UM10326

LPC32x0 and LPC32x0/01 User manual

Rev. 3 — 22 July 2011 **User manual**

Document information

Info	Content
Keywords	LPC3220, LPC3230, LPC3240, LPC3250, ARM9, LPC3220/01, LPC3230/01, LPC3240/01, LPC3250/01, 16/32-bit ARM microcontroller.
Abstract	User manual for LPC32x0.

Revision history

Rev	Date	Description
3	20110722	LPC32x0 user manual
		Modifications: System control chapter added.
2	20110504	LPC32x0 user manual
		Modifications: • Voltage domains updated in Table 671, Table 696, and Table 697. • LCDENAB signal updated in LCD timing diagrams Figure 35 and Figure 36. • Definition of Bus Keeper (BK) pins updated in Table 670. • GPI_6 updated to input with bus keeper function. • Power supply domain for pins SYSX_IN and SYSX_OUT updated in Table 671. • Description of DBGEN pin function updated in Section 4.6.1 and Table 671. • Explanation of simultaneous update mode for PWM Motor control Match and Limit register added in Section 30.7.7.1 “Match and Limit shadow write and operating registers”. • Registers SERIAL_ID0 to 3 added for unique serial ID number (Section 3.1.3). • Updated TAP ID and ETB ID (Table 710). • Editorial updates. • Pin functions for touch screen control controller updated: pin T14 has functions ADIN1/TS_XM and pin U15 has functions ADIN0/TS_YM (see Table 668, Table 669, Table 255, Table 229, Figure 37. • Pin selection for touch screen control controller in manual mode updated (TS_YPC values in Table 231). • Parts LPC3220/01, LPC3230/01, LPC3240/01, LPC3250/01 added.
1	20090218	Initial LPC32x0 user manual release.

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

NXP Semiconductor designed the LPC32x0 for embedded applications requiring high performance and low power consumption.

NXP achieved their performance objectives using an ARM926EJ-S CPU core with a Vector Floating Point co-processor and a large set of standard peripherals, including USB On-The-Go. Figure 1 shows a block diagram of the LPC32x0. The LPC32x0 operates at CPU frequencies up to 266 MHz. The basic ARM926EJ-S CPU Core implementation uses a Harvard architecture with a 5-stage pipeline. The ARM926EJ-S core also has an integral Memory Management Unit (MMU) to provide the virtual memory capabilities required to support the multi-programming demands of modern operating systems. The basic ARM926EJ-S core also includes a set of DSP instruction extensions including single cycle MAC operations and native Jazelle Java Byte-code execution in hardware. The NXP implementation has one 32 kB Instruction Cache and one 32 kB Data Cache.

For low power consumption, the LPC32x0 takes advantage of NXP Semiconductor's advanced technology development expertise to optimize Intrinsic Power, and software controlled architectural enhancements to optimize Power Management.

The LPC32x0 also includes 128 to 256 kB of on-chip static RAM, a NAND Flash interface, an Ethernet MAC, an LCD controller that supports STN and TFT panels, and an external bus interface that supports SDR and DDR SDRAM as well as static devices. In addition, the LPC32x0 includes a USB 2.0 Full Speed interface, seven UARTs, two I2C interfaces, two SPI/SSP ports, two I2S interfaces, two single output PWMs, a motor control PWM, four general purpose timers with capture inputs and compare outputs, a Secure Digital (SD) interface, and a 10-bit A/D converter with a touch screen sense option.

