integrator can be set to update the position counter on every edge of the PhA signal or to update on every edge of both PhA and PhB. Updating the position counter on every PhA and PhB provides more positional resolution at the cost of less range in the positional counter.

The position integrator and velocity capture can be independently enabled. Alternatively, the phase signals can be interpreted as a clock and direction signal as output by some encoders.

The position counter is automatically reset on one of three conditions. Incrementing past the maximum position value (QEIMAXPOS) will reset the position counter to zero. If the reset on index bit (RESPI) is set, sensing the index pulse for the first time will once reset the position counter to zero after the next positional increase (calibrate). If the continuously reset on index bit (CRESPI) is set, sensing the index pulse will continuously reset the position counter to zero after the next positional increase (recalibrate).

28.5.3 Velocity capture

The velocity capture has a programmable timer and a capture register. It counts the number of phase edges (using the same configuration as for the position integrator) in a given time period. When the velocity timer (QEITIME) overflows the contents of the velocity counter (QEIVEL) are transferred to the capture (QEICAP) register. The velocity counter is then cleared. The velocity timer is loaded with the contents of the velocity reload register (QEILOAD). Finally, the velocity interrupt (TIM_Int) is asserted. The number of edges counted in a given time period is directly proportional to the velocity of the encoder. Note that the velocity counter counts up regardless of the direction of rotation, and whether the direction changes.

Setting the reset velocity bit (RESV) will clear the velocity counter, reset the velocity capture register to 0xFFFF FFFF, load the velocity timer with the contents of the velocity reload register (QEILOAD).

The following equation converts the velocity counter value into an RPM value:

```
RPM = (PCLK * QEICAP * 60) ÷ (QEILOAD * PPR * Edges)
```

where:

- **PCLK** is the peripheral clock rate for the QEI block. See Section 3.3.3.5 for more on the possibilities for PCLK).
- **QEICAP** is the captured velocity counter value for the last velocity timer period.
- **QEILOAD** is the velocity timer reload value.
- **PPR** is the number of pulses per revolution of the physical encoder used in the application
- **Edges** is 2 or 4, based on the capture mode set in the QEICON register (2 for CapMode set to 0 and 4 for CapMode set to 1)

For example, consider a motor running at 600 RPM. A 2048 pulse per revolution quadrature encoder is attached to the motor, producing 8192 phase edges per revolution (PPR * Edges). This results in 81,920 pulses per second (the motor turns 10 times per second at 600 RPM and there are 8192 edges per revolution). If the timer were clocked at 10,000 Hz, and the QEILOAD was 2,500 (corresponding to ¼ of a second), it would count 20,480 pulses per update. Using the above equation:

RPM = (10000 * 1 * 20480 * 60) ÷ (2500 * 2048 * 4) = 600 RPM

Now, consider that the motor is sped up to 3000 RPM. This results in 409,600 pulses per second, or 102,400 every ¼ of a second. Again, the above equation gives:

RPM = (10000 * 1 * 102400 * 60) ÷ (2500 * 2048 * 4) = 3000 RPM

These are simple examples, real-world values will have a higher rate for PCLK, and probably a larger value for QEILOAD as well.

28.5.4 Velocity compare

In addition to velocity capture, the velocity measurement system includes a programmable velocity compare register. After every velocity capture event the contents of the velocity capture register (QEICAP) is compared with the contents of the velocity

compare register (VELCOMP). If the captured velocity is less than the compare value, an interrupt is asserted provided that the velocity compare interrupt enable bit is set. This can be used to determine if a motor shaft is either stalled or moving too slow.

28.6 Pin description

Table 600. QEI pin description

Pin name	I/O	Description
MC_FB0 [1]	I	Used as the Phase A (PhA) input to the Quadrature Encoder Interface.
MC_FB1 [1]	I	Used as the Phase B (PhB) input to the Quadrature Encoder Interface.
MC_FB2 [1]	I	Used as the Index (IDX) input to the Quadrature Encoder Interface.

[1] The Quadrature Encoder Interface uses the same pin functions as the Motor Control PWM feedback inputs. If used as part of motor control, the QEI can be used as an alternative to feedback directly to the MCPWM.

28.7 Register description

28.7.1 Register summary

Table 601. Register overview: QEI (base address 0x400B C000)

Symbol	Access	Address	Description	Reset Value	Table
Control registers					
CON	WO	0x000	Control register	NA	602
CONF	R/W	0x008	Configuration register	0	603
STAT	RO	0x004	Status register	0	604
Position, index, and timer registers					
POS	RO	0x00C	Position register	0	605
MAXPOS	R/W	0x010	Maximum position register	0	606
CMPOS0	R/W	0x014	Position compare register 0	0xFFFF FFFF	607
CMPOS1	R/W	0x018	Position compare register 1	0xFFFF FFFF	608
CMPOS2	R/W	0x01C	Position compare register 2	0xFFFF FFFF	609
INXCNT	RO	0x020	Index count register 0	0	610
INXCMP0	R/W	0x024	Index compare register 0	0xFFFF FFFF	611
INXCMP1	R/W	0x04C	Index compare register 1	0xFFFF FFFF	621
INXCMP2	R/W	0x050	Index compare register 2	0xFFFF FFFF	622
LOAD	R/W	0x028	Velocity timer reload register	0	612
TIME	RO	0x02C	Velocity timer register	0	613
VEL	RO	0x030	Velocity counter register	0	614
CAP	RO	0x034	Velocity capture register	0xFFFF FFFF	615
VELCOMP	R/W	0x038	Velocity compare register	0	616
FILTERPHA	R/W	0x03C	Digital filter register on PHA	0	617
FILTERPHB	R/W	0x040	Digital filter register on PHB	0	618
FILTERINX	R/W	0x044	Digital filter register on IDX	0	619
WINDOW	R/W	0x048	Index acceptance window register	0xF	620
Interrupt registers					
INTSTAT	RO	0xFE0	Interrupt status register	0	623
SET	WO	0xFEC	Interrupt status set register	NA	624
CLR	WO	0xFE8	Interrupt status clear register	NA	625
IE	RO	0xFE4	Interrupt enable register	0	626
IES	WO	0xFDC	Interrupt enable set register	NA	627
IEC	WO	0xFD8	Interrupt enable clear register	NA	628

28.7.2 Control registers

28.7.2.1 QEI Control register

This register contains bits which control the operation of the position and velocity counters of the QEI module.

Table 602: QEI Control register (CON - address 0x400B C000) bit description

Bit	Symbol	Description
0	RESP	Reset position counter. When 1 is written, resets the position counter to all zeros.
1	RESPI	Reset position counter on index. When 1 is written, resets the position counter to all zeros once only the first time an index pulse occurs.
2	RESV	Reset velocity. When 1 is written, resets the velocity counter to all zeros, reloads the velocity timer, and presets the velocity compare register.
3	RESI	Reset index counter. When 1 is written, resets the index counter to all zeros.
31:4	-	Reserved. Read value is undefined, only zero should be written.

28.7.2.2 QEI Configuration register

This register contains the configuration of the QEI module.

Table 603: QEI Configuration register (CONF - address 0x400B C008) bit description

Bit	Symbol	Description	Reset value
0	DIRINV	Direction invert. When 1, complements the DIR bit.	0
1	SIGMODE	Signal Mode. When 0, PhA and PhB function as quadrature encoder inputs. When 1, PhA functions as the direction signal and PhB functions as the clock signal.	0
2	CAPMODE	Capture Mode. When 0, only PhA edges are counted (2X). When 1, BOTH PhA and PhB edges are counted (4X), increasing resolution but decreasing range.	0
3	INVINX	Invert Index. When 1, inverts the sense of the index input.	0
4	CRESPI	Continuously reset the position counter on index. When 1, resets the position counter to all zeros whenever an index pulse occurs after the next position increase (recalibration).	0
15:5	-	Reserved. Read value is undefined, only zero should be written.	NA
19:16	INXGATE	Index gating configuration: When INXGATE[16] = 1, pass the index when PHA = 1 and PHB = 0, otherwise block index. When INXGATE[17] = 1, pass the index when PHA = 1 and PHB = 1, otherwise block index. When INXGATE[18] = 1, pass the index when PHA = 0 and PHB = 1, otherwise block index. When INXGATE[19] = 1, pass the index when PHA = 0 and PHB = 0, otherwise block index.	0xF
31:20	-	Reserved. Read value is undefined, only zero should be written.	NA

28.7.2.3 QEI Status register

This register provides the status of the encoder interface.

Table 604: QEI Status register (STAT - address 0x400B C004) bit description

Bit	Symbol	Description	Reset value
0	DIR	Direction bit. In combination with DIRINV bit indicates forward or reverse direction. See Table 599.	0
31:1	-	Reserved. Read value is undefined, only zero should be written.	NA

28.7.3 Position, index and timer registers

28.7.3.1 QEI Position register

This register contains the current value of the encoder position. Increments or decrements when encoder counts occur, depending on the direction of rotation.

Table 605: QEI Position register (POS - address 0x400B C00C) bit description

Bit	Symbol	Description	Reset value
31:0	POS	Current position value.	0

28.7.3.2 QEI Maximum Position register

This register contains the maximum value of the encoder position. In forward rotation the position register resets to zero when the position register exceeds this value. In reverse rotation the position register resets to this value when the position register decrements from zero.

Table 606: QEI Maximum Position register (MAXPOS - address 0x400B C010) bit description

Bit	Symbol	Description	Reset value
31:0	MAXPOS	Current maximum position value.	0

28.7.3.3 QEI Position Compare register 0

This register contains a position compare value. This value is compared against the current value of the position register. An interrupt can be generated when the compare value is equal to the current value of the position register. The compare value must take into account the fact that the starting position is zero.

Table 607: QEI Position Compare register 0 (CMPOS0 - address 0x400B C014) bit description

Bit	Symbol	Description	Reset value
31:0	PCMP0	Position compare value 0.	0

28.7.3.4 QEI Position Compare register 1

This register contains a position compare value. This value is compared against the current value of the position register. An interrupt can be generated when the compare value is equal to the current value of the position register. The compare value must take into account the fact that the starting position is zero.

Table 608: QEI Position Compare register 1 (CMPOS1 - address 0x400B C018) bit description

Bit	Symbol	Description	Reset value
31:0	PCMP1	Position compare value 1.	0

28.7.3.5 QEI Position Compare register 2

This register contains a position compare value. This value is compared against the current value of the position register. An interrupt can be generated when the compare value is equal to the current value of the position register. The compare value must take into account the fact that the starting position is zero.

Table 609: QEI Position Compare register 2 (CMPOS2 - address 0x400B C01C) bit description

Bit	Symbol	Description	Reset value
31:0	PCMP2	Position compare value 2.	0

28.7.3.6 QEI Index Count register

This register contains the current value of the index counter. It is updated when an index count occurs. This can be an increment when the position counter overflows the MAXPOS value or a decrement when the position counter underflows zero, depending on the direction of rotation. In case (re)calibration occurs due to an index pulse, the over/underflow is forced internally.

Table 610: QEI Index Count register (INXCNT - address 0x400B C020) bit description

Bit	Symbol	Description	Reset value
31:0	ENCPOS	Current index counter value.	0

28.7.3.7 QEI Index Compare register 0

This register contains an index compare value. This value is compared against the current value of the index count register. Interrupts can be enabled to interrupt when the compare value is less than, equal to, or greater than the current value of the index count register.

Table 611: QEI Index Compare register 0 (INXCMP0 - address 0x400B C024) bit description

Bit	Symbol	Description	Reset value
31:0	ICMP0	Index compare value 0.	0

28.7.3.8 QEI Velocity Timer Reload register

This register contains the reload value of the velocity timer. When the timer (QEITIME) overflows or the RESV bit is asserted, this value is loaded into the timer (QEITIME).

Table 612: QEI Timer Load register (LOAD - address 0x400B C028) bit description

Bit	Symbol	Description	Reset value
31:0	VELLOAD	Current velocity timer load value.	0

28.7.3.9 QEI Velocity Timer register

This register contains the current value of the velocity timer. When this timer overflows the value of velocity counter (QEIVEL) is stored in the velocity capture register (QEICAP), the velocity counter is reset to zero, the timer is reloaded with the value stored in the velocity reload register (QEILOAD), and the velocity interrupt (TIM_Int) is asserted.

Table 613: QEI Timer register (TIME - address 0x400B C02C) bit description

Bit	Symbol	Description	Reset value
31:0	VELVAL	Current velocity timer value.	0

28.7.3.10 QEI Velocity register

This register contains the running count of velocity pulses for the current time period. When the velocity timer (QEITIME) overflows the contents of this register is captured in the velocity capture register (QEICAP). After capture, this register is set to zero. This register is also reset when the velocity reset bit (RESV) is asserted.

Table 614: QEI Velocity register (VEL - address 0x400B C030) bit description

Bit	Symbol	Description	Reset value
31:0	VELPC	Current velocity pulse count.	0

28.7.3.11 QEI Velocity Capture register

This register contains the most recently measured velocity of the encoder. This corresponds to the number of velocity pulses counted in the previous velocity timer period.The current velocity count is latched into this register when the velocity timer overflows.

Table 615: QEI Velocity Capture register (CAP - address 0x400B C034) bit description

Bit	Symbol	Description	Reset value
31:0	VELCAP	Last velocity capture.	0

28.7.3.12 QEI Velocity Compare register

This register contains a velocity compare value. This value is compared against the captured velocity in the velocity capture register. If the capture velocity is less than the value in this compare register, a velocity compare interrupt (VELC_Int) will be asserted, if enabled.

Table 616: QEI Velocity Compare register (VELCOMP - address 0x400B C038) bit description

Bit	Symbol	Description	Reset value
31:0	VELPC	Compare velocity pulse count.	0

28.7.3.13 QEI Digital Filter on PHA

This register contains the sampling count for the digital filter. A sampling count of zero bypasses the filter.

Table 617: QEI Digital Filter ON PHA (FILTERPHA - address 0x400B C03C) bit description

Bit	Symbol	Description	Reset value
31:0	FILTA	Digital filter sampling delay for PhA.	0

28.7.3.14 QEI Digital Filter on PHB

This register contains the sampling count for the digital filter. A sampling count of zero bypasses the filter.

Table 618: QEI Digital Filter on PHB (FILTERPHB - address 0x400B C040) bit description

Bit	Symbol	Description	Reset value
31:0	FILTB	Digital filter sampling delay for PhB.	0

28.7.3.15 QEI Digital Filter on INX

This register contains the sampling count for the digital filter. A sampling count of zero bypasses the filter.

Table 619: QEI Digital Filter on INX (FILTERINX - address 0x400B C044) bit description

Bit	Symbol	Description	Reset value
31:0	FITLINX	Digital filter sampling delay for the index.	0

28.7.3.16 QEI index acceptance Window

This register contains the width of the index acceptance window, when the index and the phase/clock edges fall nearly together. If the activating phase/clock edge falls before the index, but within the window, the (re)calibration will be activated on that phase/clock edge.

Table 620: QEI index acceptance Window (WINDOW - address 0x400B C048) bit description

Bit	Symbol	Description	Reset value
31:0	WINDOW	Index acceptance window width.	0xF

28.7.3.17 QEI Index Compare register 1

This register contains an index compare value. This value is compared against the current value of the index count register. Interrupts can be enabled to interrupt when the compare value is less than, equal to, or greater than the current value of the index count register.

Table 621: QEI Index Compare register 1 (INXCMP1 - address 0x400B C04C) bit description

Bit	Symbol	Description	Reset value
31:0	ICMP1	Index compare value 1.	0

28.7.3.18 QEI Index Compare register 2

This register contains an index compare value. This value is compared against the current value of the index count register. Interrupts can be enabled to interrupt when the compare value is less than, equal to, or greater than the current value of the index count register.

Table 622: QEI Index Compare register 2 (INXCMP2 - address 0x400B C050) bit description

Bit	Symbol	Description	Reset value
31:0	ICMP2	Index compare value 2.	0

28.7.4 Interrupt registers

28.7.4.1 QEI Interrupt Status register

This register provides the status of the encoder interface and the current set of interrupt sources that are asserted to the controller. Bits set to 1 indicate the latched events that have occurred; a zero bit indicates that the event in question has not occurred.

Table 623: QEI Interrupt Status register (INTSTAT - address 0x400B CFE0) bit description

Bit	Symbol	Description	Reset value
0	INX_INT	Indicates that an index pulse was detected.	0
1	TIM_INT	Indicates that a velocity timer overflow occurred	0
2	VELC_INT	Indicates that captured velocity is less than compare velocity.	0
3	DIR_INT	Indicates that a change of direction was detected.	0
4	ERR_INT	Indicates that an encoder phase error was detected.	0
5	ENCLK_INT	Indicates that and encoder clock pulse was detected.	
6	POS0_INT	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_INT	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_INT	Indicates that the position 2 compare value is equal to the current position.	0
9	REV0_INT	Indicates that the index compare 0 value is equal to the current index count.	0
10	POS0REV_INT	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV0_Int is set.	0
11	POS1REV_INT	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV1_Int is set.	0
12	POS2REV_INT	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV2_Int is set.	0
13	REV1_INT	Indicates that the index compare 1value is equal to the current index count.	0
14	REV2_INT	Indicates that the index compare 2 value is equal to the current index count.	0
15	MAXPOS_INT	Indicates that the current position count goes through the MAXPOS value to zero in the forward direction, or through zero to MAXPOS in the reverse direction.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

28.7.4.2 QEI Interrupt Set register

Writing a one to a bit in this register sets the corresponding bit in the QEI Interrupt Status register (QEISTAT).

Table 624: QEI Interrupt Set register (SET - address 0x400B CFEC) bit description

Bit	Symbol	Description
0	INX_INT	Writing a 1 sets the INX_Int bit in QEIINTSTAT.
1	TIM_INT	Writing a 1 sets the TIN_Int bit in QEIINTSTAT.
2	VELC_INT	Writing a 1 sets the VELC_Int bit in QEIINTSTAT.
3	DIR_INT	Writing a 1 sets the DIR_Int bit in QEIINTSTAT.
4	ERR_INT	Writing a 1 sets the ERR_Int bit in QEIINTSTAT.
5	ENCLK_INT	Writing a 1 sets the ENCLK_Int bit in QEIINTSTAT.
6	POS0_INT	Writing a 1 sets the POS0_Int bit in QEIINTSTAT.
7	POS1_INT	Writing a 1 sets the POS1_Int bit in QEIINTSTAT.
8	POS2_INT	Writing a 1 sets the POS2_Int bit in QEIINTSTAT.
9	REV0_INT	Writing a 1 sets the REV0_Int bit in QEIINTSTAT.
10	POS0REV_INT	Writing a 1 sets the POS0REV_Int bit in QEIINTSTAT.
11	POS1REV_INT	Writing a 1 sets the POS1REV_Int bit in QEIINTSTAT.
12	POS2REV_INT	Writing a 1 sets the POS2REV_Int bit in QEIINTSTAT.
13	REV1_INT	Writing a 1 sets the REV1_Int bit in QEIINTSTAT.
14	REV2_INT	Writing a 1 sets the REV2_Int bit in QEIINTSTAT.
15	MAXPOS_INT	Writing a 1 sets the MAXPOS_Int bit in QEIINTSTAT.
31:16	-	Reserved. Read value is undefined, only zero should be written.

28.7.4.3 QEI Interrupt Clear register

Writing a one to a bit in this register clears the corresponding bit in the QEI Interrupt Status register (QEISTAT).

Table 625: QEI Interrupt Clear register (CLR - 0x400B CFE8) bit description

Bit	Symbol	Description
0	INX_INT	Writing a 1 clears the INX_Int bit in QEIINTSTAT.
1	TIM_INT	Writing a 1 clears the TIN_Int bit in QEIINTSTAT.
2	VELC_INT	Writing a 1 clears the VELC_Int bit in QEIINTSTAT.
3	DIR_INT	Writing a 1 clears the DIR_Int bit in QEIINTSTAT.
4	ERR_INT	Writing a 1 clears the ERR_Int bit in QEIINTSTAT.
5	ENCLK_INT	Writing a 1 clears the ENCLK_Int bit in QEIINTSTAT.
6	POS0_INT	Writing a 1 clears the POS0_Int bit in QEIINTSTAT.
7	POS1_INT	Writing a 1 clears the POS1_Int bit in QEIINTSTAT.
8	POS2_INT	Writing a 1 clears the POS2_Int bit in QEIINTSTAT.
9	REV0_INT	Writing a 1 clears the REV0_Int bit in QEIINTSTAT.
10	POS0REV_INT	Writing a 1 clears the POS0REV_Int bit in QEIINTSTAT.
11	POS1REV_INT	Writing a 1 clears the POS1REV_Int bit in QEIINTSTAT.
12	POS2REV_INT	Writing a 1 clears the POS2REV_Int bit in QEIINTSTAT.
13	REV1_INT	Writing a 1 clears the REV1_Int bit in QEIINTSTAT.
14	REV2_INT	Writing a 1 clears the REV2_Int bit in QEIINTSTAT.
15	MAXPOS_INT	Writing a 1 clears the MAXPOS_Int bit in QEIINTSTAT.
31:16	-	Reserved. Read value is undefined, only zero should be written.

28.7.4.4 QEI Interrupt Enable register

This register enables interrupt sources. Bits set to 1 enable the corresponding interrupt; a zero bit disables the corresponding interrupt.

Table 626: QEI Interrupt Enable register (IE - address 0x400B CFE4) bit description

Bit	Symbol	Description	Reset value
0	INX_INT	When 1, the INX_Int interrupt is enabled.	0
1	TIM_INT	When 1, the TIN_Int interrupt is enabled.	0
2	VELC_INT	When 1, the VELC_Int interrupt is enabled.	0
3	DIR_INT	When 1, the DIR_Int interrupt is enabled.	0
4	ERR_INT	When 1, the ERR_Int interrupt is enabled.	0
5	ENCLK_INT	When 1, the ENCLK_Int interrupt is enabled.	0
6	POS0_INT	When 1, the POS0_Int interrupt is enabled.	0
7	POS1_INT	When 1, the POS1_Int interrupt is enabled.	0
8	POS2_INT	When 1, the POS2_Int interrupt is enabled.	0
9	REV0_INT	When 1, the REV0_Int interrupt is enabled.	0
10	POS0REV_INT	When 1, the POS0REV_Int interrupt is enabled.	0
11	POS1REV_INT	When 1, the POS1REV_Int interrupt is enabled.	0
12	POS2REV_INT	When 1, the POS2REV_Int interrupt is enabled.	0
13	REV1_INT	When 1, the REV1_Int interrupt is enabled.	0
14	REV2_INT	When 1, the REV2_Int interrupt is enabled.	0
15	MAXPOS_INT	When 1, the MAXPOS_Int interrupt is enabled.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

28.7.4.5 QEI Interrupt Enable Set register

Writing a one to a bit in this register sets the corresponding bit in the QEI Interrupt Enable register (QEIIE).

Table 627: QEI Interrupt Enable Set register (IES - address 0x400B CFDC) bit description

Bit	Symbol	Description
0	INX_INT	Writing a 1 enables the INX_Int interrupt in the QEIIE register.
1	TIM_INT	Writing a 1 enables the TIN_Int interrupt in the QEIIE register.
2	VELC_INT	Writing a 1 enables the VELC_Int interrupt in the QEIIE register.
3	DIR_INT	Writing a 1 enables the DIR_Int interrupt in the QEIIE register.
4	ERR_INT	Writing a 1 enables the ERR_Int interrupt in the QEIIE register.
5	ENCLK_INT	Writing a 1 enables the ENCLK_Int interrupt in the QEIIE register.
6	POS0_INT	Writing a 1 enables the POS0_Int interrupt in the QEIIE register.
7	POS1_INT	Writing a 1 enables the POS1_Int interrupt in the QEIIE register.
8	POS2_INT	Writing a 1 enables the POS2_Int interrupt in the QEIIE register.
9	REV0_INT	Writing a 1 enables the REV0_Int interrupt in the QEIIE register.
10	POS0REV_INT	Writing a 1 enables the POS0REV_Int interrupt in the QEIIE register.
11	POS1REV_INT	Writing a 1 enables the POS1REV_Int interrupt in the QEIIE register.
12	POS2REV_INT	Writing a 1 enables the POS2REV_Int interrupt in the QEIIE register.
13	REV1_INT	Writing a 1 enables the REV1_Int interrupt in the QEIIE register.
14	REV2_INT	Writing a 1 enables the REV2_Int interrupt in the QEIIE register.
15	MAXPOS_INT	Writing a 1 enables the MAXPOS_Int interrupt in the QEIIE register.
31:16	-	Reserved. Read value is undefined, only zero should be written.

28.7.4.6 QEI Interrupt Enable Clear register

Writing a one to a bit in this register clears the corresponding bit in the QEI Interrupt Enable register (QEIIE).

Table 628: QEI Interrupt Enable Clear register (IEC - address 0x400B CFD8) bit description

Bit	Symbol	Description
0	INX_INT	Writing a 1 disables the INX_Int interrupt in the QEIIE register.
1	TIM_INT	Writing a 1 disables the TIN_Int interrupt in the QEIIE register.
2	VELC_INT	Writing a 1 disables the VELC_Int interrupt in the QEIIE register.
3	DIR_INT	Writing a 1 disables the DIR_Int interrupt in the QEIIE register.
4	ERR_INT	Writing a 1 disables the ERR_Int interrupt in the QEIIE register.
5	ENCLK_INT	Writing a 1 disables the ENCLK_Int interrupt in the QEIIE register.
6	POS0_INT	Writing a 1 disables the POS0_Int interrupt in the QEIIE register.
7	POS1_INT	Writing a 1 disables the POS1_Int interrupt in the QEIIE register.
8	POS2_INT	Writing a 1 disables the POS2_Int interrupt in the QEIIE register.
9	REV0_INT	Writing a 1 disables the REV0_Int interrupt in the QEIIE register.
10	POS0REV_INT	Writing a 1 disables the POS0REV_Int interrupt in the QEIIE register.
11	POS1REV_INT	Writing a 1 disables the POS1REV_Int interrupt in the QEIIE register.
12	POS2REV_INT	Writing a 1 disables the POS2REV_Int interrupt in the QEIIE register.
13	REV1_INT	Writing a 1 disables the REV1_Int interrupt in the QEIIE register.
14	REV2_INT	Writing a 1 disables the REV2_Int interrupt in the QEIIE register.
15	MAXPOS_INT	Writing a 1 disables the MAXPOS_Int interrupt in the QEIIE register.
31:16	-	Reserved. Read value is undefined, only zero should be written.

29.1 Basic configuration

The RTC is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCRTC (applies to both the RTC and the Event Recorder).

   Remark: On reset, the RTC is enabled. See Section 29.7 for power saving options.
2. Clock: The RTC uses the 1 Hz clock output from the RTC oscillator as the internal function clock. The peripheral clock is used for accessing RTC registers.
3. Interrupts: See Section 29.6.1 for RTC interrupt handling. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

29.2 Features

- Measures the passage of time to maintain a calendar and clock. Provides seconds, minutes, hours, day of month, month, year, day of week, and day of year.
- Ultra-low power design to support battery powered systems. Less than 1 microamp required for battery operation. Uses power from the CPU power supply whenever it is greater than V_{BAT}.
- 20 bytes of Battery-backed storage and RTC operation when power is removed from the CPU.
- Dedicated 32 kHz ultra low power oscillator.
- Dedicated battery power supply pin.
- RTC power supply is isolated from the rest of the chip.
- Calibration counter allows adjustment to better than ±1 sec/day with 1 sec resolution.
- Periodic interrupts can be generated from increments of any field of the time registers.
- Alarm interrupt can be generated for a specific date/time.

29.3 Description

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off. It uses very little power when its registers are not being accessed by the CPU, especially reduced power modes. The RTC is clocked by a separate 32 kHz oscillator that produces a 1 Hz internal time reference. The RTC is powered by its own power supply pin, V_{BAT}, which can be connected to a battery, externally tied to a 3V supply, or left floating.

The RTC power domain is shown in conceptual form in Figure 156. A detailed view of the time keeping portion of the RTC is shown in Figure 157.

29.4 Architecture

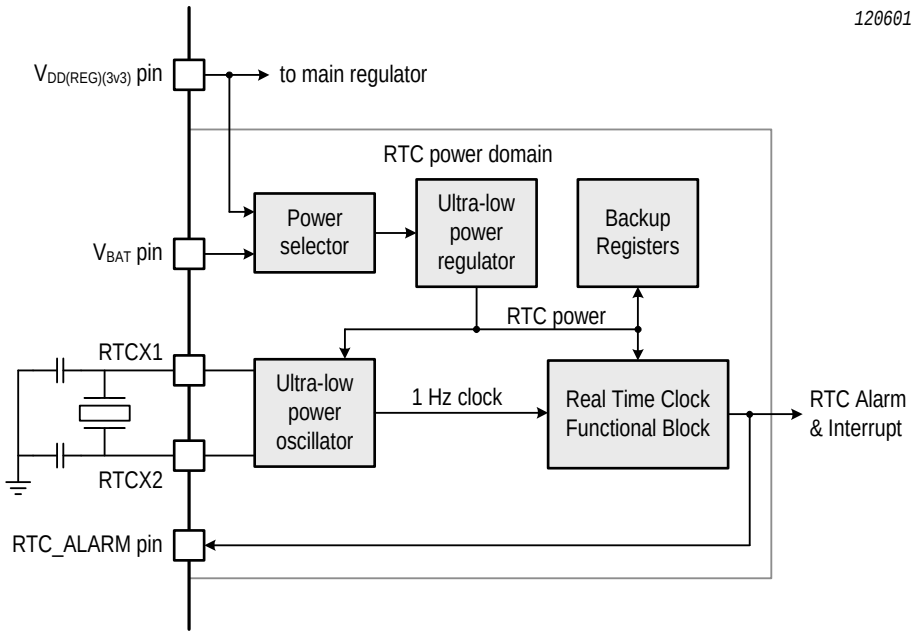


Fig 156. RTC domain conceptual diagram

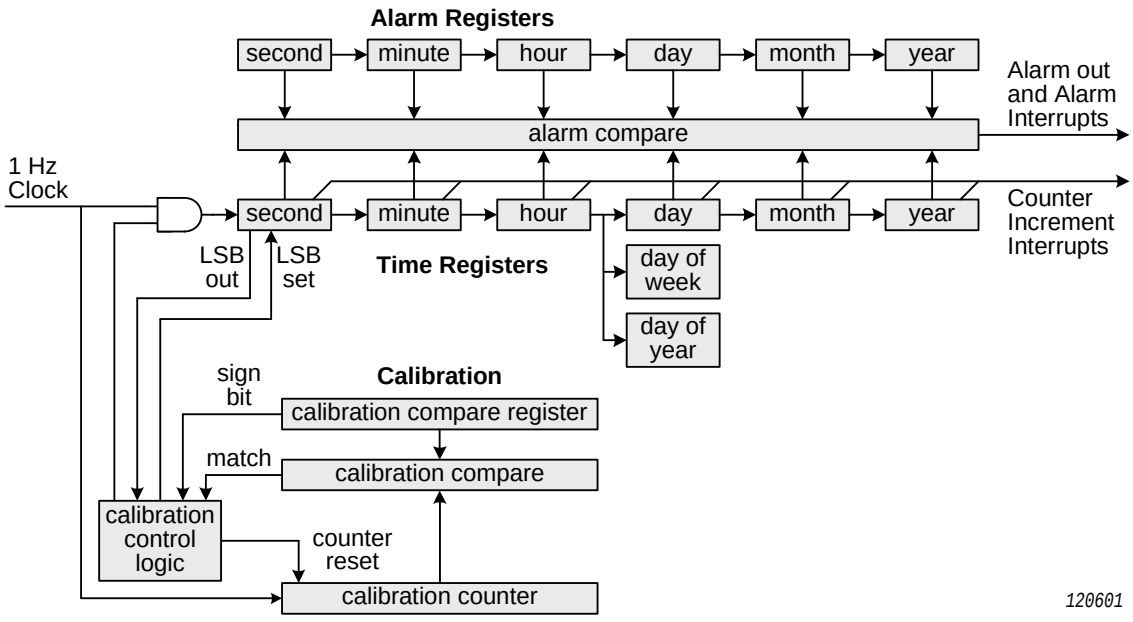


Fig 157. RTC functional block diagram

29.5 Pin description

Table 629. RTC pin description

Name	Type	Description
RTC_ALARM	Output	Alarm output from the RTC. This is an active high pin that is asserted when the internal RTC alarm occurs. The output is cleared by writing a 1 to the RTCALF bit in the Interrupt Location Register. This pin is powered by V_{BAT}. The pin can alternatively be used to indicate the Deep Power-down mode status via RTC_PDOUT in the RTC_AUX register.
RTCX1	Input	Input to the RTC oscillator circuit.
RTCX2	Output	Output from the RTC oscillator circuit. **Remark:** If the RTC is not used, the RTCX1/2 pins can be left floating.
V_{BAT}	Input	**RTC power supply:** Typically connected to an external 3V battery. If this pin is not powered, the RTC is still powered internally if $V_{DD(REG)(3V3)}$ is present.

29.6 Register description

In the register descriptions, for most of the registers the Reset Value column shows "NC", meaning that these registers are Not Changed by a Reset. Software must initialize these registers between power-on and setting the RTC into operation. The registers are split into five sections by functionality.

Table 630. Register overview: Real-Time Clock (base address 0x4002 4000)

Name	Access	Address	Description	Reset Value[1]	Table
Miscellaneous registers (see Section 29.6.2)					
ILR	R/W	0x000	Interrupt Location Register	0	631
CCR	R/W	0x008	Clock Control Register	NC	631
CIIR	R/W	0x00C	Counter Increment Interrupt Register	0	633
AMR	R/W	0x010	Alarm Mask Register	0	634
Consolidated time registers (see Section 29.6.3)					
CTIME0	RO	0x014	Consolidated Time Register 0	NC	637
CTIME1	RO	0x018	Consolidated Time Register 1	NC	638
CTIME2	RO	0x01C	Consolidated Time Register 2	NC	639
Time counter registers (see Section 29.6.4)					
SEC	R/W	0x020	Seconds Counter	NC	641
MIN	R/W	0x024	Minutes Register	NC	641
HRS	R/W	0x028	Hours Register	NC	641
DOM	R/W	0x02C	Day of Month Register	NC	641
DOW	R/W	0x030	Day of Week Register	NC	641
DOY	R/W	0x034	Day of Year Register	NC	641
MONTH	R/W	0x038	Months Register	NC	641
YEAR	R/W	0x03C	Years Register	NC	641
CALIBRATION	R/W	0x040	Calibration Value Register	NC	650
General purpose registers (see Section 29.6.6)					
GPREG0	R/W	0x044	General Purpose Register 0	NC	651
GPREG1	R/W	0x048	General Purpose Register 1	NC	651
GPREG2	R/W	0x04C	General Purpose Register 2	NC	651
GPREG3	R/W	0x050	General Purpose Register 3	NC	651
GPREG4	R/W	0x054	General Purpose Register 4	NC	651
RTC_AUX	R/W	0x05C	RTC Auxiliary control register	0x10	635
RTC_AUXEN	R/W	0x058	RTC Auxiliary Enable register	0	636
Alarm register group (see Section 29.6.7)					
ASEC	R/W	0x060	Alarm value for Seconds	NC	652
AMIN	R/W	0x64	Alarm value for Minutes	NC	652
AHRS	R/W	0x068	Alarm value for Hours	NC	652
ADOM	R/W	0x06C	Alarm value for Day of Month	NC	652
ADOW	R/W	0x070	Alarm value for Day of Week	NC	652
ADOY	R/W	0x074	Alarm value for Day of Year	NC	652
AMON	R/W	0x078	Alarm value for Months	NC	652
AYRS	R/W	0x07C	Alarm value for Year	NC	652

[1] Reset values apply only to a power-up of the RTC block, other types of reset have no effect on this block. Since the RTC is powered whenever either of the $V_{DD(REG)(3V3)}$, or V_{BAT} supplies are present, power-up reset occurs only when both supplies were absent and then one is turned on. Most registers are not affected by power-up of the RTC and must be initialized by software if the RTC is enabled. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

29.6.1 RTC interrupts

Interrupt generation is controlled through the Interrupt Location Register (ILR), Counter Increment Interrupt Register (CIIR), the alarm registers, and the Alarm Mask Register (AMR). Interrupts are generated only by the transition into the interrupt state. The ILR separately enables CIIR and AMR interrupts. Each bit in CIIR corresponds to one of the time counters. If CIIR is enabled for a particular counter, then every time the counter is incremented an interrupt is generated. The alarm registers allow the user to specify a date and time for an interrupt to be generated. The AMR provides a mechanism to mask alarm compares. If all non-masked alarm registers match the value in their corresponding time counter, then an interrupt is generated.

The RTC and Event Recorder interrupts are combined, and can bring the microcontroller out of Sleep, Deep Sleep, and Power-down modes when the RTC is running and the RTC/Event Recorder interrupt is enabled in the NVIC. For details on the RTC based wake-up process see Section 3.12.8 “Wake-up from Reduced Power Modes” on page 72 and Section 3.13 “Wake-up timer” on page 73.

29.6.2 Miscellaneous register group

29.6.2.1 Interrupt Location Register

The Interrupt Location Register is a 2-bit register that specifies which blocks are generating an interrupt (see Table 631). Writing a one to the appropriate bit clears the corresponding interrupt. Writing a zero has no effect. This allows the programmer to read this register and write back the same value to clear only the interrupt that is detected by the read.

Table 631. Interrupt Location Register (ILR - address 0x4002 4000) bit description

Bit	Symbol	Description	Reset value
0	RTCCIF	When 1, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.	0
1	RTCALF	When 1, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.	0
31:21	-	Reserved. Read value is undefined, only zero should be written.	NA

29.6.2.2 Clock Control Register

The clock register is a 4-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 632. All NC bits in this register should be initialized when the RTC is first turned on.

Table 632. Clock Control Register (CCR - address 0x4002 4008) bit description

Bit	Symbol	Value	Description	Reset value
0	CLKEN		Clock Enable.	NC
		1	The time counters are enabled.	
		0	The time counters are disabled so that they may be initialized.	
1	CTCRST		CTC Reset.	0
		1	When 1, the elements in the internal oscillator divider are reset, and remain reset until CCR[1] is changed to zero. This is the divider that generates the 1 Hz clock from the 32.768 kHz crystal. The state of the divider is not visible to software.	
		0	No effect.	

Table 632. Clock Control Register (CCR - address 0x4002 4008) bit description

Bit	Symbol	Value	Description	Reset value
3:2	-		Internal test mode controls. These bits must be 0 for normal RTC operation.	NC
4	CCALEN		Calibration counter enable.	NC
		1	The calibration counter is disabled and reset to zero.	
		0	The calibration counter is enabled and counting, using the 1 Hz clock. When the calibration counter is equal to the value of the CALIBRATION register, the counter resets and repeats counting up to the value of the CALIBRATION register. See Section 29.6.4.2 and Section 29.6.5.	
31:5	-		Reserved. Read value is undefined, only zero should be written.	NA

29.6.2.3 Counter Increment Interrupt Register

The Counter Increment Interrupt Register (CIIR) gives the ability to generate an interrupt every time a counter is incremented. This interrupt remains valid until cleared by writing a 1 to bit 0 of the Interrupt Location Register (ILR[0]).

Table 633. Counter Increment Interrupt Register (CIIR - address 0x4002 400C) bit description

Bit	Symbol	Description	Reset value
0	IMSEC	When 1, an increment of the Second value generates an interrupt.	0
1	IMMIN	When 1, an increment of the Minute value generates an interrupt.	0
2	IMHOUR	When 1, an increment of the Hour value generates an interrupt.	0
3	IMDOM	When 1, an increment of the Day of Month value generates an interrupt.	0
4	IMDOW	When 1, an increment of the Day of Week value generates an interrupt.	0
5	IMDOY	When 1, an increment of the Day of Year value generates an interrupt.	0
6	IMMON	When 1, an increment of the Month value generates an interrupt.	0
7	IMYEAR	When 1, an increment of the Year value generates an interrupt.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

29.6.2.4 Alarm Mask Register

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 634 shows the relationship between the bits in the AMR and the alarms. For the alarm function, every non-masked alarm register must match the corresponding time counter for an interrupt to be generated. The interrupt is generated only when the counter comparison first changes from no match to match. The interrupt is removed when a one is written to the appropriate bit of the Interrupt Location Register (ILR). If all mask bits are set, then the alarm is disabled.

Table 634. Alarm Mask Register (AMR - address 0x4002 4010) bit description

Bit	Symbol	Description	Reset value
0	AMRSEC	When 1, the Second value is not compared for the alarm.	0
1	AMRMIN	When 1, the Minutes value is not compared for the alarm.	0
2	AMRHOUR	When 1, the Hour value is not compared for the alarm.	0
3	AMRDOM	When 1, the Day of Month value is not compared for the alarm.	0
4	AMRDOW	When 1, the Day of Week value is not compared for the alarm.	0
5	AMRDOY	When 1, the Day of Year value is not compared for the alarm.	0

Table 634. Alarm Mask Register (AMR - address 0x4002 4010) bit description

Bit	Symbol	Description	Reset value
6	AMRMON	When 1, the Month value is not compared for the alarm.	0
7	AMRYEAR	When 1, the Year value is not compared for the alarm.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

29.6.2.5 RTC Auxiliary control register

The RTC Auxiliary Control register contains added interrupt flags for functions that are not part of the Real Time Clock itself (the part recording the passage of time and generating other time related functions). The only added interrupt flag for these devices is for failure of the RTC oscillator.

Table 635. RTC Auxiliary control register (RTC_AUX - address 0x4002 405C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	NA
4	RTC_OSCF	RTC Oscillator Fail detect flag. Read: this bit is set if the RTC oscillator stops, and when RTC power is first turned on. An interrupt will occur when this bit is set, the RTC_OSCFEN bit in RTC_AUXEN is a 1, and the RTC interrupt is enabled in the NVIC. Write: writing a 1 to this bit clears the flag.	1
5	-	Reserved. Read value is undefined, only zero should be written.	NA
6	RTC_PDOUT	When 0: the RTC_ALARM pin reflects the RTC alarm status. When 1: the RTC_ALARM pin indicates Deep Power-down mode.	0
31:7	-	Reserved. Read value is undefined, only zero should be written.	NA

29.6.2.6 RTC Auxiliary Enable register

The RTC Auxiliary Enable Register controls whether additional interrupt sources represented in the RTC Auxiliary control register are enabled.

Table 636. RTC Auxiliary Enable register (RTC_AUXEN - address 0x4002 4058) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	NA
4	RTC_OSCFEN	Oscillator Fail Detect interrupt enable. When 0: the RTC Oscillator Fail detect interrupt is disabled. When 1: the RTC Oscillator Fail detect interrupt is enabled. See Section 29.6.2.5.	0
31:5	-	Reserved. Read value is undefined, only zero should be written.	NA

29.6.3 Consolidated time registers

The values of the Time Counters can optionally be read in a consolidated format which allows the programmer to read all time counters with only three read operations. The various registers are packed into 32-bit values as shown in Table 637, Table 638, and Table 639. The least significant bit of each register is read back at bit 0, 8, 16, or 24.

The Consolidated Time Registers are read-only. To write new values to the Time Counters, the Time Counter addresses should be used.

29.6.3.1 Consolidated Time Register 0

The Consolidated Time Register 0 contains the low order time values: Seconds, Minutes, Hours, and Day of Week.

Table 637. Consolidated Time register 0 (CTIME0 - address 0x4002 4014) bit description

Bit	Symbol	Description	Reset value
5:0	SECONDS	Seconds value in the range of 0 to 59	NC
7:6	-	Reserved. The value read from a reserved bit is not defined.	NA
13:8	MINUTES	Minutes value in the range of 0 to 59	NC
15:14	-	Reserved. The value read from a reserved bit is not defined.	NA
20:16	HOURS	Hours value in the range of 0 to 23	NC
23:21	-	Reserved. The value read from a reserved bit is not defined.	NC
26:24	DOW	Day of week value in the range of 0 to 6	NA
31:27	-	Reserved. The value read from a reserved bit is not defined.	NC

29.6.3.2 Consolidated Time Register 1

The Consolidate Time Register 1 contains the Day of Month, Month, and Year values.

Table 638. Consolidated Time register 1 (CTIME1 - address 0x4002 4018) bit description

Bit	Symbol	Description	Reset value
4:0	DOM	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).	NC
7:5	-	Reserved. The value read from a reserved bit is not defined.	NA
11:8	MONTH	Month value in the range of 1 to 12.	NC
15:12	-	Reserved. The value read from a reserved bit is not defined.	NA
27:16	YEAR	Year value in the range of 0 to 4095.	NC
31:28	-	Reserved. The value read from a reserved bit is not defined.	NA

29.6.3.3 Consolidated Time Register 2

The Consolidate Time Register 2 contains just the Day of Year value.

Table 639. Consolidated Time register 2 (CTIME2 - address 0x4002 401C) bit description

Bit	Symbol	Description	Reset value
11:0	DOY	Day of year value in the range of 1 to 365 (366 for leap years).	NC
31:12	-	Reserved. The value read from a reserved bit is not defined.	NA

29.6.4 Time Counter Group

The time value consists of the eight counters shown in Table 640 and Table 641. These counters can be read or written at the locations shown in Table 641.

Table 640. Time Counter relationships and values

Counter	Size	Enabled by	Minimum value	Maximum value
Second	6	1 Hz Clock	0	59
Minute	6	Second	0	59
Hour	5	Minute	0	23
Day of Month	5	Hour	1	28, 29, 30 or 31

Table 640. Time Counter relationships and values

Counter	Size	Enabled by	Minimum value	Maximum value
Day of Week	3	Hour	0	6
Day of Year	9	Hour	1	365 or 366 (for leap year)
Month	4	Day of Month	1	12
Year	12	Month or day of Year	0	4095

Table 641. Time Counter registers

Name	Size	Description	Access	Address
SEC	6	Seconds value in the range of 0 to 59	R/W	0x4002 4020
MIN	6	Minutes value in the range of 0 to 59	R/W	0x4002 4024
HRS	5	Hours value in the range of 0 to 23	R/W	0x4002 4028
DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).[1]	R/W	0x4002 402C
DOW	3	Day of week value in the range of 0 to 6[1]	R/W	0x4002 4030
DOY	9	Day of year value in the range of 1 to 365 (366 for leap years)[1]	R/W	0x4002 4034
MONTH	4	Month value in the range of 1 to 12	R/W	0x4002 4038
YEAR	12	Year value in the range of 0 to 4095	R/W	0x4002 403C

[1] These values are simply incremented at the appropriate intervals and reset at the defined overflow point. They are not calculated and must be correctly initialized in order to be meaningful.

Table 642. Seconds register (SEC - address 0x4002 4020) bit description

Bit	Symbol	Description	Reset value
5:0	SECONDS	Seconds value in the range of 0 to 59. The register value is not changed by reset.	-
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 643. Minutes register (MIN - address 0x4002 4024) bit description

Bit	Symbol	Description	Reset value
5:0	MINUTES	Minutes value in the range of 0 to 59. The register value is not changed by reset.	-
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 644. Hours register (HRS - address 0x4002 4028) bit description

Bit	Symbol	Description	Reset value
4:0	HOURS	Hours value in the range of 0 to 23. The register value is not changed by reset.	-
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 645. Day of month register (DOM - address 0x4002 402C) bit description

Bit	Symbol	Description	Reset value
4:0	DOM	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).	-
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 646. Day of week register (DOW - address 0x4002 4030) bit description

Bit	Symbol	Description	Reset value
2:0	DOW	Day of week value in the range of 0 to 6.	-
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 647. Day of year register (DOY - address 0x4002 4034) bit description

Bit	Symbol	Description	Reset value
8:0	DOY	Day of year value in the range of 1 to 365 (366 for leap years).	-
31:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 648. Month register (MONTH - address 0x4002 4038) bit description

Bit	Symbol	Description	Reset value
3:0	MONTH	Month value in the range of 1 to 12.	-
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 649. Year register (YEAR - address 0x4002 403C) bit description

Bit	Symbol	Description	Reset value
11:0	YEAR	Year value in the range of 0 to 4095. The register value is not changed by reset.	-
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

29.6.4.1 Leap year calculation

The RTC does a simple bit comparison to see if the two lowest order bits of the year counter are zero. If true, then the RTC considers that year a leap year. The RTC considers all years evenly divisible by 4 as leap years. This algorithm is accurate from the year 1901 through the year 2099, but fails for the year 2100, which is not a leap year. The only effect of leap year on the RTC is to alter the length of the month of February for the month, day of month, and year counters.

29.6.4.2 Calibration register

The following register is used to calibrate the time counter.

Table 650. Calibration register (CALIBRATION - address 0x4002 4040) bit description

Bit	Symbol	Value	Description	Reset value
16:0	CALVAL	-	If enabled, the calibration counter counts up to this value. The maximum value is 131, 072 corresponding to about 36.4 hours. Calibration is disabled if CALVAL = 0.	NC
17	CALDIR		Calibration direction	NC
		1	Backward calibration. When CALVAL is equal to the calibration counter, the RTC timers will stop incrementing for 1 second.	
		0	Forward calibration. When CALVAL is equal to the calibration counter, the RTC timers will jump by 2 seconds.	
31:12			Reserved. Read value is undefined, only zero should be written.	NA

29.6.5 Calibration procedure

The calibration logic can periodically adjust the time counter either by not incrementing the counter, or by incrementing the counter by 2 instead of 1. This allows calibrating the RTC oscillator under some typical voltage and temperature conditions without the need to externally trim the RTC oscillator.

A recommended method for determining the calibration value is to use the CLKOUT feature to unintrusively observe the RTC oscillator frequency under the conditions it is to be trimmed for, and calculating the number of clocks that will be seen before the time is off by one second. That value is used to determine CALVAL.

If the RTC oscillator is trimmed externally, the same method of unintrusively observing the RTC oscillator frequency may be helpful in that process.

Backward calibration

Enable the RTC timer and calibration in the CCR register (set bits CLKEN = 1 and CCALEN = 0). In the CALIBRATION register, set the calibration value CALVAL $\geq$ 1 and select CALDIR = 1.

- The SEC timer and the calibration counter count up for every 1 Hz clock cycle.
- When the calibration counter reaches CALVAL, a calibration match occurs and all RTC timers will be stopped for one clock cycle so that the timers will not increment in the next cycle.
- If an alarm match event occurs in the same cycle as the calibration match, the alarm interrupt will be delayed by one cycle to avoid a double alarm interrupt.

Forward calibration

Enable the RTC timer and calibration in the CCR register (set bits CLKEN = 1 and CCALEN = 0). In the CALIBRATION register, set the calibration value CALVAL $\geq$ 1 and select CALDIR = 0.

- The SEC timer and the calibration counter count up for every 1 Hz clock cycle.
- When the calibration counter reaches CALVAL, a calibration match occurs and the RTC timers are incremented by 2.
- When the calibration event occurs, the LSB of the ALSEC register is forced to be one so that the alarm interrupt will not be missed when skipping a second.

29.6.6 General purpose registers

29.6.6.1 General purpose registers 0 to 4

These registers can be used to store important information when the main power supply is off. The value in these registers is not affected by chip reset.

Table 651. General purpose registers (GPREG[0:4] - addresses 0x4002 4044 (GPREG0) to 0x4002 4054 (GPREG4)) bit description

Bit	Symbol	Description	Reset value
31:0	GP	General purpose storage.	N/A

29.6.7 Alarm register group

The alarm registers are shown in Table 652. The values in these registers are compared with the time counters. If all the unmasked (See Section 29.6.2.4 “Alarm Mask Register” on page 775) alarm registers match their corresponding time counters then an interrupt is generated. The interrupt is cleared when a 1 is written to bit 1 of the Interrupt Location Register (ILR[1]).

Table 652. Alarm registers

Name	Size	Description	Access	Address
ALSEC	6	Alarm value for Seconds	R/W	0x4002 4060
ALMIN	6	Alarm value for Minutes	R/W	0x4002 4064
ALHOUR	5	Alarm value for Hours	R/W	0x4002 4068
ALDOM	5	Alarm value for Day of Month	R/W	0x4002 406C
ALDOW	3	Alarm value for Day of Week	R/W	0x4002 4070
ALDOY	9	Alarm value for Day of Year	R/W	0x4002 4074
ALMON	4	Alarm value for Months	R/W	0x4002 4078
ALYEAR	12	Alarm value for Years	R/W	0x4002 407C

Table 653. Alarm Seconds register (ASEC - address 0x4002 4060) bit description

Bit	Symbol	Description	Reset value
5:0	SECONDS	Seconds value in the range of 0 to 59. The register value is not changed by reset.	-
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 654. Alarm Minutes register (AMIN - address 0x4002 4064) bit description

Bit	Symbol	Description	Reset value
5:0	MINUTES	Minutes value in the range of 0 to 59. The register value is not changed by reset.	-
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 655. Alarm Hours register (AHRS - address 0x4002 4068) bit description

Bit	Symbol	Description	Reset value
4:0	HOURS	Hours value in the range of 0 to 23. The register value is not changed by reset.	-
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 656. Alarm Day of month register (ADOM - address 0x4002 406C) bit description

Bit	Symbol	Description	Reset value
4:0	DOM	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year). The register value is not changed by reset.	-
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 657. Alarm Day of week register (ADOW - address 0x4002 4070) bit description

Bit	Symbol	Description	Reset value
2:0	DOW	Day of week value in the range of 0 to 6. The register value is not changed by reset.	-
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 658. Alarm Day of year register (ADOY - address 0x4002 4074) bit description

Bit	Symbol	Description	Reset value
8:0	DOY	Day of year value in the range of 1 to 365 (366 for leap years). The register value is not changed by reset.	-
31:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 659. Alarm Month register (AMON - address 0x4002 4078) bit description

Bit	Symbol	Description	Reset value
3:0	MONTH	Month value in the range of 1 to 12. The register value is not changed by reset.	-
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 660. Alarm Year register (AYRS - address 0x4002 407C) bit description

Bit	Symbol	Description	Reset value
11:0	YEAR	Year value in the range of 0 to 4095. The register value is not changed by reset.	-
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

29.7 RTC usage notes

If the RTC is used, V_{BAT} may be connected to an independent power supply (typically an external battery), or left floating. The RTC domain will always be internally powered if $V_{DD(REG)(3V3)}$ is present, even if there is no power applied to V_{BAT}. As long as power is available on either $V_{DD(REG)(3V3)}$ or V_{BAT}, the RTC will not lose its time value and backup register contents. If both $V_{DD(REG)(3V3)}$ and V_{BAT} are not present, all RTC information will be lost. The RTC will stop incrementing or be unpredictable if the clock source is lost, interrupted, or altered. The Event Recorder is powered in the same manner as the RTC.

30.1 Basic configuration

The Event Monitor/Recorder is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCRTC (applies to both the RTC and the Event Monitor/Recorder).

   Remark: On reset, the Event Monitor/Recorder is enabled.

2. Clock: The Event Monitor/Recorder uses the 1 Hz clock output from the RTC oscillator as the internal function clock. The peripheral clock is used for accessing RTC registers.
3. Interrupts: Refer to Section 29.1 for RTC/Event Monitor/Recorder interrupt handling. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

30.2 Features

- Three digital event inputs in the VBAT power domain.
- An event is a level change at the digital event inputs.
- For each event channel, two timestamps mark the first and the last occurrence of an event. Each channel also has a dedicated counter tracking the total number of events. Timestamp values are taken from the RTC.
- Runs in VBAT power domain, independent of system power supply. Can therefore run in Deep Power Down mode.
- Very low-power consumption.
- Interrupt available if system is up.
- A qualified event can be used as a wake-up trigger.
- State of event inputs accessible by software through general purpose I/O.

30.3 Applications

Recording of tampering events in sealed product enclosures. Sensors report any attempt to open the enclosure, or to tamper with the device in any other way. The primary purpose of the Event Monitor/Recorder is to store records of such events when the device is powered only by the backup battery.

30.4 Description

The Event Monitor/Recorder relies on VBAT to be present at all times. A loss or dip of VBAT voltage causes the Real-Time Clock power fail detector to reset the event recordings. It is therefore important to have a VBAT power source that can deliver power for the longest expected mains power outage.

Once system power is restored, the CPU can check for recorded tamper events. If there were tamper events, the timestamp registers for the first and the last event would indicate the period over which they occurred.

An edge on an event input is sampled with either a 1 kHz clock, a 64 Hz clock, or a 16 Hz clock. A transition in either direction must be captured by two successive edges of this clock in order to be recognized as a valid transition. This provides a 1-2 ms rejection filter in case of the 1 kHz sample clock, a 15.6-31.2 ms rejection filter in case of the 64 Hz sample clock, and a 62.5-125ms rejection filter in case of the 16 Hz sample clock. -Such an event will set the EVx bit in the ERSTATUS register on the next rising edge of the 1 Hz clock.

If an event occurs, a timestamp will be taken from the RTC and stored in the ERLASTSTAMPx register. This timestamp will be updated with every new event. The event will also update the ERFIRSTSTAMPx register if this is the first event to occur since the last time the EVx bit in the status register was cleared.

In addition to taking the timestamp(s), a 3-bit counter (ERCOUNTERx) will be incremented on the rising edge of the 1 Hz clock (i.e. coincident with the ERLASTSTAMPx register being updated). The counter stops counting and holds when it reaches a count of seven. It will be cleared automatically when the software clears the EVx bit in the status register.

An event can be enabled to clear the backup registers in the RTC block asynchronously. This works even when the 32 kHz oscillator is not running or when Event Monitor/Recorder clocks are disabled.

The following figure shows a block diagram of the Event Monitor/Recorder.

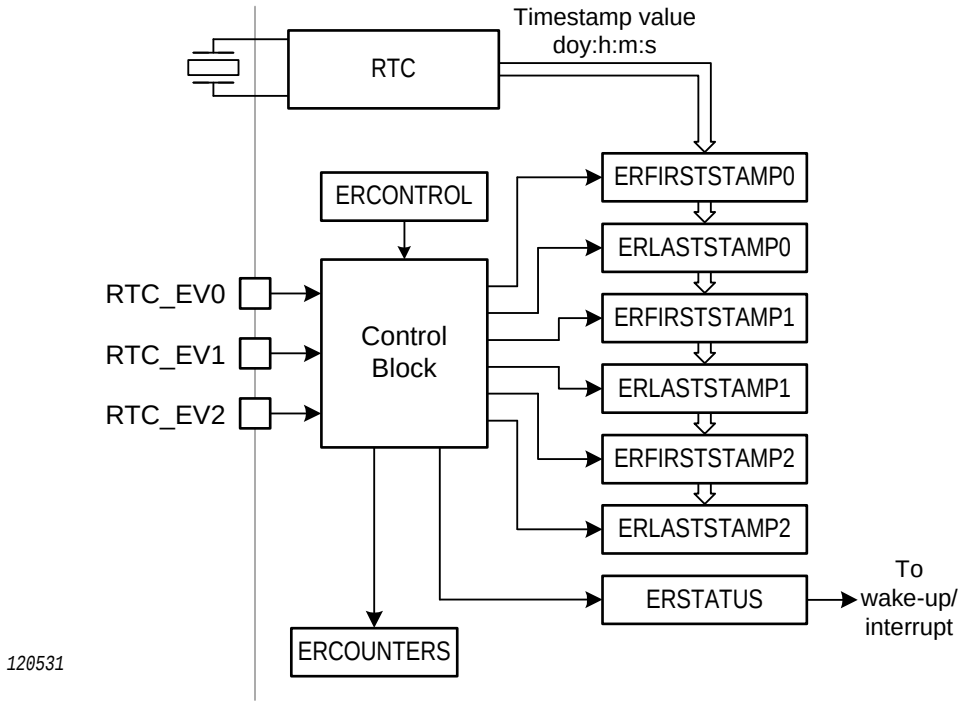


Fig 158. Event Monitor/Recorder block diagram

The CPU may at any time check the ERSTATUS register for events. If, for instance the EV0 bit is set, the corresponding ERFIRSTSTAMP0 and ERLASTSTAMP0 registers contain valid timestamps. The ERCOUNTER0 will also contain a valid count of the total number of events on channel 0 (up to a maximum of seven).

Once the (private) timestamps have been read, the CPU can clear the ERSTATUS.EVx bits by writing a 1 to it.

The CPU should ignore the timestamp registers if the ERSTATUS.EVx bit is cleared. There is no mechanism to clear or invalidate the timestamps after the event flag in the status register has been cleared. The timestamp registers will keep their old values until a new qualified event updates them. Such a qualified event will set the ERSTATUS.EVx bit and inform the CPU that the timestamp registers contain new values.

An event channel can be qualified as a wake-up trigger signal by setting the INTWAKE_ENAx bit in the ERCONTROL register. An event in that channel will then wake up the device from a power saving mode.

30.5 Pin description

Table 661. Event Monitor/Recorder pin description

Name	Type	Description
RTC_EV0	Input	Event input for Event Monitor/Recorder channel 0.
RTC_EV1	Input	Event input for Event Monitor/Recorder channel 1.
RTC_EV2	Input	Event input for Event Monitor/Recorder channel 2.

30.6 Register description

In the register descriptions, the Reset Value shown applies only to a power-up of the RTC domain, meaning that these registers are Not Changed by a device Reset.

Table 662. Register overview: Event Monitor/Recorder (base address 0x4002 4000)

Name	Access	Address	Description	Value[1]	Table
ERCONTROL	R/W	0x084	Event Monitor/Recorder Control register. Contains bits that control actions for the event channels as well as for Event Monitor/Recorder setup.	0	663
ERSTATUS	R/W	0x080	Event Monitor/Recorder Status register. Contains status flags for event channels and other Event Monitor/Recorder conditions.	0	664
ERCOUNTERS	RO	0x088	Event Monitor/Recorder Counters register. Allows reading the counters associated with the event channels.	0	665
ERFIRSTSTAMP0	RO	0x090	Event Monitor/Recorder First Stamp register for channel 0. Retains the time stamp for the first event on channel 0.	NA	666
ERFIRSTSTAMP1	RO	0x094	Event Monitor/Recorder First Stamp register for channel 1 (see ERFIRSTSTAMP0 description).	NA	666
ERFIRSTSTAMP2	RO	0x098	Event Monitor/Recorder First Stamp register for channel 2 (see ERFIRSTSTAMP0 description).	NA	666
ERLASTSTAMP0	RO	0x0A0	Event Monitor/Recorder Last Stamp register for channel 0. Retains the time stamp for the last (i.e. most recent) event on channel 0.	NA	667
ERLASTSTAMP1	RO	0x0A4	Event Monitor/Recorder Last Stamp register for channel 1 (see ERLASTSTAMP0 description).	NA	667
ERLASTSTAMP2	RO	0x0A8	Event Monitor/Recorder Last Stamp register for channel 2 (see ERLASTSTAMP0 description).	NA	667

[1] Reset values apply only to a power-up of the Event Monitor/Recorder block, other types of reset have no effect on this block. Since the Event Monitor/Recorder is powered whenever either of the $V_{DD(REG)(3V3)}$, or V_{BAT} supplies are present, power-up reset occurs only when both supplies were absent and then one is turned on. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

30.6.1 Event Monitor/Recorder Control Register

The Event Monitor/Recorder Control Register allows setup of the Event Monitor/Recorder and individual control over aspects of each channel's operation.

Table 663. Event Monitor/Recorder Control Register (ERCONTROL - 0x4002 4084) bit description

Bit	Symbol	Value	Description	Reset value
0	INTWAKE_EN0		Interrupt and wake-up enable for channel 0.	0
		0	No interrupt or wake-up will be generated by event channel 0.	
		1	An event in channel 0 will trigger an (RTC) interrupt and a wake-up request.	
1	GPCLEAR_EN0		Enables automatically clearing the RTC general purpose registers when an event occurs on channel 0.	0
		0	Channel 0 has no influence on the general purpose registers.	
		1	An event in channel 0 will clear the general purpose registers asynchronously.	
2	POL0		Selects the polarity of an event on input pin RTC_EV0.	0
		0	A channel 0 event is defined as a negative edge on RTC_EV0.	
		1	A channel 0 event is defined as a positive edge on RTC_EV0.	
3	EV0_INPUT_EN		Event enable control for channel 0.[1]	0
		0	Event 0 input is disabled and forced high internally.	
		1	Event 0 input is enabled.	
9:4	-		Reserved. Read value is undefined, only zero should be written.	NA
10	INTWAKE_EN1		Interrupt and wake-up enable for channel 1.	0
		0	No interrupt or wake-up will be generated by event channel 1.	
		1	An event in channel 1 will trigger an (RTC) interrupt and a wake-up request.	
11	GPCLEAR_EN1		Enables automatically clearing the RTC general purpose registers when an event occurs on channel 1.	0
		0	Channel 1 has no influence on the general purpose registers.	
		1	A n event in channel 1 will clear the general purpose registers asynchronously.	
12	POL1		Selects the polarity of an event on input pin RTC_EV1.	0
		0	A channel 1 event is defined as a negative edge on RTC_EV1.	
		1	A channel 1 event is defined as a positive edge on RTC_EV1.	
13	EV1_INPUT_EN		Event enable control for channel 1.[1]	0
		0	Event 1 input is disabled and forced high internally.	
		1	Event 1 input is enabled.	
19:14	-		Reserved. Read value is undefined, only zero should be written.	NA
20	INTWAKE_EN2		Interrupt and wake-up enable for channel 2.	
		0	No interrupt or wake-up will be generated by event channel 2.	
		1	An event in channel 2 will trigger an (RTC) interrupt and a wake-up request.	
21	GPCLEAR_EN2		Enables automatically clearing the RTC general purpose registers when an event occurs on channel 2.	0
		0	Channel 2 has no influence on the general purpose registers.	
		1	An event in channel 2 will clear the general purpose registers asynchronously.	

Table 663. Event Monitor/Recorder Control Register (ERCONTROL - 0x4002 4084) bit description

Bit	Symbol	Value	Description	Reset value
22	POL2		Selects the polarity of an event on input pin RTC_EV2.	0
		0	A channel 2 event is defined as a negative edge on RTC_EV2.	
		1	A channel 2 event is defined as a positive edge on RTC_EV2.	
23	EV2_INPUT_EN		Event enable control for channel 2.[1]	0
		0	Event 2 input is disabled and forced high internally.	
		1	Event 2 input is enabled.	
29:24	-		Reserved. Read value is undefined, only zero should be written.	NA
31:30	ERMODE		Controls enabling the Event Monitor/Recorder and selecting its operating frequency.[2]	0
		00	Event Monitor/Recorder clocks are disabled. Operation of the Event Monitor/Recorder is disabled except for asynchronous clearing of GP registers if selected.	
		01	Enable Event Monitor/Recorder and select a 16 Hz sample clock for event input edge detection and glitch suppression. Pulses (in either direction) shorter than 62.5 ms to 125 ms will be filtered out.	
		10	Enable Event Monitor/Recorder and select a 64 Hz sample clock for event input edge detection and glitch suppression. Pulses (in either direction) shorter than 15.6 ms to 31.2 ms will be filtered out.	
		11	Enable Event Monitor/Recorder and select a 1 kHz sample clock for event input edge detection and glitch suppression. Pulses (in either direction) shorter than 1 ms to 2 ms will be filtered out.	

[1] Event Inputs should remain DISABLED when not being used for event detection, particularly if the associated pin is being used for some other function.

[2] Event Monitor/Recorder registers can always be written to regardless of the state of these bits. Events occurring during the 1-sec interval immediately following enabling of the clocks may not be recognized.

30.6.2 Event Monitor/Recorder Status Register

The Event Monitor/Recorder Status Register contains flags for the 3 event channels, general purpose register clear flag, and the interrupt/wake-up flag.

Table 664. Event Monitor/Recorder Status Register (ERSTATUS - 0x4002 4080) bit description

Bit	Symbol	Value	Description	Reset value
0	EV0		Event flag for channel 0 (RTC_EV0 pin). Set at the end of any second if there has been an event during the preceding second. This bit is cleared by writing a 1 to it. Writing 0 has no effect.	0
		0	No event change on channel 0.	
		1	At least one event has occurred on channel 0.	
1	EV1		Event flag for channel 1 (RTC_EV1 pin). Set at the end of any second if there has been an event during the preceding second. This bit is cleared by writing a 1 to it. Writing 0 has no effect.	0
		0	No event change on channel 1.	
		1	At least one event has occurred on channel 1.	
2	EV2		Event flag for channel 2 (RTC_EV2 pin). Set at the end of any second if there has been an event during the preceding second. This bit is cleared by writing a 1 to it. Writing 0 has no effect.	0
		0	No event change on channel 2.	
		1	At least one event has occurred on channel 2.	
3	GP_CLEARED		General purpose register asynchronous clear flag. This bit is cleared by writing a 1 to it. Writing 0 has no effect.	0
		0	General purpose registers have not been asynchronous cleared.	
		1	General purpose registers have been asynchronous cleared.	
30:4	-		Reserved. Read value is undefined, only zero should be written.	NA
31	WAKEUP		Interrupt/wake-up request flag (Read-only). This bit is cleared by writing a 1 to it. Writing 0 has no effect.	0
		0	No interrupt/wake-up request is pending	
		1	An interrupt/wake-up request is pending.	

30.6.3 Event Monitor/Recorder Counters Register

The Event Monitor/Recorder Counters Register is a read-only register that allows reading counters that record the number of events on each Event Monitor/Recorder channel.

Table 665. Event Monitor/Recorder Counters Register (ERCOUNTERS - 0x4002 4088) bit description

Bit	Symbol	Description	Reset value
2:0	COUNTER0	Value of the counter for event 0. If the counter reaches full count (the value 7), it remains there if additional events occur. This counter is cleared when the corresponding EVx bit in the ERSTATUS register is cleared by software.	0
7:3	-	Reserved. The value read from a reserved bit is not defined.	NA
10:8	COUNTER1	Value of the counter for event 1. See description for COUNTER0.	0
15:11	-	Reserved. The value read from a reserved bit is not defined.	NA
18:16	COUNTER2	Value of the counter for event 2. See description for COUNTER0.	0
31:19	-	Reserved. The value read from a reserved bit is not defined.	NA

30.6.4 Event Monitor/Recorder First Stamp Register

The read-only Event Monitor/Recorder First Stamp Registers record a timestamp (from the RTC) of the first event that occurs on each Event Monitor/Recorder channel. This is when the corresponding EVx bit in the ERSTATUS register becomes set. Once that has happened, these registers will not change until software clears the corresponding EVx bit in the ERSTATUS register.

Contents of these register are only valid if the corresponding EVx bit in the ERSTATUS register = 1.

Table 666. Event Monitor/Recorder First Stamp Register (ERFIRSTSTAMP0 - 0x0x4002 4090, ERFIRSTSTAMP1 - 0x0x4002 4094, ERFIRSTSTAMP2 - 0x4002 4098) bit description

Bit	Symbol	Description	Reset value
5:0	SEC	Seconds value in the range of 0 to 59.	NA
11:6	MIN	Minutes value in the range of 0 to 59.	NA
16:12	HOUR	Hours value in the range of 0 to 23.	NA
25:17	DOY	Day of Year value in the range of 1 to 366.	NA
31:26	-	Reserved. The value read from a reserved bit is not defined.	NA

30.6.5 Event Monitor/Recorder Last Stamp Register

The read-only Event Monitor/Recorder Last Stamp Registers record a timestamp (from the RTC) whenever an event occurs on each Event Monitor/Recorder channel.

Contents of these register are only valid if the corresponding EVx bit in the ERSTATUS register = 1.

Note that after a first event on any channel, the contents of the corresponding ERFIRSTSTAMP and ERLASTSTAMP registers will be the same (the first event and the most recent event being the same). The values will diverge if a second event occurs on the same channel

Table 667. Event Monitor/Recorder Last Stamp Register (ERLASTSTAMP0 - 0x0x4002 40A0, ERLASTSTAMP1 - 0x0x4002 40A4, ERLASTSTAMP2 - 0x4002 40A8) bit description

Bit	Symbol	Description	Reset value
5:0	SEC	Seconds value in the range of 0 to 59.	NA
11:6	MIN	Minutes value in the range of 0 to 59.	NA
16:12	HOUR	Hours value in the range of 0 to 23.	NA
25:17	DOY	Day of Year value in the range of 1 to 366.	NA
31:26	-	Reserved. The value read from a reserved bit is not defined.	NA

31.1 Features

- Internally resets chip if not reloaded during the programmable timeout period.
- Optional windowed operation requires reload to occur between a minimum and maximum timeout period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog timeout.
- 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.
- A dedicated on-chip watchdog oscillator provides a reliable clock source that cannot be turned off when the Watchdog Timer is running.
- Incorrect feed sequence causes immediate watchdog reset if the watchdog is 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.

31.2 Applications

The purpose of the Watchdog Timer is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state. When enabled, a watchdog event will be generated if the user program fails to "feed" (or reload) the Watchdog within a predetermined amount of time. The Watchdog event will cause a chip reset if configured to do so.

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.

For interaction of the on-chip watchdog and other peripherals, especially the reset and boot-up procedures, please refer to Section 3.4.

31.3 Description

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:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup the Watchdog timer operating mode in WDMOD register.
- Set a value for the watchdog window time in WDWINDOW register if windowed operation is required.
- Set a value for the watchdog warning interrupt in the WDWARNINT register if a warning interrupt is required.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the WDFEED 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 in the case of 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 matches the value defined by the WDWARNINT register.

The watchdog timer block uses two clocks: PCLK and WDCLK. PCLK is used for the APB accesses to the watchdog registers. WDCLK runs the watchdog timer counter. This clock comes for a dedicated oscillator that is always on when the Watchdog Timer is enabled. This oscillator has a typical frequency of 500 kHz (see Section 3.8.4).

There is some synchronization logic between these two clock domains. When the WDMOD and WDTC registers are updated by APB operations, the new value will take effect on the logic in the WDCLK clock domain in 3 WDCLK cycles. When the watchdog timer is counting, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the PCLK when the WDTV register is read by the CPU.

The block diagram of the Watchdog is shown below in the Figure 159. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

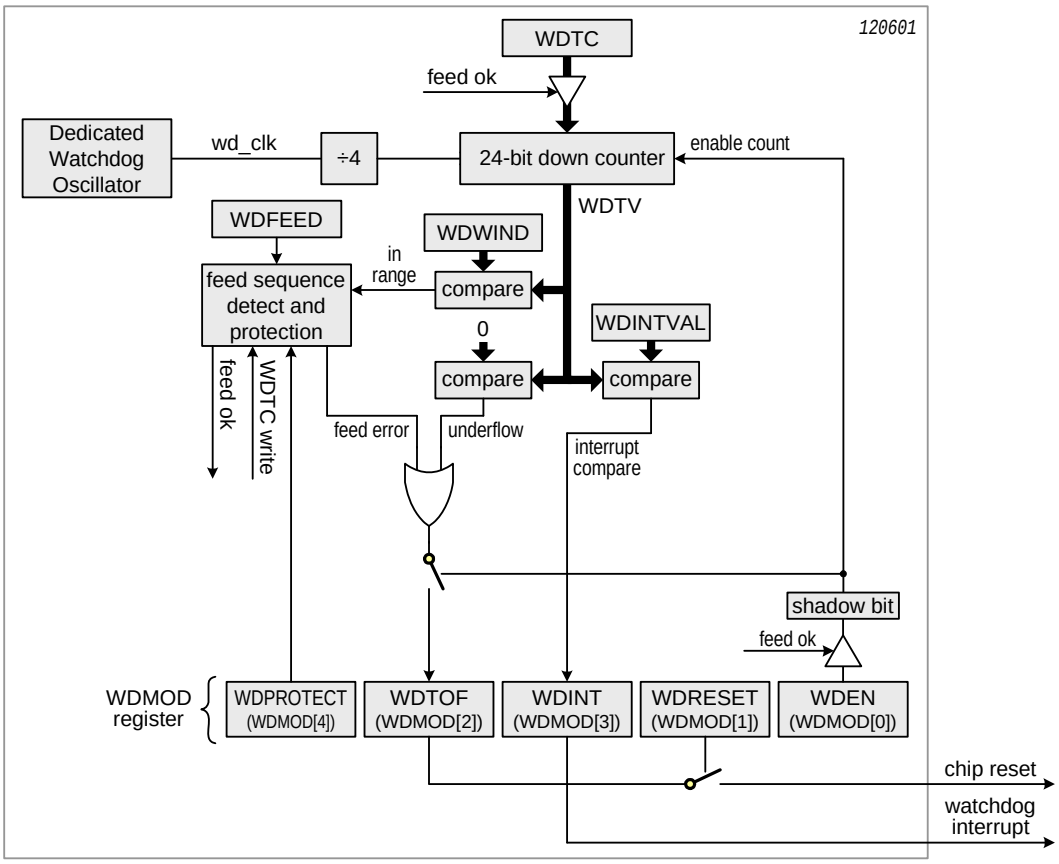


Fig 159. Watchdog Timer block diagram

31.4 Register description

Table 668. Register overview: Watchdog (base address 0x4000 0000)

Name	Access	Address	Description	Reset Value[1]	Table
MOD	R/W	0x000	Watchdog mode register. This register determines the basic mode and status of the Watchdog Timer.	0	669
TC	R/W	0x004	Watchdog timer constant register. The value in this register determines the time-out value.	0xFF	671
FEED	WO	0x008	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	NA	672
TV	RO	0x00C	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	0xFF	673
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0	674
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF	675

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

31.4.1 Watchdog Mode register

The WDMOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 669: Watchdog Mode register (MOD - 0x4000 0000) bit description

Bit	Symbol	Value	Description	Reset Value
0	WDEN		Watchdog enable bit. This bit is Set Only. See Table 670.	0
		0	The watchdog timer is stopped.	
		1	The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. This bit is Set Only. See Table 670.	0
		0	A watchdog timeout will not cause a chip reset.	
		1	A watchdog timeout 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. Causes a chip reset if WDRESET = 1. See Section "WDTOF".	0 (Only after external reset)
3	WDINT		Watchdog interrupt flag. Set when the timer reaches the value in WDWARNINT. Cleared by software. See Section "WDINT".	0
4	WDPROTECT		Watchdog update mode. This bit is Set Only. See Section "WDPROTECT".	0
		0	The watchdog reload value (WDTC) can be changed at any time.	
		1	The watchdog reload value (WDTC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW. **Note**: this mode is intended for use only when WDRESET =1.	
7:5	-		Reserved. Read value is undefined, only zero should be written.	NA

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. These flags are cleared by an external reset or a Watchdog timer reset.

Watchdog reset or interrupt will occur any time the watchdog is running. If a watchdog interrupt occurs in Sleep or Deep Sleep mode, it will wake up the device.

WDTOF

The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when WDPROTECT =1 and an attempt is made to write to the WDTC register. This flag is cleared by software writing a 0 to this bit.

WDINT

The Watchdog interrupt flag is set when the Watchdog counter reaches the value specified by WDWARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 1 to this bit.

WDPROTECT

This provides additional protection by essentially only allowing the watchdog reload value to be changed during the interrupt service of the Watchdog. The Watchdog must be running (set up and the first feed performed) before WDPROTECT is set.

Table 670. 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.

31.4.2 Watchdog Timer Constant register

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. This is pre-loaded with the value 0x00 00FF upon reset. Writing values below 0xFF will cause 0x00 00FF to be loaded into the WDTC. 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 WDTC before the watchdog counter is below the values of WDWARNINT and WDWINDOW will cause a watchdog reset and set the WDTOF flag.

Table 671: Watchdog Timer Constant register (TC - address 0x4000 0004) bit description

Bit	Symbol	Description	Reset Value
23:0	Count	Watchdog time-out interval.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

31.4.3 Watchdog Feed register

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the WDTC 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 PCLK following an incorrect access to a Watchdog register during a feed sequence.

Table 672: Watchdog Feed register (FEED - address 0x4000 0008) bit description

Bit	Symbol	Description
7:0	Feed	Feed value should be 0xAA followed by 0x55.

31.4.4 Watchdog Timer Value register

The WDTV 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 PCLK cycles, so the value of WDTV is older than the actual value of the timer when it's being read by the CPU.

Table 673: Watchdog Timer Value register (TV - address 0x4000 000C) 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.	NA

31.4.5 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 matches the value defined by WDWARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WDWARNINT 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 set to 0, the interrupt will occur at the same time as the watchdog event.

Table 674: Watchdog Timer Warning Interrupt register (WARNINT - address 0x4000 0014) bit description

Bit	Symbol	Description	Reset Value
9:0	WARNINT	Watchdog warning interrupt compare value.	0
31:10	-	Reserved. Read value is undefined, only zero should be written.	NA

31.4.6 Watchdog Timer Window register

The WDWINDOW register determines the highest WDTV value allowed when a watchdog feed is performed. If a feed valid sequence completes prior to WDTV reaching the value in WDWINDOW, a watchdog event will occur.

WDWINDOW resets to the maximum possible WDTV value, so windowing is not in effect.

Table 675: Watchdog Timer Window register (WINDOW - address 0x4000 0018) bit description

Bit	Symbol	Description	Reset Value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

31.5 Watchdog timing examples

The following figures illustrate several aspects of Watchdog Timer operation is shown below in the Figure 160.

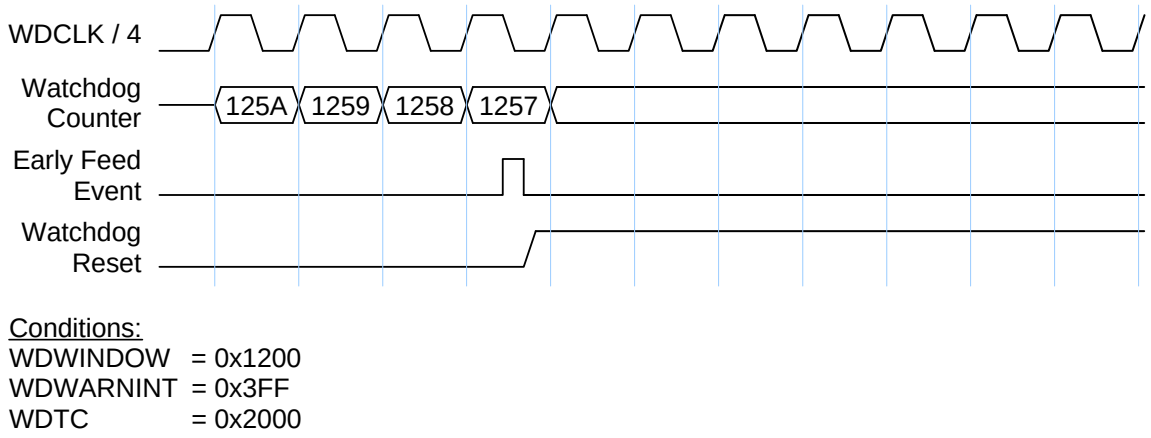


Fig 160. Early Watchdog Feed with Windowed Mode Enabled

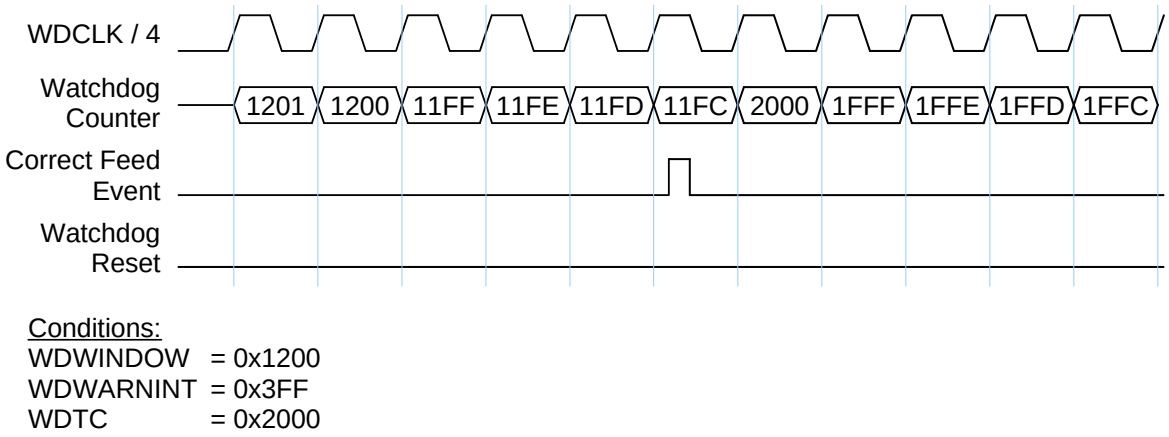


Fig 161. Correct Watchdog Feed with Windowed Mode Enabled

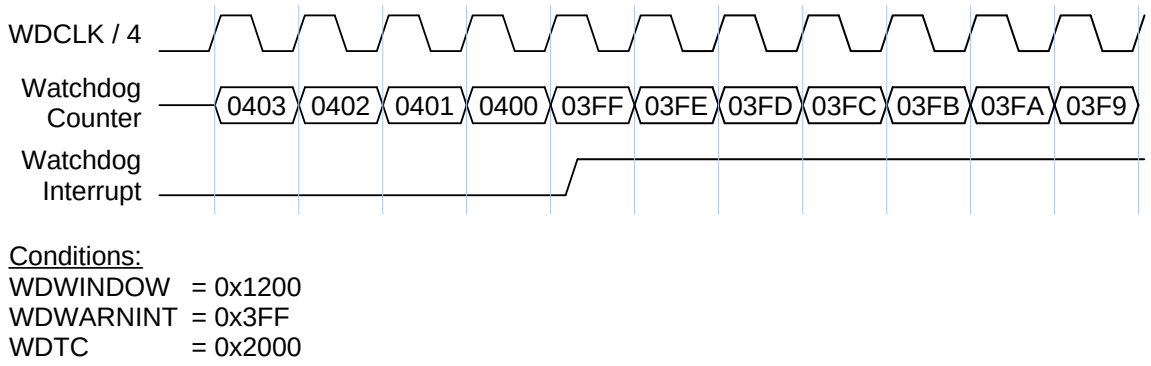


Fig 162. Watchdog Warning Interrupt

32.1 Basic configuration

The ADC is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set the PCADC bit.

   Remark: On reset, the ADC is disabled. To enable the ADC, first set the PCADC bit, and then enable the ADC in the AD0CR register (bit PDN Table 678). To disable the ADC, first clear the PDN bit, and then clear the PCADC bit.
2. Peripheral clock: The ADC operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5. To scale the clock for the ADC, see bits CLKDIV in Table 678.
3. Pins: Enable ADC0 pins and pin modes for the port pins with ADC0 functions through the relevant IOCON registers (Section 7.4.1).
4. Interrupts: To enable interrupts in the ADC, see Table 682. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. Disable the ADC interrupt in the NVIC using the appropriate Interrupt Set Enable register.
5. DMA: See Section 32.6.4. For GPDMA system connections, see Table 696.

32.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among 8 pins.
- Power-down mode.
- Measurement range V_{SS} to V_{REFP} (typically 3 V; not to exceed V_{DDA} voltage level).
- 12-bit conversion rate of 400 kHz.
- Burst conversion mode for single or multiple inputs.
- Optional conversion on transition on input pin or Timer Match signal.

32.3 Description

Basic clocking for the A/D converters is provided by the APB clock. A programmable divider is included in each converter to scale this clock to the clock (maximum 12.4 MHz) needed by the successive approximation process. A fully accurate conversion requires 31 of these clocks.

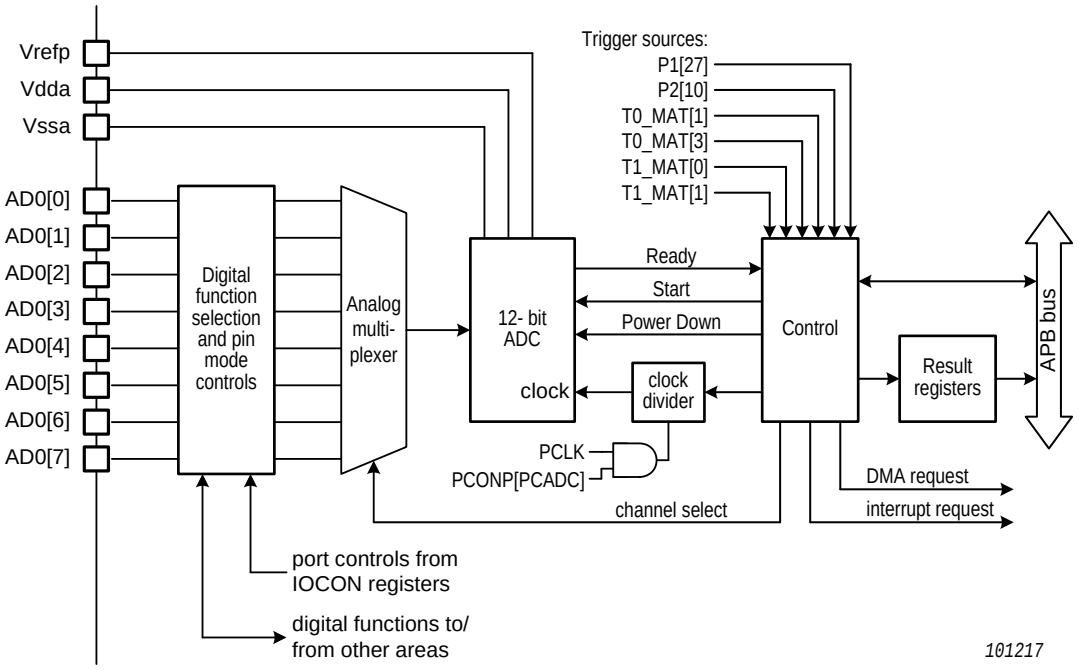


Fig 163. ADC block diagram

32.4 Pin description

Table 676 gives a brief summary of each of ADC related pins.

Table 676. ADC pin description

Pin	Type	Description
AD0[7] to AD0[0]	Input	**Analog Inputs.** The ADC cell can measure the voltage on any of these input signals. Digital signals are disconnected from the ADC input pins when the ADC function is selected on that pin in the Pin Select register. **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, A/D converter readings will be invalid. If the A/D converter is not used in an application then the pins associated with A/D inputs can be used as 5 V tolerant digital IO pins.
V_{REFP}	Reference	**Voltage Reference.** This pin provides a voltage reference level for the ADC and DAC. **Note: V_{REFP} should be tied to VDD(3V3) if the ADC and DAC are not used.**
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. **Note: VDDA should be tied to VDD(3V3) and VSSA should be tied to VSS if the ADC and DAC are not used.**

32.5 Register description

Table 677. Register overview: ADC (base address 0x4003 4000)

Generic Name	Access	Address offset	Description	Reset value[1]	Table
CR	R/W	0x000	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	1	678
GDR	R/W	0x004	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	NA	679
INTEN	R/W	0x00C	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	0x100	680
DR0	RO	0x010	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	NA	681
DR1	RO	0x014	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	NA	681
DR2	RO	0x018	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	NA	681
DR3	RO	0x01C	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	NA	681
DR4	RO	0x020	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	NA	681
DR5	RO	0x024	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	NA	681
DR6	RO	0x028	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	NA	681
DR7	RO	0x2C	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA	681
STAT	RO	0x030	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt/DMA flag.	0	682
TRM	R/W	0x034	ADC trim register.	0	683

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

32.5.1 A/D Control Register

Table 678: A/D Control Register (CR - address 0x4003 4000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	SEL		Selects which of the AD0[7:0] pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0[0], and bit 7 selects pin AD0[7]. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones is allowed. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The APB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 12.4 MHz. Typically, software should program the smallest value in this field that yields a clock of 12.4 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST		Burst mode	0
		0	Conversions are software controlled and require 31 clocks.	
		1	The AD converter does repeated conversions at up to 400 kHz, scanning (if necessary) through the pins selected by bits set to ones in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. **Remark:** START bits must be 000 when BURST = 1 or conversions will not start.	
20:17	-		Reserved. Read value is undefined, only zero should be written.	NA
21	PDN		Power down mode	0
		0	The A/D converter is in power-down mode.	
		1	The A/D converter is operational.	
23:22	-		Reserved. Read value is undefined, only zero should be written.	NA
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
		0x0	No start (this value should be used when clearing PDN to 0).	
		0x1	Start conversion now.	
		0x2	Start conversion when the edge selected by bit 27 occurs on the P2[10] pin.	
		0x3	Start conversion when the edge selected by bit 27 occurs on the P1[27] pin.	
		0x4	Start conversion when the edge selected by bit 27 occurs on MAT0.1. Note that this does not require that the MAT0.1 function appear on a device pin.	
		0x5	Start conversion when the edge selected by bit 27 occurs on MAT0.3. Note that it is not possible to cause the MAT0.3 function to appear on a device pin.	
		0x6	Start conversion when the edge selected by bit 27 occurs on MAT1.0. Note that this does not require that the MAT1.0 function appear on a device pin.	
		0x7	Start conversion when the edge selected by bit 27 occurs on MAT1.1. Note that this does not require that the MAT1.1 function appear on a device pin.	
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved. Read value is undefined, only zero should be written.	NA

32.5.2 A/D Global Data Register

The A/D Global Data Register holds the result of the most recent A/D conversion that has completed, and also includes copies of the status flags that go with that conversion.

Results of ADC conversion can be read in one of two ways. One is to use the A/D Global Data Register to read all data from the ADC. Another is to use the A/D Channel Data Registers. It is important to use one method consistently because the DONE and OVERRUN flags can otherwise get out of synch between the AD0GDR and the A/D Channel Data Registers, potentially causing erroneous interrupts or DMA activity.

Table 679: A/D Global Data Register (GDR - address 0x4003 4004) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	NA
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin selected by the SEL field, as it falls within the range of V_{REFP} to V_{SS}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{SS}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	NA
23:16	-	Reserved. Read value is undefined, only zero should be written.	NA
26:24	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 000 identifies channel 0, 001 channel 1...).	NA
29:27	-	Reserved. Read value is undefined, only zero should be written.	NA
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

32.5.3 A/D Interrupt Enable register

This register allows control over which A/D channels generate an interrupt when a conversion is complete. For example, it may be desirable to use some A/D channels to monitor sensors by continuously performing conversions on them. The most recent results are read by the application program whenever they are needed. In this case, an interrupt is not desirable at the end of each conversion for some A/D channels.

Table 680: A/D Interrupt Enable register (INTEN - address 0x4003 400C) bit description

Bit	Symbol	Value	Description	Reset value
0	ADINTEN0		Interrupt enable	0
		0	Completion of a conversion on ADC channel 0 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 0 will generate an interrupt.	
1	ADINTEN1		Interrupt enable	0
		0	Completion of a conversion on ADC channel 1 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 1 will generate an interrupt.	
2	ADINTEN2		Interrupt enable	0
		0	Completion of a conversion on ADC channel 2 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 2 will generate an interrupt.	
3	ADINTEN3		Interrupt enable	0
		0	Completion of a conversion on ADC channel 3 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 3 will generate an interrupt.	
4	ADINTEN4		Interrupt enable	0
		0	Completion of a conversion on ADC channel 4 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 4 will generate an interrupt.	
5	ADINTEN5		Interrupt enable	0
		0	Completion of a conversion on ADC channel 5 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 5 will generate an interrupt.	
6	ADINTEN6		Interrupt enable	0
		0	Completion of a conversion on ADC channel 6 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 6 will generate an interrupt.	
7	ADINTEN7		Interrupt enable	0
		0	Completion of a conversion on ADC channel 7 will not generate an interrupt.	
		1	Completion of a conversion on ADC channel 7 will generate an interrupt.	
8	ADGINTEN		Interrupt enable	1
		0	Only the individual ADC channels enabled by ADINTEN7:0 will generate interrupts.	
		1	The global DONE flag in ADDR is enabled to generate an interrupt in addition to any individual ADC channels that are enabled to generate interrupts.	
31:9	-		Reserved. Read value is undefined, only zero should be written.	NA

32.5.4 A/D Data Registers

The A/D Data Registers hold the result of the last conversion for each A/D channel, when an A/D conversion is complete. They also include the flags that indicate when a conversion has been completed and when a conversion overrun has occurred.

Results of ADC conversion can be read in one of two ways. One is to use the A/D Global Data Register to read all data from the ADC. Another is to use the A/D Channel Data Registers. It is important to use one method consistently because the DONE and OVERRUN flags can otherwise get out of synch between the AD0GDR and the A/D Channel Data Registers, potentially causing erroneous interrupts or DMA activity.

Table 681: A/D Data Registers (DR[0:7] - addresses 0x4003 4010 (DR0) to 0x4003 402C (DR7)) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	NA
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{SS}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{SS}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	NA
29:16	-	Reserved. Read value is undefined, only zero should be written.	NA
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits.This bit is cleared by reading this register.	
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.	NA

32.5.5 A/D Status register

The A/D Status register allows checking the status of all A/D channels simultaneously. The DONE and OVERRUN flags appearing in the ADDRn register for each A/D channel are mirrored in ADSTAT. The interrupt flag (the logical OR of all DONE flags) is also found in ADSTAT.

Table 682: A/D Status register (STAT - address 0x4003 4030) bit description

Bit	Symbol	Description	Reset value
0	DONE0	This bit mirrors the DONE status flag from the result register for A/D channel 0.	0
1	DONE1	This bit mirrors the DONE status flag from the result register for A/D channel 1.	0
2	DONE2	This bit mirrors the DONE status flag from the result register for A/D channel 2.	0
3	DONE3	This bit mirrors the DONE status flag from the result register for A/D channel 3.	0
4	DONE4	This bit mirrors the DONE status flag from the result register for A/D channel 4.	0
5	DONE5	This bit mirrors the DONE status flag from the result register for A/D channel 5.	0
6	DONE6	This bit mirrors the DONE status flag from the result register for A/D channel 6.	0
7	DONE7	This bit mirrors the DONE status flag from the result register for A/D channel 7.	0
8	OVERRUN0	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 0.	0
9	OVERRUN1	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 1.	0
10	OVERRUN2	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 2.	0
11	OVERRUN3	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 3.	0
12	OVERRUN4	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 4.	0
13	OVERRUN5	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 5.	0
14	OVERRUN6	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 6.	0
15	OVERRUN7	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 7.	0
16	ADINT	This bit is the A/D interrupt flag. It is one when any of the individual A/D channel Done flags is asserted and enabled to contribute to the A/D interrupt via the ADINTEN register.	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	NA

32.5.6 A/D Trim register

This register will be set by the bootcode on start-up. It contains the trim values for the DAC and the ADC. The offset trim values for the ADC can be overwritten by the user. All 12 bits are visible when this register is read.

Table 683: A/D Trim register (TRM - address 0x4003 4034) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	NA
7:4	ADCOFFS	Offset trim bits for ADC operation. Initialized by the boot code. Can be overwritten by the user.	0
11:8	TRIM	written-to by boot code. Can **not** be overwritten by the user. These bits are locked after boot code write.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

32.6 Operation

Once an ADC conversion is started, it cannot be interrupted. A new software write to launch a new conversion or a new edge-trigger event will be ignored while the previous conversion is in progress.

32.6.1 Hardware-triggered conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the ADC will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

32.6.2 Interrupts

An interrupt request is asserted to the NVIC when the DONE bit is 1. Software can use the Interrupt Enable bit for the A/D Converter in the NVIC to control whether this assertion results in an interrupt. DONE is negated when the ADDR is read.

32.6.3 Accuracy vs. digital receiver

The ADC function must be selected via the ADMODE bit in the related IOCON registers in order to obtain voltage readings on the monitored pin. The same IOCON registers should also be set to the mode for which neither pull-up nor pull-down resistor is enabled. For a pin hosting an ADC input, it is not possible to have a have a digital function selected and yet get valid ADC readings, the analog input is disabled when a digital function is selected on the pin.

32.6.4 DMA control

A DMA transfer request is generated from the ADC interrupt request line. To generate a DMA transfer the same conditions must be met as the conditions for generating an interrupt (see Section 32.6.2 and Section 32.5.3).

Remark: If the DMA is used, the ADC interrupt must be disabled in the NVIC.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register (see Section 35.5.19). 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 - see Section 35.5.19), 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 (see Section 35.5.19). Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists (see Section 35.5.18).

33.1 Basic configuration

The DAC is configured using the following registers:

1. Power: The DAC is always connected to V_{DDA}. Register access is determined by IOCON register settings (see below).
2. Peripheral clock: The DAC operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Enable the DAC pin and select the pin mode for DACOUT through the relevant IOCON register (Section 7.4.1). This must be done before accessing any DAC registers.
4. DMA: The DAC can be connected to the GPDMA controller (see Section 33.5.2). For GPDMA connections, see Table 696.

33.2 Features

- 10-bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power-down mode
- Selectable speed vs. power
- Maximum update rate of 1 MHz.