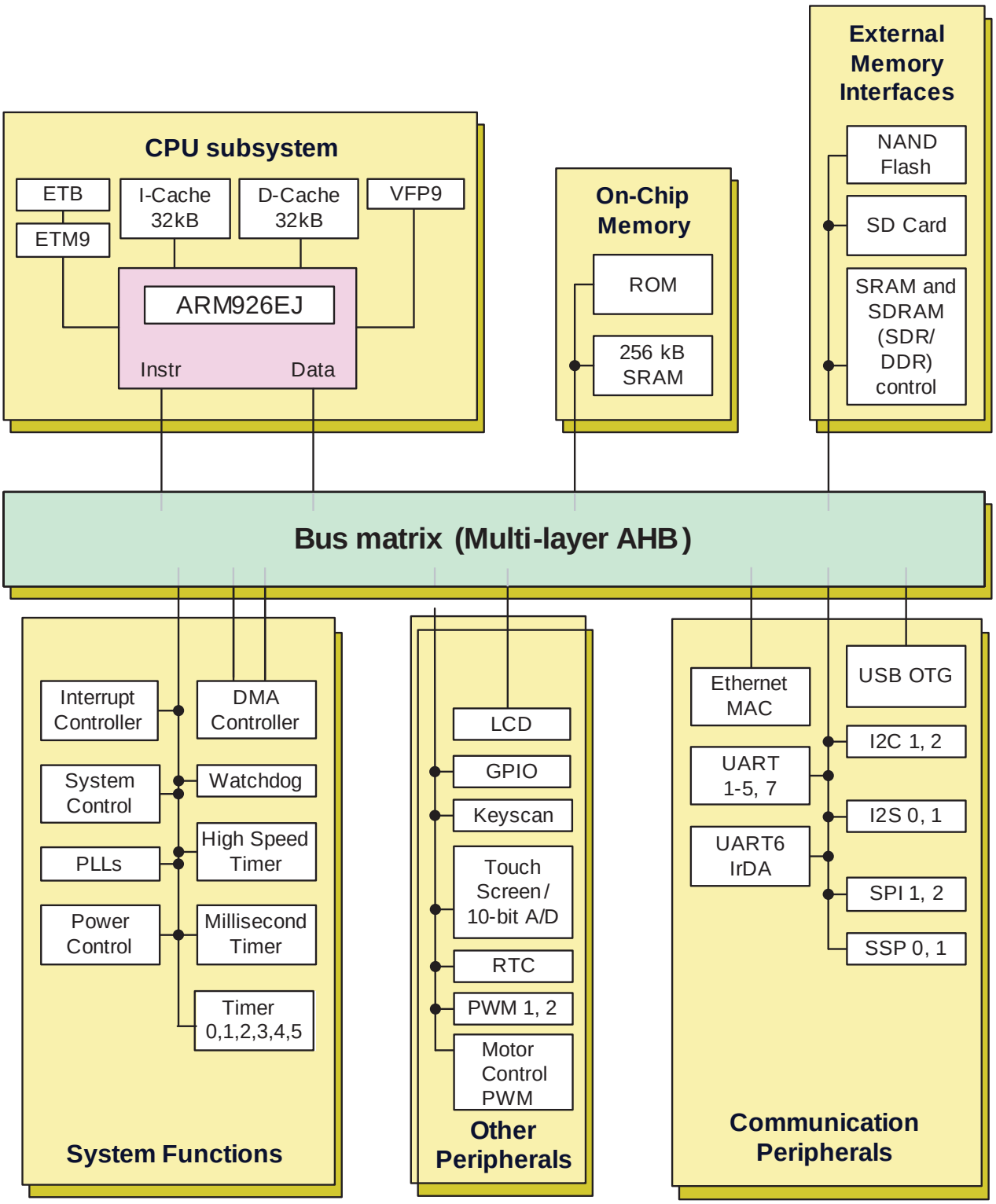


Fig 1. LPC32x0 diagram

1.2 Features

- ARM926EJS processor, running at CPU clock speeds up to 266 MHz
- A Vector Floating Point coprocessor.
- A 32 kB instruction cache and a 32 kB data cache.
- Up to 256 kB of internal SRAM.
- Selectable boot-up from various external devices: NAND Flash, SPI memory, USB, UART, or static memory.
- A Multi-layer AHB system that provides a separate bus for each AHB master, including both an instruction and data bus for the CPU, two data busses for the DMA controller, and another bus for the USB controller, one for the LCD and a final one for the Ethernet MAC. There are no arbitration delays in the system unless two masters attempt to access the same slave at the same time.
- An External memory controller for DDR and SDR SDRAM, as well as static devices.
  - The address bus provides up to 16 MB for each of the 4 static chip selects.
  - The controller provides two dynamic memory chip selects addressing up to 512 MB each.
- Two NAND Flash controllers. One for single level NAND Flash devices and the other for multi-level NAND Flash devices.
- An Interrupt Controller, supporting 73 interrupt sources.
- An eight channel General Purpose AHB DMA controller (GPDMA) that can be used with the SD card port, the 14-clock UARTs, I2S ports, and SPI interfaces, as well as memory-to-memory transfers.
- Serial Interfaces:
  - A 10/100 Ethernet MAC with dedicated DMA Controller.
  - A USB interface supporting either Device, Host (OHCI compliant), or On-The-Go (OTG) with an integral DMA controller and dedicated PLL to generate the required 48 MHz USB clock.
  - Four standard UARTs with fractional baud rate generation and 64 byte FIFOs. One of the standard UART’s supports IrDA.
  - Three additional 14-clock UARTs intended for on-board communications that support baudrates up to 921,600 bps when using a 13 MHz main oscillator.All 14-clock UARTs provide 64-byte FIFOs.
  - Two SPI controllers.
  - Two SSP controllers.
  - Two I2C-bus Interfaces with standard open drain pins. The I2C-bus Interfaces support single master, slave and multi-master I2C configurations.
  - Two I2S interfaces, each with separate input and output channels. Each channel can be operated independently on 3 pins, or both input and output with one I2S interface can be done on only 4 pins.
- Additional Peripherals:
  - LCD controller supporting both STN and TFT panels, with dedicated DMA controller. Programmable display resolution up to 1024x768.
  - Secure Digital (SD) memory card interface.

- General purpose input, output, and I/O pins. Includes 12 general purpose input pins, 24 general purpose output pins, and 51 general purpose I/O pins.
- A touch screen controller, which can alternatively operate as a 10 bit, 400kHz A/D Converter with input multiplexing from 3 pins.
- Real Time Clock (RTC) with separate power pin. This RTC has a dedicated 32 kHz oscillator. NXP implemented the RTC in an independent on-chip power domain so it can remain active while the rest of the chip is not powered. The RTC also Includes a 32 byte scratch pad memory.
- A 32-bit general purpose high speed timer with a 16-bit pre-scaler. This timer includes one external capture input pin and a capture connection to the RTC clock. Interrupts may be generated using 3 match registers.
- Six enhanced Timer/Counters which are identical except for the peripheral base address. A minimum of two capture inputs and two match outputs are pinned out for four timers. Timer 1 brings out a third match output, while Timers 2 and 3 bring out all four match outputs, Timer 4 has one capture register and Timer 5 has no pin connections.
- A 32-bit Millisecond timer driven from the RTC clock. This timer can generate interrupts using 2 match registers.
- A Watchdog Timer. The watchdog timer is clocked by PERIPH_CLK.
- Two single output PWM blocks.
- One PWM designed for Motor Control.
- Keyboard scanner function allows automatic scanning of up to an 8x8 key matrix.
- Up to 18 external interrupts.

- Standard ARM Test/Debug interface for compatibility with existing tools.
- Emulation Trace Buffer with 2K x 24 bit RAM allows trace via JTAG.
- Stop mode saves power, while allowing many peripheral functions to restart CPU activity.
- On-chip crystal oscillator.
- An on-chip PLL allows CPU operation up to the maximum CPU rate without the requirement for a high frequency crystal. Another PLL allows operation from the 32 kHz RTC clock rather than the external crystal.
- Boundary Scan for simplified board testing.
- User-accessible unique serial ID number for each chip.
- 296 pin TFBGA package.

1.3 Applications

- Consumer
- Medical
- Network Control
- Industrial
- Communications

1.4 Ordering Information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC3220FET296[1]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3230FET296[1]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3240FET296[1]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3250FET296[1]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3220FET296/01[2]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3230FET296/01[2]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3240FET296/01[2]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1
LPC3250FET296/01[2]	TFBGA296	plastic thin fine-pitch ball grid array package; 296 balls; body 15x15x0.7 mm	SOT1048-1

[1] Available in Revision “-” and “A”. F = –40 °C to +85 °C temperature range. Note that Revision “A” parts with and without the /01 suffix are identical. For example, LPC3220FET296 Revision “A” is identical to LPC3220FET296/01 Revision “A”.

[2] Available starting with Revision “A”.

1.4.1 Ordering options

Table 2. Part options

Type number	SRAM (kB)	10/100 Ethernet	LCD Controller	Temperature range (°C)	Package
LPC3220FET296	128	0	0	-40 to +85	TFBGA296
LPC3230FET296	256	0	1	-40 to +85	TFBGA296
LPC3240FET296	256	1	0	-40 to +85	TFBGA296
LPC3250FET296	256	1	1	-40 to +85	TFBGA296
LPC3220FET296/01	128	0	0	-40 to +85	TFBGA296
LPC3230FET296/01	256	0	1	-40 to +85	TFBGA296
LPC3240FET296/01	256	1	0	-40 to +85	TFBGA296
LPC3250FET296/01	256	1	1	-40 to +85	TFBGA296

1.5 Block diagram

The following block diagram shows the bus connections between the CPU, peripherals, and external memory.