33.3 Architecture

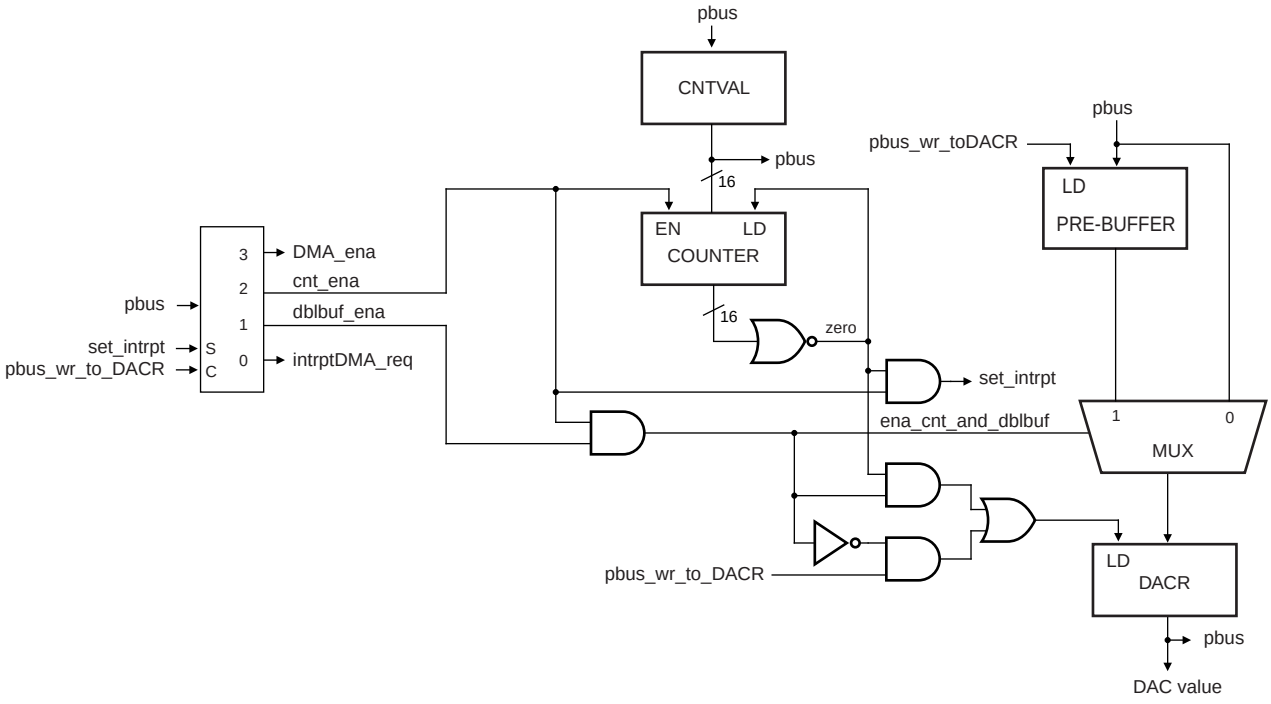


Fig 164. DAC control with DMA interrupt and timer

33.4 Pin description

Table 684 gives a brief summary of each of DAC related pins.

Table 684. D/A Pin Description

Pin	Type	Description
DAC_OUT	Output	**Analog Output.** After the selected settling time after the DACR is written with a new value, the voltage on this pin (with respect to V_{SSA}) is VALUE × V_{REFP})/1024. Note that DAC_OUT is disabled when the CPU is in Deep-sleep, Power-down, or Deep Power-down modes.
V_{REFP}	Reference	**Voltage Reference.** This pin provides a voltage reference level for the ADC and DAC. **Note: V_{REFP} should be tied to VDD(3V3) if the ADC and DAC are not used.**
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. **Note: VDDA should be tied to VDD(3V3) and VSSA should be tied to VSS if the ADC and DAC are not used.**

33.5 Register description

Note that the DAC does not have a control bit in the PCONP register. To enable the DAC, its output must be selected to appear on the related pin, P0[26], by configuring the relevant IOCON register (Section 7.4.1). See Section 7.4.1 “I/O configuration register contents (IOCON)”. the DAC must be enabled in this manner prior to accessing any DAC registers.

Table 685. Register overview: DAC (base address 0x4008 C000)

Name	Access	Address offset	Description	Reset value[1]	Table
CR	R/W	0x000	D/A Converter Register. This register contains the digital value to be converted to analog and a power control bit.	0	686
CTRL	R/W	0x004	DAC Control register. This register controls DMA and timer operation.	0	687
CNTVAL	R/W	0x008	DAC Counter Value register. This register contains the reload value for the DAC DMA/Interrupt timer.	0	688

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

33.5.1 D/A Converter Register

This read/write register includes the digital value to be converted to analog, and a bit that trades off performance vs. power. Bits 5:0 are reserved for future, higher-resolution D/A converters.

Table 686: D/A Converter Register (CR - address 0x4008 C000) bit description

Bit	Symbol	Value	Description	Reset Value
5:0	-		Reserved. Read value is undefined, only zero should be written.	NA
15:6	VALUE		After the selected settling time after this field is written with a new VALUE, the voltage on the DAC_OUT pin (with respect to V_{SSA}) is VALUE × V_{REFP}/1024.	0
16	BIAS		Settling time The settling times noted in the description of the BIAS bit are valid for a capacitance load on the DAC_OUT pin not exceeding 100 pF. A load impedance value greater than that value will cause settling time longer than the specified time. One or more graphs of load impedance vs. settling time will be included in the final data sheet.	0
		0	The settling time of the DAC is 1 μs max, and the maximum current is 700 μA. This allows a maximum update rate of 1 MHz.	
		1	The settling time of the DAC is 2.5 μs and the maximum current is 350 μA. This allows a maximum update rate of 400 kHz.	
31:17	-		Reserved. Read value is undefined, only zero should be written.	NA

33.5.2 D/A Converter Control register

This read/write register enables the DMA operation and controls the DMA timer.

Table 687. D/A Control register (CTRL - address 0x4008 C004) bit description

Bit	Symbol	Value	Description	Reset Value
0	INT_DMA_REQ		DMA interrupt request	0
		0	Clear on any write to the DACR register.	
		1	Set by hardware when the timer times out.	
1	DBLBUF_ENA		Double buffering	0
		0	Disable	
		1	Enable. When this bit and the CNT_ENA bit are both set, the double-buffering feature in the DACR register will be enabled. Writes to the DACR register are written to a pre-buffer and then transferred to the DACR on the next time-out of the counter.	
2	CNT_ENA		Time-out counter operation	0
		0	Disable	
		1	Enable	
3	DMA_ENA		DMA access	0
		0	Disable	
		1	Enable. DMA Burst Request Input 7 is enabled for the DAC (see Table 696).	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

33.5.3 D/A Converter Counter Value register

This read/write register contains the reload value for the Interrupt/DMA counter.

Table 688: D/A Converter Counter Value register (CNTVAL - address 0x4008 C008) bit description

Bit	Symbol	Description	Reset Value
15:0	VALUE	16-bit reload value for the DAC interrupt/DMA timer.	0
31:16	-	Reserved	-

33.6 Operation

33.6.1 DMA counter

When the counter enable bit CNT_ENA in DACCTRL is set, a 16-bit counter will begin counting down, at the rate selected by PCLK (see Section 3.3.3.5), from the value programmed into the DACCNTVAL register. The counter is decremented Each time the counter reaches zero, the counter will be reloaded by the value of DACCNTVAL and the DMA request bit INT_DMA_REQ will be set in hardware.

Note that the contents of the DACCTRL and DACCNTVAL registers are read and write accessible, but the timer itself is not accessible for either read or write.

If the DMA_ENA bit is set in the DACCTRL register, the DAC DMA request will be routed to the GPDMA. When the DMA_ENA bit is cleared, the default state after a reset, DAC DMA requests are blocked.

33.6.2 Double buffering

Double-buffering is enabled only if both, the CNT_ENA and the DBLBUF_ENA bits are set in DACCTRL. In this case, any write to the DACR register will only load the pre-buffer, which shares its register address with the DACR register. The DACR itself will be loaded from the pre-buffer whenever the counter reaches zero and the DMA request is set. At the same time the counter is reloaded with the COUNTVAL register value.

Reading the DACR register will only return the contents of the DACR register itself, not the contents of the pre-buffer register.

If either the CNT_ENA or the DBLBUF_ENA bits are 0, any writes to the DACR address will go directly to the DACR register.

34.1 How to read this chapter

Remark: The comparator function is not available on LPC4074 devices.

34.2 Basic configuration

The comparator block is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCCMP.

   Remark: On reset, the comparators are disabled (PCCMP = 0).
2. Peripheral clock: The comparator block operates from the common PCLK that clocks both the bus interface and functional portion of most APB peripherals. See Section 3.3.3.5.
3. Pins: Select comparator pins and pin modes through the relevant IOCON registers (Section 3.3.2.2).
4. Interrupts: If comparator interrupts are used, they must be configured in the CMP_CTRL0 and/or CMP_CTRL1 registers, see Table 692 and Table 693. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

34.3 Features

- Up to 5 selectable external sources per comparator; fully configurable on either positive or negative comparator input channels.
- 0.9 V internal bandgap reference voltage selectable as either positive or negative input on each comparator.
- 32-stage voltage ladder internal reference for selectable voltages on each comparator; configurable on either positive or negative comparator input.
- Voltage ladder source voltage is selectable from an external pin or the 3.3 V analog voltage supply.
- Voltage ladder can be separately powered down for applications only requiring the comparator function.
- Relaxation oscillator circuitry output, for a 555 style timer operation.
- Individual comparator outputs can be connected to I/O pins.
- Separate interrupt for each comparator.
- Edge and level comparator outputs connect to two timers allowing edge counting while a level match has been asserted, or measuring the time between 2 voltage trip points.

34.4 Architecture

Two embedded comparators are incorporated on-chip to compare the voltage levels on external pins or against internal voltages (see Figure 165). Up to 4 voltages on external pins and several internal reference voltages are selectable on each comparator. Additionally, two of the external inputs can be selected to drive an input common on both comparators.

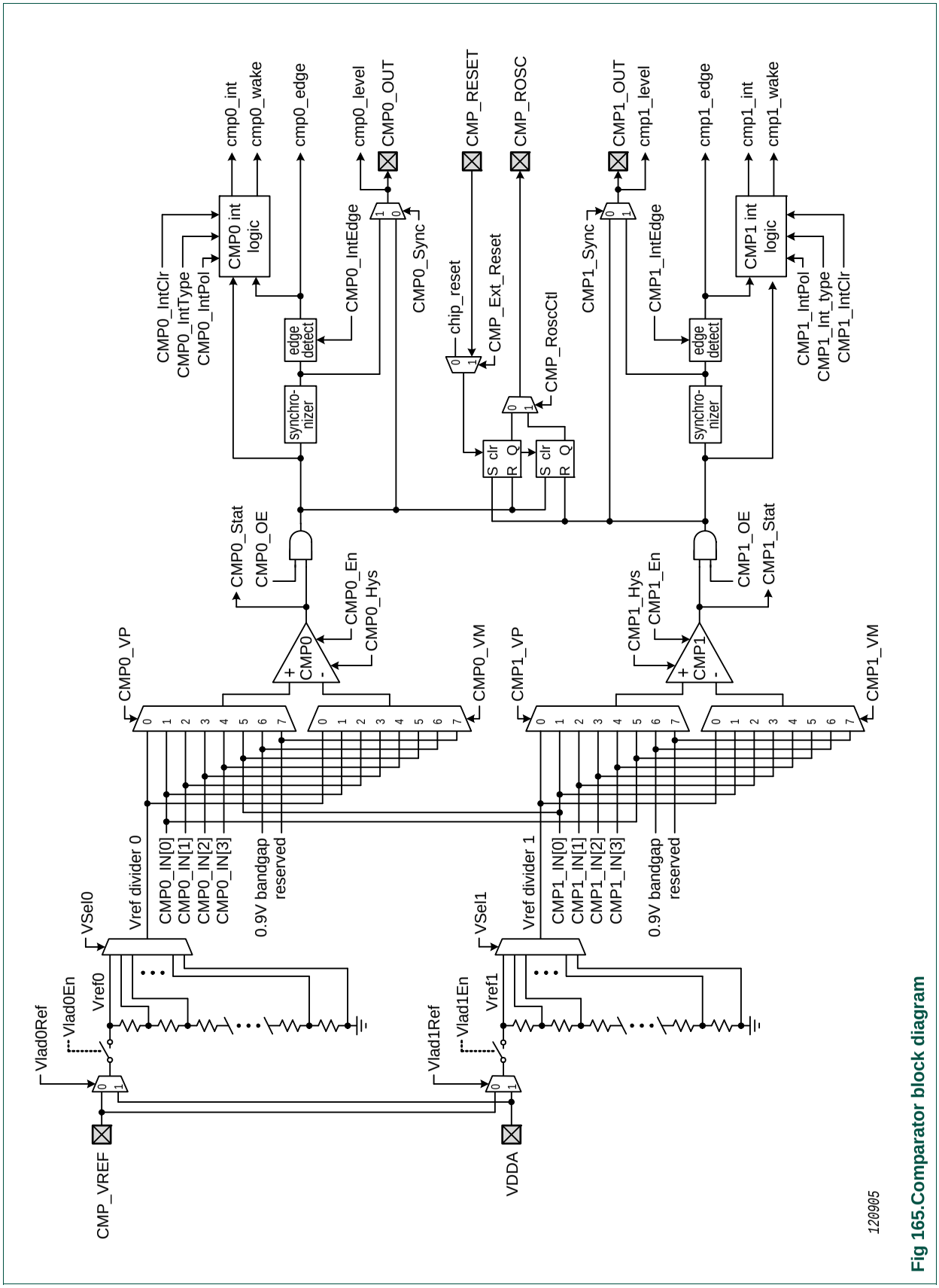


Fig 165.Comparator block diagram

34.5 Pin description

Table 689: Comparator pin description

Pin	Type	Description
CMP0_IN[3:0]	input	Comparator 0 input sources (CMP10IN[0] can also be selected as a Comparator 1 input)
CMP1_IN[3:0]	input	Comparator 1 input sources (CMP1_IN[0] can also be selected as a Comparator 0 input).
CMP0_OUT	output	Comparator 0 output
CMP1_OUT	output	Comparator 1 output
CMP_VREF	input	External reference voltage source for 32-stage Voltage Ladder
CMP_ROSC	output	Relaxation oscillator output, intended for 555 timer style applications
CMP_RESET	input	Reset to the CMP_ROSC generation logic.

34.6 Register description

Table 690. Register overview: Comparator (base address 0x4002 0000)

Name	Access	Address offset	Description	Reset value[1]	Table
CMP_CTRL	R/W	0x000	Comparator block control register	0	691
CMP_CTRL0	R/W	0x004	Comparator 0 control register	0	692
CMP_CTRL1	R/W	0x008	Comparator 1 control register	0	693

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

34.6.1 Comparator block control register

This register controls aspects of the comparator block that apply to both comparators.

Table 691. Comparator block control register (CMP_CTRL - address 0x4002 0000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CMP_PD_IREF		Controls the current source used by the comparators. The current source must be enabled when either comparator is used. When just enabled, time must be allowed for the current source to stabilize before all comparator functions will operate as expected.	00
		0x0	The comparator current source is disabled.	
		0x1	The comparator current source is disabled in Deep Sleep and Power-down modes and restored automatically when exiting those modes.	
		0x2	The comparator current source is disabled in Power-down mode and restored automatically when exiting Power-down.	
		0x3	The comparator current source is powered up.	
3:2	CMP_PD_VBG		Controls the band gap reference source that is used by the comparators. The reference must be enabled when used by either comparator.	00
		0x0	The comparator bandgap reference is disabled.	
		0x1	The comparator bandgap reference is disabled in Deep Sleep and Power-down modes and restored automatically when exiting those modes.	
		0x2	The comparator bandgap reference is disabled in Power-down mode and restored automatically when exiting Power-down.	
		0x3	The comparator bandgap reference is powered up.	
7:4	-		Reserved.	NA
8	CMP_ROSCCTL		Selects the inputs for the flip/flops that provide the CMP_ROSC output.	0
		0	The CMP_ROSC output is set by CMP1 and reset by CMP0.	
		1	The CMP_ROSC output is set by CMP0 and reset by CMP1.	
9	CMP_EXT_RESET		Selects the reset source for the CMP_ROSC output.	0
		0	The CMP_ROSC output is reset by the internal chip reset.	
		1	The CMP_ROSC output is reset by the CMP_RESET input.	
11:10	-		Reserved.	NA
12	CMP_T0CAP2		Selects the input for Timer 0 capture input 2.	0
		0	T0CAP2 is connected to comparator 0 level output.	
		1	T0CAP2 is connected to comparator 1 level output.	
13	CMP_T0CAP3		Selects the input for Timer 0 capture input 3.	0
		0	T0CAP3 is connected to comparator 0 edge output.	
		1	T0CAP3 is connected to comparator 1 edge output.	
14	CMP_T1CAP2		Selects the input for Timer 1 capture input 2.	0
		0	T1CAP2 is connected to comparator 1 edge output.	
		1	T1CAP2 is connected to comparator 0 level output.	
15	CMP_T1CAP3		Selects the input for Timer 1 capture input 3.	0
		0	T1CAP3 is connected to comparator 1 level output.	
		1	T1CAP3 is connected to comparator 0 edge output.	
31:16	-		Reserved.	NA

34.6.2 Comparator 0 control register

This register enables and configures many aspects of comparator 0 operation.

Table 692. Comparator 0 control register (CMP_CTRL0 - address 0x4002 0004) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CMP0_EN		Comparator 0 enable control.	00
		0x0	Comparator 0 disabled.	
		0x1	Comparator 0 is disabled in Deep Sleep and Power-down modes and re-enabled automatically when exiting those modes.	
		0x2	Comparator 0 is disabled in Power-down mode and re-enabled automatically when exiting Power-down.	
		0x3	Comparator 0 is enabled.	
2	CMP0_OE		Comparator 0 output enable. If Deep Sleep or Power-down mode is entered, this bit will be cleared when the device wakes up again.	0
		0	Comparator 0 output is disabled.	
		1	Comparator 0 output is enabled.	
3	CMP0_STAT		Comparator 0 status. This bit reflects the comparator 0 output, and is not affected by CMP0_OE.	0
6:4	CMP0_VM		Comparator 0 VM input select.	000
		0x0	Vref divider 0.	
		0x1	CMP0_IN[0].	
		0x2	CMP0_IN[1].	
		0x3	CMP0_IN[2].	
		0x4	CMP0_IN[3].	
		0x5	CMP1_IN[0].	
		0x6	internal 0.9 V band gap reference.	
		0x7	reserved.	
7	-		Reserved.	NA
10:8	CMP0_VP		Comparator 0 VP input select.	000
		0x0	Vref divider 0.	
		0x1	CMP0_IN[0].	
		0x2	CMP0_IN[1].	
		0x3	CMP0_IN[2].	
		0x4	CMP0_IN[3].	
		0x5	CMP1_IN[0].	
		0x6	internal 0.9 V band gap reference.	
		0x7	reserved.	
11	-		Reserved.	NA
12	CMP0_SYNC		Comparator 0 output synchronization control.	0
		0	The comparator 0 output is used directly.	
		1	The comparator 0 output is synchronized with the internal bus clock for output to other peripherals.	

Table 692. Comparator 0 control register (CMP_CTRL0 - address 0x4002 0004) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
14:13	CMP0_HYS		Comparator 0 hysteresis control. When enabled, hysteresis determines the difference required between the comparator inputs before the comparator output switches. The difference must be in the direction opposite of the current comparator output.	00
		0x0	Hysteresis is turned off, comparator output will change as the input voltages cross.	
		0x1	Hysteresis = 5 mV.	
		0x2	Hysteresis = 10 mV.	
		0x3	Hysteresis = 15 mV.	
15	CMP0_INTPOL		Selects the polarity of the CMP0 output for purposes of generating level interrupts. See Table 694.	0
		0	The CMP0 output is used as-is for generating interrupts.	
		1	The CMP0 output is used inverted for generating interrupts.	
16	CMP0_INTTYPE		Select comparator 0 interrupt type. See Table 694.	0
		0	Comparator 0 interrupt is edge triggered.	
		1	Comparator 0 interrupt is level triggered.	
18:17	CMP0_INTEDGE		Select edge triggered interrupt to be active on either high or low transitions, when CMP0_IntType = 0. See Table 694.	0
		0x0	Comparator 0 interrupt is active on falling edges.	
		0x1	Comparator 0 interrupt is active on rising edges.	
		0x2	Comparator 0 Interrupt is active on both edges.	
		0x3	reserved.	
19	CMP0_INTFLAG		Comparator 0 interrupt flag.	0
		0	The Comparator 0 interrupt is not pending.	
		1	The Comparator 0 interrupt is pending. Writing a 1 to this bit clears the flag.	
21:20	CMP0_VLADEN		Voltage ladder enable for comparator 0.	00
		0x0	The Comparator 0 voltage ladder is disabled.	
		0x1	The Comparator 0 voltage ladder is disabled in Deep Sleep and Power-down modes and re-enabled automatically when exiting those modes.	
		0x2	The Comparator 0 voltage ladder is disabled in Power-down mode and re-enabled automatically when exiting Power-down.	
		0x3	The Comparator 0 voltage ladder is enabled.	
22	CMP0_VLADREF		Voltage reference select for comparator 0 voltage ladder.	0
		0	VREF_CMP pin.	
		1	V_{DDA} pin.	

Table 692. Comparator 0 control register (CMP_CTRL0 - address 0x4002 0004) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
23	-		Reserved.	NA
28:24	CMP0_VSEL		Voltage ladder value for comparator 0. The reference voltage Vref depends on the setting of CMP0_VLADREF (either $V_{DD(3V3)}$ or voltage on pin VREF_CMP). 00000 = Vss. 00001 = 1 × Vref0 / 31. 00010 = 2 × Vref0 / 31. ... 11111 = Vref0	0x00
31:29	-		Reserved.	NA

34.6.3 Comparator 1 control register

This register enables and configures many aspects of comparator 1 operation.

Table 693. Comparator 1 control register (CMP_CTRL1 - 0x4002 0008) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CMP1_EN		Comparator 1 enable control.	00
		0x0	Comparator 1 disabled.	
		0x1	Comparator 1 is disabled in Deep Sleep and Power-down modes and re-enabled automatically when exiting those modes.	
		0x2	Comparator 1 is disabled in Power-down mode and re-enabled automatically when exiting Power-down.	
		0x3	Comparator 1 is enabled.	
2	CMP1_OE		Comparator 1 output enable. If Deep Sleep or Power-down mode is entered, this bit will be cleared when the device wakes up again.	0
		0	Comparator 1 output is disabled.	
		1	Comparator 1 output is enabled.	
3	CMP1_STAT		Comparator 1 status. This bit reflects the comparator 1 output, and is not affected by CMP1_OE.	0
6:4	CMP1_VM		Comparator 1 VM input select.	000
		0x0	Vref divider 1.	
		0x1	CMP1_IN[0].	
		0x2	CMP1_IN[1].	
		0x3	CMP1_IN[2].	
		0x4	CMP1_IN[3].	
		0x5	CMP0_IN[0].	
		0x6	internal 0.9 V band gap reference.	
		0x7	reserved.	
7	-		Reserved.	NA
10:8	CMP1_VP		Comparator 1 VP input select.	000
		0x0	Vref divider 0.	
		0x1	CMP1_IN[0].	
		0x2	CMP1_IN[1].	
		0x3	CMP1_IN[2].	
		0x4	CMP1_IN[3].	
		0x5	CMP0_IN[0].	
		0x6	internal 0.9 V band gap reference.	
		0x7	reserved.	
11	-		Reserved.	NA
12	CMP1_SYNC		Comparator 1 output synchronization control.	0
		0	The comparator 1 output is used directly.	
		1	The comparator 1 output is synchronized with the internal bus clock for output to other peripherals.	

Table 693. Comparator 1 control register (CMP_CTRL1 - 0x4002 0008) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
14:13	CMP1_HYS		Comparator 1 hysteresis control. When enabled, hysteresis determines the difference required between the comparator inputs before the comparator output switches. The difference must be in the direction opposite of the current comparator output.	00
		0x0	Hysteresis is turned off, comparator output will change as the input voltages cross.	
		0x1	Hysteresis = 5 mV.	
		0x2	Hysteresis = 10 mV.	
		0x3	Hysteresis = 15 mV.	
15	CMP1_INTPOL		Selects the polarity of the CMP1 output for purposes of generating level interrupts. See Table 694.	0
		0	The CMP1 output is used as-is for generating interrupts.	
		1	The CMP1 output is used inverted for generating interrupts.	
16	CMP1_INTTYPE		Select comparator 1 interrupt type. See Table 694.	0
		0	Comparator 1 interrupt is edge triggered.	
		1	Comparator 1 interrupt is level triggered.	
18:17	CMP1_INTEDGE		Select edge triggered interrupt to be active on either high or low transitions, when CMP1_IntType = 0. See Table 694.	0
		0x0	Comparator 1 interrupt is active on falling edges.	
		0x1	Comparator 1 interrupt is active on rising edges.	
		0x2	Comparator 1 Interrupt is active on both edges.	
		0x3	reserved.	
19	CMP1_INTFLAG		Comparator 1 interrupt flag.	0
		0	The Comparator 1 interrupt is not pending.	
		1	The Comparator 1 interrupt is pending. Writing a 1 to this bit clears the flag.	
21:20	CMP1_VLADEN		Voltage ladder enable for comparator 1.	00
		0x0	The Comparator 1 voltage ladder is disabled.	
		0x1	The Comparator 1 voltage ladder is disabled in Deep Sleep and Power-down modes and re-enabled automatically when exiting those modes.	
		0x2	The Comparator 1 voltage ladder is disabled in Power-down mode and re-enabled automatically when exiting Power-down.	
		0x3	The Comparator 1 voltage ladder is enabled.	
22	CMP1_VLADREF		Voltage reference select for comparator 1 voltage ladder.	0
		0	VREF_CMP pin.	
		1	V_{DDA} pin.	

Table 693. Comparator 1 control register (CMP_CTRL1 - 0x4002 0008) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
23	-		Reserved.	NA
28:24	CMP1_VSel		Voltage ladder value for comparator 1. The reference voltage Vref depends on the setting of CMP1_VLADREF (either $V_{DD(3V3)}$ or voltage on pin VREF_CMP). 00000 = Vss. 00001 = 1 × Vref1 / 31. 00010 = 2 × Vref1 / 31. ... 11111 = Vref1.	0x00
31:29	-		Reserved.	NA

34.6.4 Comparator interrupt configurations

Table 694: Interrupt configurations

IntEdge	IntType	IntPol	Interrupt function
00	0	See Table note [3]	Falling edge interrupt. See Table note [1]
01	0	See Table note [3]	Rising edge interrupt. See Table note [1]
1x	0	See Table note [3]	Interrupt on both edges. See Table note [1]
See Table note [4]	1	0	High level interrupt. See Table note [2]
See Table note [4]	1	1	Low level interrupt. See Table note [2]

[1] The same signal goes to timer 0 and 1 (see Table 691, bits 12 through 15)

[2] The same signal goes to the wake-up logic.

[3] CMPn_IntPol has no effect on the comparator interrupt, but can separately select a wake-up polarity.

[4] CMPn_IntEdge has no effect on the comparator interrupt, but can separately select the signal that goes to a timer from the edge output.

35.1 Basic configuration

The GPDMA is configured using the following registers:

1. Power: In the PCONP register (Section 3.3.2.2), set bit PCGPDMA.

   Remark: On reset, the GPDMA is disabled (PCGPDMA = 0).
2. Clock: The GPDMA operates at the AHB bus rate, which is the same as the CPU clock rate (CCLK).
3. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
4. Programming: see Section 35.6.
5. Select the DMA channel alternate requests in the DMA channel request select register in the system control block. See Section 3.3.7.7.

35.2 Introduction

The DMA controller allows peripheral-to memory, memory-to-peripheral, and memory-to-memory transactions. Each DMA stream provides unidirectional serial DMA transfers for a single source and destination. For example, a bidirectional port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral.

35.3 Features

- Eight DMA channels. Each channel can support one unidirectional transfer.
- 16 DMA request lines.
- Memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers are supported.
- GPDMA supports the SD card interface, all SSPs, the I^2S, all UARTs, the A/D Converter, and the D/A Converter peripherals. DMA can also be triggered by selected timer match conditions. Memory-to-memory transfers and transfers to or from GPIO are also supported.
- Scatter or gather DMA is supported through the use of linked lists. This means that the source and destination areas do not have to occupy contiguous areas of memory.
- Hardware DMA channel priority.
- AHB slave DMA programming interface. The DMA Controller is programmed by writing to the DMA control registers over the AHB slave interface.
- One AHB bus master for transferring data. The interface transfers data when a DMA request goes active.
- 32-bit AHB master bus width.
- Incrementing or non-incrementing addressing for source and destination.
- Programmable DMA burst size. The DMA burst size can be programmed to more efficiently transfer data.
- Internal four-word FIFO per channel.
- Supports 8-bit, 16-bit, and 32-bit wide transactions.
- Big-endian and little-endian support. The DMA Controller defaults to little-endian mode on reset.
- An interrupt to the processor can be generated on a DMA completion or when a DMA error has occurred.
- Raw interrupt status. The DMA error and DMA count raw interrupt status can be read prior to masking.
- DMA can operate in Sleep mode. (Note that in Sleep mode the GPDMA cannot access the flash memory or the main SRAM).

35.4 Functional description

This section describes the major functional blocks of the DMA Controller.

35.4.1 DMA controller functional description

The DMA Controller enables peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral, and memory-to-memory transactions. Each DMA stream provides unidirectional serial DMA transfers for a single source and destination. For example, a bidirectional port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral, and can be accessed through the AHB master. Figure 166 shows a block diagram of the DMA Controller.

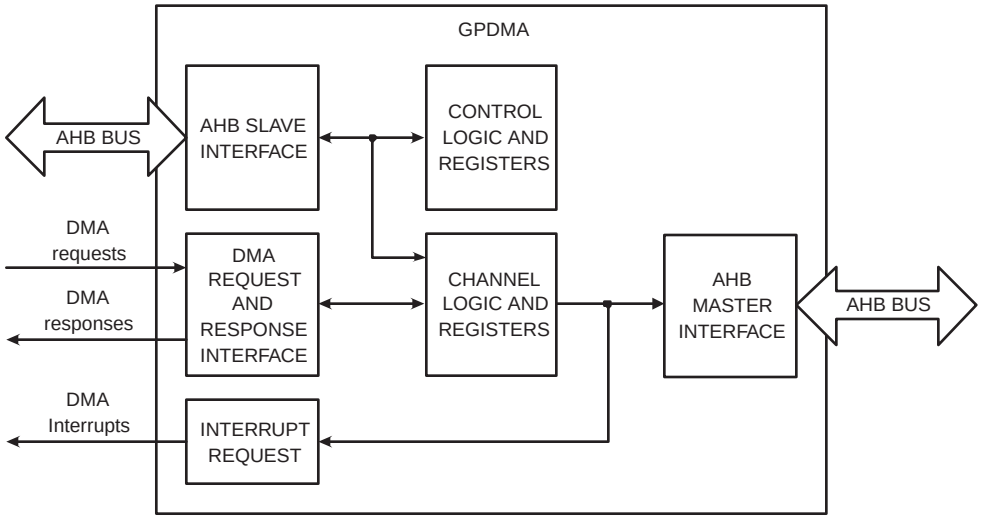


Fig 166. DMA controller block diagram

The functions of the DMA Controller are described in the following sections.

35.4.1.1 AHB slave interface

All transactions to DMA Controller registers on the AHB slave interface are 32 bits wide. 8-bit and 16-bit accesses are not supported and will result in an exception.

35.4.1.2 Control logic and register bank

The register block stores data written or to be read across the AHB interface.

35.4.1.3 DMA request and response interface

See Section 35.4.2 for information on the DMA request and response interface.

35.4.1.4 Channel logic and channel register bank

The channel logic and channel register bank contains registers and logic required for each DMA channel.

35.4.1.5 Interrupt request

The interrupt request generates the interrupt to the ARM processor.

35.4.1.6 AHB master interface

The DMA Controller contains one AHB master interface. The AHB master is capable of dealing with all types of AHB transactions, including:

- Split, retry, and error responses from slaves. If a peripheral performs a split or retry, the DMA Controller stalls and waits until the transaction can complete.
- Locked transfers for source and destination of each stream.
- Setting of protection bits for transfers on each stream.

35.4.1.6.1 Bus and transfer widths

The physical width of the AHB bus is 32 bits. Source and destination transfers can be of differing widths and can be the same width or narrower than the physical bus width. The DMA Controller packs or unpacks data as appropriate.

35.4.1.6.2 Endian behavior

The DMA Controller can cope with both little-endian and big-endian addressing.

Internally the DMA Controller treats all data as a stream of bytes instead of 16-bit or 32-bit quantities. This means that when performing mixed-endian activity, where the endianness of the source and destination are different, byte swapping of the data within the 32-bit data bus is observed.

Note: If byte swapping is not required, then use of different endianness between the source and destination addresses must be avoided. Table 695 shows endian behavior for different source and destination combinations.

Table 695. Endian behavior

Source endian	Destination endian	Source width	Destination width	Source transfer no/byte lane	Source data	Destination transfer no/byte lane	Destination data
Little	Little	8	8	1/[7:0]	21	1/[7:0]	21212121
				2/[15:8]	43	2/[15:8]	43434343
				3/[23:16]	65	3/[23:16]	65656565
				4/[31:24]	87	4/[31:24]	87878787
Little	Little	8	16	1/[7:0]	21	1/[15:0]	43214321
				2/[15:8]	43	2/[31:16]	87658765
				3/[23:16]	65		
				4/[31:24]	87		
Little	Little	8	32	1/[7:0]	21	1/[31:0]	87654321
				2/[15:8]	43		
				3/[23:16]	65		
				4/[31:24]	87		

Table 695. Endian behavior *…continued*

Source endian	Destination endian	Source width	Destination width	Source transfer no/byte lane	Source data	Destination transfer no/byte lane	Destination data
Little	Little	16	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				2/[23:16]	65	3/[23:16]	65656565
				2/[31:24]	87	4/[31:24]	87878787
Little	Little	16	16	1/[7:0]	21	1/[15:0]	43214321
				1/[15:8]	43	2/[31:16]	87658765
				2/[23:16]	65		
				2/[31:24]	87		
Little	Little	16	32	1/[7:0]	21	1/[31:0]	87654321
				1/[15:8]	43		
				2/[23:16]	65		
				2/[31:24]	87		
Little	Little	32	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				1/[23:16]	65	3/[23:16]	65656565
				1/[31:24]	87	4/[31:24]	87878787
Little	Little	32	16	1/[7:0]	21	1/[15:0]	43214321
				1/[15:8]	43	2/[31:16]	87658765
				1/[23:16]	65		
				1/[31:24]	87		
Little	Little	32	32	1/[7:0]	21	1/[31:0]	87654321
				1/[15:8]	43		
				1/[23:16]	65		
				1/[31:24]	87		
Big	Big	8	8	1/[31:24]	12	1/[31:24]	12121212
				2/[23:16]	34	2/[23:16]	34343434
				3/[15:8]	56	3/[15:8]	56565656
				4/[7:0]	78	4/[7:0]	78787878
Big	Big	8	16	1/[31:24]	12	1/[15:0]	12341234
				2/[23:16]	34	2/[31:16]	56785678
				3/[15:8]	56		
				4/[7:0]	78		
Big	Big	8	32	1/[31:24]	12	1/[31:0]	12345678
				2/[23:16]	34		
				3/[15:8]	56		
				4/[7:0]	78		
Big	Big	16	8	1/[31:24]	12	1/[31:24]	12121212
				1/[23:16]	34	2/[23:16]	34343434
				2/[15:8]	56	3/[15:8]	56565656
				2/[7:0]	78	4/[7:0]	78787878

Table 695. Endian behavior *…continued*

Source endian	Destination endian	Source width	Destination width	Source transfer no/byte lane	Source data	Destination transfer no/byte lane	Destination data
Big	Big	16	16	1/[31:24]	12	1/[15:0]	12341234
				1/[23:16]	34	2/[31:16]	56785678
				2/[15:8]	56		
				2/[7:0]	78		
Big	Big	16	32	1/[31:24]	12	1/[31:0]	12345678
				1/[23:16]	34		
				2/[15:8]	56		
				2/[7:0]	78		
Big	Big	32	8	1/[31:24]	12	1/[31:24]	12121212
				1/[23:16]	34	2/[23:16]	34343434
				1/[15:8]	56	3/[15:8]	56565656
				1/[7:0]	78	4/[7:0]	78787878
Big	Big	32	16	1/[31:24]	12	1/[15:0]	12341234
				1/[23:16]	34	2/[31:16]	56785678
				1/[15:8]	56		
				1/[7:0]	78		
Big	Big	32	32	1/[31:24]	12	1/[31:0]	12345678
				1/[23:16]	34		
				1/[15:8]	56		
				1/[7:0]	78		

35.4.1.6.3 Error conditions

An error during a DMA transfer is flagged directly by the peripheral by asserting an Error response on the AHB bus during the transfer. The DMA Controller automatically disables the DMA stream after the current transfer has completed, and can optionally generate an error interrupt to the CPU. This error interrupt can be masked.

35.4.1.7 Channel hardware

Each stream is supported by a dedicated hardware channel, including source and destination controllers, as well as a FIFO. This enables better latency than a DMA controller with only a single hardware channel shared between several DMA streams and simplifies the control logic.

35.4.1.8 DMA request priority

DMA channel priority is fixed. DMA channel 0 has the highest priority and DMA channel 7 has the lowest priority.

If the DMA Controller is transferring data for the lower priority channel and then the higher priority channel goes active, it completes the number of transfers delegated to the master interface by the lower priority channel before switching over to transfer data for the higher priority channel. Transfers delegated to the master interface are staged in the DMA channel FIFO, so the amount of data that needs to transfer could be as large as a 4 words.

It is recommended that memory-to-memory transactions use the lowest priority channel.

35.4.1.9 Interrupt generation

A combined interrupt output is generated as an OR function of the individual interrupt requests of the DMA Controller and is connected to the interrupt controller.

35.4.2 DMA system connections

35.4.2.1 DMA request signals

The DMA request signals are used by peripherals to request a data transfer. The DMA request signals indicate whether a single or burst transfer of data is required. The DMA available request signals are:

DMACBREQ[15:0] — Burst request signals. These cause a programmed burst number of data to be transferred.

DMACSREQ[15:0] — Single transfer request signals. These cause a single data to be transferred. The DMA controller transfers a single transfer to or from the peripheral.

DMACLBREQ[15:0] — Last burst request signals.

DMACLSREQ[15:0] — Last single transfer request signals.

Note that most peripherals do not support “last” request types, and many peripherals do not support the single request type. See Section 35.4.2.3.

35.4.2.2 DMA response signals

The DMA response signals indicate whether the transfer initiated by the DMA request signal has completed. The response signals can also be used to indicate whether a complete packet has been transferred. The DMA response signals from the DMA controller are:

DMACCLR[15:0] — DMA clear or acknowledge signals. The DMACCLR signal is used by the DMA controller to acknowledge a DMA request from the peripheral.

DMACTC[15:0] — DMA terminal count signals. The DMACTC signal can be used by the DMA controller to indicate to the peripheral that the DMA transfer is complete.

35.4.2.3 DMA request connections

The connection of the GPDMA to the supported peripheral devices depends on the DMA functions implemented in those peripherals. Table 696 shows the DMA Request numbers used by the supported peripherals. Alternative requests on channels 0 through 7 and 10 through 15 are chosen via the DMAReqSel register, see Section 3.3.7.7.

Table 696. DMA connections

DMA request	Burst Request	Single Request	Last Burst Request	Last Single Request	Comment
0	(unused) / T0_MAT[0]	-	-	-	Dedicated DMA request
1	SD card / T0_MAT[1]	SD card only	SD card only	SD card only	Dedicated DMA request
2	SSP0 Tx / T1_MAT[0]	SSP0 Tx only	-	-	Dedicated DMA request
3	SSP0 Rx / T1_MAT[1]	SSP0 Rx only	-	-	Dedicated DMA request
4	SSP1 Tx / T2_MAT[0]	SSP1 Tx only	-	-	Dedicated DMA request

Table 696. DMA connections

DMA request	Burst Request	Single Request	Last Burst Request	Last Single Request	Comment
5	SSP1 Rx / T2_MAT[1]	SSP1 Rx only	-	-	Dedicated DMA request
6	SSP2 TX / I^2S channel 0	SSP2 Tx only	-	-	Dedicated DMA request
7	SSP2 Rx / I^2S channel 1	SSP2 Rx only	-	-	Dedicated DMA request
8	ADC	-	-	-	ADC interrupt request [1]
9	DAC	-	-	-	Dedicated DMA request
10	UART0 Tx / UART3 Tx	-	-	-	Dedicated DMA request
11	UART0 Rx / UART3 Rx	-	-	-	Dedicated DMA request
12	UART1 Tx / UART4 Tx	-	-	-	Dedicated DMA request
13	UART1 Rx / UART4 Rx	-	-	-	Dedicated DMA request
14	UART2 Tx / T3_MAT[0]	-	-	-	Dedicated DMA request
15	UART2 Rx / T3_MAT[1]	-	-	-	Dedicated DMA request

[1] Generates an interrupt and/or DMA request depending on software setup.

35.5 Register description

The DMA Controller supports 8 channels. Each channel has registers specific to the operation of that channel. Other registers controls aspects of how source peripherals relate to the DMA Controller. There are also global DMA control and status registers.

The DMA Controller registers are shown in Table 697. In addition, the DMA request select register is located in the system control block. See Section 3.3.7.7.

Table 697. Register overview: GPDMA (base address 0x2008 0000)

Name	Access	Address offset	Description	Reset Value	Table
General registers					
INTSTAT	RO	0x000	DMA Interrupt Status Register	0	698
INTTCSTAT	RO	0x004	DMA Interrupt Terminal Count Request Status Register	0	699
INTTCCLEAR	WO	0x008	DMA Interrupt Terminal Count Request Clear Register	-	700
INTERRSTAT	RO	0x00C	DMA Interrupt Error Status Register	0	701
INTERRCLR	WO	0x010	DMA Interrupt Error Clear Register	-	702
RAWINTTCSTAT	RO	0x014	DMA Raw Interrupt Terminal Count Status Register	0	703
RAWINTERRSTAT	RO	0x018	DMA Raw Error Interrupt Status Register	0	704
ENBLDCHNS	RO	0x01C	DMA Enabled Channel Register	0	705
SOFTBREQ	R/W	0x020	DMA Software Burst Request Register	0	706
SOFTSREQ	R/W	0x024	DMA Software Single Request Register	0	707
SOFTLBREQ	R/W	0x028	DMA Software Last Burst Request Register	0	708
SOFTLSREQ	R/W	0x02C	DMA Software Last Single Request Register	0	709
CONFIG	R/W	0x030	DMA Configuration Register	0	710
SYNC	R/W	0x034	DMA Synchronization Register	0	711
Channel 0 registers					
SRCADDR0	R/W	0x100	DMA Channel 0 Source Address Register	0	712
DESTADDR0	R/W	0x104	DMA Channel 0 Destination Address Register	0	713
LLI0	R/W	0x108	DMA Channel 0 Linked List Item Register	0	714
CONTROL0	R/W	0x10C	DMA Channel 0 Control Register	0	715
CONFIG0	R/W	0x110	DMA Channel 0 Configuration Register[1]	0	716
Channel 1 registers					
SRCADDR1	R/W	0x120	DMA Channel 1 Source Address Register	0	712
DESTADDR1	R/W	0x124	DMA Channel 1 Destination Address Register	0	713
LLI1	R/W	0x128	DMA Channel 1 Linked List Item Register	0	714
CONTROL1	R/W	0x12C	DMA Channel 1 Control Register	0	715
CONFIG1	R/W	0x130	DMA Channel 1 Configuration Register[1]	0	716
Channel 2 registers					
SRCADDR2	R/W	0x140	DMA Channel 2 Source Address Register	0	712
DESTADDR2	R/W	0x144	DMA Channel 2 Destination Address Register	0	713
LLI2	R/W	0x148	DMA Channel 2 Linked List Item Register	0	714
CONTROL2	R/W	0x14C	DMA Channel 2 Control Register	0	715
CONFIG2	R/W	0x150	DMA Channel 2 Configuration Register[1]	0	716

Table 697. Register overview: GPDMA (base address 0x2008 0000)

Name	Access	Address offset	Description	Reset Value	Table
Channel 3 registers					
SRCADDR3	R/W	0x160	DMA Channel 3 Source Address Register	0	712
DESTADDR3	R/W	0x164	DMA Channel 3 Destination Address Register	0	713
LLI3	R/W	0x168	DMA Channel 3 Linked List Item Register	0	714
CONTROL3	R/W	0x16C	DMA Channel 3 Control Register	0	715
CONFIG3	R/W	0x170	DMA Channel 3 Configuration Register[1]	0	716
Channel 4 registers					
SRCADDR4	R/W	0x180	DMA Channel 4 Source Address Register	0	712
DESTADDR4	R/W	0x184	DMA Channel 4 Destination Address Register	0	713
LLI4	R/W	0x188	DMA Channel 4 Linked List Item Register	0	714
CONTROL4	R/W	0x18C	DMA Channel 4 Control Register	0	715
CONFIG4	R/W	0x190	DMA Channel 4 Configuration Register[1]	0	716
Channel 5 registers					
SRCADDR5	R/W	0x1A0	DMA Channel 5 Source Address Register	0	712
DESTADDR5	R/W	0x1A4	DMA Channel 5 Destination Address Register	0	713
LLI5	R/W	0x1A8	DMA Channel 5 Linked List Item Register	0	714
CONTROL5	R/W	0x1AC	DMA Channel 5 Control Register	0	715
CONFIG5	R/W	0x1B0	DMA Channel 5 Configuration Register[1]	0	716
Channel 6 registers					
SRCADDR6	R/W	0x1C0	DMA Channel 6 Source Address Register	0	712
DESTADDR6	R/W	0x1C4	DMA Channel 6 Destination Address Register	0	713
LLI6	R/W	0x1C8	DMA Channel 6 Linked List Item Register	0	714
CONTROL6	R/W	01CC	DMA Channel 6 Control Register	0	715
CONFIG6	R/W	0x1D0	DMA Channel 6 Configuration Register[1]	0	716
Channel 7 registers					
SRCADDR7	R/W	0x1E0	DMA Channel 7 Source Address Register	0	712
DESTADDR7	R/W	0x1E4	DMA Channel 7 Destination Address Register	0	713
LLI7	R/W	0x1E8	DMA Channel 7 Linked List Item Register	0	714
CONTROL7	R/W	0x1EC	DMA Channel 7 Control Register	0	715
CONFIG7	R/W	0x1F0	DMA Channel 7 Configuration Register[1]	0	716

[1] Bit 17 of this register is a read-only status flag.

35.5.1 DMA Interrupt Status register

The DMACIntStat Register is read-only and shows the status of the interrupts after masking. A 1 bit indicates that a specific DMA channel interrupt request is active. The request can be generated from either the error or terminal count interrupt requests. Table 698 shows the bit assignments of the DMACIntStat Register.

Table 698. DMA Interrupt Status register (INTSTAT, address 0x2008 0000) bit description

Bit	Symbol	Description
7:0	INTSTAT	Status of DMA channel interrupts after masking. Each bit represents one channel: 0 - the corresponding channel has no active interrupt request. 1 - the corresponding channel does have an active interrupt request.
31:8	-	Reserved. The value read from a reserved bit is not defined.

35.5.2 DMA Interrupt Terminal Count Request Status register

The DMACIntTCStat Register is read-only and indicates the status of the terminal count after masking. Table 699 shows the bit assignments of the DMACIntTCStat Register.

Table 699. MA Interrupt Terminal Count Request Status Register (INTTCSTAT, address 0x2008 0004) bit description

Bit	Symbol	Description
7:0	INTTCSTAT	Terminal count interrupt request status for DMA channels. Each bit represents one channel: 0 - the corresponding channel has no active terminal count interrupt request. 1 - the corresponding channel does have an active terminal count interrupt request.
31:8	-	Reserved. The value read from a reserved bit is not defined.

35.5.3 DMA Interrupt Terminal Count Request Clear register

The DMACIntTCClear Register is write-only and clears one or more terminal count interrupt requests. When writing to this register, each data bit that contains a 1 causes the corresponding bit in the status register (DMACIntTCStat) to be cleared. Data bits that are 0 have no effect. Table 700 shows the bit assignments of the DMACIntTCClear Register.

Table 700. DMA Interrupt Terminal Count Request Clear Register (INTTCCLEAR, address 0x2008 0008) bit description

Bit	Symbol	Description
7:0	INTTCCLEAR	Allows clearing the Terminal count interrupt request (IntTCStat) for DMA channels. Each bit represents one channel: 0 - writing 0 has no effect. 1 - clears the corresponding channel terminal count interrupt.
31:8	-	Reserved. Read value is undefined, only zero should be written.

35.5.4 DMA Interrupt Error Status register

The DMACIntErrStat Register is read-only and indicates the status of the error request after masking. Table 701 shows the bit assignments of the DMACIntErrStat Register.

Table 701. DMA Interrupt Error Status Register (INTERRSTAT, address 0x2008 000C) bit description

Bit	Symbol	Description
7:0	INTERRSTAT	Interrupt error status for DMA channels. Each bit represents one channel: 0 - the corresponding channel has no active error interrupt request. 1 - the corresponding channel does have an active error interrupt request.
31:8	-	Reserved. The value read from a reserved bit is not defined.

35.5.5 DMA Interrupt Error Clear register

The DMACIntErrClr Register is write-only and clears the error interrupt requests. When writing to this register, each data bit that is 1 causes the corresponding bit in the status register to be cleared. Data bits that are 0 have no effect on the corresponding bit in the register. Table 702 shows the bit assignments of the DMACIntErrClr Register.

Table 702. DMA Interrupt Error Clear Register (INTERRCLR, address 0x2008 0010) bit description

Bit	Symbol	Description
7:0	INTERRCLR	Writing a 1 clears the error interrupt request (IntErrStat) for DMA channels. Each bit represents one channel: 0 - writing 0 has no effect. 1 - clears the corresponding channel error interrupt.
31:8	-	Reserved. Read value is undefined, only zero should be written.

35.5.6 DMA Raw Interrupt Terminal Count Status register

The DMACRawIntTCStat Register is read-only and indicates which DMA channel is requesting a transfer complete (terminal count interrupt) prior to masking. (Note: the DMACIntTCStat Register contains the same information after masking.) A 1 bit indicates that the terminal count interrupt request is active prior to masking. Table 703 shows the bit assignments of the DMACRawIntTCStat Register.

Table 703. DMA Raw Interrupt Terminal Count Status Register (RAWINTTCSTAT, address 0x2008 0014) bit description

Bit	Symbol	Description
7:0	RAWINTTCSTAT	Status of the terminal count interrupt for DMA channels prior to masking. Each bit represents one channel: 0 - the corresponding channel has no active terminal count interrupt request. 1 - the corresponding channel does have an active terminal count interrupt request.
31:8	-	Reserved. The value read from a reserved bit is not defined.

35.5.7 DMA Raw Error Interrupt Status register

The DMACRawIntErrStat Register is read-only and indicates which DMA channel is requesting an error interrupt prior to masking. (Note: the DMACIntErrStat Register contains the same information after masking.) A 1 bit indicates that the error interrupt request is active prior to masking. Table 704 shows the bit assignments of register of the

DMACRawIntErrStat Register.

Table 704. DMA Raw Error Interrupt Status Register (RAWINTERRSTAT, address 0x2008 0018) bit description

Bit	Symbol	Description
7:0	RAWINTERRSTAT	Status of the error interrupt for DMA channels prior to masking. Each bit represents one channel: 0 - the corresponding channel has no active error interrupt request. 1 - the corresponding channel does have an active error interrupt request.
31:8	-	Reserved. The value read from a reserved bit is not defined.

35.5.8 DMA Enabled Channel register

The DMACEnbldChns Register is read-only and indicates which DMA channels are enabled, as indicated by the Enable bit in the Config Register. A 1 bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer. Table 705 shows the bit assignments of the DMACEnbldChns Register.

Table 705. DMA Enabled Channel Register (ENBLDCHNS, address 0x2008 001C) bit description

Bit	Symbol	Description
7:0	ENABLEDCHANNELS	Enable status for DMA channels. Each bit represents one channel: 0 - DMA channel is disabled. 1 - DMA channel is enabled.
31:8	-	Reserved. The value read from a reserved bit is not defined.

35.5.9 DMA Software Burst Request register

The DMACSoftBReq Register is read/write and enables DMA burst requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting DMA burst transfers. A request can be generated from either a peripheral or the software request register. Each bit is cleared when the related transaction has completed. Table 706 shows the bit assignments of the DMACSoftBReq Register.

Table 706. DMA Software Burst Request Register (SOFTBREQ, address 0x2008 0020) bit description

Bit	Symbol	Description
15:0	SOFTBREQ	Software burst request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral Description (refer to Table 696 for peripheral hardware connections to the DMA controller): 0 - writing 0 has no effect. 1 - writing 1 generates a DMA burst request for the corresponding request line.
31:16	-	Reserved. Read value is undefined, only zero should be written.

Note: It is recommended that software and hardware peripheral requests are not used at the same time.

35.5.10 DMA Software Single Request register

The DMACSoftSReq Register is read/write and enables DMA single transfer requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting single DMA transfers. A request can be generated from either a peripheral or the software request register. Table 707 shows the bit assignments of the DMACSoftSReq Register.

Table 707. DMA Software Single Request register bit description

Bit	Symbol	Description
15:0	SOFTSREQ	Software single transfer request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function: 0 - writing 0 has no effect. 1 - writing 1 generates a DMA single transfer request for the corresponding request line.
31:16	-	Reserved. Read value is undefined, only zero should be written.

35.5.11 DMA Software Last Burst Request register

The DMACSoftLBReq Register is read/write and enables DMA last burst requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting last burst DMA transfers. A request can be generated from either a peripheral or the software request register. Table 708 shows the bit assignments of the DMACSoftLBReq Register.

Table 708. DMA Software Last Burst Request Register (SOFTLBREQ, address 0x2008 0028) bit description

Bit	Symbol	Description
15:0	SOFTLBREQ	Software last burst request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function: 0 - writing 0 has no effect. 1 - writing 1 generates a DMA last burst request for the corresponding request line.
31:16	-	Reserved. Read value is undefined, only zero should be written.

35.5.12 DMA Software Last Single Request register

The DMACSoftLSReq Register is read/write and enables DMA last single requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting last single DMA transfers. A request can be generated from either a peripheral or the software request register. Table 709 shows the bit assignments of the DMACSoftLSReq Register.

Table 709. DMA Software Last Single Request Register (SOFTLSREQ, address 0x2008 002C) bit description

Bit	Symbol	Description
15:0	SOFTLSREQ	Software last single transfer request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function: 0 - writing 0 has no effect. 1 - writing 1 generates a DMA last single transfer request for the corresponding request line.
31:16	-	Reserved. Read value is undefined, only zero should be written.

35.5.13 DMA Configuration register

The Config Register is read/write and configures the operation of the DMA Controller. The endianness of the AHB master interface can be altered by writing to the M bit of this register. The AHB master interface is set to little-endian mode on reset. Table 710 shows the bit assignments of the Config Register.

Table 710. DMA Configuration Register (CONFIG, address 0x2008 0030) bit description

Bit	Symbol	Description
0	E	DMA Controller enable: 0 = disabled (default). Disabling the DMA Controller reduces power consumption. 1 = enabled.
1	M	AHB Master endianness configuration: 0 = little-endian mode (default). 1 = big-endian mode.
31:2	-	Reserved. Read value is undefined, only zero should be written.

35.5.14 DMA Synchronization register

The Sync Register is read/write and enables or disables synchronization logic for the DMA request signals. The DMA request signals consist of the BREQ[15:0], SREQ[15:0], LBREQ[15:0], and LSREQ[15:0]. A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests. This register is reset to 0, enabling synchronization logic by default. Table 711 shows the bit assignments of the Sync Register.

Table 711. DMA Synchronization Register (SYNC, address 0x2008 0034) bit description

Bit	Symbol	Description
15:0	DMACSYNC	Controls the synchronization logic for DMA request signals. Each bit represents one set of DMA request lines as described in the preceding text: 0 - synchronization logic for the corresponding DMA request signals are enabled. 1 - synchronization logic for the corresponding DMA request signals are disabled.
31:16	-	Reserved. Read value is undefined, only zero should be written.

35.5.15 DMA Channel registers

The channel registers are used to program the eight DMA channels. These registers consist of:

- Eight SrcAddr Registers.
- Eight DestAddr Registers.
- Eight LLI Registers.
- Eight Control Registers.
- Eight Config Registers.

When performing scatter/gather DMA, the first four of these are automatically updated.

35.5.16 DMA Channel Source Address registers

The eight read/write SrcAddr Registers contain the current source address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the appropriate channel is enabled. When the DMA channel is enabled this register is updated:

- As the source address is incremented.
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time software has processed the value read, the address may have progressed. It is intended to be read-only when the channel has stopped, in which case it shows the source address of the last item read.

Note: The source and destination addresses must be aligned to the source and destination widths.

Table 712. DMA Channel Source Address Registers (SRCADDR[0:7], 0x2008 0100 (SRCADDR0) to 0x2008 01E0 (SRCADDR7)) bit description

Bit	Symbol	Description
31:0	SRCADDR	DMA source address. Reading this register will return the current source address.

35.5.17 DMA Channel Destination Address registers

The eight read/write DestAddr Registers contain the current destination address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the channel is enabled. When the DMA channel is enabled the register is updated as the destination address is incremented and by following the linked list when a complete packet of data has been transferred. Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the address may have progressed. It is intended to be read-only when a channel has stopped, in which case it shows the destination address of the last item read.

Table 713. DMA Channel Destination Address registers (DESTADDR[0:7], 0x2008 0104 (DESTADDR0) to 0x2008 01E4 (DESTADDR7)) bit description

Bit	Symbol	Description
31:0	DESTADDR	DMA Destination address. Reading this register will return the current destination address.

35.5.18 DMA Channel Linked List Item registers

The eight read/write LLI Registers contain a word-aligned address of the next Linked List Item (LLI). If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled when all DMA transfers associated with it are completed. Programming this register when the DMA channel is enabled may have unpredictable side effects.

Table 714. DMA Channel Linked List Item registers (LLI[0:7], 0x2008 0108 (LLI0) to 0x2008 01E8 (LLI7)) bit description

Bit	Symbol	Description
1:0	-	Reserved, and must be written as 0.
31:2	LLI	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.

35.5.19 DMA channel control registers

The eight read/write Control Registers contain DMA channel control information such as the transfer size, burst size, and transfer width. Each register is programmed directly by software before the DMA channel is enabled. When the channel is enabled the register is updated by following the linked list when a complete packet of data has been transferred. Reading the register while the channel is active does not give useful information. This is because by the time software has processed the value read, the channel may have advanced. It is intended to be read-only when a channel has stopped. Table 715 shows the bit assignments of the Control Register.

35.5.19.1 Protection and access information

AHB access information is provided to the source and/or destination peripherals when a transfer occurs, although on these devices this has no effect. The transfer information is provided by programming the DMA channel (the Prot bits of the Control Register, and the Lock bit of the Config Register). These bits are programmed by software, and can be used by peripherals. Three bits of information are provided, and are used as shown in Table 715.

Table 715. DMA Channel Control registers (CONTROL[0:7], 0x2008 010C (CONTROL0) to 0x2008 01EC (CONTROL7)) bit description

Bit	Symbol	Description
11:0	TRANSFERSIZE	Transfer size. This field sets the size of the transfer when the DMA controller is the flow controller, in which case the value must be set before the channel is enabled. Transfer size is updated as data transfers are completed. A read from this field indicates the number of transfers completed on the destination bus. Reading the register when the channel is active does not give useful information because by the time that the software has processed the value read, the channel might have progressed. It is intended to be used only when a channel is enabled and then disabled. The transfer size value is not used if a peripheral is the flow controller.
14:12	SBSIZE	Source burst size. Indicates the number of transfers that make up a source burst. This value must be set to the burst size of the source peripheral, or if the source is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACBREQ signal goes active in the source peripheral. 000 - 1 001 - 4 010 - 8 011 - 16 100 - 32 101 - 64 110 - 128 111 - 256
17:15	DBSIZE	Destination burst size. Indicates the number of transfers that make up a destination burst transfer request. This value must be set to the burst size of the destination peripheral or, if the destination is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACBREQ signal goes active in the destination peripheral. 000 - 1 001 - 4 010 - 8 011 - 16 100 - 32 101 - 64 110 - 128 111 - 256
20:18	SWIDTH	Source transfer width. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required. 000 - Byte (8-bit) 001 - Halfword (16-bit) 010 - Word (32-bit) 011 to 111 - Reserved
23:21	DWIDTH	Destination transfer width. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required. 000 - Byte (8-bit) 001 - Halfword (16-bit) 010 - Word (32-bit) 011 to 111 - Reserved
25:24	-	Reserved, and must be written as 0.

Table 715. DMA Channel Control registers (CONTROL[0:7], 0x2008 010C (CONTROL0) to 0x2008 01EC (CONTROL7)) bit description *…continued*

Bit	Symbol	Description
26	SI	Source increment: 0 - the source address is not incremented after each transfer. 1 - the source address is incremented after each transfer.
27	DI	Destination increment: 0 - the destination address is not incremented after each transfer. 1 - the destination address is incremented after each transfer.
28	PROT1	This is provided to the peripheral during a DMA bus access and indicates that the access is in user mode or privileged mode. This information is not used on these devices. 0 - access is in user mode. 1 - access is in privileged mode.
29	PROT2	This is provided to the peripheral during a DMA bus access and indicates to the peripheral that the access is bufferable or not bufferable. This information is not used on these devices. 0 - access is not bufferable. 1 - access is bufferable.
30	PROT3	This is provided to the peripheral during a DMA bus access and indicates to the peripheral that the access is cacheable or not cacheable. This information is not used on these devices. 0 - access is not cacheable. 1 - access is cacheable.
31	I	Terminal count interrupt enable bit. 0 - the terminal count interrupt is disabled. 1 - the terminal count interrupt is enabled.

35.5.20 DMA Channel Configuration registers

The eight Config Registers are read/write with the exception of bit[17] which is read-only. These registers configure each DMA channel. The registers are not updated when a new LLI is requested. Table 716 shows the bit assignments of the Config Register.

Table 716. DMA Channel Configuration registers (CONFIG[0:7], 0x2008 0110 (CONFIG0) to 0x2008 01F0 (CONFIG7)) bit description

Bit	Symbol	Description
0	E	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0 = channel disabled. 1 = channel enabled. The Channel Enable bit status can also be found by reading the DMACEnbldChns Register. A channel is enabled by setting this bit. A channel can be disabled by clearing the Enable bit. This causes the current AHB transfer (if one is in progress) to complete and the channel is then disabled. Any data in the FIFO of the relevant channel is lost. Restarting the channel by setting the Channel Enable bit has unpredictable effects, the channel must be fully re-initialized. The channel is also disabled, and Channel Enable bit cleared, when the last LLI is reached, the DMA transfer is completed, or if a channel error is encountered. If a channel must be disabled without losing data in the FIFO, the Halt bit must be set so that further DMA requests are ignored. The Active bit must then be polled until it reaches 0, indicating that there is no data left in the FIFO. Finally, the Channel Enable bit can be cleared. **Remark:** it is important to be aware that for memory-to-peripheral or memory-to-memory transfers, the DMA controller starts filling the related channel FIFO as soon as the channel is enabled. Therefore the source data must be set up in memory prior to enabling channels using those transfer types.
5:1	SRCPERIPHERAL	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory. See Table 696 for peripheral identification.
10:6	DESTPERIPHERAL	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory. See Table 696 for peripheral identification.
13:11	TRANSFERTYPE	This value indicates the type of transfer and specifies the flow controller. The transfer type can be memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral. Flow can be controlled by the DMA controller, the source peripheral, or the destination peripheral. Refer to Table 717 for the encoding of this field.
14	IE	Interrupt error mask. When cleared, this bit masks out the error interrupt of the relevant channel.
15	ITC	Terminal count interrupt mask. When cleared, this bit masks out the terminal count interrupt of the relevant channel.
16	L	Lock. When set, this bit enables locked transfers. This information is not used on these devices.

Table 716. DMA Channel Configuration registers (CONFIG[0:7], 0x2008 0110 (CONFIG0) to 0x2008 01F0 (CONFIG7)) bit description *…continued*

Bit	Symbol	Description
17	A	Active: 0 = there is no data in the FIFO of the channel. 1 = the channel FIFO has data. This value can be used with the Halt and Channel Enable bits to cleanly disable a DMA channel. This is a read-only bit.
18	H	Halt: 0 = enable DMA requests. 1 = ignore further source DMA requests. The contents of the channel FIFO are drained. This value can be used with the Active and Channel Enable bits to cleanly disable a DMA channel.
31:19	-	Reserved. Read value is undefined, only zero should be written.

35.5.20.1 Lock control

The lock control may set the lock bit by writing a 1 to bit 16 of the Config Register. When a burst occurs, the AHB arbiter will not de-grant the master during the burst until the lock is de-asserted. The DMA Controller can be locked for a a single burst such as a long source fetch burst or a long destination drain burst. The DMA Controller does not usually assert the lock continuously for a source fetch burst followed by a destination drain burst.

There are situations when the DMA Controller asserts the lock for source transfers followed by destination transfers. This is possible when internal conditions in the DMA Controller permit it to perform a source fetch followed by a destination drain back-to-back.

35.5.20.2 Transfer type

Table 717 lists the bit values of the transfer type bits identified in Table 716.

Table 717. Transfer type bits

Bit value	Transfer type	Flow controller
000	Memory to memory	DMA controller
001	Memory to peripheral	DMA controller
010	Peripheral to memory	DMA controller
011	Source peripheral to destination peripheral	DMA controller
100	Source peripheral to destination peripheral	Destination peripheral
101	Memory to peripheral	Destination peripheral
110	Peripheral to memory	Source peripheral
111	Source peripheral to destination peripheral	Source peripheral

35.6 Using the DMA controller

35.6.1 Programming the DMA controller

All accesses to the DMA Controller internal register must be word (32-bit) reads and writes.

35.6.1.1 Enabling the DMA controller

To enable the DMA controller set the Enable bit in the Config register.

35.6.1.2 Disabling the DMA controller

To disable the DMA controller:

- Read the EnbldChns register and ensure that all the DMA channels have been disabled. If any channels are active, see Disabling a DMA channel.
- Disable the DMA controller by writing 0 to the DMA Enable bit in the Config register.

35.6.1.3 Enabling a DMA channel

To enable the DMA channel set the channel enable bit in the relevant DMA channel configuration register. Note that the channel must be fully initialized before it is enabled.

35.6.1.4 Disabling a DMA channel

A DMA channel can be disabled in three ways:

- By writing directly to the channel enable bit. Any outstanding data in the FIFO's is lost if this method is used.
- By using the active and halt bits in conjunction with the channel enable bit.
- By waiting until the transfer completes. This automatically clears the channel.

Disabling a DMA channel and losing data in the FIFO

Clear the relevant channel enable bit in the relevant channel configuration register. The current AHB transfer (if one is in progress) completes and the channel is disabled. Any data in the FIFO is lost. The channel must be fully re-initialized before it is enabled again.

Disabling the DMA channel without losing data in the FIFO

- Set the halt bit in the relevant channel configuration register. This causes any future DMA request to be ignored.
- Poll the active bit in the relevant channel configuration register until it reaches 0. This bit indicates whether there is any data in the channel that has to be transferred.
- Clear the channel enable bit in the relevant channel configuration register

35.6.1.5 Setting up a new DMA transfer

To set up a new DMA transfer:

If the channel is not set aside for the DMA transaction:

1. Read the EnbldChns controller register and find out which channels are inactive.
2. Choose an inactive channel that has the required priority.

3. Program the DMA controller

35.6.1.6 Halting a DMA channel

Set the halt bit in the relevant DMA channel configuration register. The current source request is serviced. Any further source DMA request is ignored until the halt bit is cleared.

35.6.1.7 Programming a DMA channel

1. Choose a free DMA channel with the priority needed. DMA channel 0 has the highest priority and DMA channel 7 the lowest priority.
2. Clear any pending interrupts on the channel to be used by writing to the IntTCClear and IntErrClear register. The previous channel operation might have left interrupt active.
3. Write the source address into the SrcAddr register.
4. Write the destination address into the DestAddr register.
5. Write the address of the next LLI into the LLI register. If the transfer comprises of a single packet of data then 0 must be written into this register.
6. Write the control information into the Control register.
7. Write the channel configuration information into the Config register. If the enable bit is set then the DMA channel is automatically enabled.

35.6.2 Flow control

The device that controls the length of the packet is known as the flow controller. The flow controller is usually the DMA controller, where the packet length is programmed by software before the DMA channel is enabled. Most peripherals are not able to be the flow controller, but when this feature is supported, it can be used by either the source or destination peripheral.

When the DMA transfer is completed:

1. The DMA Controller issues an acknowledge to the peripheral in order to indicate that the transfer has finished.
2. A TC interrupt is generated, if enabled.
3. The DMA Controller moves on to the next LLI.

The following sections describe the DMA Controller data flow sequences for the four allowed transfer types:

- Memory-to-peripheral.
- Peripheral-to-memory.
- Memory-to-memory.
- Peripheral-to-peripheral.

Each transfer type other than memory-to-memory can have either the peripheral or the DMA controller as the flow controller, resulting in 8 possible control scenarios.

Table 718 indicates the request signals used for each type of transfer.

Table 718. DMA request signal usage

Transfer direction	Request generator	Flow controller
Memory-to-peripheral	Destination peripheral	DMA Controller
Peripheral-to-memory	Source peripheral	DMA Controller
Memory-to-memory	DMA Controller	DMA Controller
Source peripheral to destination peripheral	Both source and destination peripherals	DMA Controller
Memory-to-peripheral	Destination peripheral	Destination peripheral
Peripheral-to-memory	Source peripheral	Source peripheral
Source peripheral to destination peripheral	Both source and destination peripherals	Source peripheral
Source peripheral to destination peripheral	Both source and destination peripherals	Destination peripheral

35.6.2.1 Peripheral-to-memory or memory-to-peripheral DMA flow

For a peripheral-to-memory or memory-to-peripheral DMA flow, the following sequence occurs:

1. Program and enable the DMA channel.
2. Wait for a DMA request.
3. The DMA Controller starts transferring data when:
   - The DMA request goes active.
   - The DMA stream has the highest pending priority.
   - The DMA Controller is the bus master of the AHB bus.
4. If an error occurs while transferring the data, an error interrupt is generated and disables the DMA stream, and the flow sequence ends.
5. Decrement the transfer count.
6. If the transfer has completed (indicated by the transfer count reaching 0 if the DMA controller is performing flow control, or by the peripheral sending a DMA request if the peripheral is performing flow control):
   - The DMA Controller responds with a DMA acknowledge.
   - The terminal count interrupt is generated (this interrupt can be masked).
   - If the LLI Register is not 0, then reload the SrcAddr, DestAddr, LLI, and Control registers and go to back to step 2. However, if LLI is 0, the DMA stream is disabled and the flow sequence ends.

35.6.2.2 Peripheral-to-peripheral DMA flow

For a peripheral-to-peripheral DMA flow, the following sequence occurs:

1. Program and enable the DMA channel.
2. Wait for a source DMA request.
3. The DMA Controller starts transferring data when:
   - The DMA request goes active.
   - The DMA stream has the highest pending priority.
   - The DMA Controller is the bus master of the AHB bus.

4. If an error occurs while transferring the data an error interrupt is generated, the DMA stream is disabled, and the flow sequence ends.
5. Decrement the transfer count.
6. If the transfer has completed (indicated by the transfer count reaching 0 if the DMA controller is performing flow control, or by the peripheral sending a DMA request if the peripheral is performing flow control):
   - The DMA Controller responds with a DMA acknowledge to the source peripheral.
   - Further source DMA requests are ignored.
7. When the destination DMA request goes active and there is data in the DMA Controller FIFO, transfer data into the destination peripheral.
8. If an error occurs while transferring the data, an error interrupt is generated, the DMA stream is disabled, and the flow sequence ends.
9. If the transfer has completed it is indicated by the transfer count reaching 0 if the DMA controller is the flow controller. or by the peripheral sending a DMA request if the peripheral is performing flow control. The following happens:
   - The DMA Controller responds with a DMA acknowledge to the destination peripheral.
   - The terminal count interrupt is generated (this interrupt can be masked).
   - If the LLI Register is not 0, then reload the SrcAddr, DestAddr, LLI, and Control Registers and go to back to step 2. However, if LLI is 0, the DMA stream is disabled and the flow sequence ends.

35.6.2.3 Memory-to-memory DMA flow

For a memory-to-memory DMA flow the following sequence occurs:

1. Program and enable the DMA channel.
2. Transfer data whenever the DMA channel has the highest pending priority and the DMA Controller gains mastership of the AHB bus.
3. If an error occurs while transferring the data, generate an error interrupt and disable the DMA stream.
4. Decrement the transfer count.
5. If the count has reached zero:
   - Generate a terminal count interrupt (the interrupt can be masked).
   - If the LLI Register is not 0, then reload the SrcAddr, DestAddr, LLI, and Control Registers and go to back to step 2. However, if LLI is 0, the DMA stream is disabled and the flow sequence ends.

Note: Memory-to-memory transfers should be programmed with a low channel priority, otherwise other DMA channels cannot access the bus until the memory-to-memory transfer has finished, or other AHB masters cannot perform any transaction.

35.6.3 Interrupt requests

Interrupt requests can be generated when an AHB error is encountered or at the end of a transfer (terminal count), after all the data corresponding to the current LLI has been transferred to the destination. The interrupts can be masked by programming bits in the relevant Control and Config Channel Registers. The interrupt requests from all DMA

channels can be found in the RawIntTCStat and RawIntErrStat registers. The masked versions of the DMA interrupt data is contained in the IntTCStat and IntErrStat registers. The IntStat register then combines the IntTCStat and IntErrStat requests into a single register to enable the source of an interrupt to be found quickly. Writing to the IntTCClear or the IntErrClr Registers with a bit set to 1 enables selective clearing of interrupts.

35.6.3.1 Hardware interrupt sequence flow

When a DMA interrupt request occurs, the Interrupt Service Routine needs to:

1. Read the IntTCStat Register to determine whether the interrupt was generated due to the end of the transfer (terminal count). A 1 bit indicates that the transfer completed. If more than one request is active, it is recommended that the highest priority channels be checked first.
2. Read the IntErrStat Register to determine whether the interrupt was generated due to an error occurring. A 1 bit indicates that an error occurred.
3. Service the interrupt request. The channel that caused the interrupt can be determined by reading the IntStat register. If more than one request is active, the one with the highest priority should generally be serviced first.
4. For a terminal count interrupt, write a 1 to the relevant bit of the IntTCClr Register. For an error interrupt write a 1 to the relevant bit of the IntErrClr Register to clear the interrupt request.

35.6.4 Address generation

Address generation can be either incrementing or non-incrementing (address wrapping is not supported).

Some devices, especially memories, disallow burst accesses across certain address boundaries. The DMA controller assumes that this is the case with any source or destination area that is configured for incrementing addressing. This boundary is assumed to be aligned with the specified burst size. For example, if the channel is set for 16-transfer burst to a 32-bit wide device then the boundary is 64-bytes aligned (that is address bits [5:0] equal 0). If a DMA burst is to cross one of these boundaries, then, instead of a burst, that transfer is split into separate AHB transactions.

35.6.4.1 Word-aligned transfers across a boundary

The channel is configured for 16-transfer bursts, each transfer 32-bits wide, to a destination for which address incrementing is enabled. The start address for the current burst is 0x0C00 0024, the next boundary (calculated from the burst size and transfer width) is 0x0C00 0040.

The transfer will be split into two AHB transactions:

- a 7-transfer burst starting at address 0x0C00 0024
- a 9-transfer burst starting at address 0x0C00 0040.

35.6.5 Scatter/gather

Scatter/gather is supported through the use of linked lists. This means that the source and destination areas do not have to occupy contiguous areas in memory. Where scatter/gather is not required, the LLI Register must be set to 0.

The source and destination data areas are defined by a series of linked lists. Each Linked List Item (LLI) controls the transfer of one block of data, and then optionally loads another LLI to continue the DMA operation, or stops the DMA stream. The first LLI is programmed into the DMA Controller.

The data to be transferred described by an LLI (referred to as the packet of data) usually requires one or more DMA bursts (to each of the source and destination).

35.6.5.1 Linked list items

A Linked List Item (LLI) consists of four words. These words are organized in the following order:

1. SrcAddr.
2. DestAddr.
3. LLI.
4. Control.

Note: The Config DMA channel Configuration Register is not part of the linked list item.

35.6.5.1.1 Programming the DMA controller for scatter/gather DMA

To program the DMA Controller for scatter/gather DMA:

1. Write the LLIs for the complete DMA transfer to memory. Each linked list item contains four words:
   - Source address.
   - Destination address.
   - Pointer to next LLI.
   - Control word.

   The last LLI has its linked list word pointer set to 0.
2. Choose a free DMA channel with the priority required. DMA channel 0 has the highest priority and DMA channel 7 the lowest priority.
3. Write the first linked list item, previously written to memory, to the relevant channel in the DMA Controller.
4. Write the channel configuration information to the channel Configuration Register and set the Channel Enable bit. The DMA Controller then transfers the first and then subsequent packets of data as each linked list item is loaded.
5. An interrupt can be generated at the end of each LLI depending on the Terminal Count bit in the Control Register. If this bit is set an interrupt is generated at the end of the relevant LLI. The interrupt request must then be serviced and the relevant bit in the IntTCClear Register must be set to clear the interrupt.

35.6.5.1.2 Example of scatter/gather DMA

See Figure 167 for an example of an LLI. A section of memory is to be transferred to a peripheral. The addresses of each LLI entry are given, in hexadecimal, at the left-hand side of the figure. In this example, the LLIs describing the transfer are to be stored contiguously from address 0x2002 0000, but they could be located anywhere. The right side of the figure shows the memory containing the data to be transferred.

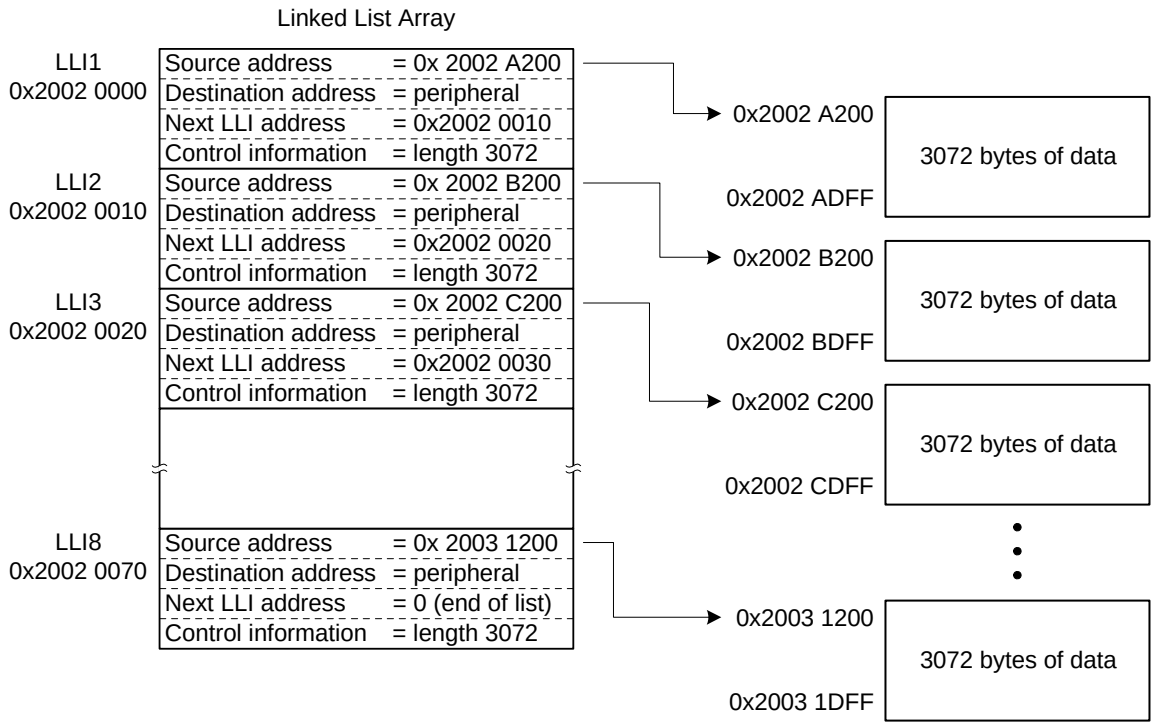


Fig 167. LLI example

The first LLI, stored at 0x2002 0000, defines the first block of data to be transferred, which is the data stored from address 0x2002 A200 to 0x2002 ADFF:

- Source start address 0x2002 A200.
- Destination address set to the destination peripheral address.
- Transfer width, word (32-bit).
- Transfer size, 3072 bytes (0XC00).
- Source and destination burst sizes, 16 transfers.
- Next LLI address, 0x2002 0010.

The second LLI, stored at 0x2002 0010, describes the next block of data to be transferred:

- Source start address 0x2002 B200.
- Destination address set to the destination peripheral address.
- Transfer width, word (32-bit).
- Transfer size, 3072 bytes (0xC00).
- Source and destination burst sizes, 16 transfers.
- Next LLI address, 0x2002 0020.

A chain of descriptors is built up, each one pointing to the next in the series. To initialize the DMA stream, the first LLI, 0x2002 0000, is programmed into the DMA Controller. When the first packet of data has been transferred the next LLI is automatically loaded.

The final LLI is stored at 0x2002 0070 and contains:

- Source start address 0x2003 1200.
- Destination address set to the destination peripheral address.
- Transfer width, word (32-bit).
- Transfer size, 3072 bytes (0xC00).
- Source and destination burst sizes, 16 transfers.
- Next LLI address, 0x0.

Because the next LLI address is set to zero, this is the last descriptor, and the DMA channel is disabled after transferring the last item of data. The channel is probably set to generate an interrupt at this point to indicate to the ARM processor that the channel can be reprogrammed.

36.1 Introduction

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used. To save system power and bus bandwidth, the CRC engine supports DMA transfers in addition to software PIO operations using the CPU.

36.2 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 or DMA 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)

36.3 Description

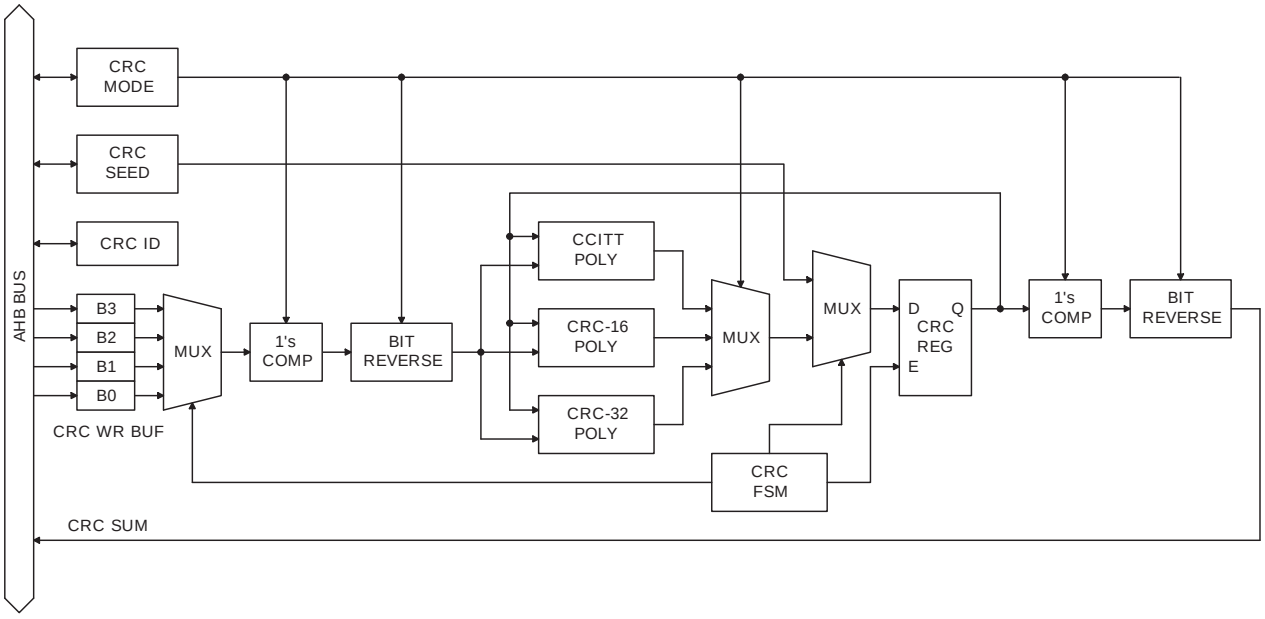


Fig 168. CRC block diagram

36.4 Register description

Table 719. Register overview: CRC engine (base address 0x2009 0000)

Name	Access	Address offset	Description	Reset value[1]	Table
MODE	R/W	0x000	CRC mode register	0	720
SEED	R/W	0x004	CRC seed register	0xFFFF	721
SUM	RO	0x008	CRC checksum register	0xFFFF	722
DATA	WO	0x008	CRC data register	-	723

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

36.4.1 CRC mode register

Table 720. CRC mode register (MODE - address 0x2009 0000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CRC_POLY		Select CRC polynomial	0
		0x0	CRC-CCITT polynomial	
		0x1	CRC-16 polynomial	
		0x2	CRC-32 polynomial	
2	BIT_RVS_WR		Select bit order for CRC_WR_DATA	0
		0	No bit order reverse for CRC_WR_DATA (per byte)	
		1	Bit order reverse for CRC_WR_DATA (per byte)	
3	CMPL_WR		Select one's complement for CRC_WR_DATA	0
		0	No one's complement for CRC_WR_DATA	
		1	One's complement for CRC_WR_DATA	
4	BIT_RVS_SUM		Select bit order revers for CRC_SUM	0
		0	No bit order reverse for CRC_SUM	
		1	Bit order reverse for CRC_SUM	
5	CMPL_SUM		Select one's complement for CRC_SUM	0
		0	No one's complement for CRC_SUM	
		1	One's complement for CRC_SUM	
31:6	Reserved		Always '0' when read	0

36.4.2 CRC seed register

Table 721. CRC seed register (SEED - address 0x2009 0004) 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:** Writing a new seed value to this register essentially starts a new CRC with that seed.	0xFFFF

36.4.3 CRC checksum register

This register is a read-only register containing the most recent checksum.

Table 722. CRC checksum register (SUM - address 0x2009 0008) 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.	0xFFFF

36.4.4 CRC data register

This register is a write-only register containing the data block for which the CRC sum will be calculated.

Table 723. CRC data register (DATA - address 0x2009 0008) bit description

Bit	Symbol	Description
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.

36.5 Functional description

The following sections describe the register settings for each supported CRC standard:

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

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

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

37.1 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., or to save power. See Section 37.5.1.7).
2. Clocking: Timing for the EEPROM must be set up before it can be used. See Section 37.5.1.5 and Section 37.5.1.6.
3. Interrupts: Interrupts are controlled using a set of registers, see Section 37.5.2. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register, see Table 51.

37.2 Description

EEPROM is a non-volatile memory mainly used for storing relatively small amounts of data, for example for application settings. The EEPROM is indirectly accessed through address and data registers, so the CPU cannot execute code from EEPROM memory.

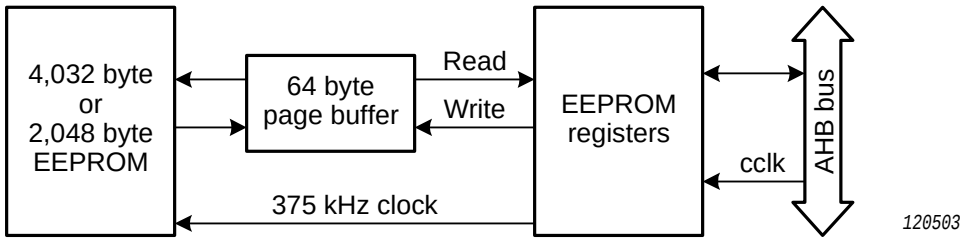


Fig 169. EEPROM block diagram

All communication with the actual EEPROM block is done through the 64 byte page buffer.

37.3 Features

- 4,032 bytes EEPROM on most devices
- Access via address and data registers on the AHB bus
- Less than 3 ms erase / program time
- Endurance of > 100k erase / program cycles

37.4 EEPROM operation

37.4.1 EEPROM device description

EEPROM is a non-volatile memory mainly used for storing relatively small amounts of data, for example for storing settings.

There are three operations for accessing the memory: reading, writing and erase/program. "Writing" to memory is split up into two separate operations, writing and erase/program. The first operation, which will be called "writing" in this document, is not really updating the memory, but only updating the temporary data register called the "page register". The page register needs to be written with a minimum 1 byte and a maximum 64 bytes before the second operation, which is called "erase/program" in this document, can be used to actually update the non-volatile memory. Note that the data written to the page register is not "cached"; it cannot be read before it is actually programmed into non-volatile memory.

The 64-byte page register is the same size as a page in EEPROM memory. The 4,032 bytes EEPROM on most devices contains 63 pages. Devices with a 2 kB EEPROM provide 2,048 bytes on 32 pages.

37.4.2 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. The page register (64 bytes) will temporarily hold write data. But as soon as this data needs to be read from the EEPROM or data needs to be written to another page, the contents of the page register first needs to be programmed into the EEPROM memory.

The following sections explain the EEPROM operations (read, write and erase/program) in more detail.

37.4.2.1 Writing

The EEPROM controller supports writing of 8-bit, 16-bit, or 32-bit elements. Since the EEPROM device doesn’t directly support 32-bit write operations, the controller splits the operation into two 16-bit operations.

For doing a write operation first an address needs to be written into the address register and the kind of write operation needs to be selected in the command register. This can be done in any order. After this the data is written to the write data register, which automatically starts the write operation on the EEPROM device.

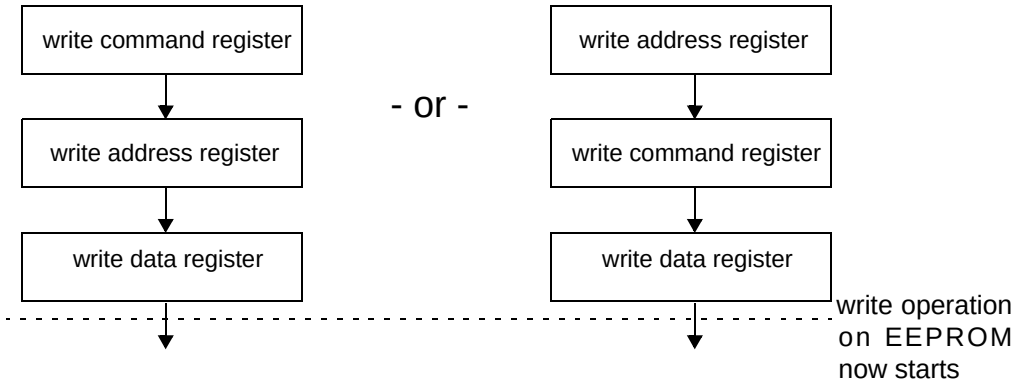


Fig 170. Starting a write operation

A write operation causes an automatic post-increment of the address. This allows consecutive writes to the page register without the need of writing a new address for every write operation. Of course the address register could be written with another address value to write to another location.

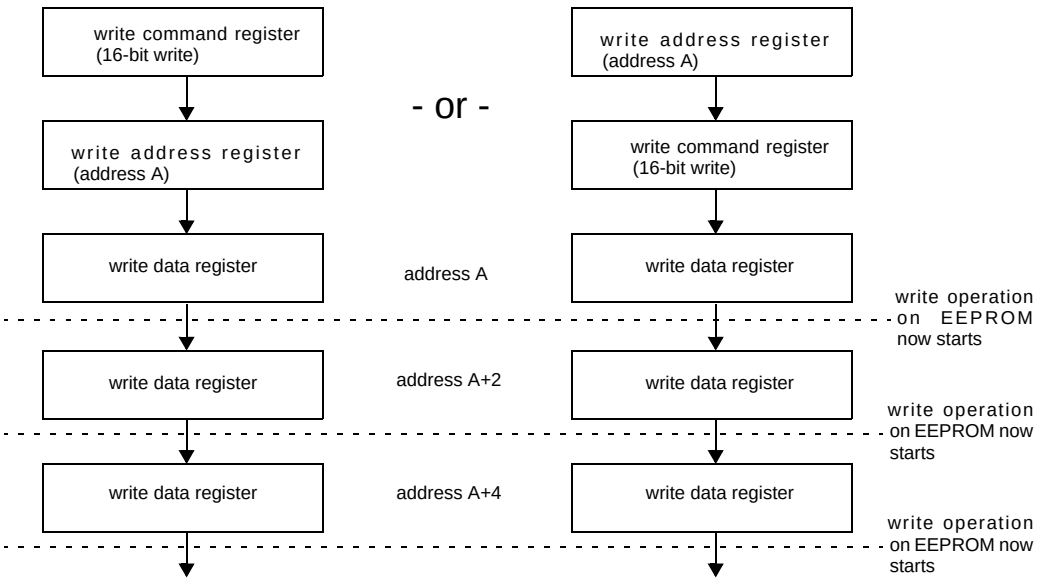


Fig 171. (16-bit) write operations with post-incrementing of address

If the data register is written while a previous EEPROM operation is still pending, the write transfer on the system bus is stalled until the previous operation is finished. This can be avoided by polling the interrupt status register to see if an operation is still pending before starting the write operation.

Software has to make sure that the following rules are followed:

- overwriting (writing it two times before an erase/program operation) one of the locations in a 64-byte page register is not allowed. It will cause the loss of the previously written data

- in case the default address post-incrementing is used, the upper boundary of the page register may not be crossed
- the contents of the page register needs to be programmed into non-volatile memory before it can be read back
- write operations to a misaligned address will result in an error response on the write transfer to the write data register (for example a 32-bit write operation to an address other than a multiple of 0x4). The operation will not be performed.

37.4.2.2 Erase/Programming

After the page register has been written with user data, it still has to programmed into non-volatile memory. This is a separate step. Only writing to the page register will not write the EEPROM memory.

Programming the page into memory takes a relatively long time, therefore the corresponding interrupt can be enabled, or the interrupt status bit can be polled to avoid stalling of the system bus.

An erase/program operation starts by providing the MSBs of the address that selects the page in memory. The 6 LSBs are "don't care". The operation is started by writing the command register (selecting the erase/program operation). Before beginning a programming operation, the EEPROM status should be polled to insure that the last write operation has been completed.

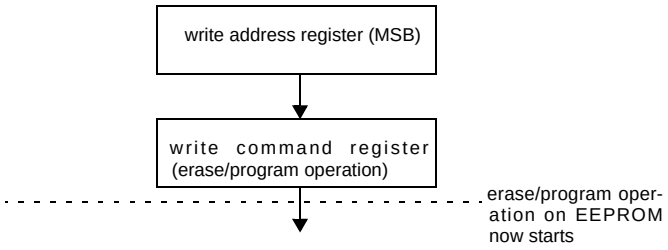


Fig 172. Programming a page into memory

37.4.2.3 Reading

The EEPROM controller supports reading 8-bit, 16-bit, or 32-bit elements. Since the EEPROM device doesn't support 32-bit operations the controller splits the operation into two 16-bit operations.

For doing a read operation, an address first needs to be written into the address register. Then the operation needs to be selected in the command register. Writing the command register will automatically start the read operation on the EEPROM device.

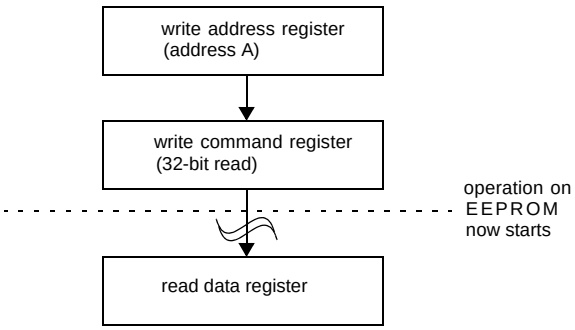


Fig 173. Starting a read operation (32-bit read from address A)

If the read data register is read while the read operation is still pending, then the read transfer on the system bus is stalled until the previous read operation is finished. This can be avoided by polling the interrupt status register to see if the operation is still pending before reading the read data register.

Read operations will automatically post-increment the address register. This allows consecutive reads from the EEPROM memory without the need of writing a new address for every read operation. By setting the read data prefetch bit in the command register, reading from the read data register automatically starts up a read operation from the next (incremented) address location. When doing consecutive reads in this way, the first read operation is started as result of writing the command register. The following read operations are started as result of reading the read data register to obtain the result of the previous read operations.

Read operations from a misaligned address will result on an error response on the write transfer to the command register (for example a 32-bit read operation from an address other than a multiple of 0x4). The operation will not be performed.

37.4.2.4 Exceptions

The controller can generate exceptions in the following situations:

- Writing a read-only register or reading a write-only register
- A transfer to a non-existing register location

37.5 Register description

Table 724 shows the registers related to EEPROM operation. Details of each register follow.

Table 724. Register overview: EEPROM controller (base address 0x0020 0000)

Name	Access	Address offset	Description	Reset value[1]	Table
EEPROM registers					
CMD	R/W	0x080	EEPROM command register	0	725
ADDR	R/W	0x084	EEPROM address register	0	726
WDATA	WO	0x088	EEPROM write data register	-	727
RDATA	RO	0x08C	EEPROM read data register	-	728
WSTATE	R/W	0x090	EEPROM wait state register	0	729
CLKDIV	R/W	0x094	EEPROM clock divider register	0	730
PWRDWN	R/W	0x098	EEPROM power-down register	0	731
EEPROM interrupt registers:					
INTSTAT	RO	0xFE0	EEPROM interrupt status	0	732
INTSTATCLR	WO	0xFE8	EEPROM interrupt status clear	-	733
INTSTATSET	WO	0xFEC	EEPROM interrupt status set	-	734
INTEN	RO	0xFE4	EEPROM interrupt enable	0	735
INTENCLR	WO	0xFD8	EEPROM interrupt enable clear	-	736
INTENSET	WO	0xFDC	EEPROM interrupt enable set	-	737

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

37.5.1 EEPROM control registers

37.5.1.1 EEPROM command register

The EEPROM command register is used to select and start a read, write or erase/program operation. Read and erase/program operations are started on the EEPROM device as a side-effect of writing to this register. (Write operations are started as a side-effect of writing to the write data register).

Table 725. EEPROM command register (CMD - address 0x0020 0080) bit description

Bits	Symbol	Description	Reset value
2:0	CMD	Command. 000: 8-bit read 001: 16-bit read 010: 32-bit read 011: 8-bit write 100: 16-bit write 101: 32-bit write 110: erase/program page 111: reserved	0
3	RDPREFETCH	Read data prefetch bit. 0: do not prefetch next read data as result of reading from the read data register. 1: prefetch read data as result of reading from the read data register. When this bit is set multiple consecutive data elements can be read without the need of programming new address values in the address register. The address post-increment and the automatic read data prefetch (if enabled) allow only reading from the read data register to be done to read the data.	0
31:4	-	Reserved. Read value is undefined, only zero should be written.	NA

37.5.1.2 EEPROM address register

The EEPROM address register is used to program the address for read, write or erase/program operations.

Table 726. EEPROM address register (ADDR - address 0x0020 0084) bit description

Bits	Symbol	Description	Reset value
11:0	ADDR	EEPROM Address. Lower 6 bits are don't care.	0
31:12	-	Reserved. Read value is undefined, only zero should be written.	NA

37.5.1.3 EEPROM write data register

The EEPROM write data register is used to write data into the page register (write operations).

Writing this register will start the write operation as a side-effect. The address is post-incremented, so consecutive writes to this register can be done to write a burst of data, up to a maximum of 64 bytes. The address will be incremented automatically according to the data size of the write operation.

If data is written to this register while a previous operation (read, write or an erase/program) is still pending or being processed, the write command on the AHB is stalled until the previous operation is finished. To avoid stalling of the system bus, the interrupt status register can be used for polling the status of pending operations.

Table 727. EEPROM write data register (WDATA - address 0x0020 0088) bit description

Bits	Symbol	Description
31:0	WDATA	Write data. In case of: 8-bit write operations: bits [7:0] must contain valid write data. 16-bit write operations: bits [15:0] must contain valid write data. 32-bit write operations: bits [31:0] must contain valid write data.

37.5.1.4 EEPROM read data register

The EEPROM read data register is used to read data from the non-volatile memory.

Reading this register will start the next read operation, and the address will be post-incremented. Consecutive reads from this register can be done to read a burst of data. The address will be incremented automatically according to the data size of the read operation.

If data is read from this register while the read operation is still pending the read command on the AHB bus is stalled until the pending operation is finished. To avoid stalling of the system bus, the interrupt status register can be used for polling the status of pending operations.

Table 728. EEPROM read data register (RDATA - address 0x0020 008C) bit description

Bits	Symbol	Description
31:0	RDATA	Read data. 8-bit read operations: bits [7:0] contain the valid read data, others are zero. 16-bit read operations: bits [15:0] contain the valid read data, others are zero. 32-bit read operations: bits [31:0] contain the valid read data.

37.5.1.5 EEPROM wait state register

The EEPROM controller uses the times specified by this register to perform various internal timing functions. The user must program the wait state fields with appropriate values for proper EEPROM erase/program operation. Programming a zero will result in a one cycle wait state.

Example with cclk=120 MHz:

PHASE3 = (15ns * cclk) - 1 = (15ns * 120MHz) - 1 = 1.8

Therefore, PHASE3 = 2

PHASE2=(55ns * cclk) - 1 = (55ns * 120MHz) - 1 = 5.8

Therefore, PHASE2 = 6

PHASE1=(35ns * cclk) - 1 = (35ns * 120MHz) - 1 = 3.2

Therefore, PHASE3 = 4

Table 729. EEPROM wait state register (WSTATE - address 0x0020 0090) bit description

Bits	Symbol	Description	Reset value
7:0	PHASE3	Wait states 3 (minus 1 encoded). The number of system clock periods required to give a minimum time of 15 ns.	0
15:8	PHASE2	Wait states 2 (minus 1 encoded). The number of system clock periods required to give a minimum time of 55 ns.	0
23:16	PHASE1	Wait states 1 (minus 1 encoded). The number of system clock periods required to give a minimum time of 35 ns.	0
31:24	-	Reserved. Read value is undefined, only zero should be written.	NA

37.5.1.6 EEPROM clock divider register

The EEPROM requires a 375 kHz ±6.67% clock. 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 will be IDLE to save power.

$$CLKDIV = \frac{cclk}{375kHz} - 1$$

For example, if the CPU clock is 80 MHz:

CLKDIV = (cclk / 375 kHz) - 1 = (80 MHz / 375 kHz) - 1 = 212

Table 730. EEPROM clock divider register (CLKDIV - address 0x0020 0094) bit description

Bits	Symbol	Value	Description	Reset value
15:0	CLKDIV		EEPROM division factor.	0
		0	The EEPROM clock will be idled in order to save power	
		1	cclk is divided by 2	
		2	cclk is divided by 3	
		:	:	
		0xFFFF	cclk is divided by 65536	
31:16	-		Reserved. Read value is undefined, only zero should be written.	NA

37.5.1.7 EEPROM power down register

The EEPROM power-down register can be used to put the EEPROM in power-down mode.