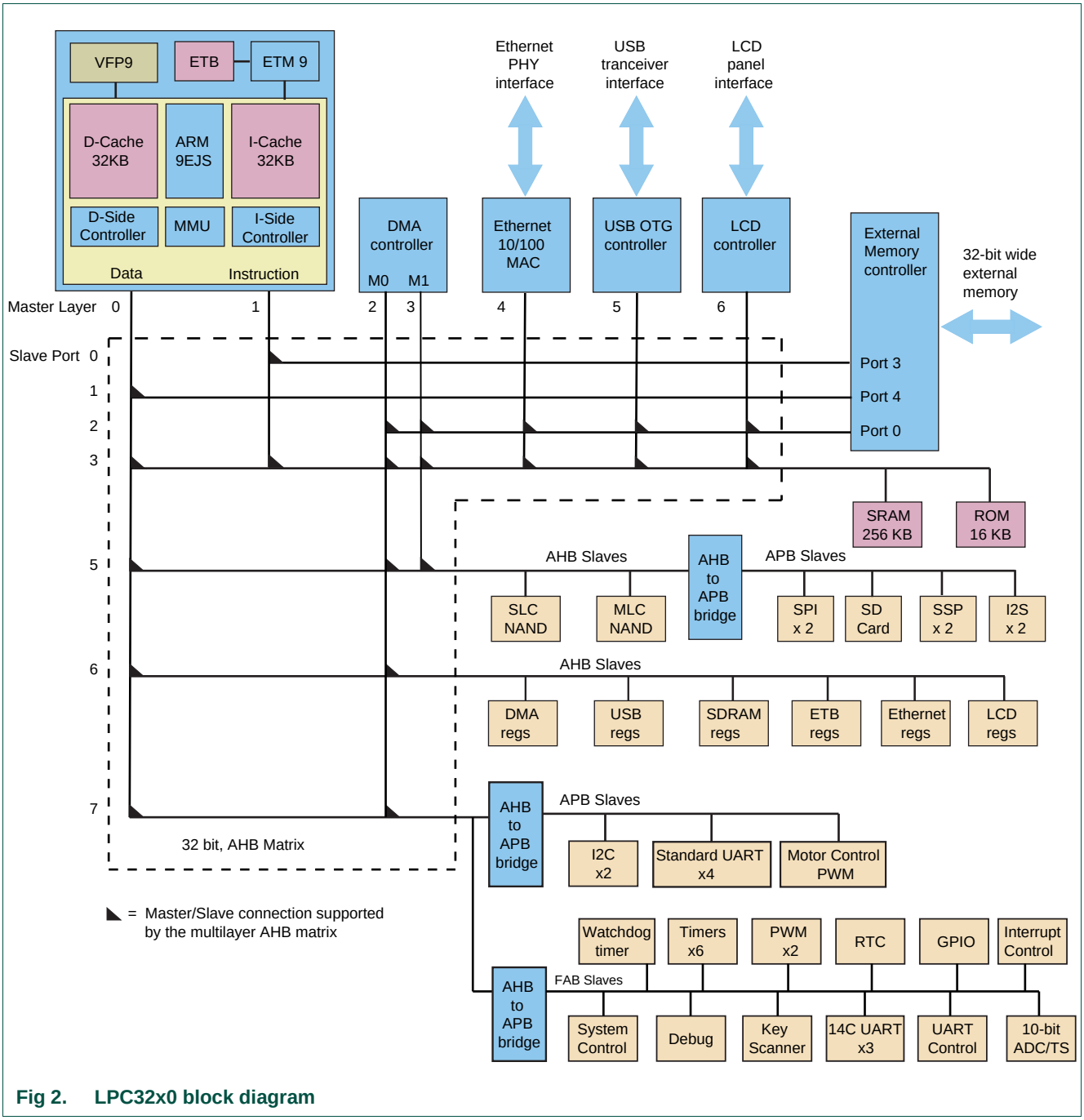


Fig 2. LPC32x0 block diagram

1.6 CPU Subsystem

1.6.1 CPU

NXP implemented the LPC32x0 using an ARM926EJ-S CPU core that has a Harvard architecture and a 5-stage pipeline. To the ARM926EJ-S CPU core, NXP added a 32 kB Instruction Cache, a 32 kB Data Cache and a Vector Floating Point coprocessor.

The ARM926EJ-S core also has an integral Memory Management Unit (MMU) to provide the virtual memory capabilities required to support the multi-programming demands of modern operating systems. The ARM926EJ-S core V5TE instruction set includes DSP instruction extensions, and can execute native Jazelle Java Byte-code in hardware. The LPC32x0 operates at CPU frequencies up to 266 MHz.

1.6.2 Vector Floating Point (VFP) coprocessor

The LPC32x0 has a VFP co-processor providing full support for single-precision and double-precision add, subtract, multiply, divide, and multiply-accumulate operations at CPU clock speeds. It is compliant with the IEEE 754 standard for binary Floating-Point Arithmetic. This hardware floating point capability makes the micro controller suitable for advanced Motor control and DSP applications. The VFP has 3 separate pipelines for Floating-point MAC operations, divide or square root operations, and Load/Store operations. These pipelines operate in parallel and can complete execution out of order. All single-precision instructions execute in one cycle, except the divide and square root instructions. All double-precision multiply and multiply-accumulate instructions take two cycles. The VFP also provides format conversions between floating-point and integer word formats.

1.6.3 Emulation and debugging

The LPC32x0 supports emulation and debugging via a dedicated JTAG serial port. An Embedded Trace Buffer allows tracing program execution. The dedicated JTAG port allows debugging of all chip features without impact to any pins that may be used in the application.

EmbeddedICE

Standard ARM EmbeddedICE logic provides on-chip debug support. The debugging of the target system requires a host computer running the debugger software and an EmbeddedICE protocol converter. The EmbeddedICE protocol converter converts the Remote Debug Protocol commands to the JTAG data needed to access the ARM core.

The ARM core has a Debug Communication Channel function built-in. The debug communication channel allows a program running on the target to communicate with the host debugger or another separate host without stopping the program flow or entering the debug state.

Embedded Trace Buffer

The Embedded Trace Module (ETM) is connected directly to the ARM core. It compresses the trace information and exports it through a narrow trace port. An internal Embedded Trace Buffer of 2 k $\times$ 24 bits captures the trace information under software debugger control. Data from the Embedded Trace Buffer is recovered by the debug software through the JTAG port.

The trace contains information about when the ARM core switches between states. Instruction trace (or PC trace) shows the flow of execution of the processor and provides a list of all the instructions that were executed. Instruction trace is significantly compressed by only broadcasting branch addresses as well as a set of status signals that indicate the pipeline status on a cycle by cycle basis. For data accesses either data or address or both can be traced.

1.7 AHB Bus Architecture

The LPC32x0 has a multi-layer AHB matrix for inter-block communication. AHB is an ARM defined high-speed bus, which is part of the ARM bus architecture. AHB is a high-bandwidth low-latency bus that supports multi-master arbitration and a bus grant/request mechanism. For systems that have only one (CPU), or two (CPU and DMA) bus masters a simple AHB works well. However, if a system requires multiple bus masters and the CPU needs access to external memory, a single AHB bus can cause a bottleneck. In addition to the AHB matrix the LPC32x0 has two peripheral busses the APB and the FAB. A short description of the busses are described in the following sections.

1.7.1 AHB Matrix

To increase performance, the LPC32x0 uses an expanded AHB architecture known as Multi-layer AHB. A Multi-layer AHB replaces the request/grant and arbitration mechanism used in a simple AHB with an interconnect matrix that moves arbitration out toward the slave devices. Thus, if a CPU and a DMA controller want access to the same memory, the interconnect matrix arbitrates between the two when granting access to the memory. This advanced architecture allows simultaneous access by bus masters to different resources with an increase in arbitration complexity. In this architectural implementation, removing guaranteed central arbitration and allowing more than one bus master to be active at the same time provides better overall micro controller performance.

In the LPC32x0, the Multi-Layer AHB system has a separate bus for each of seven AHB Masters:

- CPU Data bus
- CPU Instruction bus
- General purpose DMA Master 0
- General purpose DMA Master 1
- Ethernet Controller
- USB Controller
- LCD Controller

There are no arbitration delays unless two masters attempt to access the same slave at the same time.