The EEPROM may not be put 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 731. EEPROM power down/DCM register (PWRDWN - address 0x0020 0098) 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

37.5.2 Interrupt registers

These registers control interrupts from the EEPROM.

37.5.2.1 Interrupt status register

Table 732. Interrupt status register (INTSTAT - address 0x0020 0FE0) bit description

Bits	Symbol	Description	Reset value
25:0	-	Reserved. The value read from a reserved bit is not defined.	NA
26	END_OF_RDWR	Interrupt status bit for EEPROM read/write operation finished. This bit is set when this operation has finished OR when '1' is written in the corresponding bit of the INTSTATSET register. This bit is cleared when '1' is written to the corresponding bit of the INTSTATCLR register.	0
27	-	Reserved. The value read from a reserved bit is not defined.	NA
28	END_OF_PROG1	Interrupt status bit for EEPROM program operation finished. This bit is set when this operation has finished OR when '1' is written to the corresponding bit of the INTSTATSET register. This bit is cleared when '1' is written to the corresponding bit of the INTSTATCLR register.	0
31:29	-	Reserved. The value read from a reserved bit is not defined.	NA

The interrupt request output is asserted when the bit-wise AND of INTSTAT and INTEN is nonzero. For the EEPROM read/write operation finished interrupt it is better not to enable the interrupt, but to only poll the bit in the INTSTAT register. This is because these operations are relatively fast operations that do not justify calling a interrupt service subroutine in software.

37.5.2.2 Interrupt status clear register

Table 733. Interrupt status clear register (INTSTATCLR - address 0x0020 0FE8) bit description

Bits	Symbol	Description
25:0	-	Reserved. Read value is undefined, only zero should be written.
26	RDWR_CLR_ST	Clear read/write operation finished interrupt status bit for the EEPROM. 0 leave corresponding bit unchanged. 1 clear corresponding bit.
27	-	Reserved. Read value is undefined, only zero should be written.
28	PROG1_CLR_ST	Clear program operation finished interrupt status bit for the EEPROM. 0 leave corresponding bit unchanged. 1 clear corresponding bit.
31:29	-	Reserved. Read value is undefined, only zero should be written.

37.5.2.3 Interrupt status set

Table 734. Interrupt status set register (INTSTATSET - address 0x0020 0FEC)

Bits	Symbol	Description
25:0	-	Reserved. Read value is undefined, only zero should be written.
26	RDWR_SET_ST	Set read/write operation finished interrupt status bit for the EEPROM. 0 leave the corresponding bit in the INTSTAT register unchanged. 1 set the corresponding bit in the INTSTAT register.
27	-	Reserved. Read value is undefined, only zero should be written.
28	PROG1_SET_ST	Set program operation finished interrupt status bit for the EEPROM. 0 leave the corresponding bit in the INTSTAT register unchanged. 1 set the corresponding bit in the INTSTAT register.
31:29	-	Reserved. Read value is undefined, only zero should be written.

37.5.2.4 Interrupt enable register

Table 735. Interrupt enable register (INTEN - address 0x0020 0FE4) bit description

Bits	Symbol	Description	Reset value
25:0	-	Reserved. The value read from a reserved bit is not defined.	NA
26	EE_RW_DONE	Interrupt enable bit for EEPROM read/write operation finished. This bit is set when ‘1’ is written to the corresponding bit of the INTENSET register. This bit is cleared when ‘1’ is written to the corresponding bit of the INTENCLR register.	0
27	-	Reserved. The value read from a reserved bit is not defined.	NA
28	EE_PROG_DONE	Interrupt enable bit for EEPROM program operation finished. This bit is set when ‘1’ is written to the corresponding bit of the INTENSET register. This bit is cleared when ‘1’ is written to the corresponding bit of the INTENCLR register.	0
31:29	-	Reserved. The value read from a reserved bit is not defined.	NA

This is a Read-only register. Changes to this register must be made using the INTENSET and INTENCLR registers. The interrupt request output is asserted when the bit-wise AND of INTSTAT and INTEN is a non-zero value. For EEPROM read/write completed operations, it is better not to enable the interrupt, but to poll the bit in the INTSTAT register. This is because these operations are relatively fast and do not justify calling an interrupt service subroutine in software.

37.5.2.5 Interrupt enable clear register

Table 736. Interrupt enable clear register (INTENCLR - address 0x0020 0FD8) bit description

Bits	Symbol	Description
25:0	-	Reserved. Read value is undefined, only zero should be written.
26	RDWR_CLR_EN	Clear the read/write operation finished interrupt enable bit for the EEPROM. 0: leave the corresponding bit in the INTEN register unchanged. 1: clear the corresponding bit in the INTEN register.

Table 736. Interrupt enable clear register (INTENCLR - address 0x0020 0FD8) bit description

Bits	Symbol	Description
27	-	Reserved. Read value is undefined, only zero should be written.
28	PROG1_CLR_EN	Clear the program operation finished interrupt enable bit for the EEPROM. 0: leave the corresponding bit in the INTEN register unchanged. 1: clear the corresponding bit in the INTEN register.
31:29	-	Reserved. Read value is undefined, only zero should be written.

37.5.2.6 Interrupt enable set register

Table 737. Interrupt enable set register (INTENSET - address 0x0020 0FDC) bit description

Bits	Symbol	Description
25:0	-	Reserved. Read value is undefined, only zero should be written.
26	RDWR_SET_EN	Set the read/write operation finished interrupt enable bit for the EEPROM. 0: leave the corresponding bit in the INTEN register unchanged. 1: set the corresponding bit in the INTEN register.
27	-	Reserved. Read value is undefined, only zero should be written.
28	PROG1_SET_EN	Set the program operation finished interrupt enable bit for the EEPROM. 0: leave the corresponding bit in the INTEN register unchanged. 1: set the corresponding bit in the INTEN register.
31:29	-	Reserved. Read value is undefined, only zero should be written.

38.1 Introduction

The boot loader controls initial operation after reset and also provides the tools for programming 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.

38.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and UART0 serial port. 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.
- Flash signature generation: built-in hardware can generate a signature for a range of flash addresses, or for the entire flash memory.

38.3 Description

The flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A LOW level after reset at pin P2[10] is considered an external hardware request to start the ISP command handler using UART0 pins P0[2] (U0_TXD) and P0[3] (U0_RXD). Assuming that power supply pins are on their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before P2[10] is sampled and the decision on whether to continue with user code or ISP handler is made. If P2[10] is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P2[10] is sampled HIGH after reset), a search is made for a valid user program. 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.

Pin P2[10] is used as a hardware request signal for ISP and therefore requires special attention. Since P2[10] is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

When ISP mode is entered after a power on reset, the IRC frequency of 12 MHz is used to operate the CPU and peripherals. The baud rates that can easily be obtained in this case are: 9600 baud, 19200 baud, 38400 baud, 57600 baud, and 115200 baud.

A hardware flash signature generation capability is built into the flash memory. this feature can be used to create a signature that can then be used to verify flash contents. Details of flash signature generation are in Section 38.10.

38.3.1 Memory map after any reset

When a user program begins execution after reset, the interrupt vectors are set to point to the beginning of flash memory.

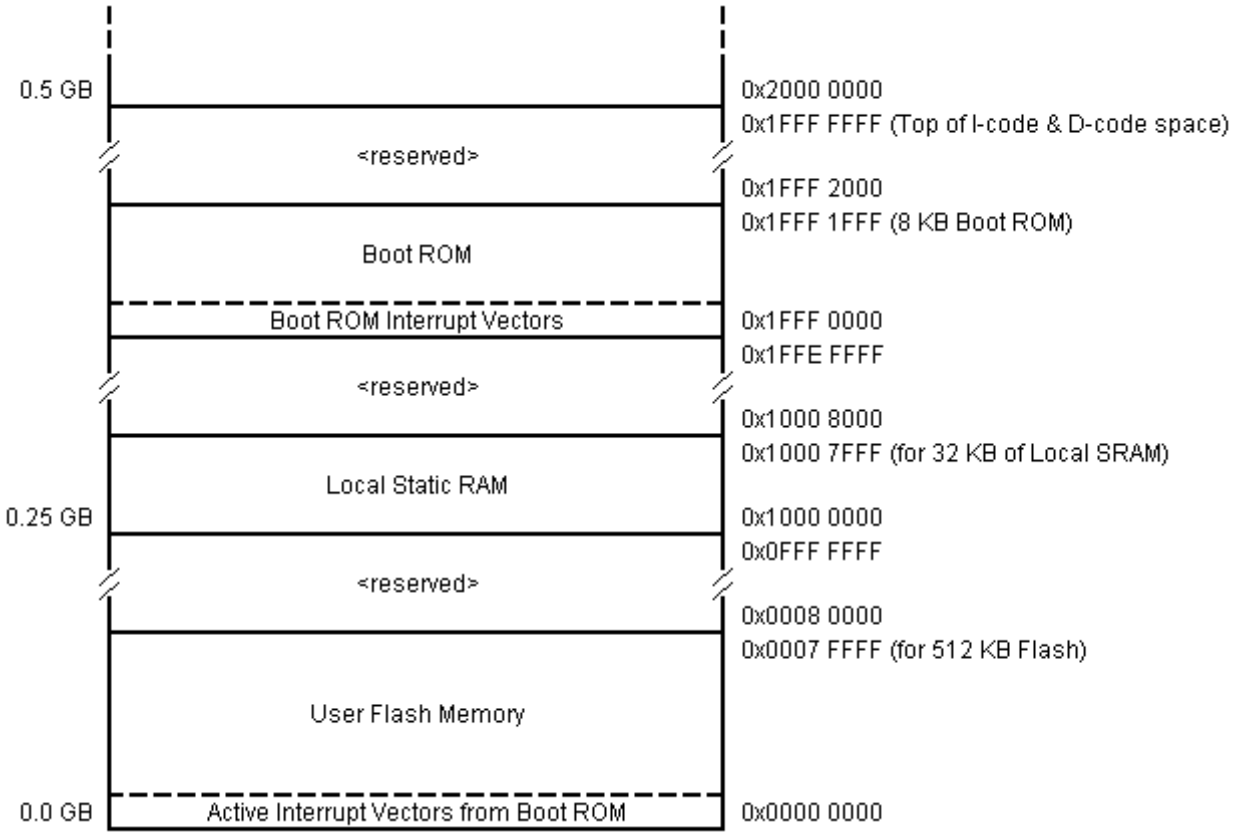


Fig 174. Map of lower memory

38.3.1.1 Criterion for Valid User Code

The reserved Cortex-M4 exception vector location 7 (offset 0x001C in the vector table) should contain the 2's complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The boot loader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. 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 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. For example, if the part is running at 10 MHz, the response from the host should be "10000<CR><LF>". "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 CCLK frequency should be greater than or equal to 10 MHz.

For more details on Reset, PLL and startup/boot code interaction see Section 3.10.1 “PLL and startup/boot code interaction”.

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 38.7 “ISP commands” on page 884.

38.3.2 Communication protocol

All 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 UU-encoded format.

38.3.2.1 ISP command format

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

38.3.2.2 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).

38.3.2.3 ISP data format

The data stream is in UU-encoded format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

38.3.2.4 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (0x13) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (0x11). The host should also support the same flow control scheme.

38.3.2.5 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC" (0x1B). This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

38.3.2.6 Interrupts during IAP

The on-chip flash memory is not accessible during IAP operations. When the user application code starts executing, the interrupt vectors from the user flash area are active. The user should either disable interrupts, or ensure that user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM, before making an IAP call (see Section 5.4 “Vector table remapping”). The IAP code does not use or disable interrupts.

38.3.2.7 Addresses in IAP and ISP commands

IAP and ISP commands that reference memory addresses have a limited range. The command descriptions in Section 38.7 “ISP commands” and Section 38.8 “IAP commands” note RAM address or flash address or both. RAM addresses must be located

in on-chip RAM, addresses outside those ranges will be flagged as errors. Flash addresses must be located in on-chip Flash memory, addresses outside that range will be flagged as errors.

38.3.2.8 RAM used by ISP command

ISP commands use on-chip RAM from 0x1000 0118 to 0x1000 01FF. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 256 bytes, growing downwards.

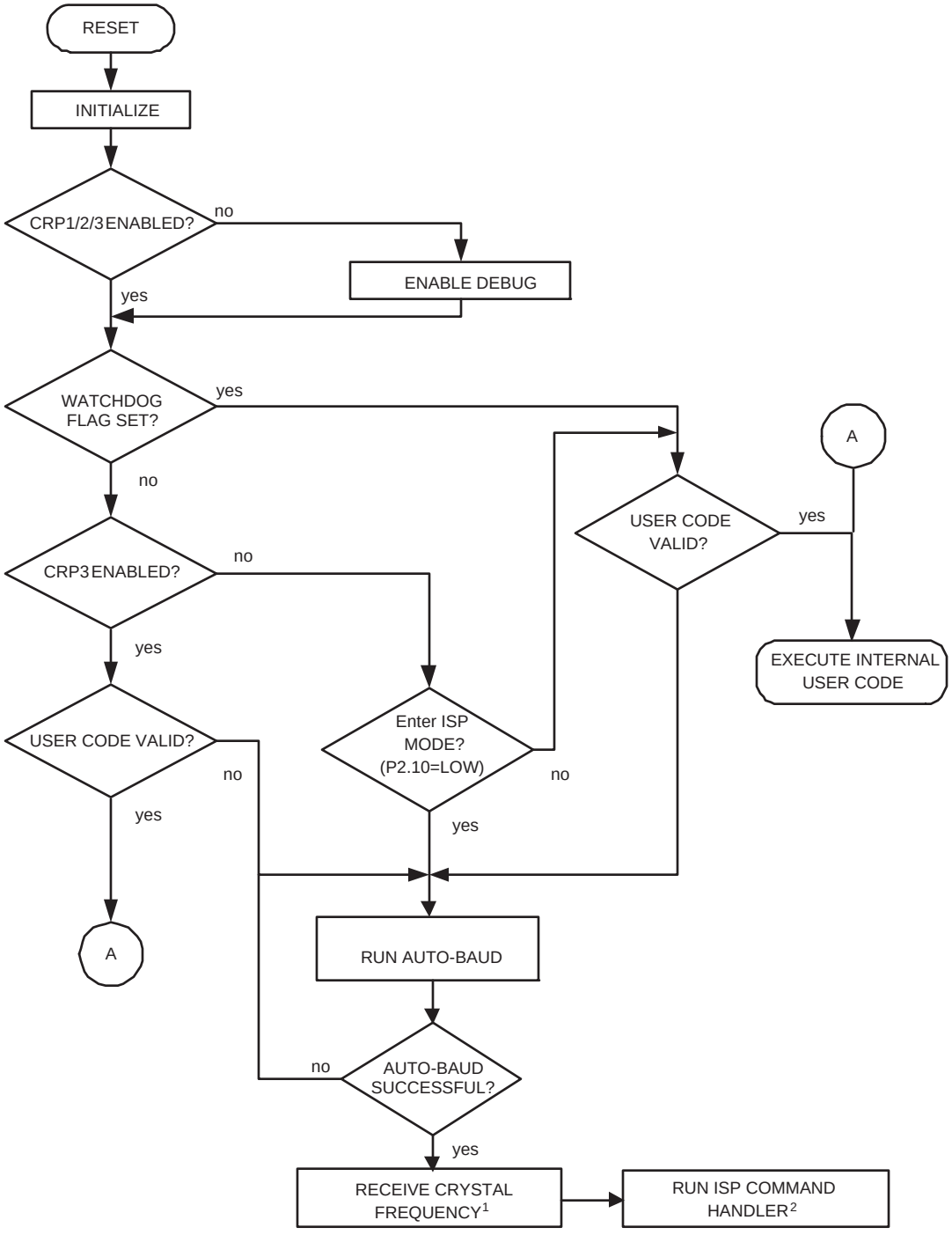
38.3.2.9 RAM used by Boot process prior to entering user program

Following chip reset, the Boot program uses a subset of the RAM that is used for ISP command handling. This includes location 0x1000 0120 and also parts of the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 32 bytes. If the user program assumes that RAM is unchanged during a reset where power is not removed from the device, it is important to be aware of these exceptions.

38.3.2.10 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes, growing downwards.

38.4 Boot process flowchart



(1) For details on handling the crystal frequency, see Section 38.8.9 “Re-invoke ISP” on page 898

(2) For details on available ISP commands based on the CRP settings see Section 38.6 “Code Read Protection (CRP)”

Fig 175. Boot process flowchart

38.5 Sector numbers

Some IAP and ISP commands operate on "sectors" and specify sector numbers. The following table indicate the correspondence between sector numbers and memory addresses for devices containing 32, 64, 128, 256 and 512 kB of flash respectively. IAP and ISP routines are located in the Boot ROM.

Table 738. Flash sectors details

Sector Number	Sector Size [kB]	Start Address	End Address	32 kB Device	64 kB Device	128 kB Device	256 kB Device	512 kB Device
0	4	0X0000 0000	0X0000 0FFF	x	x	x	x	x
1	4	0X0000 1000	0X0000 1FFF	x	x	x	x	x
2	4	0X0000 2000	0X0000 2FFF	x	x	x	x	x
3	4	0X0000 3000	0X0000 3FFF	x	x	x	x	x
4	4	0X0000 4000	0X0000 4FFF	x	x	x	x	x
5	4	0X0000 5000	0X0000 5FFF	x	x	x	x	x
6	4	0X0000 6000	0X0000 6FFF	x	x	x	x	x
7	4	0X0000 7000	0X0000 7FFF	x	x	x	x	x
8	4	0X0000 8000	0X0000 8FFF		x	x	x	x
9	4	0X0000 9000	0X0000 9FFF		x	x	x	x
10 (0x0A)	4	0X0000 A000	0X0000 AFFF		x	x	x	x
11 (0x0B)	4	0X0000 B000	0X0000 BFFF		x	x	x	x
12 (0x0C)	4	0X0000 C000	0X0000 CFFF		x	x	x	x
13 (0x0D)	4	0X0000 D000	0X0000 DFFF		x	x	x	x
14 (0X0E)	4	0X0000 E000	0X0000 EFFF		x	x	x	x
15 (0x0F)	4	0X0000 F000	0X0000 FFFF		x	x	x	x
16 (0x10)	32	0x0001 0000	0x0001 7FFF			x	x	x
17 (0x11)	32	0x0001 8000	0x0001 FFFF			x	x	x
18 (0x12)	32	0x0002 0000	0x0002 7FFF				x	x
19 (0x13)	32	0x0002 8000	0x0002 FFFF				x	x
20 (0x14)	32	0x0003 0000	0x0003 7FFF				x	x
21 (0x15)	32	0x0003 8000	0x0003 FFFF				x	x
22 (0x16)	32	0x0004 0000	0x0004 7FFF					x
23 (0x17)	32	0x0004 8000	0x0004 FFFF					x
24 (0x18)	32	0x0005 0000	0x0005 7FFF					x
25 (0x19)	32	0x0005 8000	0x0005 FFFF					x
26 (0x1A)	32	0x0006 0000	0x0006 7FFF					x
27 (0x1B)	32	0x0006 8000	0x0006 FFFF					x
28 (0x1C)	32	0x0007 0000	0x0007 7FFF					x
29 (0x1D)	32	0x0007 8000	0x0007 FFFF					x

38.6 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in flash location at 0x0000 02FC. IAP commands are not affected by the code read protection.

Important: Any CRP change becomes effective only after the device has gone through a power cycle.

Table 739. Code Read Protection options

Name	Pattern programmed in 0x0000 02FC	Description
CRP1	0x1234 5678	Access to chip via the JTAG pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command can not access RAM below 0x1000 0200. This is due to use of the RAM by the ISP code, see Section 38.3.2.8. • Read Memory command: disabled. • Copy RAM to Flash command: cannot write to Sector 0. • Go command: disabled. • Erase sector(s) command: can erase any individual sector except sector 0 only, or can erase all sectors at once. • Compare command: disabled This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. The compare command is disabled, so in the case of partial flash updates the secondary loader should implement a checksum mechanism to verify the integrity of the flash.
CRP2	0x8765 4321	This is similar to CRP1 with the following additions: • Write to RAM command: disabled. • Copy RAM to Flash: disabled. • Erase command: only allows erase of all sectors.
CRP3	0x4321 8765	This is similar to CRP2, but ISP entry by pulling P2[10] LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using the P2[10] pin. It is up to the user's application to provide for flash updates by using IAP calls or by invoking ISP with UART0. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

Table 740. Code Read Protection hardware/software interaction

CRP option	User Code Valid	P2[10] pin at reset	JTAG enabled	Device enters ISP mode	Partial flash update in ISP mode
None	No	X	Yes	Yes	Yes
	Yes	High	Yes	No	NA
	Yes	Low	Yes	Yes	Yes
CRP1	No	x	No	Yes	Yes
	Yes	High	No	No	NA
	Yes	Low	No	Yes	Yes
CRP2	No	x	No	Yes	No
	Yes	High	No	No	NA
	Yes	Low	No	Yes	No
CRP3	No	x	No	Yes	No
	Yes	x	No	No	NA

If any CRP mode is enabled and access to the chip is allowed via the ISP, an unsupported or restricted ISP command will be terminated with return code CODE_READ_PROTECTION_ENABLED.

38.7 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 741. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 742
Set Baud Rate	B <Baud Rate> <stop bit>	Table 743
Echo	A <setting>	Table 744
Write to RAM	W <start address> <number of bytes>	Table 745
Read Memory	R <address> <number of bytes>	Table 746
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 747
Copy RAM to Flash	C <flash address> <RAM address> <number of bytes>	Table 748
Go	G <address> <Mode>	Table 749
Erase sector(s)	E <start sector number> <end sector number>	Table 750
Blank check sector(s)	I <start sector number> <end sector number>	Table 751
Read Part ID	J	Table 752
Read Boot Code version	K	Table 754
Read serial number	N	Table 755
Compare	M <address1> <address2> <number of bytes>	Table 756

38.7.1 Unlock <Unlock code>

Table 742. ISP Unlock command

Command	U
Input	Unlock code: 23130 decimal
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command is used to unlock Flash Write, Erase, and Go commands.
Example	"U 23130<CR><LF>" unlocks the Flash Write/Erase & Go commands.

38.7.2 Set Baud Rate <Baud Rate> <stop bit>

Table 743. 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 is used to change 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.

When the Set Baud Rate command is used after ISP has been re-invoked by a user program (using the “Re-invoke ISP” IAP command, see Section 38.8.9), the clocking setup is returned to the initial state, i.e. running from the IRC with the PLL disconnected.

38.7.3 Echo <setting>

Table 744. 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.

38.7.4 Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less than 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 745. ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. The address should be on a word boundary. The source address must be within an on-chip RAM (see Section 38.3.2.7). **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. Data should be in UU-encoded format. This command is blocked when code read protection levels CRP2 or CRP3 are enabled.
Example	"W 268435968 4<CR><LF>" writes 4 bytes of data to address 0x1000 0200.

38.7.5 Read Memory <address> <no. of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less than 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 746. ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. The address should be on a word boundary. The address must be within on-chip flash or on-chip RAM (see Section 38.3.2.7). **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (UU-encoded)> | 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 any level of code read protection is enabled.
Example	"R 268435968 4<CR><LF>" reads 4 bytes of data from address 0x1000 0200.

38.7.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 747. ISP Prepare sector(s) 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 0<CR><LF>" prepares the flash sector 0.

38.7.7 Copy RAM to Flash <flash address> <RAM address> <no of bytes>

Table 748. ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be on a 256 byte boundary. The destination address must be within the on-chip flash memory (see Section 38.3.2.7). **RAM Address(SRC):** Source RAM address from where data bytes are to be read. The source address must be within an on-chip RAM (see Section 38.3.2.7). **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 levels CRP2 or CRP3 are enabled. When code read protection level CRP1 is enabled, individual sectors other than sector 0 can be written.
Example	"C 0 268468224 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 8000 to the flash address 0.

38.7.8 Go <address> <mode>

Table 749. 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. The address must be within on-chip flash or on-chip RAM (see Section 38.3.2.7). **Mode (retained for backward compatibility):** T (Execute program in Thumb Mode) | A (not allowed).
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. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when any level of code read protection is enabled.
Example	"G 0 T<CR><LF>" branches to address 0x0000 0000.

When the GO command is used, execution begins at the specified address (assuming it is an executable address) with the device left as it was configured for the ISP code. This means that some things are different than they would be for entering user code directly following a chip reset. The CPU will be running from the 12 MHz IRC with the PLLs turned off.

38.7.9 Erase sector(s) <start sector number> <end sector number>

Table 750. 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 is blocked when code read protection level CRP3 is enabled. When code read protection level CRP1 is enabled, individual sectors other than sector 0 can be erased. All sectors can be erased at once in CRP1 and CRP2.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

38.7.10 Blank check sector(s) <sector number> <end sector number>

Table 751. 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.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

38.7.11 Read Part Identification number

Table 752. ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 753 “Part identification numbers”).
Description	This command is used to read the part identification number. The part identification number maps to a feature subset within a device family. This number will not normally change as a result of technical revisions.

Table 753. Part identification numbers

Device	ASCII/dec coding	Hex coding
LPC4088	1209876295	0x481D 3F47
LPC4078	1192836935	0x4719 3F47
LPC4076	1192828739	0x4719 1F43
LPC4074	1191252274	0x4701 1132

38.7.12 Read Boot Code version number

Table 754. 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.

38.7.13 Read device serial number

Table 755. ISP Read device serial number command

Command	N
Input	None.
Return Code	CMD_SUCCESS followed by the device serial number in 4 decimal ASCII groups, each representing a 32-bit value.
Description	This command is used to read the device serial number. The serial number may be used to uniquely identify a single unit among all devices of the same part number.

38.7.14 Compare <address1> <address2> <no of bytes>

Table 756. ISP Compare command

Command	M
Input	**Address1 (DST):** Starting flash or RAM address of data bytes to be compared. The address should be on a word boundary. The address must be within on-chip flash or on-chip RAM (see Section 38.3.2.7). **Address2 (SRC):** Starting flash or RAM address of data bytes to be compared. The address should be on a word boundary. The address must be within on-chip flash or on-chip RAM (see Section 38.3.2.7). **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 |
Description	This command is used to compare the memory contents at two locations. This command is blocked when any level of code read protection is enabled.
Example	"M 8192 268435968 4<CR><LF>" compares 4 bytes from the RAM address 0x1000 0200 to the 4 bytes from the flash address 0x2000.

38.7.15 ISP Return Codes

Table 757. ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on 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 or end sector number is greater than start sector number.
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 not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.
19	CODE_READ_PROTECTION_ENABLED	Code read protection enabled.

38.8 IAP commands

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 from the IAP command is returned in the table pointed to by register r1. The user can reuse the command table for the result by passing the same pointer in registers r0 and r1. The parameter table should be large enough to hold all of the results in case the number of results are greater than the number of parameters. Parameter passing is illustrated in the Figure 176. 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 4, returned by the "Read device serial number" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at location 0x1FFF 1FF0.

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

Define the IAP location entry point. Bit 0 of the IAP location is set since the Cortex-M4 uses only Thumb mode.

```
#define IAP_LOCATION 0x1FFF1FF1
```

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

```
unsigned long command[5];
unsigned long output[5];
```

or

```
unsigned long * command;
unsigned long * output;
command=(unsigned long *) 0x...
output= (unsigned long *) 0x...
```

Define a 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 function pointer:

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, output);
```

The IAP call could be simplified further by using the symbol definition file feature supported by ARM Linker in ADS (ARM Developer Suite). You could also call the IAP routine using assembly code.

Note that the first entry in the command table is the IAP command, followed by any required command parameters, starting with Param0. The first entry in the output table is the Return Code, followed by any other results, starting with Result0.

As per the ARM specification (The ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05) up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively. 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 758. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50 decimal	Table 759
Copy RAM to Flash	51 decimal	Table 760
Erase sector(s)	52 decimal	Table 761
Blank check sector(s)	53 decimal	Table 762
Read part ID	54 decimal	Table 763
Read Boot Code version	55 decimal	Table 764
Read device serial number	58 decimal	Table 765
Compare	56 decimal	Table 766
Reinvoke ISP	57 decimal	Table 767

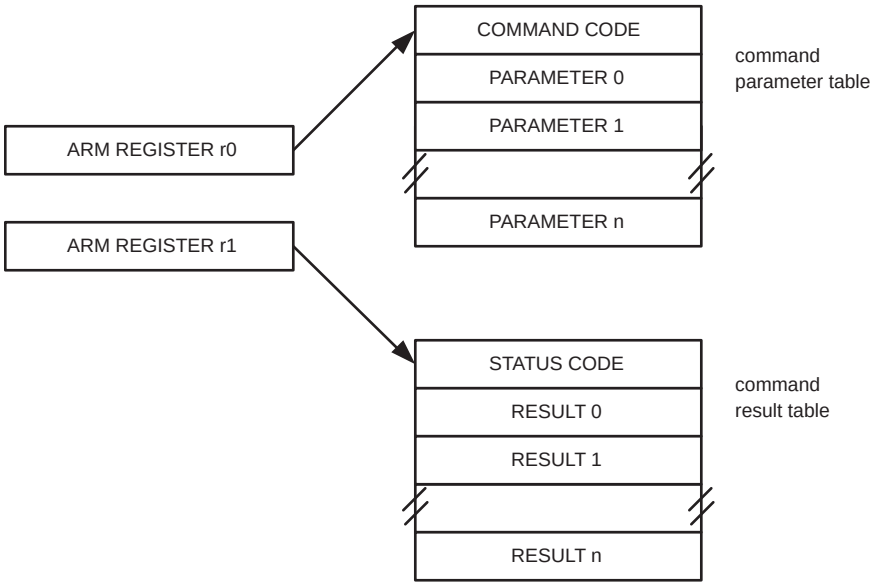


Fig 176. IAP parameter passing

38.8.1 Prepare sector(s) for write operation

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

Table 759. 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).
Return 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.

38.8.2 Copy RAM to Flash

Table 760. 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 should be on a 256 byte boundary. The destination address must be within the on-chip flash memory (see Section 38.3.2.7). **Param1(SRC):** Source RAM address from which data bytes are to be read. The address should be on a word boundary. The source address must be within an on-chip RAM (see Section 38.3.2.7). **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **Param3:** CPU Clock Frequency (CCLK) in kHz.
Return 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 |
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.

38.8.3 Erase Sector(s)

Table 761. IAP Erase Sector(s) command

Command	Erase Sector(s)
Input	**Command code: 52 decimal** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number). **Param2:** CPU Clock Frequency (CCLK) in kHz.
Return 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.

38.8.4 Blank check sector(s)

Table 762. IAP Blank check sector(s) command

Command	Blank check sector(s)
Input	**Command code: 53 decimal** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Return 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.

38.8.5 Read part identification number

Table 763. IAP Read part identification number command

Command	Read part identification number
Input	**Command code: 54 decimal** **Parameters:** None
Return Code	CMD_SUCCESS
Result	**Result0:** Part Identification Number.
Description	This command is used to read the part identification number. The value returned is the hexadecimal version of the part ID. See Table 753 “Part identification numbers”.

38.8.6 Read Boot Code version number

Table 764. IAP Read Boot Code version number command

Command	Read boot code version number
Input	**Command code: 55 decimal**
	Parameters: None
Return Code	CMD_SUCCESS
Result	**Result0:** 2 bytes of boot code version number. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

38.8.7 Read device serial number

Table 765. IAP Read device serial number command

Command	Read device serial number
Input	**Command code: 58 decimal**
	Parameters: None
Return Code	CMD_SUCCESS
Result	**Result0:** First 32-bit word of Device Identification Number (at the lowest address)
	Result1: Second 32-bit word of Device Identification Number
	Result2: Third 32-bit word of Device Identification Number
	Result3: Fourth 32-bit word of Device Identification Number
Description	This command is used to read the device identification number. The serial number may be used to uniquely identify a single unit among all devices of the same part number.

38.8.8 Compare <address1> <address2> <no of bytes>

Table 766. IAP Compare command

Command	Compare
Input	**Command code: 56 decimal**
	Param0(DST): Starting flash or RAM address of data bytes to be compared. The address should be on a word boundary. The address must be within on-chip flash or on-chip RAM (see Section 38.3.2.7).
	Param1(SRC): Starting flash or RAM address of data bytes to be compared. The address should be on a word boundary. The address must be within on-chip flash or on-chip RAM (see Section 38.3.2.7).
	Param2: Number of bytes to be compared; should be a multiple of 4.
Return 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.
	The result may not be correct when the source or destination includes any of the first 64 bytes starting from address zero. The first 64 bytes can be re-mapped to RAM.

38.8.9 Re-invoke ISP

Table 767. Re-invoke ISP

Command	Compare
Input	**Command code: 57 decimal**
Return Code	None
Result	**None.**
Description	This command is used to invoke the boot loader in ISP mode. It maps boot vectors, resets the clocking configuration, configures UART0 pins Rx and Tx, resets TIMER1 and resets the U0FDR (see Section 18.6.11). This command may be used when a valid user program is present in the internal flash memory and the P2[10] pin is not accessible to force the ISP mode.

38.8.10 IAP Status Codes

Table 768. 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 in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to 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.

38.9 JTAG flash programming interface

Debug tools can write parts of the flash image to the RAM and then execute the IAP call "Copy RAM to Flash" repeatedly with proper offset.

38.10 Flash signature generation

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 (e.g. during programming).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 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 (e.g. 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.

38.10.1 Register description for signature generation

Table 769. Register overview: Flash controller (base address 0x0020 0000)

Name	Access	Address offset	Description	Reset Value	Reference
FMSSTART	R/W	0x020	Signature start address register	0	Table 770
FMSSTOP	R/W	0x024	Signature stop-address register	0	Table 771
FMSW0	RO	0x02C	128-bit signature Word 0	-	Table 772
FMSW1	RO	0x030	128-bit signature Word 1	-	Table 773
FMSW2	RO	0x034	128-bit signature Word 2	-	Table 774
FMSW3	RO	0x038	128-bit signature Word 3	-	Table 775
STAT	RO	0xFE0	Signature generation status register	0	Table 776
STATCLR	WO	0xFE8	Signature generation status clear register	-	Table 777

38.10.1.1 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.

Table 770 and Table 771 show the bit assignments in the FMSSTART and FMSSTOP registers respectively.

Table 770. Flash Module Signature Start register (FMSSTART - 0x0020 0020) 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

Table 771. Flash Module Signature Stop register (FMSSTOP - 0x0020 0024) bit description

Bit	Symbol	Value	Description	Reset Value
16:0	STOP		BIST stop address divided by 16 (corresponds to AHB byte address [20:4]).	0
17	SIG_START		Start control bit for signature generation.	0
		0	Signature generation is stopped	
		1	Initiate signature generation	
31:18	-		Reserved. Read value is undefined, only zero should be written.	NA

38.10.1.2 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 makes saves time and code space. The method for generating the signature is described in Section 38.10.2.

Table 775 show bit assignment of the FMSW0 and FMSW1, FMSW2, FMSW3 registers respectively.

Table 772. FMSW0 register bit description (FMSW0, address: 0x0020 002C)

Bit	Symbol	Description	Reset Value
31:0	SW0_31_0	Word 0 of 128-bit signature (bits 31 to 0).	-

Table 773. FMSW1 register bit description (FMSW1, address: 0x0020 0030)

Bit	Symbol	Description	Reset Value
31:0	SW1_63_32	Word 1 of 128-bit signature (bits 63 to 32).	-

Table 774. FMSW2 register bit description (FMSW2, address: 0x0020 0034)

Bit	Symbol	Description	Reset Value
31:0	SW2_95_64	Word 2 of 128-bit signature (bits 95 to 64).	-

Table 775. FMSW3 register bit description (FMSW3, address: 0x0020 0038)

Bit	Symbol	Description	Reset Value
31:0	SW3_127_96	Word 3 of 128-bit signature (bits 127 to 96).	-

38.10.1.3 Flash Module 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 776. Flash module Status register (STAT - 0x0020 0FE0) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. The value read from a reserved bit is not defined.	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. The value read from a reserved bit is not defined.	NA

38.10.1.4 Flash Module Status Clear register

The FMSTATCLR register is used to clear the signature generation completion flag.

Table 777. Flash Module Status Clear register (STATCLR - 0x0x0020 0FE8) bit description

Bit	Symbol	Description
1:0	-	Reserved. Read value is undefined, only zero should be written.
2	SIG_DONE_CLR	Writing a 1 to this bits clears the signature generation completion flag (SIG_DONE) in the FMSTAT register.
31:2	-	Reserved. Read value is undefined, only zero should be written.

38.10.2 Algorithm and procedure for signature generation

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.

The signature generation is started by writing a '1' to FMSSTOP.MISR_START. Starting the signature generation is typically combined with defining the stop address, which is done in another field FMSSTOP.FMSSTOP of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. Reading of the flash memory for signature generation uses a self-timed read mechanism and does not depend on any configurable timing settings for the flash. A safe estimation for the duration of the signature generation is:

Duration = int((60 / 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.

If signature generation is triggered via JTAG, the duration is in JTAG tck cycles, and tcy is the time in ns for one JTAG clock. Polling the SIG_DONE bit in FMSTAT is not possible in this case.

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.

Content verification

The signature as it is read from the FMSW0 to FMSW3 registers must be equal to the reference signature. The algorithms to derive the reference signature is given in Figure 177.