1.7.2 APB bus

Many peripheral functions are accessed by on-chip APB busses that are attached to the higher speed AHB bus. The APB bus performs reads and writes to peripheral registers in three peripheral clocks.

1.7.3 FAB bus

Some peripherals are placed on a special bus called FAB that allows faster CPU access to those peripheral functions. A write access to FAB peripherals takes a single AHB clock and a read access to FAB peripherals takes two AHB clocks.

1.8 Internal Memory

On-Chip SRAM (IRAM)

On-chip SRAM may be used for code and/or data storage. The SRAM may be accessed as 8, 16, or 32 bit memory. The LPC32x0 provides 256 kB of internal SRAM.

On-Chip ROM

An on-chip 16 kB ROM contains the necessary code to load code from one of four sources, UART5, SSP0 (in SPI mode), EMC (CS0 SRAM) memory, or NAND FLASH, and copy it to IRAM. The code loaded to IRAM will typically be FLASH programming software. The boot process is handled by an on-chip bootloader that looks at the state of a pin (GPI_1) to see if it should attempt to download a program over a serial link (UART5) or one of three alternate sources, SSP0, EMC (CS0) memory, and NAND Flash and then branch to its entry point and execute. If the GPI_1 pin is low it will boot from UART5 (SERVICE_N boot), if it is set high, it will test in sequence each of the remaining possible sources for the presence of download code. The bootloader will first test for boot code in the SSP0, if this fails it will test for boot code in EMC memory and if this fails it will test the NAND Flash for boot code.

The bootloader reserves a small portion (8kB from 0x0000 E000 to 0x0000 FFFF) on the Internal SRAM during the boot process. This space is no longer used once the boot loader has passed control to the downloaded code.

1.9 External Memory Interfaces

The LPC32x0 includes three external memory interfaces.

- an interface supporting two NAND Flash controllers, an Multi-level and a Single-level.
- an external memory controller (EMC) interface for SDRAM, DDR SDRAM, and Static Memory devices.
- a Secure Digital Memory Controller (SDCARD) interface.

1.9.1 NAND flash controller Interface

The LPC32x0 includes two NAND flash controllers, one for multi-level NAND flash devices and one for single-level NAND flash devices. The two NAND flash controllers use the same pins to interface to external NAND flash devices, so only one interface is active at a time.

Multi-Level Cell (MLC) NAND flash controller

The MLC NAND flash controller interfaces to either multi-level or single-level NAND flash devices. An external NAND flash device is used to allow the bootloader to automatically load a portion of the application code into internal SRAM for execution following reset.

The MLC NAND flash controller supports up to 2 Gbit devices with small (528 byte) or large (2114 byte) pages. Programmable NAND timing parameters allow support for a variety of NAND flash devices. A built-in Reed-Solomon encoder/decoder provides error detection and correction capability. A 528 byte data buffer reduces the need for CPU supervision during loading. The MLC NAND flash controller also provides DMA support.

Single-Level Cell (SLC) NAND flash controller

The SLC NAND flash controller interfaces to single-level NAND flash devices up to 2 Gbit in size. DMA page transfers are supported, including a 20 byte DMA read and write FIFO. Hardware support for ECC (Error Checking and Correction) is included for the main data area. Software can correct a single bit error.

1.9.2 External Memory Controller Interface

The LPC32x0 includes an External Memory Controller (EMC) that supports SDRAM, DDR SDRAM, and Static memory devices. The EMC provides an interface between the system bus and external (off-chip) memory devices.

The controller supports SDR SDRAM devices of 64/128/256/512 Mbit in size, as well as 16-bit wide data bus DDR SDRAM devices of 64/128/256/512 Mbit in size. Two dynamic memory chip selects are supplied, supporting two groups of SDRAM

- EMC_DYCS0_N in the address range 0x8000 0000 to 0x9FFF FFFF
- EMC_DYCS1_N in the address range 0xA000 0000 to 0xBFFF FFFF

The memory controller also supports 8-bit, 16-bit, and 32-bit wide asynchronous static memory devices, including RAM, ROM, and Flash, with or without asynchronous page mode. Four static memory chip selects are supplied for SRAM devices.