```
sign = 0
FOR address = FMSTART.FMSTART TO FMSTOP.FMSTOP
{
        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
```

Fig 177. Algorithm for generating a 128 bit signature

39.1 Features

- Supports both standard JTAG and ARM Serial Wire Debug modes.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Trace port provides CPU instruction trace capability.
- Eight Breakpoints. 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 that can also be used as trace triggers.
- Instrumentation Trace Macrocell allows additional software controlled trace capability.

39.2 Introduction

Debug and trace functions are integrated into the ARM Cortex-M4. Serial wire debug and trace functions are supported in addition to a standard JTAG debug and parallel trace functions. The ARM Cortex-M4 is configured to support up to eight breakpoints and four watchpoints.

39.3 Description

Debugging with the LPC408x/407x defaults to JTAG. Once in the JTAG debug mode, the debug tool can switch to Serial Wire Debug mode.

Instruction trace is supported by a 4-bit parallel interface using 5 pins. Note that the trace function available for the Cortex-M4 is functionally very different than the trace that was available for previous ARM7 based devices, using only 5 pins instead of 10.

39.4 Pin description

The tables below indicate the various pin functions related to debug and trace. Some of these functions share pins with other functions which therefore may not be used at the same time. Use of the JTAG port excludes use of Serial Wire Debug and Serial Wire Output. Use of the parallel trace requires 5 pins that may be part of the user application, limiting debug possibilities for those features.

Table 778. JTAG pin description

Pin Name	Type	Description
JTAG_TCK	Input	**JTAG Test Clock.** This pin is the clock for debug logic when in the JTAG debug mode.
JTAG_TMS	Input	**JTAG Test Mode Select.** The TMS pin selects the next state in the TAP state machine. This pin includes an internal pull-up for compliance with IEEE 1149.1.
JTAG_TDI	Input	**JTAG Test Data In.** This is the serial data input for the shift register. This pin includes an internal pull-up for compliance with IEEE 1149.1.
JTAG_TDO	Output	**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.
$\overline{\text{JTAG\_TRST}}$	Input	**JTAG Test Reset.** The $\overline{\text{JTAG\_TRST}}$ pin can be used to reset the test logic within the debug logic. This pin includes an internal pull-up for compliance with IEEE 1149.1.

Table 779. Serial Wire Debug pin description

Pin Name	Type	Description
SWDCLK	Input	**Serial Wire Clock.** This pin is the clock for debug logic when in the Serial Wire Debug mode. This is an internally selected alternate function for the JTAG_TCK pin.
SWDIO	Input / Output	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the Cortex-M4 CPU. This is an internally selected alternate function for the JTAG_TMS pin.
SWO	Output	**Serial Wire Output.** The SWO pin optionally provides data from the ITM and/or the ETM for an external debug tool to evaluate. This is an internally selected alternate function for the JTAG_TDO pin.

Table 780. Parallel Trace pin description

Pin Name	Type	Description
TRACECLK	Input	**Trace Clock.** This pin provides the sample clock for trace data on the TRACEDATA pins when tracing is enabled by an external debug tool.
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.

39.5 Debug connections

The LPC408x/407x supplies dedicated pins for JTAG and Serial Wire Debug (SWD). When a debug session is started, the part will be in JTAG debug mode as recommended by ARM Ltd at the time of design. Once in debug mode, the debugger can switch the device to SWD mode.

Connections from a target board to the debugger can vary. Selecting a debug connector to add to a new board design depends on the debug tools that will be used. For example, debug tools for ARM-based devices in the past have used a standard connection as shown in Figure 178. This diagram has been adapted to fit the LPC408x/407x, taking into account the pins that have built-in pull-ups.

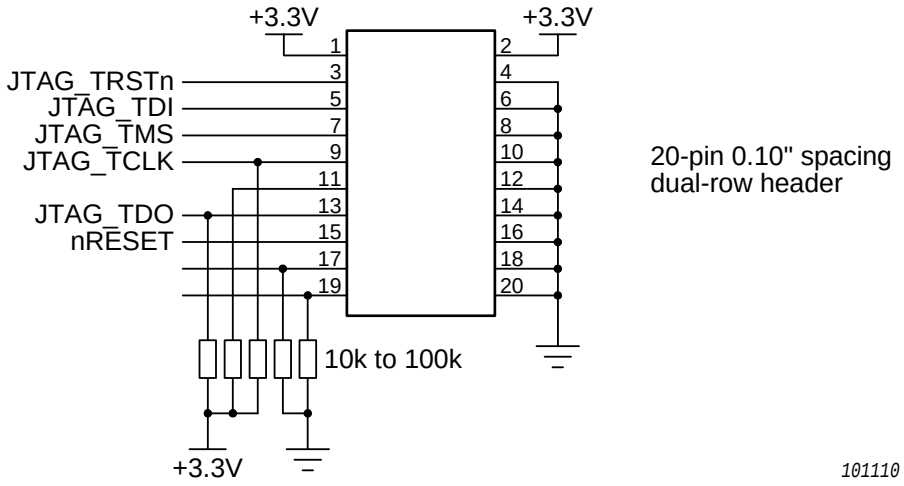


Fig 178. ARM Standard JTAG Connector

Newer tools may use a small debug-only connector as shown in Figure 179. If the debug trace feature will be used, there is also a debug-with-trace connector specification as shown in Figure 180. These 2 connector pinouts are defined in ARM Ltd's CoreSight™ Components Technical Reference Manual. Please note that any debug connection scheme should be checked with the tool vendor before an application board is designed.

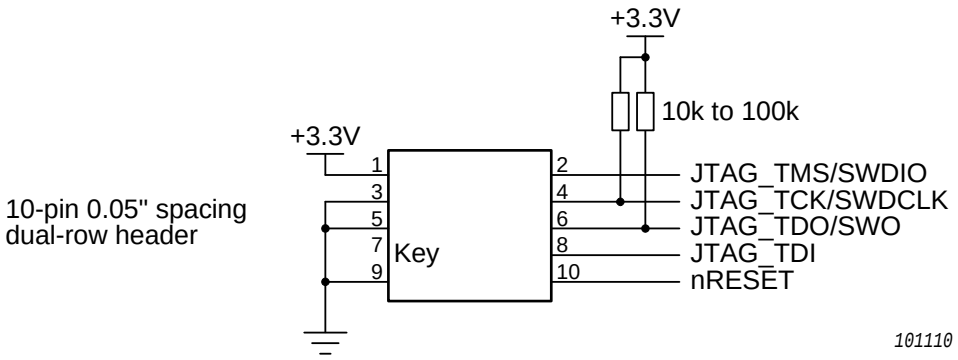


Fig 179. Cortex Debug Connector

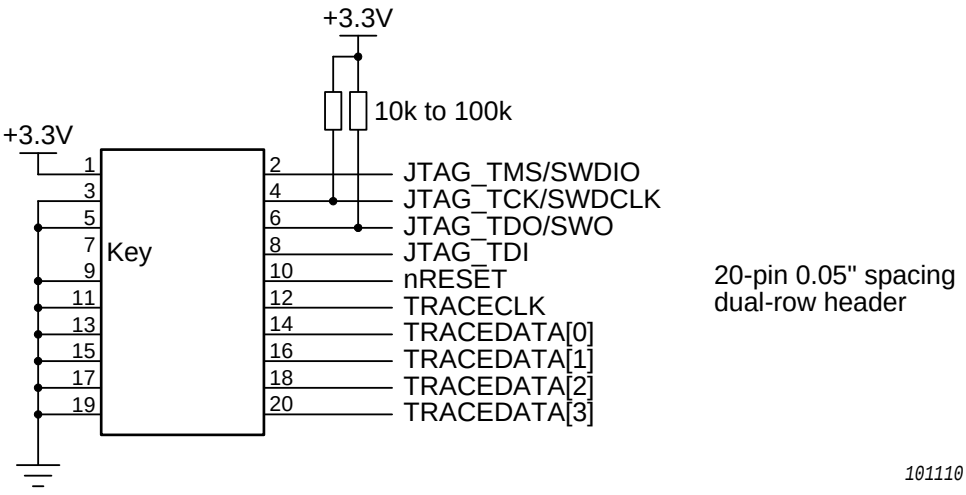


Fig 180. Cortex Debug & ETM Connector

39.6 JTAG TAP Identification

The JTAG TAP controller contains device ID that can be used by debugging software to identify the general type of device. More detailed device information is available through ISP/IAP commands (see Section 38.7 and Section 38.8). For the LPC408x/407x family, this ID value is 0x410F C241.

39.7 Debug Notes

Important: The user should be aware of certain limitations during debugging. The most important is that, due to limitations of the Cortex-M4 integration, the device cannot wake up in the usual manner from Deep Sleep and Power-down modes. It is recommended not to use these modes during debug.

Once an application is downloaded via JTAG/SWD interface, the USB to SWD/JTAG debug adapter should be removed from the target board, and thereafter, power cycle the device to allow wake-up from Deep Sleep and Power-down modes.

Another issue is that debug mode changes the way in which reduced power modes are handled by the Cortex-M4 CPU. This causes power modes at the debug level to be different from normal mode operation. These differences mean that power measurements should not be made while debugging, the results will be higher than during normal operation in an application.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

Debugging is disabled if code read protection is enabled.

39.8 Debug memory re-mapping

Following chip reset, a portion of the Boot ROM is mapped to address 0 so that it will be automatically executed. The Boot ROM switches the map to point to Flash memory prior to user code being executed. In this way a user normally does not need to know that this re-mapping occurs.

However, when a debugger halts CPU execution immediately following reset, the Boot ROM is still mapped to address 0 and can cause confusion. Ideally, the debugger should correct the mapping automatically in this case, so that a user does not need to deal with it.

39.8.1 Memory Mapping Control register

The MEMMAP register allows switch the mapping of the bottom of memory, including default reset and interrupt vectors, between the Boot ROM and the bottom of on-chip Flash memory.

Table 781. Memory Mapping Control register (MEMMAP - 0x400F C040) bit description

Bit	Symbol	Value	Description	Reset value
0	MAP		Memory map control.	0
		0	Boot mode. A portion of the Boot ROM is mapped to address 0.	
		1	User mode. The on-chip Flash memory is mapped to address 0.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	NA

40.1 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 searchable, hyperlinked version:

  `infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0553a/index.html`

- for the Adobe PDF formatted file:

  `infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf`

This section of this manual describes the Cortex-M4 implementation options and any other distinctions that apply for the LPC408x/407x devices.

40.1.1 Cortex-M4 implementation options

The ARM document “Cortex™-M4 Devices Generic User Guide” lists a number of implementation options. These options and the selections for the LPC408x/407x are given below.

- Inclusion of MPU: LPC408x/407x devices include the MPU. 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.
- Inclusion of FPU: LPC408x/407x devices include the FPU. 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.
- Number of interrupts: LPC408x/407x devices implement 44 interrupts. Not all interrupts are available on all part numbers.
- Number of priority bits: LPC408x/407x devices implement 5 interrupt priority bits.
- Inclusion of the WIC: LPC408x/407x devices include the WIC.
- Sleep mode power-saving: NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M4. Details all available reduced power modes and wake-up possibilities on the LPC408x/407x can be found in Section 3.12 “Power control”.
- Register reset values: LPC408x/407x devices do not reset the register bank when the device is reset.
- Endianness: LPC408x/407x devices use little endian memory organization. Specific peripheral blocks (such as an external memory controller, DMA controller, or LCD controller) may support little endian organization for special purposes.
- Memory features: The memory map for LPC408x/407x devices can be found in Section 2.2 “Memory maps”.
- Bit-banding: LPC408x/407x devices include bit banding. APB peripherals and the Peripheral SRAMs are located in bit-band space.

- SysTick timer: The SysTick Calibration register is implemented on LPC408x/407x devices, for details see Section 25.5 “Register description”.

In addition, there are debug and trace options:

- Debug options: see Section 39.1.
- Trace options: see Section 39.1.

41.1 Abbreviations

Table 782. Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
CAN	Controller Area Network
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
EMC	External Memory Controller
EOP	End Of Packet
ETM	Embedded Trace Macrocell
GPIO	General Purpose Input/Output
I2C	Inter-IC Control bus
I2S	Inter-IC Sound bus
IrDA	Infrared Data Association
JTAG	Joint Test Action Group
MII	Media Independent Interface (Ethernet related)
MIIM	Media Independent Interface Management (Ethernet related)
PHY	Physical layer interface
PLL	Phase-Locked Loop
PWM	Pulse Width Modulator
QEI	Quadrature Encoder Interface
RMII	Reduced Media Independent Interface (Ethernet related)
SE0	Single Ended Zero (USB related)
SPI	Serial Peripheral Interface
SPIFI	SPI Flash Interface
SSI	Serial Synchronous Interface
SSP	Synchronous Serial Port
SWD	Serial Wire Debug
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

41.2 References

[1] LPC408x/7x data sheet:
http://www.nxp.com/documents/data_sheet/LPC408X_7X.pdf

[2] LPC408x/7x Errata sheet:
http://www.nxp.com/documents/errata_sheet/ES_LPC407X_8X.pdf

41.3 Legal information

41.3.1 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.

41.3.2 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

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 and its suppliers accept 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.

41.3.3 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.

41.4 Tables

Table 1. Ordering information 8
Table 2. Ordering options 8
Table 3. Memory usage and details 14
Table 4. AHB peripherals and base addresses 17
Table 5. APB0 peripherals and base addresses 17
Table 6. APB1 peripherals and base addresses 18
Table 7. Matrix Arbitration register (MATRIXARB - 0x400F C188) bit description. 19
Table 8. Pin summary. 22
Table 9. Register overview: System control (base address 0x400F C000). 23
Table 10. PLL Control registers (PLLCON[0:1] - addresses 0x400F C080 (PLLCON0) and 0x400F C0A0 (PLLCON1)) bit description. 25
Table 11. PLL Configuration registers (PLLCFG[0:1] - addresses 0x400F C084 (PLLCFG0) and 0x400F C0A4 (PLLCFG1)) bit description 25
Table 12. PLL Status registers (PLLSTAT[0:1] - addresses 0x400F C088 (PLLSTAT0) and 0x400F C0A8 (PLLSTAT1)) bit description 26
Table 13. PLL Feed registers (PLLFEED[0:1] - addresses 0x400F C08C (PLLFEED0) and 0x400F C0AC (PLLFEED1)) bit description. 27
Table 14. Power Mode Control register (PCON - address 0x400F C0C0) bit description. 28
Table 15. Encoding of reduced power modes 29
Table 16. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description 30
Table 17. Power Control for Peripherals register (PCONP1 - address 0x400F C0C8) bit description 31
Table 18. Power Boost control register (PBOOST - address 0x400F C1B0) bit description 31
Table 19. EMC Clock Selection register (EMCCLKSEL - address 0x400F C100) bit description 32
Table 20. CPU Clock Selection register (CCLKSEL - address 0x400F C104) bit description 33
Table 21. USB Clock Selection register (USBCLKSEL - address 0x400F C108) bit description 33
Table 22. Clock Source Selection register (CLKSRCSEL - address 0x400F C10C) bit description 34
Table 23. Peripheral Clock Selection register (PCLKSEL - address 0x400F C1A8) bit description 34
Table 24. SPIFI Clock Selection register (SPIFICLKSEL - address 0x400F C1B4) bit description 35
Table 25. External Interrupt Flag register (EXTINT - address 0x400F C140) bit description 36
Table 26. External Interrupt Mode register (EXTMODE - address 0x400F C148) bit description 37
Table 27. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description 38
Table 28. Reset Source Identification register (RSID - address 0x400F C180) bit description 39
Table 29. Reset control register 0 (RSTCON0 - address 0x400F C1CC) bit description. 40
Table 30. Reset control register 1 (RSTCON1 - address 0x400F C1D0) bit description 41
Table 31. Delay Control register (EMCDLYCTL - 0x400F C1DC) bit description. 42
Table 32. EMC Calibration register (EMCCAL - 0x400F C1E0) bit description 43
Table 33. System Controls and Status register (SCS - address 0x400F C1A0) bit description. 45
Table 34. IRC control register (IRCCTRL - address 0x400F C1A4) bit description. 46
Table 35. LCD Configuration register (LCD_CFG, address - 0x400F C1B8) bit description. 46
Table 36. CAN Sleep Clear register (CANSLEEPCLR - address 0x400F C110) bit description 47
Table 37. CAN Wake-up Flags register (CANWAKEFLAGS - address 0x400F C114) bit description 47
Table 38. USB Interrupt Status register - (USBINTST - address 0x400F C1C0) bit description. 48
Table 39. DMA Request Select register bit description . 49
Table 40. Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8) bit description 51
Table 41. External Interrupt registers. 57
Table 42. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) low frequency mode (OSCRANGE = 0, see Table 33) 59
Table 43. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) high frequency mode (OSCRANGE = 1, see Table 33) 59
Table 44. PLL1 registers 63
Table 45. Elements determining PLL frequency 65
Table 46. PLL Multiplier values 66
Table 47. PLL Divider values 66
Table 48. Power Control registers 72
Table 49. Summary of flash accelerator registers 77
Table 50. Flash Accelerator Configuration register (FLASHCFG - address 0x400F C000) bit description 77
Table 51. Connection of interrupt sources to the Vectored Interrupt Controller. 81
Table 52. NVIC register map 84
Table 53. Interrupt Set-Enable Register 0 register 85
Table 54. Interrupt Set-Enable Register 1 register 86
Table 55. Interrupt Clear-Enable Register 0 87
Table 56. Interrupt Clear-Enable Register 1 register . . . 88
Table 57. Interrupt Set-Pending Register 0 register 89
Table 58. Interrupt Set-Pending Register 1 register 90
Table 59. Interrupt Clear-Pending Register 0 register . . 91
Table 60. Interrupt Clear-Pending Register 1 register . . 92
Table 61. Interrupt Active Bit Register 0 93
Table 62. Interrupt Active Bit Register 1 94
Table 63. Interrupt Priority Register 0 95
Table 64. Interrupt Priority Register 1 95
Table 65. Interrupt Priority Register 2 95
Table 66. Interrupt Priority Register 3 96
Table 67. Interrupt Priority Register 4 96
Table 68. Interrupt Priority Register 5 96

Table 69. Interrupt Priority Register 6 97
Table 70. Interrupt Priority Register 7 97
Table 71. Interrupt Priority Register 8 97
Table 72. Interrupt Priority Register 9 98
Table 73. Interrupt Priority Register 10 98
Table 74. Software Trigger Interrupt Register 98
Table 75. Pin description 99
Table 76. Summary of I/O pin configuration registers . . 121
Table 77. I/O Control registers for port 0 125
Table 78. I/O Control registers for port 1 126
Table 79. I/O Control registers for port 2 127
Table 80. I/O Control registers for port 3 128
Table 81. I/O Control registers for port 4 129
Table 82. I/O Control registers for port 5 130
Table 83. Type D IOCON registers bit description 132
Table 84. Type D I/O Control registers: FUNC values and pin functions 133
Table 85. Type A IOCON registers bit description 138
Table 86. Type A I/O Control registers: FUNC values and pin functions 139
Table 87. Type U IOCON registers bit description 140
Table 88. Type U I/O Control registers: FUNC values and pin functions 140
Table 89. Type I IOCON registers bit description 141
Table 90. Type I I/O Control registers: FUNC values and pin functions 141
Table 91. Type W IOCON registers bit description 142
Table 92. Type W I/O Control registers: FUNC values and pin functions 143
Table 93. GPIO pin description 145
Table 94. Register overview: GPIO (base address 0x2009 8000) 146
Table 95. Register overview: GPIO interrupt (base address 0x4002 8000) 147
Table 96. GPIO port Direction register (DIR[0:5] - addresses 0x2009 8000 (DIR0) to 0x200980A0 (DIR5)) bit description 148
Table 97. Fast GPIO port Mask register (MASK[0:5] - addresses 0x2009 8010 (MASK0) to 0x2009 80B0 (MASK5)) bit description. 148
Table 98. Fast GPIO port Pin value register (PIN[0:5] - addresses 0x2009 8014 (PIN0) to 0x2009 80B4 (PIN5)) bit description. 149
Table 99. Fast GPIO port output Set register (SET[0:5] - addresses 0x2009 8018 (SET0) to 0x2009 80B8 (SET5)) bit description 149
Table 100. Fast GPIO port output Clear register (CLR[0:5] - addresses 0x2009 801C (CLR0) to 0x2009 80BC (CLR5)) bit description 150
Table 101. GPIO overall Interrupt Status register (STATUS - address 0x4002 8080) bit description. 151
Table 102. GPIO Interrupt Status for port 0 Rising Edge Interrupt (STATR0 - 0x4002 8084) bit description 152
Table 103. GPIO Interrupt Status for port 0 Falling Edge Interrupt (STATF0 - 0x4002 8088) bit description 153
Table 104. GPIO Interrupt Clear register for port 0 (CLR0 - 0x4002 808C) bit description 154
Table 105. GPIO Interrupt Enable for port 0 Rising Edge (ENR0 - 0x4002 8090) bit description 155
Table 106. GPIO Interrupt Enable for port 0 Falling Edge (ENF0 - address 0x4002 8094) bit description156
Table 107. GPIO Interrupt Status for port 2 Rising Edge Interrupt (STATR2 - 0x4002 80A4) bit description 157
Table 108. GPIO Interrupt Status for port 2 Falling Edge Interrupt (STATF2 - 0x4002 80A8) bit description 158
Table 109. GPIO Interrupt Clear register for port 0 (CLR2 - 0x4002 80AC) bit description. 159
Table 110. GPIO Interrupt Enable for port 2 Rising Edge (ENR2 - 0x4002 80B0) bit description 160
Table 111. GPIO Interrupt Enable for port 2 Falling Edge (ENF2 - 0x4002 80B4) bit description 161
Table 112. EMC configuration. 163
Table 113. Memory bank selection 171
Table 114. Pad interface and control signal descriptions 175
Table 115. Register overview: EMC (base address 0x2009 C000) 176
Table 116. EMC Control register (Control - address 0x2009 C000) bit description 178
Table 117. EMC Status register (STATUS - address 0x2009 C008) bit description 179
Table 118. EMC Configuration register (CONFIG - address 0x2009 C008) bit description 179
Table 119. Dynamic Control register (DYNAMICCONTROL - address 0x2009 C020) bit description 180
Table 120. Dynamic Memory Refresh Timer register (DYNAMICREFRESH - address 0x2009 C024) bit description 181
Table 121. Dynamic Memory Read Configuration register (DYNAMICREADCONFIG - address 0x2009 C028) bit description 182
Table 122. Dynamic Memory Precharge Command Period register (DYNAMICRP - address 0x2009 C030) bit description. 182
Table 123. Dynamic Memory Active to Precharge Command Period register (DYNAMICRAS - address 0x2009 C034) bit description 183
Table 124. Dynamic Memory Self Refresh Exit Time register (DYNAMICSREX - address 0x2009 C038) bit description 183
Table 125. Dynamic Memory Last Data Out to Active Time register (DYNAMICAPR - address 0x2009 C03C) bit description. 184
Table 126. Dynamic Memory Data In to Active Command Time register (DYNAMICDAL - address 0x2009 C040) bit description 184
Table 127. Dynamic Memory Write Recovery Time register (DYNAMICWR - address 0x2009 C044) bit description 185
Table 128. Dynamic Memory Active to Active Command Period register (DYNAMICRC - address 0x2009 C048) bit description 185
Table 129. Dynamic Memory Auto Refresh Period register

(DYNAMICRFC - address 0x2009 C04C) bit description 186
Table 130. Dynamic Memory Exit Self Refresh register (DYNAMICXSR - address 0x2009 C050) bit description 186
Table 131. Dynamic Memory Active Bank A to Active Bank B Time register (DYNAMICRRD - address 0x2009 C054) bit description 187
Table 132. Dynamic Memory Load Mode register to Active Command Time (DYNAMICMRD - address 0x2009 C058) bit description 187
Table 133. Static Memory Extended Wait register (STATICEXTENDEDWAIT - address 0x2009 C080) bit description 188
Table 134. Dynamic Memory Configuration registers (DYNAMICCONFIG[0:3], address 0x2009 C100 (DYNAMICCONFIG0), 0x2009 C120 (DYNAMICCONFIG1), 0x2009 C140 (DYNAMICCONFIG2), 0x2009 C160 (DYNAMICCONFIG3)) bit description 189
Table 135. Address mapping 190
Table 136. Dynamic Memory RASCAS Delay registers (DYNAMICRASCAS[0:3], address 0x2009 C104 (DYNAMICRASCAS0), 0x2009 C124 (DYNAMICRASCAS1), 0x2009 C144 (DYNAMICRASCAS2), 0x2009 C164 (DYNAMICRASCAS3)) bit description 192
Table 137. Static Memory Configuration registers (STATICCONFIG[0:3], address 0x2009 C200 (STATICCONFIG0), 0x2009 C220 (STATICCONFIG1), 0x2009 C240 (STATICCONFIG2), 0x2009 C260 (STATICCONFIG3)) bit description 193
Table 138. Static Memory Write Enable Delay registers (STATICWAITWEN[0:3], address 0x2009 C204 (STATICWAITWEN0), 0x2009 C224 (STATICWAITWEN1),0x2009 C244 (STATICWAITWEN2), 0x2009 C264 (STATICWAITWEN3)) bit description 194
Table 139. Static Memory Output Enable delay registers (STATICWAITOEN[0:3], address 0x2009 C208 (STATICWAITOEN0), 0x0x2009 C228 (STATICWAITOEN1), 0x0x2009 C248 (STATICWAITOEN2), 0x0x2009 C268 (STATICWAITOEN3)) bit description 195
Table 140. Static Memory Read Delay registers (STATICWAITRD[0:3], address 0x2009 C20C (STATICWAITRD0), 0x2009 C22C (STATICWAITRD1), 0x2009 C24C (STATICWAITRD2), 0x2009 C26C (STATICWAITRD3)) bit description 195
Table 141. Static Memory Page Mode Read Delay registers (STATICWAITPAGE[0:3], address 0x2009 C210 (STATICWAITPAGE0), 2009 C230 (STATICWAITPAGE1), 0x2009 C250 (STATICWAITPAGE2), 0x2009 C270 (STATICWAITPAGE3)) bit description 196
Table 142. Static Memory Write Delay registers (STATICWAITWR[0:3], address 0x2009 C214 (STATICWAITWR0), 0x2009 C234 (STATICWAITWR1), 0x2009 C254 (STATICWAITWR2), 0x2009 C274 (STATICWAITWR3)) bit description 196
Table 143. Static Memory Turn-around Delay registers (STATICWAITTURN[0:3], address 0x2009 C218 (STATICWAITTURN0),0x2009 C238 (STATICWAITTURN1), 0x2009 C258 (STATICWAITTURN2), 0x2009 C278 (STATICWAITTURN3)) bit description 197
Table 144. Ethernet acronyms, abbreviations, and definitions 202
Table 145. Example PHY Devices 208
Table 146. Ethernet MII pin descriptions 209
Table 147. Ethernet RMII pin descriptions 209
Table 148. Ethernet MIIM pin descriptions 209
Table 149. Register overview: Ethernet (base address 0x2008 4000) 210
Table 150. MAC Configuration register 1 (MAC1 - address 0x2008 4000) bit description 212
Table 151. MAC Configuration register 2 (MAC2 - address 0x2008 4004) bit description 213
Table 152. Pad operation 214
Table 153. Back-to-back Inter-packet-gap register (IPGT - address 0x2008 4008) bit description 214
Table 154. Non Back-to-back Inter-packet-gap register (IPGR - address 0x2008 400C) bit description 214
Table 155. Collision Window / Retry register (CLRT - address 0x2008 4010) bit description 215
Table 156. Maximum Frame register (MAXF - address 0x2008 4014) bit description 215
Table 157. PHY Support register (SUPP - address 0x2008 4018) bit description 215
Table 158. Test register (TEST - address 0x2008 401C) bit description 216
Table 159. MII Mgmt Configuration register (MCFG - address 0x2008 4020) bit description 216
Table 160. Clock select encoding 217
Table 161. MII Mgmt Command register (MCMD - address 0x2008 4024) bit description 217
Table 162. MII Mgmt Address register (MADR - address 0x2008 4028) bit description 217
Table 163. MII Mgmt Write Data register (MWTD - address 0x2008 402C) bit description 218
Table 164. MII Mgmt Read Data register (MRDD - address 0x2008 4030) bit description 218
Table 165. MII Mgmt Indicators register (MIND - address 0x2008 4034) bit description 218
Table 166. Station Address register (SA0 - address 0x2008 4040) bit description 219
Table 167. Station Address register (SA1 - address 0x2008 4044) bit description 219
Table 168. Station Address register (SA2 - address 0x2008 4048) bit description 219
Table 169. Command register (COMMAND - address 0x2008 4100) bit description 220

Table 170. Status register (STATUS - address 0x2008 4104) bit description 220
Table 171. Receive Descriptor Base Address register (RXDESCRIPTOR - address 0x2008 4108) bit description 221
Table 172. Receive Status Base Address register (RXSTATUS - address 0x2008 410C) bit description 221
Table 173. Receive Number of Descriptors register (RXDESCRIPTORNUMBER - address 0x2008 4110) bit description 221
Table 174. Receive Produce Index register (RXPRODUCEINDEX - address 0x2008 4114) bit description 221
Table 175. Receive Consume Index register (RXCONSUMEINDEX - address 0x2008 4118) bit description 222
Table 176. Transmit Descriptor Base Address register (TXDESCRIPTOR - address 0x2008 411C) bit description 222
Table 177. Transmit Status Base Address register (TXSTATUS - address 0x2008 4120) bit description 222
Table 178. Transmit Number of Descriptors register (TXDESCRIPTORNUMBER - address 0x2008 4124) bit description 222
Table 179. Transmit Produce Index register (TXPRODUCEINDEX - address 0x2008 4128) bit description 223
Table 180. Transmit Consume Index register (TXCONSUMEINDEX - address 0x2008 412C) bit description 223
Table 181. Transmit Status Vector 0 register (TSV0 - address 0x2008 4158) bit description. 224
Table 182. Transmit Status Vector 1 register (TSV1 - address 0x2008 415C) bit description 225
Table 183. Receive Status Vector register (RSV - address 0x2008 4160) bit description 226
Table 184. Flow Control Counter register (FLOWCONTROLCOUNTER - address 0x2008 4170) bit description 227
Table 185. Flow Control Status register (FLOWCONTROLSTATUS - address 0x2008 4174) bit description 227
Table 186. Receive Filter Control register (RXFILTERCTRL - address 0x2008 4200) bit description. 228
Table 187. Receive Filter WoL Status register (RXFILTERWOLSTATUS - address 0x2008 4204) bit description 228
Table 188. Receive Filter WoL Clear register (RxFilterWoLClear - address 0x2008 4208) bit description 229
Table 189. Hash Filter Table LSBs register (HASHFILTERL - address 0x2008 4210) bit description. 229
Table 190. Hash Filter MSBs register (HASHFILTERH - address 0x2008 4214) bit description. 229
Table 191. Interrupt Status register (INTSTATUS - address 0x2008 4FE0) bit description 230
Table 192. Interrupt Enable register (INTENABLE - address 0x2008 4FE4) bit description 231
Table 193. Interrupt Clear register (INTCLEAR - address 0x2008 4FE8) bit description 231
Table 194. Interrupt Set register (INTSET - address 0x2008 4FEC) bit description. 232
Table 195. Power-Down register (POWERDOWN - address 0x2008 4FF4) bit description 232
Table 196. Receive Descriptor Fields 234
Table 197. Receive Descriptor Control Word 234
Table 198. Receive Status Fields 234
Table 199. Receive Status HashCRC Word 235
Table 200. Receive status information word 235
Table 201. Transmit descriptor fields 237
Table 202. Transmit descriptor control word 237
Table 203. Transmit status fields 237
Table 204. Transmit status information word 238
Table 205. LCD controller pins 280
Table 206. Pins used for single panel STN displays 280
Table 207. Pins used for dual panel STN displays 281
Table 208. Pins used for TFT displays 281
Table 209. FIFO bits for Little-endian Byte, Little-endian Pixel order. 285
Table 210. FIFO bits for Big-endian Byte, Big-endian Pixel order. 286
Table 211. FIFO bits for Little-endian Byte, Big-endian Pixel order. 287
Table 212. RGB mode data formats 288
Table 213. Palette data storage for TFT modes. 289
Table 214. Palette data storage for STN color modes. . . 289
Table 215. Palette data storage for STN monochrome mode. 290
Table 216. Palette data storage for STN monochrome mode. 291
Table 217. Addresses for 32 x 32 cursors 293
Table 218. Buffer to pixel mapping for 32 x 32 pixel cursor format. 294
Table 219. Buffer to pixel mapping for 64 x 64 pixel cursor format. 294
Table 220. Pixel encoding. 295
Table 221. Color display driven with 2 2/3 pixel data . . . 295
Table 222. Register overview: LCD controller (base address 0x2008 8000) 299
Table 223. Horizontal Timing register (TIMH, address 0x2008 8000) bit description 300
Table 224. Vertical Timing register (TIMV, address 0x2008 8004) bit description. 301
Table 225. Clock and Signal Polarity register (POL, address 0x2008 8008) bit description 302
Table 226. Line End Control register (LE, address 0x2008 800C) bit description 303
Table 227. Upper Panel Frame Base register (UPBASE, address 0x2008 8010) bit description 304
Table 228. Lower Panel Frame Base register (LPBASE, address 0x2008 8014) bit description 304
Table 229. LCD Control register (CTRL, address 0x2008 8018) bit description. 305
Table 230. Interrupt Mask register (INTMSK, address

0x2008 801C) bit description 307
Table 231. Raw Interrupt Status register (INTRAW, address 0x2008 8020) bit description 308
Table 232. Masked Interrupt Status register (INTSTAT, address 0x2008 8024) bit description. 308
Table 233. Interrupt Clear register (INTCLR, address 0x2008 8028) bit description 309
Table 234. Upper Panel Current Address register (UPCURR, address 0x2008 802C) bit description 309
Table 235. Lower Panel Current Address register (LPCURR, address 0x2008 8030) bit description. 309
Table 236. Color Palette registers (PAL[0:127], address 0x2008 8200 (PAL0) to 0x2008 83FC (PAL127)) bit description 310
Table 237. Cursor Image registers (CRSR_IMG[0:255], address 0x2008 8800 (CRSR_IMG0) to 0x2008 8BFC (CRSR_IMG255)) bit description 310
Table 238. Cursor Control register (CRSR_CTRL, address 0x2008 8C00) bit description 311
Table 239. Cursor Configuration register (CRSR_CFG, address 0x2008 8C04) bit description 311
Table 240. Cursor Palette register 0 (CRSR_PAL0, address 0x2008 8C08) bit description 312
Table 241. Cursor Palette register 1 (CRSR_PAL1, address 0x2008 8C0C) bit description 312
Table 242. Cursor XY Position register (CRSR_XY, address 0x2008 8C10) bit description 313
Table 243. Cursor Clip Position register (CRSR_CLIP, address 0x2008 8C14) bit description 313
Table 244. Cursor Interrupt Mask register (CRSR_INTMSK, RW - 0x2008 8C20) 314
Table 245. Cursor Interrupt Clear register (CRSR_INTCLR, address 0x2008 8C24) bit description 314
Table 246. Cursor Raw Interrupt Status register (CRSR_INTRAW, address 0x2008 8C28) bit description 314
Table 247. Cursor Masked Interrupt Status register (CRSR_INTSTAT, address 0x2008 8C2C) bit description 315
Table 248. LCD panel connections for STN single panel mode. 319
Table 249. LCD panel connections for STN dual panel mode. 320
Table 250. LCD panel connections for TFT panels 321
Table 251. USB related acronyms, abbreviations, and definitions used in this chapter 323
Table 252. Fixed endpoint configuration 324
Table 253. USB external interface. 327
Table 254. USB device controller clock sources 328
Table 255. Register overview: USB device controller (base address 0x2008 C000) 330
Table 256. USB Port Select register (PORTSEL - address 0x2008 C110) bit description 332
Table 257. USB Device Interrupt Status register (DEVINTST - address 0x2008 C200) bit description 332
Table 258. USB Device Interrupt Enable register (DEVINTEN - address 0x2008 C204) bit description 333
Table 259. USB Device Interrupt Clear register (DEVINTCLR - address 0x2008 C208) bit description 334
Table 260. USB Device Interrupt Set register (DEVINTSET - address 0x2008 C20C) bit description. 335
Table 261. USB Device Interrupt Priority register (DEVINTPRI - address 0x2008 C22C) bit description 336
Table 262. USB Endpoint registers bit allocation 337
Table 263. USB Endpoint Interrupt Status register (EPINTST - address 0x2008 C230) bit description 337
Table 264. USB Endpoint Interrupt Enable register (EPINTEN - address 0x2008 C234) bit description 338
Table 265. USB Endpoint Interrupt Clear register (EPINTCLR - address 0x2008 C238) bit description 338
Table 266. USB Endpoint Interrupt Set register (EPINTSET - address 0x2008 C23C) bit description. 339
Table 267. USB Endpoint Interrupt Priority register (EPINTPRI - address 0x2008 C240) bit description 339
Table 268. USB Realize Endpoint register (REEP - address 0x2008 C244) bit description 341
Table 269. USB Endpoint Index register (EPIN - address 0x2008 C248) bit description 342
Table 270. USB MaxPacketSize register (MAXPSIZE - address 0x2008 C24C) bit description. 342
Table 271. USB Receive Data register (RXDATA - address 0x2008 C218) bit description 343
Table 272. USB Receive Packet Length register (RXPLEN - address 0x2008 C220) bit description 343
Table 273. USB Transmit Data register (TXDATA - address 0x2008 C21C) bit description. 344
Table 274. USB Transmit Packet Length register (TXPLEN - address 0x2008 C224) bit description 344
Table 275. USB Control register (CTRL - address 0x2008 C228) bit description 345
Table 276. USB Command Code register (CMDCODE - address 0x2008 C210) bit description 346
Table 277. USB Command Data register (CMDDATA - address 0x2008 C214) bit description 346
Table 278. USB DMA Request Status register (DMARST - address 0x2008 C250) bit description 347
Table 279. USB DMA Request Clear register (DMARCLR - address 0x2008 C254) bit description 348
Table 280. USB DMA Request Set register (DMARSET - address 0x2008 C258) bit description 348
Table 281. USB UDCA Head register (UDCAH - address 0x2008 C280) bit description 349
Table 282. USB EP DMA Status register (EPDMAST - address 0x2008 C284) bit description 349
Table 283. USB EP DMA Enable register (EPDMAEN - address 0x2008 C288) bit description 349
Table 284. USB EP DMA Disable register (EPDMADIS - address 0x2008 C28C) bit description. 350
Table 285. USB DMA Interrupt Status register (DMAINTST - address 0x2008 C290) bit description 350
Table 286. USB DMA Interrupt Enable register (DMAINTEN

- address 0x2008 C294) bit description 351
Table 287. USB End of Transfer Interrupt Status register (EOTINTST - address 0x2008 C2A0) bit description 351
Table 288. USB End of Transfer Interrupt Clear register (EOTINTCLR - address 0x2008 C2A4) bit description 351
Table 289. USB End of Transfer Interrupt Set register (EOTINTSET - address 0x2008 C2A8) bit description 352
Table 290. USB New DD Request Interrupt Status register (NDDRINTST - address 0x2008 C2AC) bit description 352
Table 291. USB New DD Request Interrupt Clear register (NDDRINTCLR - address 0x2008 C2B0) bit description 352
Table 292. USB New DD Request Interrupt Set register (NDDRINTSET - address 0x2008 C2B4) bit description 352
Table 293. USB System Error Interrupt Status register (SYSERRINTST - address 0x2008 C2B8) bit description 353
Table 294. USB System Error Interrupt Clear register (SYSERRINTCLR - address 0x2008 C2BC) bit description 353
Table 295. USB System Error Interrupt Set register (SYSERRINTSET - address 0x2008 C2C0) bit description 353
Table 296. ClkCtrl register (CLKCTRL - address 0x2008 CFF4) bit description 354
Table 297. USB Clock Status register (CLKST - address 0x2008 CFF8) bit description 354
Table 298. SIE command code table. 359
Table 299. Set Address command bit description 359
Table 300. Configure Device command bit description . . .360
Table 301. Set Mode command bit description 360
Table 302. Set Device Status command bit description. . .362
Table 303. Get Error Code command bit description. . . .363
Table 304. Read Error Status command bit description . .364
Table 305. Select Endpoint command bit description . . .365
Table 306. Set Endpoint Status command bit description366
Table 307. Clear Buffer command bit description 367
Table 308. DMA descriptor 374
Table 309. USB (OHCI) related acronyms and abbreviations used in this chapter 388
Table 310. USB OTG port pins 390
Table 311. USB Host register address definitions 391
Table 312. USB OTG port 1 pins. 395
Table 313. Register overview: USB OTG controller (base address 0x2008 C000) 400
Table 314. OTG Interrupt Status register (INTST - address 0x2008 C100) bit description 400
Table 315. OTG Interrupt enable register (INTEN - address 0x2008 C104) bit description 401
Table 316. OTG Interrupt enable register (INTSET - address 0x2008 C108) bit description 401
Table 317. OTG Interrupt enable register (INTCLR - address 0x2008 C10C) bit description 402
Table 318. OTG Status Control register (STCTRL - address 0x2008 C110) bit description 403
Table 319. OTG Timer register (TMR - address 0x2008 C114) bit description 404
Table 320. I2C Receive register (I2C_RX - address 0x2008 C300) bit description 404
Table 321. I2C Transmit register (I2C_TX - address 0x2008 C300) bit description 405
Table 322. I2C status register (I2C_STS - address 0x2008 C304) bit description 405
Table 323. I2C Control register (I2C_CTL - address 0x2008 C308) bit description 407
Table 324. I2C_CLKHI register (I2C_CLKHI - address 0x2008 C30C) bit description. 408
Table 325. I2C_CLKLO register (I2C_CLKLO - address 0x2008 C310) bit description 408
Table 326. OTG clock control register (CLKCTRL - address 0x2008 CFF4) bit description. 409
Table 327. OTG clock status register (CLKST - address 0x2008 CFF8) bit description. 410
Table 328. SPIFI flash memory map. 426
Table 329. SPIFI Pin description. 426
Table 330. Supported QSPI devices. 427
Table 331. Register overview: SPIFI (base address 0x2009 4000) 428
Table 332. SPIFI control register (CTRL, address 0x2009 4000) bit description. 428
Table 333. SPIFI command register (CMD, address 0x2009 4004) bit description. 430
Table 334. SPIFI address register (ADDR, address 0x2009 4008) bit description. 431
Table 335. SPIFI intermediate data register (IDATA, address 0x2009 400C) bit description 431
Table 336. SPIFI cache limit register (CLIMIT, address 0x2009 4010) bit description 432
Table 337. SPIFI Data register (DATA, address 0x2009 4014) bit description. 432
Table 338. SPIFI memory command register (MCMD, address 0x2009 4018) bit description 433
Table 339. SPIFI status register (STAT, address 0x2009 401C) bit description 434
Table 340. SD/MMC card interface pin description. 440
Table 341. Command format 444
Table 342. Simple response format 444
Table 343. Long response format 445
Table 344. Command path status flags. 445
Table 345. CRC token status 448
Table 346. Data path status flags 449
Table 347. Transmit FIFO status flags 450
Table 348. Receive FIFO status flags. 450
Table 349. Register overview: SD card interface (base address 0x400C 0000). 452
Table 350: Power Control register (PWR - address 0x400C 0000) bit description 452
Table 351: MCI Clock Control register (CLOCK - address 0x400C 0004) bit description 453
Table 352: MCI Argument register (ARGUMENT - address 0x400C 0008) bit description 453

Table 353: MCI Command register (COMMAND - address 0x400C 000C) bit description 454
Table 354: Command Response Types. 454
Table 355: MCI Command Response register (RESPCMD - address 0x400C 0010) bit description 454
Table 356: MCI Response registers (RESPONSE[0:3] - addresses 0x400C 0014, 0x400C 0018, 0x400C 001C and 0x400C 0020) bit description 455
Table 357: Response Register Type 455
Table 358: MCI Data Timer register (DATATIMER - address 0x400C 0024) bit description 455
Table 359: MCI Data Length register (DATALENGTH - address 0x400C 0028) bit description 455
Table 360: Data Control register (DATACTRL - address 0x400C 002C) bit description 456
Table 361: Data Block Length 456
Table 362: MCI Data Counter register (DATACNT - address 0x400C 0030) bit description 457
Table 363: MCI Status register (STATUS - address 0x400C 0034) bit description 457
Table 364: MCI Clear register (CLEAR - address 0x400C 0038) bit description 458
Table 365: MCI Interrupt Mask registers (MASK0 - address 0x400C 003C) bit description 459
Table 366: MCI FIFO Counter register (FIFOCNT - address 0x400C 0048) bit description 459
Table 367: MCI Data FIFO register (FIFO - address 0x400C 0080 to 0x400C 00BC) bit description. . . 460
Table 368: UART1 Pin Description 464
Table 369: Register overview: UART1 (base address 0x4001 0000) 465
Table 370: UART1 Receiver Buffer Register when DLAB = 0 (RBR - address 0x4001 0000) bit description 466
Table 371: UART1 Transmitter Holding Register when DLAB = 0 (THR - address 0x4001 0000) bit description 466
Table 372: UART1 Divisor Latch LSB Register when DLAB = 1 (DLL - address 0x4001 0000) bit description 467
Table 373: UART1 Divisor Latch MSB Register when DLAB = 1 (DLM - address 0x4001 0004) bit description 467
Table 374: UART1 Interrupt Enable Register when DLAB = 0 (IER - address 0x4001 0004) bit description .468
Table 375: UART1 Interrupt Identification Register (IIR - address 0x4001 0008) bit description. 469
Table 376: UART1 Interrupt Handling 470
Table 377: UART1 FIFO Control Register (FCR - address 0x4001 0008) bit description 472
Table 378: UART1 Line Control Register (LCR - address 0x4001 000C) bit description 474
Table 379: UART1 Modem Control Register (MCR - address 0x4001 0010) bit description 475
Table 380: Modem status interrupt generation 476
Table 381: UART1 Line Status Register (LSR - address 0x4001 0014) bit description 478
Table 382: UART1 Modem Status Register (MSR - address 0x4001 0018) bit description 479
Table 383: UART1 Scratch Pad Register (SCR - address 0x4001 0014) bit description 481
Table 384: Auto-baud Control Register (ACR - address 0x4001 0020) bit description 481
Table 385: UART1 Fractional Divider Register (FDR - address 0x4001 0028) bit description 485
Table 386. Fractional Divider setting look-up table 487
Table 387: UART1 Transmit Enable Register (TER - address 0x4001 0030) bit description 488
Table 388: UART1 RS485 Control register (RS485CTRL - address 0x4001 004C) bit description 489
Table 389. UART1 RS-485 Address Match register (RS485ADRMATCH - address 0x4001 0050) bit description 489
Table 390. UART1 RS-485 Delay value register (RS485DLY - address 0x4001 0054) bit description 490
Table 391: UARTn Pin description 494
Table 392. Register overview: UART0/2/3 (base address: 0x4000 C000, 0x4008 8000, 0x4009 C000) . 495
Table 393: UARTn Receiver Buffer Register when DLAB = 0, read only (RBR - address 0x4000 C000 (UART0), 0x4009 8000 (UART2), 04009 C000 (UART3)) bit description 496
Table 394: UARTn Transmit Holding Register when DLAB = 0, write only (THR - address 0x4000 C000 (UART0), 0x4009 8000 (UART2), 0x4009 C000 (UART3)) bit description. 496
Table 395: UARTn Divisor Latch LSB register when DLAB = 1 (DLL - address 0x4000 C000 (UART0), 0x4009 8000 (UART2), 0x4009 C000 (UART3)) bit description. 497
Table 396: UARTn Divisor Latch MSB register when DLAB = 1 (DLM - address 0x4000 C004 (UART0), 0x4009 8004 (UART2), 0x4009 C004 (UART3)) bit description. 497
Table 397: UARTn Interrupt Enable Register when DLAB = 0 (IER - address 0x4000 C004 (UART0), 0x4009 8004 (UART2), 0x4009 C004 (UART3)) bit description. 498
Table 398: UARTn Interrupt Identification Register, read only (IIR - address 0x4000 C008 (UART0), 0x4009 8008 (UART2), 0x4009 C008 (UART3)) bit description. 499
Table 399: UARTn Interrupt Handling. 500
Table 400: UARTn FIFO Control Register, write only (FCR - address 0x4000 C008 (UART0), 0x4009 8008 (UART2), 0x4007 C008 (UART3)) bit description 501
Table 401: UARTn Line Control Register (LCR - address 0x4000 C00C (UART0), 0x4009 800C (UART2), 0x4009 C00C (UART3)) bit description 502
Table 402: UARTn Line Status Register (LSR - address 0x4000 C014 (UART0), 0x4009 8014 (UART2), 0x4009 C014 (UART3)) bit description 503
Table 403: UARTn Scratch Pad Register (SCR - address 0x4000 C01C (UART0), 0x4009 801C (UART2),

0x4009 C01C (UART3)) bit description 504
Table 404: UARTn Auto-baud Control Register (ACR - address 0x4000 C020 (UART0), 0x4009 8020 (UART2), 0x4009 C020 (UART3)) bit description 505
Table 405: UARTn Fractional Divider Register (FDR - address 0x4000 C028 (UART0), 0x4009 8028 (UART2), 0x4009 C028 (UART3)) bit description 508
Table 406. Fractional Divider setting look-up table 510
Table 407: UARTn Transmit Enable Register (TER - address 0x4000 C030 (UART0), 0x4009 8030 (UART2), 0x4009 C030 (UART3)) bit description. 511
Table 408: UARTn RS485 Control register (RS485CTRL - address 0x4000 C04C (UART0), 0x4009 804C (UART2), 0x4009 C04C (UART3)) bit description 512
Table 409. UARTn RS-485 Address Match register (RS485ADRMATCH - address 0x4000 C050 (UART0), RS485ADRMATCH - 0x4009 8050 (UART2), RS485ADRMATCH - 0x4009 C050 (UART3)) bit description. 512
Table 410. UARTn RS-485 Delay value register (RS485DLY - address 0x4000 0054 (UART0), RS485DLY - 0x4009 8054 (UART2), RS485DLY - 0x4009 C054 (UART3)) bit description. 513
Table 411: UART4 Pin description. 517
Table 412. Register overview: UART4 (base address: 0x400A 4000) 518
Table 413: UART4 Receiver Buffer Register when DLAB = 0 (RBR - address 0x400A 4000) bit description 519
Table 414: UART4 Transmit Holding Register when DLAB = 0 (THR -address 0x400A 4000) bit description 519
Table 415: UART4 Divisor Latch LSB register when DLAB = 1 (DLL - address 0x400A 4000) bit description 520
Table 416: UART4 Divisor Latch MSB register when DLAB = 1 (DLM - address 0x400A 4004) bit description 520
Table 417: UART4 Interrupt Enable Register when DLAB = 0 (IER - address 0x400A 4004) bit description. 521
Table 418: UART4 Interrupt Identification Register (IIR - address 0x400A 4008) bit description 522
Table 419: UART4 Interrupt Handling 523
Table 420: UART4 FIFO Control Register (FCR - address 0x400A 4008) bit description 525
Table 421: UART4 Line Control Register (LCR - address 0x400A 400C) bit description 526
Table 422: UART4 Line Status Register (LSR - address 0x400A 4014) bit description 527
Table 423: UART4 Scratch Pad Register (SCR - address 0x400A 401C) bit description 528
Table 424: UART4 Auto-baud Control Register (ACR - 0x400A 4020) bit description 529
Table 425: UART4 IrDA Control Register (ICR - address 0x400A 4024) bit description 532
Table 426: IrDA Pulse Width 532
Table 427: UART4 Fractional Divider Register (FDR - address 0x400A 4028) bit description 533
Table 428. Fractional Divider setting look-up table 535
Table 429. UART4 Oversampling Register (OSR - address 0x400A 402C) bit description. 536
Table 430. UART4 Smart Card Interface Control register (SCICTRL - address 0x400A 4048) bit description 537
Table 431: UART4 RS485 Control register (RS485CTRL - address 0x400A 404C) bit description. 538
Table 432. UART4 RS-485 Address Match register (RS485ADRMATCH - address 0x400A 4050) bit description 539
Table 433. UART4 RS-485 Delay value register (RS485DLY - address 0x400A 4054) bit description 539
Table 434. UART4 Synchronous mode control register (SYNCCTRL - address 0x400A 4058) bit description 541
Table 435. CAN Pin descriptions 544
Table 436. Memory map of the CAN block 549
Table 437. Register overview: CAN acceptance filter (base address 0x4003 C000). 549
Table 438. Register overview: central CAN (base address 0x4004 0000) 549
Table 439. Register overview: CAN (base address 0x4004 4000 (CAN1) and 0x4004 8000 (CAN2)) 549
Table 440. CAN1 and CAN2 controller register summary550
Table 441. Register overview: CAN Wake and Sleep (base address 0x400F C000) 551
Table 442. CAN Mode register (CAN1MOD - address 0x4004 4000, CAN2MOD - address 0x4004 8000) bit description 552
Table 443. CAN Command Register (CAN1CMR - address 0x4004 4004, CAN2CMR - address 0x4004 8004) bit description. 554
Table 444. CAN Global Status Register (CAN1GSR - address 0x4004 4008, CAN2GSR - address 0x4004 8008) bit description 555
Table 445. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description 558
Table 446. CAN Interrupt Enable Register (CAN1IER - address 0x4004 4010, CAN2IER - address 0x4004 8010) bit description 561
Table 447. CAN Bus Timing Register (CAN1BTR - address 0x4004 4014, CAN2BTR - address 0x4004 8014) bit description. 562
Table 448. CAN Error Warning Limit register (CAN1EWL - address 0x4004 4018, CAN2EWL - address 0x4004 8018) bit description 563
Table 449. CAN Status Register (CAN1SR - address 0x4004 401C, CAN2SR - address 0x4004 801C) bit description. 564
Table 450. CAN Receive Frame Status register (CAN1RFS - address 0x4004 4020, CAN2RFS - address 0x4004 8020) bit description 565
Table 451. CAN Receive Identifier register (CAN1RID - address 0x4004 4024, CAN2RID - address

0x4004 8024) bit description 566
Table 452. RX Identifier register when FF = 1 566
Table 453. CAN Receive Data register A (CAN1RDA - address 0x4004 4028, CAN2RDA - address 0x4004 8028) bit description 567
Table 454. CAN Receive Data register B (CAN1RDB - address 0x4004 402C, CAN2RDB - address 0x4004 802C) bit description 567
Table 455. CAN Transmit Frame Information register (CAN1TFI[1/2/3] - address 0x4004 40[30/40/50], CAN2TFI[1/2/3] - 0x4004 80[30/40/50]) bit description 568
Table 456. CAN Transfer Identifier register (CAN1TID[1/2/3] - address 0x4004 40[34/44/54], CAN2TID[1/2/3] - address 0x4004 80[34/44/54]) bit description. 569
Table 457. Transfer Identifier register when FF = 1. 569
Table 458. CAN Transmit Data register A (CAN1TDA[1/2/3] - address 0x4004 40[38/48/58], CAN2TDA[1/2/3] - address 0x4004 80[38/48/58]) bit description. 569
Table 459. CAN Transmit Data register B (CAN1TDB[1/2/3] - address 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3] - address 0x4004 80[3C/4C/5C]) bit description. . . 570
Table 460. Central Transit Status Register (TXSR - address 0x4004 0000) bit description 571
Table 461. Central Receive Status Register (RXSR - address 0x4004 0004) bit description 572
Table 462. Central Miscellaneous Status Register (MSR - address 0x4004 0008) bit description. 572
Table 463. Acceptance filter modes and access control . 573
Table 464. Section configuration register settings 574
Table 465. Acceptance Filter Mode Register (AFMR - address 0x4003 C000) bit description 576
Table 466. Standard Frame Individual Start Address register (SFF_SA - address 0x4003 C004) bit description 577
Table 467. Standard Frame Group Start Address register (SFF_GRP_SA - address 0x4003 C008) bit description 578
Table 468. Extended Frame Start Address register (EFF_SA - address 0x4003 C00C) bit description 578
Table 469. Extended Frame Group Start Address register (EFF_GRP_SA - address 0x4003 C010) bit description 578
Table 470. End of AF Tables register (ENDOFTABLE - address 0x4003 C014) bit description 579
Table 471. LUT Error Address register (LUTERRAD - address 0x4003 C018) bit description 579
Table 472. LUT Error register (LUTERR - address 0x4003 C01C) bit description 579
Table 473. Global FullCAN Enable register (FCANIE - address 0x4003 C020) bit description 580
Table 474. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0x4003 C024) bit description 580
Table 475. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0x4003 C028) bit description 580
Table 476. Format of automatically stored Rx messages 583
Table 477. FullCAN semaphore operation 583
Table 478. Example of Acceptance Filter Tables and ID index Values 594
Table 479. Used ID-Look-up Table sections 596
Table 480. Used ID-Look-up Table sections 597
Table 481. SSP pin descriptions 602
Table 482. Register overview: SSP (base address 0x4008 8000 (SSP0), 0x4003 0000 (SSP1), 0x400A C000 (SSP2)) 610
Table 483: SSPn Control Register 0 (CR0 - address 0x4008 8000 (SSP0), 0x4003 0000 (SSP1) , 0x400A C000 (SSP2)) bit description 611
Table 484: SSPn Control Register 1 (CR1 - address 0x4008 8004 (SSP0), 0x4003 0004 (SSP1), 0x400A C004 (SSP2)) bit description 612
Table 485: SSPn Data Register (DR - address 0x4008 8008 (SSP0), 0x4003 0008 (SSP1), 0x400A C008 (SSP2)) bit description 612
Table 486: SSPn Status Register (SR - address 0x4008 800C (SSP0), 0x4003 000C (SSP1), 0x400A C00C (SSP2)) bit description 613
Table 487: SSPn Clock Prescale Register (CPSR - address 0x4008 8010 (SSP0), 0x4003 0010 (SSP1), 0x400A C010 (SSP2)) bit description 613
Table 488: SSPn Interrupt Mask Set/Clear register (IMSC - address 0x4008 8014 (SSP0), 0x4003 0014 (SSP1), 0x400A C014 (SSP2)) bit description614
Table 489: SSPn Raw Interrupt Status register (RIS - address 0x4008 8018 (SSP0), 0x4003 0018 (SSP1), 0x400A C018 (SSP2)) bit description614
Table 490: SSPn Masked Interrupt Status register (MIS - address 0x4008 801C (SSP0), 0x4003 001C (SSP1), 0x400A C01C (SSP2)) bit description 615
Table 491: SSPn interrupt Clear Register (ICR - address 0x4008 8020 (SSP0), 0x4003 0020 (SSP1) , 0x400A C020 (SSP2)) bit description 615
Table 492: SSPn DMA Control Register (DMACR - address 0x4008 8024 (SSP0), 0x4003 0024 (SSP1), 0x400A C024 (SSP2)) bit description 615
Table 493. I^2C Pin Description 619
Table 494. I2C0CONSET and I2C1CONSET used to configure Master mode 620
Table 495. I2C0CONSET and I2C1CONSET used to configure Slave mode 622
Table 496. Register overview: I2C-bus interface (base address 0x4001 C000 (I2C0), 0x4005 C000 (I2C1), 0x400A 0000 (I2C2)) 628
Table 497. I^2C Control Set register (CONSET - addresses 0x4001 C000 (I2C0), 0x4005 C000 (I2C1) , 0x400A 0000 (I2C2)) bit description 629
Table 498. I^2C Control Clear register (CONCLR - addresses 0x4001 C018 (I2C0), 0x4005 C018 (I2C1), 0x400A 0018 (I2C2)) bit description 631
Table 499. I^2C Status register (STAT - addresses 0x4001 C004 (I2C0), 0x4005 C004 (I2C1), 0x400A 0004 (I2C2)) bit description 631

Table 500. I2C Data register (DAT- addresses 0x4001 C008 (I2C0), 0x4005 C008 (I2C1), 0x400A 0008 (2C2)) bit description 632
Table 501. I2C Monitor mode control register (MMCTRL - addresses 0x4001 C01C (I2C0), 0x4005 C01C (I2C1), 0x400A 001C (I2C2)) bit description. .632
Table 502. I2C Data buffer register (DATA_BUFFER - addresses 0x4001 C02C (I2C0), 0x4005 C02C (I2C1), 0x400A 002C (I2C2)) bit description. .633
Table 503. I2C Slave Address register 0 (ADR0 - address 0x4001 C00C (I2C0), 0x4005 C00C (I2C1), 0x400A 000C (I2C2)) bit description. 634
Table 504. I2C Slave Address registers (ADR[1:3] - address 0x4001 C020 (ADR1) to 0x4001 C028 (ADR3) (I2C0), 0x4005 C020 (ADR1) to 0x4005 C028 (ADR3) (I2C1), 0x400A 0020 (ADR1) to 0x400A 0028 (ADR3) (I2C2)) bit description .634
Table 505. I2C Mask registers (MASK[0:3] - address 0x4001 C030 (MASK0) to 0x4001 C03C (MASK3) (I2C0), 0x4005 C030 (MASK0) to 0x4005 C03C (MASK3) (I2C1), 0x400A 0030 (MASK0) to 0x400A 003C (MASK3) (I2C1)) bit description 634
Table 506. I2C SCL HIGH Duty Cycle register (SCLH - address 0x4001 C010 (I2C0), 0x4005 C010 (I2C1), 0x400A 0010(I2C2)) bit description . .635
Table 507. I2C SCL Low duty cycle register (SCLL - address 0x4001 C014 (I2C0), 0x4005 C014 (I2C1), 0x400A 0014 (I2C2)) bit description. 635
Table 508. Example I2C clock rates[1] 635
Table 509. Abbreviations used to describe an I2C operation. 636
Table 510. I2CONSET used to initialize Master Transmitter mode. 637
Table 511. I2CONSET used to initialize Slave Receiver mode. 641
Table 512. Master Transmitter mode. 644
Table 513. Master Receiver mode. 645
Table 514. Slave Receiver mode. 646
Table 515. Slave Transmitter mode. 648
Table 516. Miscellaneous States. 649
Table 517. Pin descriptions 666
Table 518. Register overview: I2S (base address 0x400A 8000) 668
Table 519: Digital Audio Output register (DAO - address 0x400A 8000) bit description 669
Table 520: Digital Audio Input register (DAI - address 0x400A 8004) bit description 669
Table 521: Transmit FIFO register (TXFIFO - address 0x400A 8008) bit description 670
Table 522: Receive FIFO register (RXFIFO - address 0x400A 800C) bit description 670
Table 523: Status Feedback register (STATE - address 0x400A 8010) bit description 670
Table 524: DMA Configuration register 1 (DMA1 - address 0x400A 8014) bit description 671
Table 525: DMA Configuration register 2 (DMA2 - address 0x400A 8018) bit description 671
Table 526: Interrupt Request Control register (IRQ - address 0x400A 801C) bit description. 672
Table 527: Transmit Clock Rate register (TXRATE - address 0x400A 8020) bit description 672
Table 528: Receive Clock Rate register (RXRATE - address 0x400A 8024) bit description 673
Table 529: Transmit Clock Bit Rate register (TXBITRATE - address 0x400A 8028) bit description 674
Table 530: Receive Clock Rate Bit register (RXBITRATE - address 0x400A 802C) bit description. 674
Table 531: Transmit Mode Control register (TXMODE - 0x400A 8030) bit description 674
Table 532: Receive Mode Control register (RXMODE - 0x400A 8034) bit description 675
Table 533: I2S transmit modes 678
Table 534: I2S receive modes. 681
Table 535. Conditions for FIFO level comparison. 685
Table 536. DMA and interrupt request generation 685
Table 537. Status feedback in the I2SSTATE register . . .685
Table 538. Timer/Counter pin description. 690
Table 539. Register overview: Timer0/1/2/3 (register base addresses 0x4000 4000 (TIMER0), 0x4000 8000 (TIMER1), 0x4009 0000 (TIMER2), 0x4009 4000 (TIMER3)) 691
Table 540. Interrupt Register (IR - addresses 0x4000 4000 (TIMER0), 0x4000 8000 (TIMER1), 0x4009 0000 (TIMER2), 0x4009 4000 (TIMER3)) bit description 692
Table 541. Timer Control Register (TCR - addresses 0x4000 4004 (TIMER0), 0x4000 8004 (TIMER1), 0x4009 0004 (TIMER2), 0x4009 4004 (TIMER3)) bit description. 692
Table 542. Timer counter registers (TC - addresses 0x400 4008 (TIMER0), 0x4000 8008 (TIMER1), 0x4009 0008 (TIMER2), 0x4009 4008 (TIMER3)) bit description. 693
Table 543. Timer prescale registers (PR - addresses 0x4000 400C (TIMER0), 0x4000 800C (TIMER1), 0x4009 000C (TIMER2), 0x4009 400C (TIMER3)) bit description. 693
Table 544. Timer prescale counter registers (PC - addresses 0x4000 4010 (TIMER0), 0x4000 8010 (TIMER1), 0x4009 0010 (TIMER2), 0x4009 4010 (TIMER3)) bit description. 693
Table 545. Match Control Register (MCR - addresses 0x4000 4014 (TIMER0), 0x4000 8014 (TIMER1), 0x4009 0014 (TIMER2), 0x4009 4014 (TIMER3)) bit description. 694
Table 546. Timer match registers (MR[0:3], addresses 0x4000 4018 (MR0) to 0x4000 4024 (MR3) (TIMER0), 0x4000 8018 (MR0) to 0x4000 8024 (MR3) (TIMER1), 0x4009 0018 (MR0) to 0x4009 0024 (MR3) (TIMER2), 0x4009 4018 (MR0) to 0x4009 4024 (MR3)(TIMER3)) bit description 695
Table 547. Capture Control Register (CCR - addresses 0x4000 4028 (TIMER0), 0x4000 8020 (TIMER1), 0x4009 0028 (TIMER2), 0x4009 4028 (TIMER3)) bit description. 695

Table 548. Timer capture registers (CR[0:1], address 0x4000 402C (CR0) to 0x4000 4030 (CR1) (TIMER0), 0x4000 802C (CR0) to 0x4000 0030 (CR1) (TIMER1), 0x4009 002C (CR0) to 0x4009 0030 (CR1) (TIMER2), 0x4009 402C (CR0) to 0x4000 4030 (CR1) (TIMER3)) bit description696
Table 549. Timer external match registers (EMR - addresses 0x4000 403C (TIMER0), 0x4000 803C (TIMER1), 0x4009 403C (TIMER2), 0x400C 403C (TIMER3)) bit description 697
Table 550. Count Control Register (CTCR - addresses 0x4000 4070 (TIMER0), 0x4000 8070 (TIMER1), 0x4009 0070 (TIMER2), 0x4009 4070 (TIMER3)) bit description 698
Table 551. System Tick Timer register map 703
Table 552. System Timer Control and status register (STCTRL - 0xE000 E010) bit description 703
Table 553. System Timer Reload value register (STRELOAD - 0xE000 E014) bit description 703
Table 554. System Timer Current value register (STCURR - 0xE000 E018) bit description 704
Table 555. System Timer Calibration value register (STCALIB - 0xE000 E01C) bit description . . .704
Table 556. Set and reset inputs for PWM Flip-Flops 710
Table 557. Pin summary 711
Table 558. Register overview: PWM (base addresses 0x4001 4000 (PWM0) and 0x4001 8000 (PWM1)) 712
Table 559: PWM Interrupt Register (IR - address 0x4001 4000 (PWM0) and 0x4001 8000 (PWM1)) bit description 713
Table 560: PWM Timer Control Register (TCR - address 0x4001 4004 (PWM0) and 0x4001 8004 (PWM1)) bit description 714
Table 561. PWM Timer counter registers (TC - addresses 0x4001 4008 (PWM0), 0x4001 8008 (PWM1)) bit description 714
Table 562. PWM prescale registers (PR - addresses 0x4001 400C (PWM0), 0x4001 800C (PWM1)) bit description 715
Table 563. PWM prescale counter registers (PC - addresses 0x4001 4010 (PWM0), 0x4001 8010 (PWM1)) bit description 715
Table 564. Match Control Register (MCR - address 0x4001 4014 (PWM0) and 0x4001 8014 (PWM1)) bit description 715
Table 565. PWM match registers (MR[0:3], addresses 0x4001 4018 (MR0) to 0x4001 4024 (MR3) (PWM0), 0x4001 8018 (MR0) to 0x4001 5024 (MR3) (PWM1)) bit description 717
Table 566. PWM match registers (MR[4:6], addresses 0x4001 4040 (MR4) to 0x4001 4048 (MR6) (PWM0), 0x4001 8040 (MR4) to 0x4001 5048 (MR6) (PWM1)) bit description 717
Table 567: PWM Capture Control Register (CCR - address 0x4001 4028 (PWM0) and 0x4001 8028 (PWM1)) bit description 718
Table 568. PWM capture registers (CR[0:1], address 0x4001 402C (CR0) to 0x4001 4038 (CR3) (PWM0), 0x4001 802C (CR0) to 0x4001 8038 (CR3) (PWM1)) bit description. 719
Table 569: PWM Control Registers (PCR - address 0x4001 404C (PWM0) and 0x4001 804C (PWM1)) bit description 719
Table 570: PWM Latch Enable Register (LER - address 0x4001 4050 (PWM0) and 0x4001 8050 (PWM1)) bit description. 720
Table 571: PWM Count control Register (CTCR - address 0x4001 4070 (PWM0) and 0x4001 8070 (PWM1)) bit description. 722
Table 572. Pin summary 724
Table 573. Register overview: Motor Control Pulse Width Modulator (MCPWM) (base address 0x400B 8000) 727
Table 574. MCPWM Control read address (CON - 0x400B 8000) bit description 728
Table 575. MCPWM Control set address (CON_SET - 0x400B 8004) bit description 729
Table 576. MCPWM Control clear address (CON_CLR - 0x400B 8008) bit description 730
Table 577. MCPWM Capture Control read address (CAPCON - 0x400B 800C) bit description . . . 731
Table 578. MCPWM Capture Control set address (CAPCON_SET - 0x400B 8010) bit description 732
Table 579. MCPWM Capture control clear register (CAPCON_CLR - address 0x400B 8014) bit description 733
Table 580. MCPWM Timer/Counter 0 to 2 registers (TC[0:2] - 0x400B 8018 (TC0), 0x400B 801C (TC1), 0x400B 8020) (TC2)bit description 733
Table 581. MCPWM Limit 0 to 2 registers (LIM[0:2] - 0x400B 8024 (LIM0), 0x400B 8028 (LIM1), 0x400B 802C (LIM2)) bit description 734
Table 582. MCPWM Match 0 to 2 registers (MAT[0:2] - addresses 0x400B 8030 (MAT0), 0x400B 8034 (MAT1), 0x400B 8038 (MAT2)) bit description 734
Table 583. MCPWM Dead-time register (DT - address 0x400B 803C) bit description. 736
Table 584. MCPWM Communication Pattern register (CP - address 0x400B 8040) bit description 736
Table 585. MCPWM Capture read addresses (CAP[0:2] - 0x400B 8044 (CAP0), 0x400B 8048 (CAP1), 0x400B 804C (CAP2)) bit description 737
Table 586. Motor Control PWM interrupts. 737
Table 587. MCPWM Interrupt Enable read address (INTEN - 0x400B 8050) bit description 737
Table 588. MCPWM interrupt enable set register (INTEN_SET - address 0x400B 8054) bit description 738
Table 589. PWM interrupt enable clear register (INTEN_CLR - address 0x400B 8058) bit description 739
Table 590. MCPWM Count Control read address (CNTCON - 0x400B 805C) bit description. 739
Table 591. MCPWM Count Control set address (CNTCON_SET - 0x400B 8060) bit description. .

741
Table 592. MCPWM Count Control clear address (CNTCON_CLR - 0x400B 8064) bit description . . 742
Table 593. MCPWM Interrupt flags read address (INTF - 0x400B 8068) bit description 743
Table 594. MCPWM Interrupt Flags set address (INTF_SET - 0x400B 806C) bit description 744
Table 595. MCPWM Interrupt Flags clear address (INTF_CLR - 0x400B 8070) bit description. . .745
Table 596. MCPWM Capture clear address (CAP_CLR - 0x400B 8074) bit description 745
Table 597. Encoder states 754
Table 598. Encoder state transitions[1] 754
Table 599. Encoder direction 755
Table 600. QEI pin description. 757
Table 601. Register overview: QEI (base address 0x400B C000) 758
Table 602: QEI Control register (CON - address 0x400B C000) bit description 759
Table 603: QEI Configuration register (CONF - address 0x400B C008) bit description 759
Table 604: QEI Status register (STAT - address 0x400B C004) bit description 759
Table 605: QEI Position register (POS - address 0x400B C00C) bit description. 760
Table 606: QEI Maximum Position register (MAXPOS - address 0x400B C010) bit description 760
Table 607: QEI Position Compare register 0 (CMPOS0 - address 0x400B C014) bit description 760
Table 608: QEI Position Compare register 1 (CMPOS1 - address 0x400B C018) bit description 760
Table 609: QEI Position Compare register 2 (CMPOS2 - address 0x400B C01C) bit description 761
Table 610: QEI Index Count register (INXCNT - address 0x400B C020) bit description 761
Table 611: QEI Index Compare register 0 (INXCMP0 - address 0x400B C024) bit description 761
Table 612: QEI Timer Load register (LOAD - address 0x400B C028) bit description 761
Table 613: QEI Timer register (TIME - address 0x400B C02C) bit description. 761
Table 614: QEI Velocity register (VEL - address 0x400B C030) bit description 762
Table 615: QEI Velocity Capture register (CAP - address 0x400B C034) bit description 762
Table 616: QEI Velocity Compare register (VELCOMP - address 0x400B C038) bit description 762
Table 617: QEI Digital Filter ON PHA (FILTERPHA - address 0x400B C03C) bit description. 762
Table 618: QEI Digital Filter on PHB (FILTERPHB - address 0x400B C040) bit description 762
Table 619: QEI Digital Filter on INX (FILTERINX - address 0x400B C044) bit description 763
Table 620: QEI index acceptance Window (WINDOW - address 0x400B C048) bit description 763
Table 621: QEI Index Compare register 1 (INXCMP1 - address 0x400B C04C) bit description. 763
Table 622: QEI Index Compare register 2 (INXCMP2 - address 0x400B C050) bit description. 763
Table 623: QEI Interrupt Status register (INTSTAT - address 0x400B CFE0) bit description 764
Table 624: QEI Interrupt Set register (SET - address 0x400B CFEC) bit description 765
Table 625: QEI Interrupt Clear register (CLR - 0x400B CFE8) bit description 766
Table 626: QEI Interrupt Enable register (IE - address 0x400B CFE4) bit description 767
Table 627: QEI Interrupt Enable Set register (IES - address 0x400B CFDC) bit description 768
Table 628: QEI Interrupt Enable Clear register (IEC - address 0x400B CFD8) bit description 769
Table 629. RTC pin description. 772
Table 630. Register overview: Real-Time Clock (base address 0x4002 4000) 773
Table 631. Interrupt Location Register (ILR - address 0x4002 4000) bit description 774
Table 632. Clock Control Register (CCR - address 0x4002 4008) bit description 774
Table 633. Counter Increment Interrupt Register (CIIR - address 0x4002 400C) bit description 775
Table 634. Alarm Mask Register (AMR - address 0x4002 4010) bit description 775
Table 635. RTC Auxiliary control register (RTC_AUX - address 0x4002 405C) bit description 776
Table 636. RTC Auxiliary Enable register (RTC_AUXEN - address 0x4002 4058) bit description 776
Table 637. Consolidated Time register 0 (CTIME0 - address 0x4002 4014) bit description 777
Table 638. Consolidated Time register 1 (CTIME1 - address 0x4002 4018) bit description 777
Table 639. Consolidated Time register 2 (CTIME2 - address 0x4002 401C) bit description 777
Table 640. Time Counter relationships and values 777
Table 641. Time Counter registers 778
Table 642. Seconds register (SEC - address 0x4002 4020) bit description. 778
Table 643. Minutes register (MIN - address 0x4002 4024) bit description 778
Table 644. Hours register (HRS - address 0x4002 4028) bit description 778
Table 645. Day of month register (DOM - address 0x4002 402C) bit description 779
Table 646. Day of week register (DOW - address 0x4002 4030) bit description 779
Table 647. Day of year register (DOY - address 0x4002 4034) bit description 779
Table 648. Month register (MONTH - address 0x4002 4038) bit description. 779
Table 649. Year register (YEAR - address 0x4002 403C) bit description 779
Table 650. Calibration register (CALIBRATION - address 0x4002 4040) bit description 780
Table 651. General purpose registers (GPREG[0:4] - addresses 0x4002 4044 (GPREG0) to 0x4002 4054 (GPREG4)) bit description 781

Table 652. Alarm registers 781
Table 653. Alarm Seconds register (ASEC - address 0x4002 4060) bit description 781
Table 654. Alarm Minutes register (AMIN - address 0x4002 4064) bit description 781
Table 655. Alarm Hours register (AHRS - address 0x4002 4068) bit description 782
Table 656. Alarm Day of month register (ADOM - address 0x4002 406C) bit description 782
Table 657. Alarm Day of week register (ADOW - address 0x4002 4070) bit description 782
Table 658. Alarm Day of year register (ADOY - address 0x4002 4074) bit description 782
Table 659. Alarm Month register (AMON - address 0x4002 4078) bit description 782
Table 660. Alarm Year register (AYRS - address 0x4002 407C) bit description 782
Table 661. Event Monitor/Recorder pin description 787
Table 662. Register overview: Event Monitor/Recorder (base address 0x4002 4000) 787
Table 663. Event Monitor/Recorder Control Register (ERCONTROL - 0x4002 4084) bit description 788
Table 664. Event Monitor/Recorder Status Register (ERSTATUS - 0x4002 4080) bit description . . 790
Table 665. Event Monitor/Recorder Counters Register (ERCOUNTERS - 0x4002 4088) bit description 791
Table 666. Event Monitor/Recorder First Stamp Register (ERFIRSTSTAMP0 - 0x0x4002 4090, ERFIRSTSTAMP1 - 0x0x4002 4094, ERFIRSTSTAMP2 - 0x4002 4098) bit description 791
Table 667. Event Monitor/Recorder Last Stamp Register (ERLASTSTAMP0 - 0x0x4002 40A0, ERLASTSTAMP1 - 0x0x4002 40A4, ERLASTSTAMP2 - 0x4002 40A8) bit description 792
Table 668. Register overview: Watchdog (base address 0x4000 0000) 796
Table 669: Watchdog Mode register (MOD - 0x4000 0000) bit description 797
Table 670. Watchdog operating modes selection 798
Table 671: Watchdog Timer Constant register (TC - address 0x4000 0004) bit description 798
Table 672: Watchdog Feed register (FEED - address 0x4000 0008) bit description 798
Table 673: Watchdog Timer Value register (TV - address 0x4000 000C) bit description 799
Table 674: Watchdog Timer Warning Interrupt register (WARNINT - address 0x4000 0014) bit description 799
Table 675: Watchdog Timer Window register (WINDOW - address 0x4000 0018) bit description. 799
Table 676. ADC pin description 803
Table 677. Register overview: ADC (base address 0x4003 4000) 804
Table 678: A/D Control Register (CR - address 0x4003 4000) bit description 805
Table 679: A/D Global Data Register (GDR - address 0x4003 4004) bit description 806
Table 680: A/D Interrupt Enable register (INTEN - address 0x4003 400C) bit description 807
Table 681: A/D Data Registers (DR[0:7] - addresses 0x4003 4010 (DR0) to 0x4003 402C (DR7)) bit description 808
Table 682: A/D Status register (STAT - address 0x4003 4030) bit description 809
Table 683: A/D Trim register (TRM - address 0x4003 4034) bit description. 810
Table 684. D/A Pin Description 813
Table 685. Register overview: DAC (base address 0x4008 C000) 814
Table 686: D/A Converter Register (CR - address 0x4008 C000) bit description 814
Table 687. D/A Control register (CTRL - address 0x4008 C004) bit description 815
Table 688: D/A Converter Counter Value register (CNTVAL - address 0x4008 C008) bit description 815
Table 689: Comparator pin description 820
Table 690. Register overview: Comparator (base address 0x4002 0000) 820
Table 691. Comparator block control register (CMP_CTRL - address 0x4002 0000) bit description 821
Table 692. Comparator 0 control register (CMP_CTRL0 - address 0x4002 0004) bit description 822
Table 693. Comparator 1 control register (CMP_CTRL1 - 0x4002 0008) bit description 825
Table 694: Interrupt configurations 827
Table 695. Endian behavior 831
Table 696. DMA connections 834
Table 697. Register overview: GPDMA (base address 0x2008 0000) 836
Table 698. DMA Interrupt Status register (INTSTAT, address 0x2008 0000) bit description 838
Table 699. MA Interrupt Terminal Count Request Status Register (INTTCSTAT, address 0x2008 0004) bit description 838
Table 700. DMA Interrupt Terminal Count Request Clear Register (INTTCCLEAR, address 0x2008 0008) bit description. 838
Table 701. DMA Interrupt Error Status Register (INTERRSTAT, address 0x2008 000C) bit description. 839
Table 702. DMA Interrupt Error Clear Register (INTERRCLR, address 0x2008 0010) bit description 839
Table 703. DMA Raw Interrupt Terminal Count Status Register (RAWINTTCSTAT, address 0x2008 0014) bit description 839
Table 704. DMA Raw Error Interrupt Status Register (RAWINTERRSTAT, address 0x2008 0018) bit description 840
Table 705. DMA Enabled Channel Register (ENBLDCHNS, address 0x2008 001C) bit description 840
Table 706. DMA Software Burst Request Register (SOFTBREQ, address 0x2008 0020) bit

description 840
Table 707. DMA Software Single Request register bit description 841
Table 708. DMA Software Last Burst Request Register (SOFTLBREQ, address 0x2008 0028) bit description 841
Table 709. DMA Software Last Single Request Register (SOFTLSREQ, address 0x2008 002C) bit description 842
Table 710. DMA Configuration Register (CONFIG, address 0x2008 0030) bit description 842
Table 711. DMA Synchronization Register (SYNC, address 0x2008 0034) bit description 842
Table 712. DMA Channel Source Address Registers (SRCADDR[0:7], 0x2008 0100 (SRCADDR0) to 0x2008 01E0 (SRCADDR7)) bit description . 843
Table 713. DMA Channel Destination Address registers (DESTADDR[0:7], 0x2008 0104 (DESTADDR0) to 0x2008 01E4 (DESTADDR7)) bit description 844
Table 714. DMA Channel Linked List Item registers (LLI[0:7], 0x2008 0108 (LLI0) to 0x2008 01E8 (LLI7)) bit description 844
Table 715. DMA Channel Control registers (CONTROL[0:7], 0x2008 010C (CONTROL0) to 0x2008 01EC (CONTROL7)) bit description 845
Table 716. DMA Channel Configuration registers (CONFIG[0:7], 0x2008 0110 (CONFIG0) to 0x2008 01F0 (CONFIG7)) bit description . . . 847
Table 717. Transfer type bits 848
Table 718. DMA request signal usage 851
Table 719. Register overview: CRC engine (base address 0x2009 0000) 859
Table 720. CRC mode register (MODE - address 0x2009 0000) bit description 859
Table 721. CRC seed register (SEED - address 0x2009 0004) bit description 859
Table 722. CRC checksum register (SUM - address 0x2009 0008) bit description 860
Table 723. CRC data register (DATA - address 0x2009 0008) bit description 860
Table 724. Register overview: EEPROM controller (base address 0x0020 0000) 867
Table 725. EEPROM command register (CMD - address 0x0020 0080) bit description 868
Table 726. EEPROM address register (ADDR - address 0x0020 0084) bit description 868
Table 727. EEPROM write data register (WDATA - address 0x0020 0088) bit description 869
Table 728. EEPROM read data register (RDATA - address 0x0020 008C) bit description 869
Table 729. EEPROM wait state register (WSTATE - address 0x0020 0090) bit description 870
Table 730. EEPROM clock divider register (CLKDIV - address 0x0020 0094) bit description. 871
Table 731. EEPROM power down/DCM register (PWRDWN - address 0x0020 0098) bit description 871
Table 732. Interrupt status register (INTSTAT - address 0x0020 0FE0) bit description 872
Table 733. Interrupt status clear register (INTSTATCLR - address 0x0020 0FE8) bit description 872
Table 734. Interrupt status set register (INTSTATSET - address 0x0020 0FEC) 873
Table 735. Interrupt enable register (INTEN - address 0x0020 0FE4) bit description 873
Table 736. Interrupt enable clear register (INTENCLR - address 0x0020 0FD8) bit description 873
Table 737. Interrupt enable set register (INTENSET - address 0x0020 0FDC) bit description. 874
Table 738. Flash sectors details 881
Table 739. Code Read Protection options 882
Table 740. Code Read Protection hardware/software interaction 883
Table 741. ISP command summary 884
Table 742. ISP Unlock command 884
Table 743. ISP Set Baud Rate command 885
Table 744. ISP Echo command 885
Table 745. ISP Write to RAM command 886
Table 746. ISP Read Memory command 886
Table 747. ISP Prepare sector(s) for write operation command 887
Table 748. ISP Copy command 887
Table 749. ISP Go command 888
Table 750. ISP Erase sector command 888
Table 751. ISP Blank check sector command 889
Table 752. ISP Read Part Identification command 889
Table 753. Part identification numbers 889
Table 754. ISP Read Boot Code version number command 889
Table 755. ISP Read device serial number command . . 890
Table 756. ISP Compare command 890
Table 757. ISP Return Codes Summary. 891
Table 758. IAP Command Summary 893
Table 759. IAP Prepare sector(s) for write operation command 895
Table 760. IAP Copy RAM to Flash command 895
Table 761. IAP Erase Sector(s) command 896
Table 762. IAP Blank check sector(s) command 896
Table 763. IAP Read part identification number command 896
Table 764. IAP Read Boot Code version number command 897
Table 765. IAP Read device serial number command . . 897
Table 766. IAP Compare command 897
Table 767. Re-invoke ISP 898
Table 768. IAP Status Codes Summary 898
Table 769. Register overview: Flash controller (base address 0x0020 0000) 899
Table 770. Flash Module Signature Start register (FMSSTART - 0x0020 0020) bit description . 900
Table 771. Flash Module Signature Stop register (FMSSTOP - 0x0020 0024) bit description 900
Table 772. FMSW0 register bit description (FMSW0, address: 0x0020 002C) 900
Table 773. FMSW1 register bit description (FMSW1, address: 0x0020 0030) 900

Table 774. FMSW2 register bit description (FMSW2, address: 0x0020 0034) 900
Table 775. FMSW3 register bit description (FMSW3, address: 0x0020 0038) 900
Table 776. Flash module Status register (STAT - 0x0020 0FE0) bit description 901
Table 777. Flash Module Status Clear register (STATCLR - 0x0x0020 0FE8) bit description 901
Table 778. JTAG pin description 904
Table 779. Serial Wire Debug pin description 904
Table 780. Parallel Trace pin description. 904
Table 781. Memory Mapping Control register (MEMMAP - 0x400F C040) bit description 907
Table 782. Abbreviations 910

41.5 Figures

Fig 1. LPC408x/407x simplified block diagram 10
Fig 2. LPC408x/407x block diagram, CPU and buses . . 13
Fig 3. System memory map 16
Fig 4. Clock generation 21
Fig 5. EMC programmable delays 42
Fig 6. EMC delay calibration 43
Fig 7. Reset block diagram 52
Fig 8. Example of start-up after reset 53
Fig 9. External interrupt logic 56
Fig 10. Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for $C_{X1}/_{X2}$ evaluation 59
Fig 11. PLL0 and PLL1 block diagram 63
Fig 12. CLKOUT selection 74
Fig 13. Simplified block diagram of the flash accelerator showing potential bus connections 75
Fig 14. I/O configurations 122
Fig 15. GPIO interrupt block diagram 151
Fig 16. EMC block diagram 166
Fig 17. SDRAM mode register 173
Fig 18. 32 bit bank external memory interfaces (bits MW = 10) 199
Fig 19. 16 bit bank external memory interfaces (bits MW = 01) 199
Fig 20. 8 bit bank external memory interface (bits MW = 00) 200
Fig 21. Typical memory configuration diagram 201
Fig 22. Ethernet block diagram 204
Fig 23. Ethernet packet fields 206
Fig 24. Receive descriptor memory layout 233
Fig 25. Transmit descriptor memory layout 236
Fig 26. Transmit example memory and registers 249
Fig 27. Receive Example Memory and Registers 255
Fig 28. Transmit Flow Control 260
Fig 29. Receive filter block diagram 262
Fig 30. Receive Active/Inactive state machine 266
Fig 31. Transmit Active/Inactive state machine 267
Fig 32. LCD controller block diagram 283
Fig 33. Cursor movement 291
Fig 34. Cursor clipping 292
Fig 35. Cursor image format 293
Fig 36. Power-up and power-down sequences 298
Fig 37. Horizontal timing for STN displays 316
Fig 38. Vertical timing for STN displays 317
Fig 39. Horizontal timing for TFT displays 317
Fig 40. Vertical timing for TFT displays 318
Fig 41. USB device controller block diagram 325
Fig 42. USB MaxPacketSize register array indexing . . . 342
Fig 43. Interrupt event handling 357
Fig 44. UDCA Head register and DMA Descriptors 373
Fig 45. Isochronous OUT endpoint operation example . 380
Fig 46. Data transfer in ATLE mode 381
Fig 47. USB Host controller block diagram 389
Fig 48. USB OTG controller block diagram 394
Fig 49. USB OTG port configuration: port U1 OTG dual-role device, port U2 host 396
Fig 50. USB OTG port configuration: VP_VM mode . . . 397
Fig 51. USB host port configuration: port U1 and U2 as hosts 398
Fig 52. USB device port configuration: port U1 host and port U2 device 399
Fig 53. Port selection for PORT_FUNC bit 0 = 0 and PORT_FUNC bit 1 = 0 404
Fig 54. USB OTG interrupt handling 411
Fig 55. USB OTG controller with software stack 413
Fig 56. Hardware support for B-device switching from peripheral state to host state 414
Fig 57. State transitions implemented in software during B-device switching from peripheral to host 415
Fig 58. Hardware support for A-device switching from host state to peripheral state 417
Fig 59. State transitions implemented in software during A-device switching from host to peripheral 418
Fig 60. Clocking and power control 422
Fig 61. Opcode only commands 435
Fig 62. Read commands 436
Fig 63. Secure digital memory card connection 440
Fig 64. Multimedia card system 441
Fig 65. SD card interface 442
Fig 66. Command path state machine 443
Fig 67. Command transfer 444
Fig 68. Data path state machine 446
Fig 69. Pending command start 448
Fig 70. UART1 block diagram 463
Fig 71. Auto-RTS Functional Timing 476
Fig 72. Auto-CTS Functional Timing 477
Fig 73. Auto-baud a) mode 0 and b) mode 1 waveform 484
Fig 74. Algorithm for setting UART dividers 486
Fig 75. UART0, 2, and 3 block diagram 494
Fig 76. Auto-baud a) mode 0 and b) mode 1 waveform 507
Fig 77. Algorithm for setting UART dividers 509
Fig 78. UART 4 block diagram 517
Fig 79. Auto-baud a) mode 0 and b) mode 1 waveform 531
Fig 80. Algorithm for setting UART dividers 534
Fig 81. CAN controller block diagram 545
Fig 82. Transmit buffer layout for standard and extended frame format configurations 546
Fig 83. Receive buffer layout for standard and extended frame format configurations 547
Fig 84. Global Self-Test (high-speed CAN Bus example) 548
Fig 85. Local self test (high-speed CAN Bus example) . 548
Fig 86. Entry in FullCAN and individual standard identifier tables 574
Fig 87. Entry in standard identifier range table 575
Fig 88. Entry in either extended identifier table 575
Fig 89. ID Look-up table example explaining the search algorithm 581
Fig 90. Semaphore procedure for reading an auto-stored message 585
Fig 91. FullCAN section example of the ID look-up table 587

Fig 92. FullCAN message object layout 587
Fig 93. Normal case, no messages lost 589
Fig 94. Message lost 589
Fig 95. Message gets overwritten 590
Fig 96. Message overwritten indicated by semaphore bits and message lost 591
Fig 97. Message overwritten indicated by message lost 592
Fig 98. Clearing message lost 593
Fig 99. Detailed example of acceptance filter tables and ID index values 595
Fig 100. ID Look-up table configuration example (no FullCAN) 597
Fig 101. ID Look-up table configuration example (FullCAN activated and enabled) 599
Fig 102. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer 603
Fig 103. SPI frame format with CPOL=0 and CPHA=0 (a) Single and b) Continuous Transfer) 604
Fig 104. SPI frame format with CPOL=0 and CPHA=1 . . 605
Fig 105. SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer) 606
Fig 106. SPI Frame Format with CPOL = 1 and CPHA = 1. . 607
Fig 107. Microwire frame format (single transfer) 608
Fig 108. Microwire frame format (continuos transfers) . . 609
Fig 109. Microwire frame format setup and hold details . 609
Fig 110. I^2C-bus configuration 618
Fig 111. Format in the Master Transmitter mode 620
Fig 112. Format of Master Receiver mode 621
Fig 113. A Master Receiver switches to Master Transmitter after sending repeated START 621
Fig 114. Format of Slave Receiver mode 622
Fig 115. Format of Slave Transmitter mode 623
Fig 116. I^2C serial interface block diagram 624
Fig 117. Arbitration procedure 626
Fig 118. Serial clock synchronization 626
Fig 119. Format and states in the Master Transmitter mode 638
Fig 120. Format and states in the Master Receiver mode 640
Fig 121. Format and states in the Slave Receiver mode. 642
Fig 122. Format and states in the Slave Transmitter mode 643
Fig 123. Simultaneous repeated START conditions from two masters 651
Fig 124. Forced access to a busy I^2C-bus 651
Fig 125. Recovering from a bus obstruction caused by a LOW level on SDA 652
Fig 126. Simple I^2S configurations and bus timing 667
Fig 127. Typical transmitter master mode, with or without MCLK output 679
Fig 128. Transmitter master mode sharing the receiver reference clock 679
Fig 129. 4-wire transmitter master mode sharing the receiver bit clock and WS 679
Fig 130. Typical transmitter slave mode 679
Fig 131. Transmitter slave mode sharing the receiver reference clock 680
Fig 132. 4-wire transmitter slave mode sharing the receiver bit clock and WS 680
Fig 133. Typical receiver master mode, with or without MCLK output 682
Fig 134. Receiver master mode sharing the transmitter reference clock 682
Fig 135. 4-wire receiver master mode sharing the transmitter bit clock and WS 682
Fig 136. Typical receiver slave mode 682
Fig 137. Receiver slave mode sharing the transmitter reference clock 683
Fig 138. 4-wire receiver slave mode sharing the transmitter bit clock and WS 683
Fig 139. I^2S clocking and pin connections 684
Fig 140. FIFO contents for various I^2S modes 686
Fig 141. Timer block diagram 689
Fig 142. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled. 700
Fig 143. A timer Cycle in Which PR=2, MRx=6, and both interrupt and stop on match are enabled 700
Fig 144. System Tick Timer block diagram 702
Fig 145. PWM block diagram 709
Fig 146. Sample PWM waveforms 710
Fig 147. MCPWM Block Diagram 725
Fig 148. Edge-aligned PWM waveform without dead time, POLA = 0. 746
Fig 149. Center-aligned PWM waveform without dead time, POLA = 0. 747
Fig 150. Edge-aligned PWM waveform with dead time, POLA = 0. 747
Fig 151. Center-aligned waveform with dead time, POLA = 0. 748
Fig 152. Three-phase DC mode sample waveforms . . . 750
Fig 153. Three-phase AC mode sample waveforms, edge aligned PWM mode 751
Fig 154. Encoder interface block diagram 753
Fig 155. Quadrature Encoder Basic Operation. 755
Fig 156. RTC domain conceptual diagram 771
Fig 157. RTC functional block diagram. 771
Fig 158. Event Monitor/Recorder block diagram 786
Fig 159. Watchdog Timer block diagram 795
Fig 160. Early Watchdog Feed with Windowed Mode Enabled 800
Fig 161. Correct Watchdog Feed with Windowed Mode Enabled 800
Fig 162. Watchdog Warning Interrupt 801
Fig 163. ADC block diagram 803
Fig 164. DAC control with DMA interrupt and timer 813
Fig 165. Comparator block diagram 819
Fig 166. DMA controller block diagram. 830
Fig 167. LLI example 855
Fig 168. CRC block diagram. 858
Fig 169. EEPROM block diagram. 862
Fig 170. Starting a write operation 864
Fig 171. (16-bit) write operations with post-incrementing of address 864
Fig 172. Programming a page into memory 865

Fig 173. Starting a read operation (32-bit read from address A) 866
Fig 174. Map of lower memory 876
Fig 175. Boot process flowchart 880
Fig 176. IAP parameter passing 894
Fig 177. Algorithm for generating a 128 bit signature . . . 902
Fig 178. ARM Standard JTAG Connector 905
Fig 179. Cortex Debug Connector. 906
Fig 180. Cortex Debug & ETM Connector. 906

41.6 Contents

Chapter 1: Introductory information

1.1 Introduction . . . 3
1.2 Features . . . 4
1.3 Applications . . . 7
1.4 Ordering information . . . 8
1.4.1 Part options summary . . . 8
1.5 Simplified block diagram . . . 10
1.6 Architectural overview . . . 11
1.7 ARM Cortex-M4 processor . . . 11
1.8 On-chip flash memory system . . . 11
1.9 On-chip Static RAM . . . 11
1.10 On-chip EEPROM . . . 12
1.11 Detailed block diagram . . . 13

Chapter 2: LPC408x/407x Memory map

2.1 Memory map and peripheral addressing . . . 14
2.2 Memory maps . . . 15
2.3 On-chip peripherals . . . 17
2.3.1 AHB peripherals . . . 17
2.3.2 APB peripheral addresses . . . 17
2.4 Memory re-mapping . . . 18
Boot ROM re-mapping . . . 18
2.5 AHB arbitration . . . 19
2.5.1 Matrix Arbitration register . . . 19

Chapter 3: LPC408x/407x System and clock control

3.1 Introduction . . . 20
3.1.1 Summary of clocking and power control functions . . . 20
3.2 Pin description . . . 21
3.3 Register description . . . 23
3.3.1 PLL registers . . . 25
3.3.1.1 PLL Control registers . . . 25
3.3.1.2 PLL Configuration registers . . . 25
3.3.1.3 PLL Status registers . . . 26
3.3.1.4 PLL Interrupts: PLOCK0 and PLOCK1 . . . 26
3.3.1.5 PLL Feed registers . . . 26
3.3.2 Power control . . . 28
3.3.2.1 Power Mode Control register . . . 28
3.3.2.1.1 Encoding of Reduced Power Modes . . . 29
3.3.2.2 Power Control for Peripherals registers . . . 29
3.3.2.3 Power Boost control register . . . 31
3.3.3 Clock selection and divider registers . . . 32
3.3.3.1 EMC Clock Selection register . . . 32
3.3.3.2 CPU Clock Selection register . . . 32
3.3.3.3 USB Clock Selection register . . . 33
3.3.3.4 Clock Source Selection register . . . 34
3.3.3.5 Peripheral Clock Selection register . . . 34
3.3.3.6 SPIFI Clock Selection register . . . 34
3.3.4 External interrupts . . . 36
3.3.4.1 External Interrupt flag register . . . 36
3.3.4.2 External Interrupt Mode register . . . 37
3.3.4.3 External Interrupt Polarity register . . . 38
3.3.5 Device and Peripheral Reset . . . 39
3.3.5.1 Reset Source Identification Register . . . 39
3.3.5.2 Reset control register 0 . . . 40
3.3.5.3 Reset control register 1 . . . 41
3.3.6 EMC delay control and calibration . . . 42
3.3.6.1 EMC Delay Control register . . . 42
3.3.6.2 EMC Calibration register . . . 43
Procedure for calibrating programmable delays . . . 44
3.3.7 Miscellaneous system control registers . . . 45
3.3.7.1 System Controls and Status register . . . 45
3.3.7.2 IRC control register . . . 46
3.3.7.3 LCD Configuration register . . . 46
3.3.7.4 CAN Sleep Clear register . . . 47
3.3.7.5 CAN Wake-up Flags register . . . 47
3.3.7.6 USB Interrupt Status Register . . . 48
3.3.7.7 DMA Request Select register . . . 49
3.3.7.7.1 Timer DMA requests . . . 50
3.3.7.8 Clock Output Configuration register . . . 50
3.4 Chip reset . . . 52
3.5 Peripheral reset control . . . 54
3.6 Brown-out detection . . . 55
3.7 External interrupt inputs . . . 56
3.7.1 Register description . . . 57
3.8 Oscillators . . . 58
3.8.1 Internal RC oscillator . . . 58
3.8.2 Main oscillator . . . 58
3.8.2.1 Main oscillator startup . . . 60
3.8.3 RTC oscillator . . . 60
3.8.4 Watchdog oscillator . . . 60
3.9 Clock source selection multiplexer . . . 61
3.10 PLL0 and PLL1 (Phase Locked Loops) . . . 62
3.10.1 PLL and startup/boot code interaction . . . 62
3.10.2 PLL register description . . . 63
3.10.3 PLLs and Power-down mode . . . 64
3.10.4 PLL frequency calculation . . . 65
3.10.5 Procedure for determining PLL settings . . . 65
3.10.6 PLL configuration sequence . . . 67
To set up a PLL and switch clocks to its output: . . . 67
To switch clocks away from a PLL output: . . . 67
3.10.7 PLL configuration examples . . . 68
3.11 Clock selection and division . . . 69

3.12 Power control 69
3.12.1 Sleep mode 69
3.12.2 Deep Sleep mode 70
3.12.3 Power-down mode 71
3.12.4 Deep Power-down mode 71
3.12.5 Peripheral power control 71
3.12.6 Power boost 71
3.12.7 Register description 72
3.12.8 Wake-up from Reduced Power Modes 72
3.12.9 Power control usage notes 72
3.12.10 Power domains 73
3.13 Wake-up timer 73
3.14 External clock output pin 74

Chapter 4: LPC408x/407x Flash accelerator

4.1 Introduction 75
4.2 Flash accelerator blocks 75
4.2.1 Flash memory bank 75
4.2.2 Flash programming Issues 76
4.3 Register description 77
4.4 Flash Accelerator Configuration register . . . 77
4.5 Operation 78

Chapter 5: LPC408x/407x Nested Vectored Interrupt Controller (NVIC)

5.1 Features 80
5.2 Description 80
5.3 Interrupt sources 80
5.4 Vector table remapping 83
Examples: 83
5.5 Register description 84
5.5.1 . . . . . Interrupt Set-Enable Register 0 register 85
5.5.2 . . . . . Interrupt Set-Enable Register 1 register 86
5.5.3 Interrupt Clear-Enable Register 0 87
5.4 . . . Interrupt Clear-Enable Register 1 register 88
5.5.5 . . . . Interrupt Set-Pending Register 0 register 89
5.5.6 . . . . Interrupt Set-Pending Register 1 register 90
5.5.7 . . Interrupt Clear-Pending Register 0 register 91
5.5.8 . . Interrupt Clear-Pending Register 1 register 92
5.5.9 Interrupt Active Bit Register 0 93
5.5.10 Interrupt Active Bit Register 1 94
5.5.11 Interrupt Priority Register 0 95
5.5.12 Interrupt Priority Register 1 95
5.5.13 Interrupt Priority Register 2 95
5.5.14 Interrupt Priority Register 3 96
5.5.15 Interrupt Priority Register 4 96
5.5.16 Interrupt Priority Register 5 96
5.5.17 Interrupt Priority Register 6 97
5.5.18 Interrupt Priority Register 7 97
5.5.19 Interrupt Priority Register 8 97
5.5.20 Interrupt Priority Register 9 98
5.5.21 Interrupt Priority Register 10 98
5.5.22 Software Trigger Interrupt Register 98

Chapter 6: LPC408x/407x Pin configuration

6.1 Pin configuration 99

Chapter 7: LPC408x/407x I/O configuration

7.1 Introduction 121
7.2 Description 121
7.3 IOCON registers 121
Multiple connections 122
7.3.1 Pin function 122
7.3.2 Pin mode 123
7.3.3 Hysteresis 123
7.3.4 Input Inversion 123
7.3.5 Analog/digital mode 123
7.3.6 Input filter 123
7.3.7 Output slew rate 123
7.3.8 I^2C modes 124
7.3.9 Open-Drain Mode 124
7.3.10 DAC enable 124
7.4 Register description 125
7.4.1 I/O configuration register contents (IOCON) 131
7.4.1.1 Type D IOCON registers (applies to most GPIO port pins) 132
7.4.1.2 Type A IOCON registers (applies to pins that include an analog function) 138
7.4.1.3 Type U IOCON registers (applies to pins that include a USB D+ or D- function) 140
7.4.1.4 Type I IOCON registers (applies to pins that include a specialized I^2C function) 141
7.4.1.5 Type W IOCON registers (these pins are otherwise the same as Type D, but include a selectable input glitch filter 142

Chapter 8: LPC408x/407x GPIO

8.1 Basic configuration 144
8.2 Features 144
8.2.1 Digital I/O ports 144
8.2.2 Interrupt generating digital ports 144
8.3 Applications 145
8.4 Pin description 145

8.5 Register description 146
8.5.1 GPIO port registers. 148
8.5.1.1 GPIO port Direction register 148
8.5.1.2 Fast GPIO port Mask register 148
8.5.1.3 GPIO port Pin value register 148
8.5.1.4 GPIO port output Set register 149
8.5.1.5 GPIO port output Clear register 150
8.5.2 GPIO interrupt registers 151
8.5.2.1 GPIO overall Interrupt Status register 151
8.5.2.2 GPIO Interrupt Status for port 0 Rising Edge Interrupt 152
8.5.2.3 GPIO Interrupt Status for port 0 Falling Edge Interrupt 153
8.5.2.4 GPIO Interrupt Clear register for port 0 154
8.5.2.5 GPIO Interrupt Enable for port 0 Rising Edge 155
8.5.2.6 GPIO Interrupt Enable for port 0 Falling Edge 156
8.5.2.7 GPIO Interrupt Status for port 2 Rising Edge Interrupt 157
8.5.2.8 GPIO Interrupt Status for port 2 Falling Edge Interrupt 158
8.5.2.9 GPIO Interrupt Clear register for port 2 159
8.5.2.10 GPIO Interrupt Enable for port 2 Rising Edge 160
8.5.2.11 GPIO Interrupt Enable for port 2 Falling Edge 161
8.6 GPIO usage notes 162
8.6.1 Example: An instantaneous output of 0s and 1s on a GPIO port 162
8.6.2 Writing to FIOSET/FIOCLR vs. FIOPIN 162

Chapter 9: LPC408x/407x External Memory Controller (EMC)

9.1 How to read this chapter 163
9.2 Basic configuration 164
9.3 Introduction 164
9.4 Features 165
9.5 EMC functional description 166
9.5.1 AHB slave register interface 167
9.5.2 AHB slave memory interface 167
9.5.2.1 Memory transaction endianness. 167
9.5.2.2 Memory transaction size. 167
9.5.2.3 Write protected memory areas 167
9.5.3 Pad interface 167
9.5.4 Data buffers 167
9.5.4.1 Write buffers 167
9.5.4.2 Read buffers 168
9.5.5 Memory controller state machine 169
9.5.6 Timing control with programmable delay elements 169
9.6 Low-power operation. 170
9.6.1 Low-power SDRAM Deep-sleep Mode. 170
9.6.2 Low-power SDRAM partial array refresh . . . 170
9.7 Memory bank select 171
9.8 EMC Reset 171
9.9 Address shift mode 172
9.10 Memory mapped I/O and burst disable 172
9.11 Using the EMC with SDRAM 173
9.11.1 Mode register setup 173
Example 174
9.12 Pin description. 175
9.13 Register description 176
9.13.1 EMC Control register 178
9.13.2 EMC Status register 179
9.13.3 EMC Configuration register 179
9.13.4 Dynamic Memory Control register 180
9.13.5 Dynamic Memory Refresh Timer register . . 181
9.13.6 Dynamic Memory Read Configuration register 182
9.13.7 Dynamic Memory Precharge Command Period register 182
9.13.8 Dynamic Memory Active to Precharge Command Period register 183
9.13.9 Dynamic Memory Self-refresh Exit Time register 183
9.13.10 Dynamic Memory Last Data Out to Active Time register 184
9.13.11 Dynamic Memory Data-in to Active Command Time register 184
9.13.12 Dynamic Memory Write Recovery Time register 185
9.13.13 Dynamic Memory Active to Active Command Period register 185
9.13.14 Dynamic Memory Auto-refresh Period register. 186
9.13.15 Dynamic Memory Exit Self-refresh register 186
9.13.16 Dynamic Memory Active Bank A to Active Bank B Time register 187
9.13.17 Dynamic Memory Load Mode register to Active Command Time 187
9.13.18 Static Memory Extended Wait register 188
9.13.19 Dynamic Memory Configuration registers . . 189
9.13.20 Dynamic Memory RAS & CAS Delay registers 191
9.13.21 Static Memory Configuration registers 193
9.13.22 Static Memory Write Enable Delay registers 194
9.13.23 Static Memory Output Enable Delay registers 195
9.13.24 Static Memory Read Delay registers 195
9.13.25 Static Memory Page Mode Read Delay registers 196
9.13.26 Static Memory Write Delay registers 196
9.13.27 Static Memory Turn Round Delay registers . 197
9.14 External static memory interface. 198
9.14.1 32-bit wide memory bank connection 198
9.14.2 16-bit wide memory bank connection 199
9.14.3 8-bit wide memory bank connection 200
9.14.4 Memory configuration example 201

Chapter 10: LPC408x/407x Ethernet

10.1 Basic configuration 202
10.2 Introduction 202
10.3 Features 203
10.4 Architecture and operation 204
10.5 DMA engine functions 205
10.6 Overview of DMA operation 205
10.7 Ethernet packet 206
10.8 Overview 207
10.8.1 Partitioning 207
10.8.2 Example PHY Devices 208
10.9 Pin description 209
10.10 Register description 210
10.10.1 Ethernet MAC register definitions 212
10.10.1.1 MAC Configuration Register 1 212
10.10.1.2 MAC Configuration Register 2 213
10.10.1.3 Back-to-Back Inter-Packet-Gap Register . . . 214
10.10.1.4 Non Back-to-Back Inter-Packet-Gap Register 214
10.10.1.5 Collision Window / Retry Register 215
10.10.1.6 Maximum Frame Register 215
10.10.1.7 PHY Support Register 215
10.10.1.8 Test Register 216
10.10.1.9 MII Mgmt Configuration Register 216
10.10.1.10 MII Mgmt Command Register 217
10.10.1.11 MII Mgmt Address Register 217
10.10.1.12 MII Mgmt Write Data Register 218
10.10.1.13 MII Mgmt Read Data Register 218
10.10.1.14 MII Mgmt Indicators Register 218
10.10.1.15 Station Address 0 Register 219
10.10.1.16 Station Address 1 Register 219
10.10.1.17 Station Address 2 Register 219
10.10.2 Control register definitions 220
10.10.2.1 Command Register 220
10.10.2.2 Status Register 220
10.10.2.3 Receive Descriptor Base Address Register . 221
10.10.2.4 Receive Status Base Address Register 221
10.10.2.5 Receive Number of Descriptors Register . . . 221
10.10.2.6 Receive Produce Index Register 221
10.10.2.7 Receive Consume Index Register 222
10.10.2.8 Transmit Descriptor Base Address Register 222
10.10.2.9 Transmit Status Base Address Register . . . 222
10.10.2.10 Transmit Number of Descriptors Register . 222
10.10.2.11 Transmit Produce Index Register 223
10.10.2.12 Transmit Consume Index Register 223
10.10.2.13 Transmit Status Vector 0 Register 223
10.10.2.14 Transmit Status Vector 1 Register 225
10.10.2.15 Receive Status Vector Register 226
10.10.2.16 Flow Control Counter Register 227
10.10.2.17 Flow Control Status Register 227
10.10.3 Receive filter register definitions 228
10.10.3.1 Receive Filter Control Register 228
10.10.3.2 Receive Filter WoL Status Register 228
10.10.3.3 Receive Filter WoL Clear Register 229
10.10.3.4 Hash Filter Table LSBs Register 229
10.10.3.5 Hash Filter Table MSBs Register 229
10.10.4 Module control register definitions 230
10.10.4.1 Interrupt Status Register 230
10.10.4.2 Interrupt Enable Register 231
10.10.4.3 Interrupt Clear Register 231
10.10.4.4 Interrupt Set Register 232
10.10.4.5 Power-Down Register 232
10.11 Descriptor and status formats 233
10.11.1 Receive descriptors and statuses 233
10.11.2 Transmit descriptors and statuses 236
10.12 Ethernet block functional description 239
10.12.1 Overview 239
10.12.2 AHB interface 239
10.13 Interrupts 241
10.13.1 Direct Memory Access (DMA) 241
10.13.2 Initialization 243
10.13.3 Transmit process 245
10.13.4 Receive process 251
10.13.5 Transmission retry 257
10.13.6 Status hash CRC calculations 257
10.13.7 Duplex modes 258
10.13.8 IEE 802.3/Clause 31 flow control 258
10.13.9 Half-Duplex mode backpressure 260
10.13.10 Receive filtering 261
10.13.11 Power management 263
10.13.12 Wake-up on LAN 264
10.13.13 Enabling and disabling receive and transmit 265
10.13.14 Transmission padding and CRC 267
10.13.15 Huge frames and frame length checking . . . 268
10.13.16 Statistics counters 268
10.13.17 MAC status vectors 268
10.13.18 Reset 269
10.13.19 Ethernet errors 270
10.14 AHB bandwidth 271
10.14.1 DMA access 271
10.14.2 Types of CPU access 272
10.14.3 Overall bandwidth 272
10.15 CRC calculation 274

Chapter 11: LPC408x/407x LCD controller

11.1 How to read this chapter 276
11.2 Basic configuration 276
11.3 Introduction 276
11.4 Features 277
11.4.1 Programmable parameters 277
11.4.2 Hardware cursor support 278
11.4.3 Types of LCD panels supported 278
11.4.4 TFT panels 278
11.4.5 Color STN panels 279
11.4.6 Monochrome STN panels 279
11.5 Pin description 280

11.5.1 Signal usage 280
11.5.1.1 Signals used for single panel STN displays . 280
11.5.1.2 Signals used for dual panel STN displays . . 280
11.5.1.3 Signals used for TFT displays 281
11.6 LCD controller functional description 282
11.6.1 AHB interfaces 283
11.6.1.1 AMBA AHB slave interface 283
11.6.1.2 AMBA AHB master interface 283
11.6.2 Dual DMA FIFOs and associated control logic 284
11.6.3 Pixel serializer 284
11.6.4 RAM palette 288
11.6.5 Hardware cursor 290
11.6.5.1 Cursor operation 290
11.6.5.2 Cursor sizes 290
11.6.5.3 Cursor movement 291
11.6.5.4 Cursor XY positioning 291
11.6.5.5 Cursor clipping 292
11.6.5.6 Cursor image format 292
11.6.6 Gray scaler 295
11.6.7 Upper and lower panel formatters 295
11.6.8 Panel clock generator 296
11.6.9 Timing controller 296
11.6.10 STN and TFT data select 296
11.6.10.1 STN displays 296
11.6.10.2 TFT displays 296
11.6.11 Interrupt generation 296
11.6.11.1 Master bus error interrupt 297
11.6.11.2 Vertical compare interrupt 297
11.6.11.2.1 Next base address update interrupt 297
11.6.11.2.2 FIFO underflow interrupt 297
11.6.12 LCD power-up and power-down sequence . 297
11.7 Register description 299
11.7.1 Horizontal Timing register 300
11.7.1.1 Horizontal timing restrictions 300
11.7.2 Vertical Timing register 301
11.7.3 Clock and Signal Polarity register 302
11.7.4 Line End Control register 303
11.7.5 Upper Panel Frame Base Address register . 304
11.7.6 Lower Panel Frame Base Address register . 304
11.7.7 LCD Control register 305
11.7.8 Interrupt Mask register 307
11.7.9 Raw Interrupt Status register 308
11.7.10 Masked Interrupt Status register 308
11.7.11 Interrupt Clear register 309
11.7.12 Upper Panel Current Address register 309
11.7.13 Lower Panel Current Address register 309
11.7.14 Color Palette registers 310
11.7.15 Cursor Image registers 310
11.7.16 Cursor Control register 311
11.7.17 Cursor Configuration register 311
11.7.18 Cursor Palette register 0 312
11.7.19 Cursor Palette register 1 312
11.7.20 Cursor XY Position register 313
11.7.21 Cursor Clip Position register 313
11.7.22 Cursor Interrupt Mask register 314
11.7.23 Cursor Interrupt Clear register 314
11.7.24 Cursor Raw Interrupt Status register 314
11.7.25 Cursor Masked Interrupt Status register . . . 315
11.8 LCD timing diagrams 316
11.9 LCD panel signal usage 319

Chapter 12: LPC408x/407x USB device controller

12.1 How to read this chapter 322
12.2 Basic configuration 322
12.3 Introduction 322
12.4 Features 323
12.5 Fixed endpoint configuration 323
12.6 Functional description 325
12.6.1 Analog transceiver 325
12.6.2 Serial Interface Engine (SIE) 325
12.6.3 Endpoint RAM (EP_RAM) 325
12.6.4 EP_RAM access control 326
12.6.5 DMA engine and bus master interface 326
12.6.6 Register interface 326
12.6.7 SoftConnect 326
12.6.8 GoodLink 326
12.7 Operational overview 327
12.8 Pin description 327
12.9 Clocking and power management 328
12.9.1 Power requirements 328
12.9.2 Clocks 328
12.9.3 Power management support 328
12.9.4 Remote wake-up 329
12.10 Register description 330
12.10.1 Port select register 332
12.10.1.1 USB Port Select register 332
12.10.2 Device interrupt registers 332
12.10.2.1 USB Device Interrupt Status register 332
12.10.2.2 USB Device Interrupt Enable register 333
12.10.2.3 USB Device Interrupt Clear register 334
12.10.2.4 USB Device Interrupt Set register 335
12.10.2.5 USB Device Interrupt Priority register 336
12.10.3 Endpoint interrupt registers 337
12.10.3.1 USB Endpoint Interrupt Status register 337
12.10.3.2 USB Endpoint Interrupt Enable register . . . 338
12.10.3.3 USB Endpoint Interrupt Clear register 338
12.10.3.4 USB Endpoint Interrupt Set register 339
12.10.3.5 USB Endpoint Interrupt Priority register 339
12.10.4 Endpoint realization registers 340
12.10.4.1 EP RAM requirements 340
12.10.4.2 USB Realize Endpoint register 341
12.10.4.3 USB Endpoint Index register 342
12.10.4.4 USB MaxPacketSize register 342
12.10.5 USB transfer registers 343
12.10.5.1 USB Receive Data register 343
12.10.5.2 USB Receive Packet Length register 343
12.10.5.3 USB Transmit Data register 344
12.10.5.4 USB Transmit Packet Length register 344
12.10.5.5 USB Control register 345

12.10.6 SIE command code registers 346
12.10.6.1 USB Command Code register 346
12.10.6.2 USB Command Data register 346
12.10.7 DMA registers 347
12.10.7.1 USB DMA Request Status register 347
12.10.7.2 USB DMA Request Clear register 348
12.10.7.3 USB DMA Request Set register 348
12.10.7.4 USB UDCA Head register 349
12.10.7.5 USB EP DMA Status register 349
12.10.7.6 USB EP DMA Enable register 349
12.10.7.7 USB EP DMA Disable register 350
12.10.7.8 USB DMA Interrupt Status register 350
12.10.7.9 USB DMA Interrupt Enable register 351
12.10.7.10 USB End of Transfer Interrupt Status register 351
12.10.7.11 USB End of Transfer Interrupt Clear register 351
12.10.7.12 USB End of Transfer Interrupt Set register . 352
12.10.7.13 USB New DD Request Interrupt Status register 352
12.10.7.14 USB New DD Request Interrupt Clear register 352
12.10.7.15 USB New DD Request Interrupt Set register 352
12.10.7.16 USB System Error Interrupt Status register 353
12.10.7.17 USB System Error Interrupt Clear register . 353
12.10.7.18 USB System Error Interrupt Set register . . 353
12.10.8 Clock control registers 354
12.10.8.1 USB Clock Control register 354
12.10.8.2 USB Clock Status register 354
12.11 Interrupt handling 355
Slave mode 355
DMA mode 355
12.12 Serial interface engine command description . . 358
12.12.1 Set Address (Command: 0xD0, Data: write 1 byte) 359
12.12.2 Configure Device (Command: 0xD8, Data: write 1 byte) 360
12.12.3 Set Mode (Command: 0xF3, Data: write 1 byte) 360
12.12.4 Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes) 361
12.12.5 Read Test Register (Command: 0xFD, Data: read 2 bytes) 361
12.12.6 Set Device Status (Command: 0xFE, Data: write 1 byte) 362
12.12.7 Get Device Status (Command: 0xFE, Data: read 1 byte) 363
12.12.8 Get Error Code (Command: 0xFF, Data: read 1 byte) 363
12.12.9 Read Error Status (Command: 0xFB, Data: read 1 byte) 364
12.12.10 Select Endpoint (Command: 0x00 - 0x1F, Data: read 1 byte (optional)) 365
12.12.11 Select Endpoint/Clear Interrupt (Command: 0x40 - 0x5F, Data: read 1 byte) 366
12.12.12 Set Endpoint Status (Command: 0x40 - 0x55, Data: write 1 byte (optional)) 366
12.12.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional)) 367
12.12.14 Validate Buffer (Command: 0xFA, Data: none) 367
12.13 USB device controller initialization 368
12.14 Slave mode operation 370
12.14.1 Interrupt generation 370
12.14.2 Data transfer for OUT endpoints 370
12.14.3 Data transfer for IN endpoints 370
12.15 DMA operation 372
12.15.1 Transfer terminology 372
12.15.2 USB device communication area 372
12.15.3 Triggering the DMA engine 373
12.15.4 The DMA descriptor 373
12.15.4.1 Next_DD_pointer 374
12.15.4.2 DMA_mode 374
12.15.4.3 Next_DD_valid 375
12.15.4.4 Isochronous_endpoint 375
12.15.4.5 Max_packet_size 375
12.15.4.6 DMA_buffer_length 375
12.15.4.7 DMA_buffer_start_addr 375
12.15.4.8 DD_retired 375
12.15.4.9 DD_status 375
12.15.4.10 Packet_valid 376
12.15.4.11 LS_byte_extracted 376
12.15.4.12 MS_byte_extracted 376
12.15.4.13 Present_DMA_count 376
12.15.4.14 Message_length_position 376
12.15.4.15 Isochronous_packetsize_memory_address 376
12.15.5 Non-isochronous endpoint operation 377
12.15.5.1 Setting up DMA transfers 377
12.15.5.2 Finding DMA Descriptor 377
12.15.5.3 Transferring the data 377
12.15.5.4 Optimizing descriptor fetch 377
12.15.5.5 Ending the packet transfer 378
12.15.5.6 No_Packet DD 378
12.15.6 Isochronous endpoint operation 378
12.15.6.1 Setting up DMA transfers 378
12.15.6.2 Finding the DMA Descriptor 379
12.15.6.3 Transferring the Data 379
OUT endpoints 379
IN endpoints 379
12.15.6.4 DMA descriptor completion 379
12.15.6.5 Isochronous OUT Endpoint Operation Example 379
12.15.7 Auto Length Transfer Extraction (ATLE) mode operation 380
OUT transfers in ATLE mode 380
IN transfers in ATLE mode 382
12.15.7.1 Setting up the DMA transfer 382
12.15.7.2 Finding the DMA Descriptor 382
12.15.7.3 Transferring the Data 382
OUT endpoints 382
IN endpoints 382
12.15.7.4 Ending the packet transfer 383
OUT endpoints 383
IN endpoints 383

12.16 Double buffered endpoint operation 384
12.16.1 Bulk endpoints 384
12.16.2 Isochronous endpoints 385

Chapter 13: LPC408x/407x USB Host controller

13.1 How to read this chapter 387
13.2 Basic configuration 387
13.3 Introduction 388
13.3.1 Features 388
13.3.2 Architecture 389
13.4 Interfaces 390
13.4.1 Pin description 390
13.4.1.1 USB host usage note 391
13.4.2 Software interface 391
13.4.2.1 Register map 391
13.4.2.2 USB Host Register Definitions 392

Chapter 14: LPC408x/407x USB OTG controller

14.1 How to read this chapter 393
14.2 Basic configuration 393
14.3 Introduction 393
14.4 Features 393
14.5 Architecture 394
14.6 Modes of operation 394
14.7 Pin configuration 395
14.7.1 Using port U1 for OTG operation 396
14.7.2 Using both ports U1 and U2 for host operation 398
14.7.3 Using U1 for host operation and U2 for device operation 399
14.8 Register description 400
14.8.1 OTG Interrupt Status Register 400
14.8.2 OTG Interrupt Enable Register 401
14.8.3 OTG Interrupt Set Register 401
14.8.4 OTG Interrupt Clear Register 401
14.8.5 OTG Status and Control Register 403
14.8.6 OTG Timer Register 404
14.8.7 I2C Receive Register 404
14.8.8 I2C Transmit Register 405
14.8.9 I2C Status Register 405
14.8.10 I2C Control Register 407
14.8.11 I2C Clock High Register 408
14.8.12 I2C Clock Low Register 408
14.8.13 OTG Clock Control Register 409
14.8.14 OTG Clock Status Register 410
14.8.15 Interrupt handling 411
14.9 HNP support 412
14.9.1 B-device: peripheral to host switching 413
Remove D+ pull-up 415
Add D+ pull-up 416
14.9.2 A-device: host to peripheral HNP switching . 416
Set BDIS_ACON_EN in external OTG transceiver 419
Clear BDIS_ACON_EN in external OTG transceiver 419
Discharge V_{BUS} 419
Load and enable OTG timer 420
Stop OTG timer 420
Suspend host on port 1 420
14.10 Clocking and power management 421
14.10.1 Device clock request signals 423
14.10.1.1 Host clock request signals 423
14.10.2 Power-down mode support 423
14.11 USB OTG controller initialization 424

Chapter 15: LPC408x/407x SPI flash interface (SPIFI)

15.1 Basic configuration 425
15.2 Features 425
15.3 General description 426
15.4 Pin description 426
15.5 Supported devices 427
15.6 SPIFI hardware 428
15.7 Register description 428
15.7.1 SPIFI control register 428
15.7.2 SPIFI command register 429
15.7.3 SPIFI address register 431
15.7.4 SPIFI intermediate data register 431
15.7.5 SPIFI cache limit register 432
15.7.6 SPIFI data register 432
15.7.7 SPIFI memory command register 432
15.7.8 SPIFI status register 434
15.8 Functional description 434
15.8.1 Data transfer 434
15.8.2 Software requirements and capabilities 436
15.8.3 Peripheral mode DMA operation 437

Chapter 16: LPC408x/407x SD card interface

16.1 How to read this chapter 439
16.2 Basic configuration 439
16.3 Introduction 439
16.4 Features 439
16.5 Pin description 440
16.6 Functional overview 440
16.6.1 Secure digital memory card 440
16.6.1.1 Secure digital memory card bus signals . . . 440
16.6.2 Multimedia card 440

16.6.3 SD card interface details . . . 441
16.6.3.1 Adapter register block . . . 442
16.6.3.2 Control unit . . . 442
16.6.3.3 Command path . . . 443
16.6.3.4 Command path state machine . . . 443
16.6.3.5 Command format . . . 444
16.6.3.6 Data path . . . 445
16.6.3.7 Data path state machine . . . 446
16.6.3.8 Data counter . . . 447
16.6.3.9 Bus mode . . . 448
16.6.3.10 CRC Token status . . . 448
16.6.3.11 Status flags . . . 449
16.6.3.12 CRC generator . . . 449
16.6.3.13 Data FIFO . . . 449
16.6.3.14 Transmit FIFO . . . 450
16.6.3.15 Receive FIFO . . . 450
16.6.3.16 APB interfaces . . . 451
16.6.3.17 Interrupt logic . . . 451

16.7 Register description . . . 452
16.7.1 Power Control Register . . . 452
16.7.2 Clock Control Register . . . 453
16.7.3 Argument Register . . . 453
16.7.4 Command Register . . . 454
16.7.5 Command Response Register . . . 454
16.7.6 Response Registers . . . 454
16.7.7 Data Timer Register . . . 455
16.7.8 Data Length Register . . . 455
16.7.9 Data Control Register . . . 456
16.7.10 Data Counter Register . . . 456
16.7.11 Status Register . . . 457
16.7.12 Clear Register . . . 458
16.7.13 Interrupt Mask Registers . . . 459
16.7.14 FIFO Counter Register . . . 459
16.7.15 Data FIFO Register . . . 460

Chapter 17: LPC408x/407x UART1

17.1 Basic configuration . . . 461
17.2 Features . . . 461
17.3 Architecture . . . 462
17.4 Pin description . . . 464
17.5 Register description . . . 465
17.5.1 UART1 Receiver Buffer Register . . . 466
17.5.2 UART1 Transmitter Holding Register . . . 466
17.5.3 UART1 Divisor Latch LSB and MSB Registers . . . 467
17.5.4 UART1 Interrupt Enable Register . . . 468
17.5.5 UART1 Interrupt Identification Register . . . 469
17.5.6 UART1 FIFO Control Register . . . 472
17.5.6.1 DMA Operation . . . 472
UART receiver DMA . . . 472
UART transmitter DMA . . . 473
17.5.7 UART1 Line Control Register . . . 474
17.5.8 UART1 Modem Control Register . . . 475
17.5.9 Auto-flow control . . . 475
17.5.9.1 Auto-RTS . . . 475
17.5.9.2 Auto-CTS . . . 476
17.5.10 UART1 Line Status Register . . . 478
17.5.11 UART1 Modem Status Register . . . 479
17.5.12 UART1 Scratch Pad Register . . . 481
17.5.13 UART1 Auto-baud Control Register . . . 481
17.5.14 Auto-baud . . . 482
17.5.15 Auto-baud modes . . . 483
17.5.16 UART1 Fractional Divider Register . . . 485
17.5.16.1 Baud rate calculation . . . 486
17.5.16.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600 . . . 487
17.5.16.1.2 Example 2: PCLK = 12 MHz, BR = 115200 487
17.5.17 UART1 Transmit Enable Register . . . 488
17.5.18 UART1 RS485 Control register . . . 489
17.5.19 UART1 RS-485 Address Match register . . . 489
17.5.20 UART1 RS-485 Delay value register . . . 490
17.5.21 RS-485/EIA-485 modes of operation . . . 490
RS-485/EIA-485 Normal Multidrop Mode (NMM) . . . 490
RS-485/EIA-485 Auto Address Detection (AAD) mode . . . 490
RS-485/EIA-485 Auto Direction Control . . . 491
RS485/EIA-485 driver delay time . . . 491
RS485/EIA-485 output inversion . . . 491

Chapter 18: LPC408x/407x UART0/2/3

18.1 How to read this chapter . . . 492
18.2 Basic configuration . . . 492
18.3 Features . . . 493
18.4 Architecture . . . 493
18.5 Pin description . . . 494
18.6 Register description . . . 495
18.6.1 UARTn Receiver Buffer Register . . . 496
18.6.2 UARTn Transmit Holding Register . . . 496
18.6.3 UARTn Divisor Latch LSB register . . . 497
18.6.4 UARTn Interrupt Enable Register . . . 498
18.6.5 UARTn Interrupt Identification Register . . . 499
18.6.6 UARTn FIFO Control Register . . . 501
18.6.6.1 DMA Operation . . . 501
UART receiver DMA . . . 501
UART transmitter DMA . . . 501
18.6.7 UARTn Line Control Register . . . 502
18.6.8 UARTn Line Status Register . . . 503
18.6.9 UARTn Scratch Pad Register . . . 504
18.6.10 UARTn Auto-baud Control Register . . . 505
18.6.10.1 Auto-baud . . . 505
18.6.10.2 Auto-baud modes . . . 506
18.6.11 UARTn Fractional Divider Register . . . 508
18.6.11.1 Baud rate calculation . . . 509
18.6.11.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600 . . . 510

18.6.11.1.2 Example 2: PCLK = 12 MHz, BR = 115200 510
18.6.12 UARTn Transmit Enable Register 511
18.6.13 UARTn RS485 Control register 512
18.6.14 UARTn RS-485 Address Match register . . . 512
18.6.15 UARTn RS-485 Delay value register 513
18.6.16 RS-485/EIA-485 modes of operation 513
RS-485/EIA-485 Normal Multidrop Mode (NMM) 513
RS-485/EIA-485 Auto Address Detection (AAD) mode 513
RS-485/EIA-485 Auto Direction Control. 514
RS485/EIA-485 driver delay time. 514
RS485/EIA-485 output inversion 514

Chapter 19: LPC408x/407x UART4

19.1 How to read this chapter. 515
19.2 Basic configuration 515
19.3 Features 516
19.4 Architecture 516
19.5 Pin description. 517
19.6 Register description 518
19.6.1 UART4 Receiver Buffer Register 519
19.6.2 UART4 Transmit Holding Register 519
19.6.3 UART4 Divisor Latch LSB register 520
19.6.4 UART4 Interrupt Enable Register. 521
19.6.5 UART4 Interrupt Identification Register 522
19.6.6 UART4 FIFO Control Register 525
19.6.6.1 DMA Operation. 525
UART receiver DMA. 525
UART transmitter DMA. 525
19.6.7 UART4 Line Control Register. 526
19.6.8 UART4 Line Status Register. 527
19.6.9 UART4 Scratch Pad Register 528
19.6.10 UART4 Auto-baud Control Register 529
19.6.10.1 Auto-baud. 529
19.6.10.2 Auto-baud modes 530
19.6.11 UART4 IrDA Control Register 532
19.6.12 UART4 Fractional Divider Register 533
19.6.12.1 Baud rate calculation 534
19.6.12.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600 535
19.6.12.1.2 Example 2: PCLK = 12 MHz, BR = 115200 535
19.6.13 UART4 Oversampling Register 536
19.6.14 UART4 Smart Card Interface Control register 537
19.6.14.1 Smart card Connection 537
19.6.14.2 Smart card Setup. 537
19.6.15 UART4 RS485 Control register 538
19.6.16 UART4 RS-485 Address Match register . . . 539
19.6.17 UART4 RS-485 Delay value register 539
19.6.18 RS-485/EIA-485 modes of operation. 539
RS-485/EIA-485 Normal Multidrop Mode (NMM) 539
RS-485/EIA-485 Auto Address Detection (AAD) mode 539
RS-485/EIA-485 Auto Direction Control. 540
RS485/EIA-485 driver delay time. 540
RS485/EIA-485 output inversion 540
19.6.19 UART4 Synchronous mode control register 541

Chapter 20: LPC408x/407x CAN controller

20.1 Basic configuration 543
20.2 CAN controllers 543
20.3 Features 543
20.3.1 General CAN features 543
20.3.2 CAN controller features 544
20.3.3 Acceptance filter features 544
20.4 Pin description. 544
20.5 CAN controller architecture 544
20.5.1 APB Interface Block (AIB) 545
20.5.2 Interface Management Logic (IML). 545
20.5.3 Transmit Buffers (TXB) 545
20.5.4 Receive Buffer (RXB) 546
20.5.5 Error Management Logic (EML) 547
20.5.6 Bit Timing Logic (BTL) 547
20.5.7 Bit Stream Processor (BSP) 547
20.5.8 CAN controller self-tests. 547
Global self test 548
Local self test 548
20.6 Memory map of the CAN block. 549
20.7 Register description 549
20.7.1 CAN Mode register 551
20.7.2 CAN Command Register 553
20.7.3 CAN Global Status Register 555
RX error counter 556
TX error counter. 557
20.7.4 CAN Interrupt and Capture Register 557
20.7.5 CAN Interrupt Enable Register 561
20.7.6 CAN Bus Timing Register 562
Baud rate prescaler 562
Synchronization jump width 563
Time segment 1 and time segment 2. 563
20.7.7 CAN Error Warning Limit register 563
20.7.8 CAN Status Register 563
20.7.9 CAN Receive Frame Status register 565
20.7.9.1 ID index field 566
20.7.10 CAN Receive Identifier register 566
20.7.11 CAN Receive Data register A 566
20.7.12 CAN Receive Data register B 567
20.7.13 CAN Transmit Frame Information register . . 567
Automatic transmit priority detection 568
Tx DLC. 568
20.7.14 CAN Transmit Identifier register 569
20.7.15 CAN Transmit Data register A 569
20.7.16 CAN Transmit Data register B 569
20.8 CAN controller operation 570
20.8.1 Error handling 570
20.8.2 Sleep mode 570

20.8.3 Interrupts. 571
20.8.4 Transmit priority 571
20.9 Centralized CAN registers. 571
20.9.1 Central Transmit Status Register 571
20.9.2 Central Receive Status Register 572
20.9.3 Central Miscellaneous Status Register 572
20.10 Global acceptance filter 572
20.11 Acceptance filter modes 572
20.11.1 Acceptance filter Off mode 573
20.11.2 Acceptance filter Bypass mode 573
20.11.3 Acceptance filter Operating mode 573
20.11.4 FullCAN mode 573
20.12 Sections of the ID look-up table RAM 573
20.13 ID look-up table RAM. 574
20.14 Acceptance filter registers 576
20.14.1 Acceptance Filter Mode Register 576
20.14.2 Section configuration registers 577
20.14.3 Standard Frame Individual Start Address register 577
20.14.4 Standard Frame Group Start Address register 577
20.14.5 Extended Frame Start Address register 578
20.14.6 Extended Frame Group Start Address register 578
20.14.7 End of AF Tables register 578
20.14.8 Status registers. 579
20.14.9 LUT Error Address register 579
20.14.10 LUT Error register 579
20.14.11 Global FullCANInterrupt Enable register . . . 580
20.14.12 FullCAN Interrupt and Capture registers . . . 580
20.15 Configuration and search algorithm 580
20.15.1 Acceptance filter search algorithm 580
20.16 FullCAN mode 582
20.16.1 FullCAN message layout 583
20.16.2 FullCAN interrupts 586
20.16.2.1 FullCAN message interrupt enable bit 586
20.16.2.2 Message lost bit and CAN channel number . 587
20.16.2.3 Setting the interrupt pending bits (IntPnd 63 to 0) 588
20.16.2.4 Clearing the interrupt pending bits (IntPnd 63 to 0) 588
20.16.2.5 Setting the message lost bit of a FullCAN message object (MsgLost 63 to 0). 588
20.16.2.6 Clearing the message lost bit of a FullCAN message object (MsgLost 63 to 0) 588
20.16.3 Set and clear mechanism of the FullCAN interrupt 588
20.16.3.1 Scenario 1: Normal case, no message lost . 588
20.16.3.2 Scenario 2: Message lost. 589
20.16.3.3 Scenario 3: Message gets overwritten indicated by Semaphore bits 590
20.16.3.4 Scenario 3.1: Message gets overwritten indicated by Semaphore bits and Message Lost. 590
20.16.3.5 Scenario 3.2: Message gets overwritten indicated by Message Lost 591
20.16.3.6 Scenario 4: Clearing Message Lost bit 592
20.17 Examples of acceptance filter tables and ID index values. 593
20.17.1 Example 1: only one section is used 593
20.17.2 Example 2: all sections are used 593
20.17.3 Example 3: more than one but not all sections are used 593
20.17.4 Configuration example 4 594
20.17.5 Configuration example 5 594
20.17.6 Configuration example 6 595
Explicit standard frame format identifier section (11-bit CAN ID): 596
Group of standard frame format identifier section (11-bit CAN ID): 596
Explicit extended frame format identifier section (29-bit CAN ID, Figure 100) 596
Group of extended frame format identifier section (29-bit CAN ID, Figure 100) 596
20.17.7 Configuration example 7 597
FullCAN explicit standard frame format identifier section (11-bit CAN ID) 598
Explicit standard frame format identifier section (11-bit CAN ID) 598
FullCAN message object data section 598
20.17.8 Look-up table programming guidelines 599

Chapter 21: LPC408x/407x SSP interfaces

21.1 Basic configuration 601
21.2 Features 601
21.3 Description 601
21.4 Pin descriptions 602
21.5 Bus description 603
21.5.1 Texas Instruments synchronous serial frame format 603
21.5.2 SPI frame format 603
21.5.2.1 Clock Polarity (CPOL) and Phase (CPHA) control 604
21.5.2.2 SPI format with CPOL=0,CPHA=0 604
21.5.2.3 SPI format with CPOL=0,CPHA=1 605
21.5.2.4 SPI format with CPOL = 1,CPHA = 0 606
21.5.2.5 SPI format with CPOL = 1,CPHA = 1. 607
21.5.3 National Semiconductor Microwire frame format. 607
21.5.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode 609
21.6 Register description 610
21.6.1 SSPn Control Register 0 611
21.6.2 SSPn Control Register 1 612
21.6.3 SSPn Data Register 612
21.6.4 SSPn Status Register 613
21.6.5 SSPn Clock Prescale Register 613
21.6.6 SSPn Interrupt Mask Set/Clear Register . . 614
21.6.7 SSPn Raw Interrupt Status Register 614

21.6.8 SSPn Masked Interrupt Status Register . . . 615
21.6.9 SSPn Interrupt Clear Register 615
21.6.10 SSPn DMA Control Register 615

Chapter 22: LPC408x/407x I2C-bus interfaces

22.1 Basic configuration 616
22.2 Features 617
22.3 Applications 617
22.4 Description 617
22.4.1 I2C FAST Mode Plus. 618
22.5 Pin description 619
22.6 I2C operating modes 619
22.6.1 Master Transmitter mode 619
22.6.2 Master Receiver mode 620
22.6.3 Slave Receiver mode 621
22.6.4 Slave Transmitter mode 622
22.7 I2C implementation and operation 623
22.7.1 Input filters and output stages. 623
22.7.2 Address Registers, I2ADR0 to I2ADR3 625
22.7.3 Address mask registers, I2MASK0 to I2MASK3. . 625
22.7.4 Comparator. 625
22.7.5 Shift register, I2DAT 625
22.7.6 Arbitration and synchronization logic 625
22.7.7 Serial clock generator. 626
22.7.8 Timing and control 627
22.7.9 Control register, I2CONSET and I2CONCLR 627
22.7.10 Status decoder and status register 627
22.8 Register description 628
22.8.1 I2C Control Set register. 629
22.8.2 I2C Control Clear register 631
22.8.3 I2C Status register 631
22.8.4 I2C Data register 632
22.8.5 I2C Monitor mode control register 632
22.8.5.1 Interrupt in Monitor mode 633
22.8.5.2 Loss of arbitration in Monitor mode 633
22.8.6 I2C Data buffer register 633
22.8.7 I2C Slave Address registers 634
22.8.8 I2C Mask registers 634
22.8.9 I2C SCL HIGH duty cycle register. 635
22.8.10 I2C SCL Low duty cycle register 635
22.8.11 Selecting the appropriate I2C data rate and duty cycle 635
22.9 Details of I2C operating modes. 636
22.9.1 Master Transmitter mode 637
22.9.2 Master Receiver mode 639
22.9.3 Slave Receiver mode 641
22.9.4 Slave Transmitter mode 643
22.9.5 Detailed state tables 644
22.9.6 Miscellaneous states 649
22.9.6.1 I2STAT = 0xF8 649
22.9.6.2 I2STAT = 0x00 649
22.9.7 Some special cases 650
22.9.7.1 Simultaneous repeated START conditions from two masters 650
22.9.7.2 Data transfer after loss of arbitration 650
22.9.7.3 Forced access to the I2C-bus. 650
22.9.7.4 I2C-bus obstructed by a LOW level on SCL or SDA 650
22.9.7.5 Bus error 651
22.9.8 I2C state service routines. 653
22.9.8.1 Initialization 653
22.9.8.2 I2C interrupt service 653
22.9.8.3 The state service routines 653
22.9.8.4 Adapting state services to an application. . . 653
22.10 Software example 654
22.10.1 Initialization routine 654
22.10.2 Start Master Transmit function 654
22.10.3 Start Master Receive function 654
22.10.4 I2C interrupt routine 654
22.10.5 Non mode specific states. 655
22.10.5.1 State: 0x00. 655
22.10.5.2 Master States 655
22.10.5.3 State: 0x08. 655
22.10.5.4 State: 0x10. 655
22.10.6 Master Transmitter states 656
22.10.6.1 State: 0x18. 656
22.10.6.2 State: 0x20. 656
22.10.6.3 State: 0x28. 656
22.10.6.4 State: 0x30. 656
22.10.6.5 State: 0x38. 657
22.10.7 Master Receiver states 658
22.10.7.1 State: 0x40. 658
22.10.7.2 State: 0x48. 658
22.10.7.3 State: 0x50. 658
22.10.7.4 State: 0x58. 658
22.10.8 Slave Receiver states 659
22.10.8.1 State: 0x60. 659
22.10.8.2 State: 0x68. 659
22.10.8.3 State: 0x70. 659
22.10.8.4 State: 0x78. 659
22.10.8.5 State: 0x80. 660
22.10.8.6 State: 0x88. 660
22.10.8.7 State: 0x90. 660
22.10.8.8 State: 0x98. 660
22.10.8.9 State: 0xA0. 660
22.10.9 Slave Transmitter states 661
22.10.9.1 State: 0xA8. 661
22.10.9.2 State: 0xB0. 661
22.10.9.3 State: 0xB8. 661
22.10.9.4 State: 0xC0 661
22.10.9.5 State: 0xC8 662

Chapter 23: LPC408x/407x I2S interface

23.1 Basic configuration 663
23.2 Features 664

23.3 Description 665
23.4 Pin descriptions 666
23.5 Register description 668
23.5.1 Digital Audio Output register 669
23.5.2 Digital Audio Input register 669
23.5.3 Transmit FIFO register 670
23.5.4 Receive FIFO register 670
23.5.5 Status Feedback register 670
23.5.6 DMA Configuration Register 1 671
23.5.7 DMA Configuration Register 2 671
23.5.8 Interrupt Request Control register 672
23.5.9 Transmit Clock Rate register 672
23.5.9.1 Notes on fractional rate generators 673
23.5.10 Receive Clock Rate register 673
23.5.11 Transmit Clock Bit Rate register 674
23.5.12 Receive Clock Bit Rate register 674
23.5.13 Transmit Mode Control register 674
23.5.14 Receive Mode Control register 675
23.6 I^2S transmit and receive interfaces 676
23.7 I^2S operating modes 677
23.7.1 I^2S transmit modes 678
23.7.2 I^2S receive modes 681
23.7.2.1 Overall clocking and pin connections 684
23.8 FIFO controller 685

Chapter 24: LPC408x/407x Timer0/1/2/3

24.1 Basic configuration 687
24.2 Features 687
24.3 Applications 688
24.4 Description 688
24.5 Pin description 690
24.5.1 Multiple CAP and MAT pins 690
24.6 Register description 691
24.6.1 Interrupt Register 692
24.6.2 Timer Control Register 692
24.6.3 Timer Counter registers 693
24.6.4 Prescale register 693
24.6.5 Prescale Counter register 693
24.6.6 Match Control Register 693
24.6.7 Match Registers (MR0 to MR3) 695
24.6.8 Capture Control Register 695
24.6.9 Capture Registers 696
24.6.10 External Match Register 697
24.6.11 Count Control Register 698
24.6.12 DMA operation 699
24.7 Example timer operation 700

Chapter 25: LPC408x/407x System Tick timer

25.1 Basic configuration 701
25.2 Features 701
25.3 Description 701
25.4 Operation 701
25.5 Register description 703
25.5.1 System Timer Control and status register . . 703
25.5.2 System Timer Reload value register 703
25.5.3 System Timer Current value register 704
25.5.4 System Timer Calibration value register . . . 704
25.6 Example timer calculations 705
Example 1) 705
Example 2) 705
Example 3) 705
Example 4) 705

Chapter 26: LPC408x/407x Pulse Width Modulators (PWM0/1)

26.1 Basic configuration 706
26.2 Features 707
26.3 Description 708
26.4 Sample waveform with rules for single and double edge control 710
26.4.1 Rules for Single Edge Controlled PWM Outputs 711
26.4.2 Rules for Double Edge Controlled PWM Outputs 711
26.5 Pin description 711
26.6 Register description 712
26.6.1 PWM Interrupt Register 712
26.6.2 PWM Timer Control Register 714
26.6.3 PWM Timer Counter 714
26.6.4 PWM Prescale Register 715
26.6.5 PWM Prescale Counter Register 715
26.6.6 PWM Match Control Register 715
26.6.7 PWM Match Registers 717
26.6.8 PWM Capture Control Register 718
26.6.9 PWM Capture Registers 718
26.6.10 PWM Control Registers 719
26.6.11 PWM Latch Enable Register 720
26.6.12 PWM Count Control Register 722

Chapter 27: LPC408x/407x Motor control PWM

27.1 Basic configuration 723
27.2 Introduction 723
27.3 Description 723
27.4 Pin description 724
27.5 Block diagram 725
27.6 Configuring other modules for MCPWM use 726
27.7 General operation 726
27.8 Register description 727

27.8.1 MCPWM Control register 728
27.8.1.1 MCPWM Control read address 728
27.8.1.2 MCPWM Control set address 729
27.8.1.3 MCPWM Control clear address 730
27.8.2 PWM Capture Control register 731
27.8.2.1 MCPWM Capture Control read address . . . 731
27.8.2.2 MCPWM Capture Control set address 732
27.8.2.3 MCPWM Capture control clear address . . . 732
27.8.3 MCPWM Timer/Counter 0-2 registers 733
27.8.4 MCPWM Limit 0-2 registers 734
27.8.5 MCPWM Match 0-2 registers 734
27.8.5.1 Match register in Edge-Aligned mode. 735
27.8.5.2 Match register in Center-Aligned mode 735
27.8.5.3 0 and 100% duty cycle 735
27.8.6 MCPWM Dead-time register 735
27.8.7 MCPWM Communication Pattern register . . 736
27.8.8 MCPWM Capture read addresses 736
27.8.9 MCPWM Interrupt registers 737
8.9.1 MCPWM Interrupt Enable read address . . . 737
27.8.9.2 MCPWM Interrupt Enable set address 738
27.8.9.3 MCPWM Interrupt Enable clear address . . . 738
27.8.10 MCPWM Count Control register 739
27.8.10.1 MCPWM Count Control read address 739
27.8.10.2 MCPWM Count Control set address 741
27.8.10.3 MCPWM Count Control clear address 742
27.8.11 MCPWM Interrupt flag registers. 742
27.8.11.1 MCPWM Interrupt Flags read address 742
27.8.11.2 MCPWM Interrupt Flags set address 744
27.8.11.3 MCPWM Interrupt Flags clear address 744
27.8.12 MCPWM Capture clear address 745
27.9 PWM operation 746
27.9.1 Pulse-width modulation 746
Edge-aligned PWM without dead-time. 746
Center-aligned PWM without dead-time 746
Dead-time counter 747
27.9.2 Shadow registers and simultaneous updates 748
27.9.3 Fast Abort (ABORT). 748
27.9.4 Capture events. 748
27.9.5 External event counting (Counter mode) . . . 749
27.9.6 Three-phase DC mode 749
27.9.7 Three phase AC mode. 750
27.9.8 Interrupts 751

Chapter 28: LPC408x/407x Quadrature Encoder Interface (QEI)

28.1 How to read this chapter. 752
28.2 Basic configuration 752
28.3 Features 752
28.4 Introduction 753
28.5 Functional description 754
28.5.1 Input signals 754
28.5.1.1 Quadrature input signals 754
28.5.1.2 Digital input filtering 755
28.5.2 Position capture 755
28.5.3 Velocity capture 756
28.5.4 Velocity compare 756
28.6 Pin description. 757
28.7 Register description 758
28.7.1 Register summary 758
28.7.2 Control registers 759
28.7.2.1 QEI Control register 759
28.7.2.2 QEI Configuration register 759
28.7.2.3 QEI Status register 759
28.7.3 Position, index and timer registers 760
28.7.3.1 QEI Position register 760
28.7.3.2 QEI Maximum Position register 760
28.7.3.3 QEI Position Compare register 0 760
28.7.3.4 QEI Position Compare register 1 760
28.7.3.5 QEI Position Compare register 2 761
28.7.3.6 QEI Index Count register 761
28.7.3.7 QEI Index Compare register 0 761
28.7.3.8 QEI Velocity Timer Reload register 761
28.7.3.9 QEI Velocity Timer register 761
28.7.3.10 QEI Velocity register 762
28.7.3.11 QEI Velocity Capture register 762
28.7.3.12 QEI Velocity Compare register. 762
28.7.3.13 QEI Digital Filter on PHA 762
28.7.3.14 QEI Digital Filter on PHB 762
28.7.3.15 QEI Digital Filter on INX 763
28.7.3.16 QEI index acceptance Window 763
28.7.3.17 QEI Index Compare register 1 763
28.7.3.18 QEI Index Compare register 2 763
28.7.4 Interrupt registers. 764
28.7.4.1 QEI Interrupt Status register 764
28.7.4.2 QEI Interrupt Set register 765
28.7.4.3 QEI Interrupt Clear register 766
28.7.4.4 QEI Interrupt Enable register 767
28.7.4.5 QEI Interrupt Enable Set register. 768
28.7.4.6 QEI Interrupt Enable Clear register 769

Chapter 29: LPC408x/407x Real Time Clock (RTC)

29.1 Basic configuration 770
29.2 Features 770
29.3 Description 770
29.4 Architecture 771
29.5 Pin description. 772
29.6 Register description 772
29.6.1 RTC interrupts 774
29.6.2 Miscellaneous register group. 774
29.6.2.1 Interrupt Location Register. 774
29.6.2.2 Clock Control Register 774
29.6.2.3 Counter Increment Interrupt Register 775
29.6.2.4 Alarm Mask Register 775
29.6.2.5 RTC Auxiliary control register 776
29.6.2.6 RTC Auxiliary Enable register 776
29.6.3 Consolidated time registers 776

29.6.3.1 Consolidated Time Register 0 777
29.6.3.2 Consolidated Time Register 1 777
29.6.3.3 Consolidated Time Register 2 777
29.6.4 Time Counter Group 777
29.6.4.1 Leap year calculation 779
29.6.4.2 Calibration register 779
29.6.5 Calibration procedure 780
Backward calibration 780
Forward calibration 780
29.6.6 General purpose registers 781
29.6.6.1 General purpose registers 0 to 4 781
29.6.7 Alarm register group 781
29.7 RTC usage notes 783

Chapter 30: LPC408x/407x Event Monitor/Recorder

30.1 Basic configuration 784
30.2 Features 784
30.3 Applications 784
30.4 Description 785
30.5 Pin description 787
30.6 Register description 787
30.6.1 Event Monitor/Recorder Control Register . . 788
30.6.2 Event Monitor/Recorder Status Register . . . 790
30.6.3 Event Monitor/Recorder Counters Register . 791
30.6.4 Event Monitor/Recorder First Stamp Register 791
30.6.5 Event Monitor/Recorder Last Stamp Register 791

Chapter 31: LPC408x/407x Windowed Watchdog Timer (WWDT)

31.1 Features 793
31.2 Applications 793
31.3 Description 794
31.4 Register description 796
31.4.1 Watchdog Mode register 797
WDTOF 797
WDINT 797
WDPROTECT 797
31.4.2 Watchdog Timer Constant register 798
31.4.3 Watchdog Feed register 798
31.4.4 Watchdog Timer Value register 799
31.4.5 Watchdog Timer Warning Interrupt register 799
31.4.6 Watchdog Timer Window register 799
31.5 Watchdog timing examples 800

Chapter 32: LPC408x/407x Analog-to-Digital Converter (ADC)

32.1 Basic configuration 802
32.2 Features 802
32.3 Description 803
32.4 Pin description 803
32.5 Register description 804
32.5.1 A/D Control Register 805
32.5.2 A/D Global Data Register 806
32.5.3 A/D Interrupt Enable register 807
32.5.4 A/D Data Registers 808
32.5.5 A/D Status register 809
32.5.6 A/D Trim register 810
32.6 Operation 811
32.6.1 Hardware-triggered conversion 811
32.6.2 Interrupts 811
32.6.3 Accuracy vs. digital receiver 811
32.6.4 DMA control 811

Chapter 33: LPC408x/407x Digital-to-Analog Converter (DAC)

33.1 Basic configuration 812
33.2 Features 812
33.3 Architecture 813
33.4 Pin description 813
33.5 Register description 814
33.5.1 D/A Converter Register 814
33.5.2 D/A Converter Control register 814
33.5.3 D/A Converter Counter Value register 815
33.6 Operation 816
33.6.1 DMA counter 816
33.6.2 Double buffering 816

Chapter 34: LPC408x/407x Comparators

34.1 How to read this chapter 817
34.2 Basic configuration 817
34.3 Features 817
34.4 Architecture 818
34.5 Pin description 820
34.6 Register description 820
34.6.1 Comparator block control register 821
34.6.2 Comparator 0 control register 822
34.6.3 Comparator 1 control register 825
34.6.4 Comparator interrupt configurations 827

Chapter 35: LPC408x/407x General Purpose DMA controller

35.1 Basic configuration 828
35.2 Introduction 828

35.3 Features 829
35.4 Functional description 830
35.4.1 DMA controller functional description 830
35.4.1.1 AHB slave interface 830
35.4.1.2 Control logic and register bank 830
35.4.1.3 DMA request and response interface 830
35.4.1.4 Channel logic and channel register bank . . . 830
35.4.1.5 Interrupt request 831
35.4.1.6 AHB master interface 831
35.4.1.6.1 Bus and transfer widths 831
35.4.1.6.2 Endian behavior 831
35.4.1.6.3 Error conditions 833
35.4.1.7 Channel hardware 833
35.4.1.8 DMA request priority 833
35.4.1.9 Interrupt generation 834
35.4.2 DMA system connections 834
35.4.2.1 DMA request signals 834
35.4.2.2 DMA response signals 834
35.4.2.3 DMA request connections 834
35.5 Register description 836
35.5.1 DMA Interrupt Status register 838
35.5.2 DMA Interrupt Terminal Count Request Status register 838
35.5.3 DMA Interrupt Terminal Count Request Clear register 838
35.5.4 DMA Interrupt Error Status register 838
35.5.5 DMA Interrupt Error Clear register 839
35.5.6 DMA Raw Interrupt Terminal Count Status register 839
35.5.7 DMA Raw Error Interrupt Status register . . . 839
35.5.8 DMA Enabled Channel register 840
35.5.9 DMA Software Burst Request register 840
35.5.10 DMA Software Single Request register 841
35.5.11 DMA Software Last Burst Request register . 841
35.5.12 DMA Software Last Single Request register 841
35.5.13 DMA Configuration register 842
35.5.14 DMA Synchronization register 842
35.5.15 DMA Channel registers 843
35.5.16 DMA Channel Source Address registers . . 843
35.5.17 DMA Channel Destination Address registers 843
35.5.18 DMA Channel Linked List Item registers . . . 844
35.5.19 DMA channel control registers 844
35.5.19.1 Protection and access information 844
35.5.20 DMA Channel Configuration registers 847
35.5.20.1 Lock control 848
35.5.20.2 Transfer type 848
35.6 Using the DMA controller 849
35.6.1 Programming the DMA controller 849
35.6.1.1 Enabling the DMA controller 849
35.6.1.2 Disabling the DMA controller 849
35.6.1.3 Enabling a DMA channel 849
35.6.1.4 Disabling a DMA channel 849
Disabling a DMA channel and losing data in the FIFO 849
Disabling the DMA channel without losing data in the FIFO 849
35.6.1.5 Setting up a new DMA transfer 849
35.6.1.6 Halting a DMA channel 850
35.6.1.7 Programming a DMA channel 850
35.6.2 Flow control 850
35.6.2.1 Peripheral-to-memory or memory-to-peripheral DMA flow 851
35.6.2.2 Peripheral-to-peripheral DMA flow 851
35.6.2.3 Memory-to-memory DMA flow 852
35.6.3 Interrupt requests 852
35.6.3.1 Hardware interrupt sequence flow 853
35.6.4 Address generation 853
35.6.4.1 Word-aligned transfers across a boundary . 853
35.6.5 Scatter/gather 853
35.6.5.1 Linked list items 854
35.6.5.1.1 Programming the DMA controller for scatter/gather DMA 854
35.6.5.1.2 Example of scatter/gather DMA 854

Chapter 36: LPC408x/407x CRC engine

36.1 Introduction 857
36.2 Features 857
36.3 Description 858
36.4 Register description 859
36.4.1 CRC mode register 859
36.4.2 CRC seed register 859
36.4.3 CRC checksum register 860
36.4.4 CRC data register 860
36.5 Functional description 861
CRC-CCITT set-up 861
CRC-16 set-up 861
CRC-32 set-up 861

Chapter 37: LPC408x/407x EEPROM memory

37.1 Basic configuration 862
37.2 Description 862

continued >>

37.3 Features 862
37.4 EEPROM operation 863
37.4.1 EEPROM device description 863
37.4.2 EEPROM operations 863
37.4.2.1 Writing. 863
37.4.2.2 Erase/Programming 865
37.4.2.3 Reading 865
37.4.2.4 Exceptions 866
37.5 Register description 867
37.5.1 EEPROM control registers 868
37.5.1.1 EEPROM command register 868
37.5.1.2 EEPROM address register 868
37.5.1.3 EEPROM write data register. 869
37.5.1.4 EEPROM read data register 869
37.5.1.5 EEPROM wait state register 870
Example with cclk=120 MHz: 870
37.5.1.6 EEPROM clock divider register 870
37.5.1.7 EEPROM power down register 871
37.5.2 Interrupt registers. 872
37.5.2.1 Interrupt status register 872
37.5.2.2 Interrupt status clear register 872
37.5.2.3 Interrupt status set 873
37.5.2.4 Interrupt enable register. 873
37.5.2.5 Interrupt enable clear register 873
37.5.2.6 Interrupt enable set register 874

Chapter 38: LPC408x/407x Flash memory

38.1 Introduction 875
38.2 Features 875
38.3 Description 875
38.3.1 Memory map after any reset. 876
38.3.1.1 Criterion for Valid User Code 876
38.3.2 Communication protocol. 878
38.3.2.1 ISP command format 878
38.3.2.2 ISP response format. 878
38.3.2.3 ISP data format. 878
38.3.2.4 ISP flow control. 878
38.3.2.5 ISP command abort 878
38.3.2.6 Interrupts during IAP. 878
38.3.2.7 Addresses in IAP and ISP commands 878
38.3.2.8 RAM used by ISP command 879
38.3.2.9 RAM used by Boot process prior to entering user program 879
38.3.2.10 RAM used by IAP command handler 879
38.4 Boot process flowchart. 880
38.5 Sector numbers 881
38.6 Code Read Protection (CRP) 882
38.7 ISP commands 884
38.7.1 Unlock <Unlock code> 884
38.7.2 Set Baud Rate <Baud Rate> <stop bit> 885
38.7.3 Echo <setting> 885
38.7.4 Write to RAM <start address> <number of bytes>. 885
38.7.5 Read Memory <address> <no. of bytes> . . . 886
38.7.6 Prepare sector(s) for write operation <start sector number> <end sector number> 887
38.7.7 Copy RAM to Flash <flash address> <RAM address> <no of bytes> 887
38.7.8 Go <address> <mode>. 888
38.7.9 Erase sector(s) <start sector number> <end sector number> 888
38.7.10 Blank check sector(s) <sector number> <end sector number> 889
38.7.11 Read Part Identification number 889
38.7.12 Read Boot Code version number. 889
38.7.13 Read device serial number 890
38.7.14 Compare <address1> <address2> <no of bytes> 890
38.7.15 ISP Return Codes 891
38.8 IAP commands 892
38.8.1 Prepare sector(s) for write operation 895
38.8.2 Copy RAM to Flash 895
38.8.3 Erase Sector(s) 896
38.8.4 Blank check sector(s). 896
38.8.5 Read part identification number. 896
38.8.6 Read Boot Code version number. 897
38.8.7 Read device serial number 897
38.8.8 Compare <address1> <address2> <no of bytes> 897
38.8.9 Re-invoke ISP 898
38.8.10 IAP Status Codes. 898
38.9 JTAG flash programming interface 899
38.10 Flash signature generation 899
38.10.1 Register description for signature generation 899
38.10.1.1 Signature generation address and control registers 899
38.10.1.2 Signature generation result registers. 900
38.10.1.3 Flash Module Status register 901
38.10.1.4 Flash Module Status Clear register 901
38.10.2 Algorithm and procedure for signature generation 902
Signature generation 902
Content verification 902

Chapter 39: LPC408x/407x JTAG, Debug, and Trace

39.1 Features 903
39.2 Introduction 903
39.3 Description 903
39.4 Pin description. 904
39.5 Debug connections. 905
39.6 JTAG TAP Identification 906
39.7 Debug Notes 907
39.8 Debug memory re-mapping 907

39.8.1 Memory Mapping Control register 907

Chapter 40: ARM Cortex-M4 Appendix

40.1 ARM Cortex-M4 Details 908

40.1.1 Cortex-M4 implementation options 908

Chapter 41: Supplementary information

41.1 Abbreviations 910

41.2 References 911

41.3 Legal information 912

41.3.1 Definitions 912

41.3.2 Disclaimers 912

41.3.3 Trademarks 912

41.4 Tables 913

41.5 Figures 928

41.6 Contents 931

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12 March 2014

Document identifier: UM10562