- EMC_CS0_N in the address range 0xE000 0000 to 0xE0FF FFFF
- EMC_CS1_N in the address range 0xE100 0000 to 0xE1FF FFFF
- EMC_CS2_N in the address range 0xE200 0000 to 0xE2FF FFFF
- EMC_CS3_N in the address range 0xE300 0000 to 0xE3FF FFFF

The SDRAM controller uses three data ports to allow simultaneous requests from multiple on-chip AHB bus masters and has the following features.

- Dynamic memory interface supports SDRAM, DDR-SDRAM, and low-power variants. It also supports Micron SyncFlash types of memory
- Read and write buffers to reduce latency and improve performance
- Static memory features include
  - asynchronous page mode read
  - programmable wait states
  - bus turnaround cycles
  - output enable and write enable delays
  - extended wait
- Power-saving modes dynamically control EMC_CKE and EMC_CLK
- Dynamic memory self-refresh mode supported by software

- Controller supports 2K, 4K, and 8K row address synchronous memory parts. That is, typical 512Mb, 256Mb, 128Mb, and 16Mb parts, with 8, 16, or 32 DQ (data) bits per device
- Two reset domains enable dynamic memory contents to be preserved over a soft reset
- This controller does NOT Support Synchronous static memory devices (burst mode devices)

1.9.3 SD card controller

The SD interface allows access to external SD memory cards. The SD card interface conforms to the SD Memory Card Specification Version 1.01.

Features

- Conformance to the SD Memory Card Specification Version 1.01.
- DMA is supported through the system DMA controller.
- Provides all functions specific to the SD memory card. These include the clock generation unit, power management control, command and data transfer.

1.10 AHB Master Peripherals

The LPC32x0 implements four AHB master peripherals, which include a General Purpose Direct Memory Access (GPDMA) controller, a 10/100 Ethernet Media Access Controller (MAC), a Universal Serial Bus (USB) controller, and an LCD Controller. Each of these four peripherals contain an integral DMA controller optimized to support the demands of the peripheral.

1.10.1 General purpose DMA controller (GPDMA)

The GPDMA controller allows 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 receives. The source and destination areas can each be either a memory region or a peripheral, and can be accessed through the same AHB master or one area by each master. The DMA controller supports the following peripheral device transfers.

- Secure Digital (SD) Memory interface
- 14-clock UART’s
- I2S0 and I2S1 Ports
- SPI1 and SPI2 Interfaces
- SSP0 and SSP1 interfaces
- Memory

The DMA controls eight DMA channels with hardware prioritization. The DMA controller interfaces to the system via two AHB bus masters, each with a full 32-bit data bus width. DMA operations may be set up for 8-bit, 16-bit, and 32-bit data widths, and can be either big-endian or little-endian. Incrementing or non-incrementing addressing for source and

destination are supported, as well as programmable DMA burst size. 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.

1.10.2 Ethernet MAC

The Ethernet block contains a full featured 10 Mbit/s or 100 Mbit/s Ethernet MAC 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 and the CPU share a dedicated AHB subsystem that is used to access the Ethernet SRAM for Ethernet data, control, and status information. All other AHB traffic in the LPC32x0 takes place on a different AHB subsystem, effectively separating Ethernet activity from the rest of the system. The Ethernet DMA can also access off-chip memory via the EMC, as well as the SRAM located on another AHB. However, using memory other than the Ethernet SRAM, especially off-chip memory, will slow Ethernet access to memory and increase the loading of its AHB. The Ethernet block interfaces between an off-chip Ethernet PHY using the Media Independent Interface (MII) or Reduced MII (RMII) protocol and the on-chip Media Independent Interface Management (MIIM) serial bus.

Features

- Ethernet standards support:
  - Supports 10 Mbit/s or 100 Mbit/s 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
  - Virtual Local Area Network (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 pre-fetching
- Enhanced Ethernet features:
  - Receive filtering
  - Multicast and broadcast frame support for both transmit and receive
  - Optional automatic Frame Check Sequence (FCS) insertion with Circular Redundancy Check (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 back-off 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 MII or RMII interface
  - PHY register access is available via the MIIM interface

1.10.3 USB interface

The LPC32x0 supports USB in either DEVICE, HOST, or OTG configuration.

1.10.3.1 USB DEVICE controller

The USB device controller enables 12 Mbit/s data exchange with a USB host controller. It consists of register interface, serial interface engine, endpoint buffer memory and DMA controller. The serial interface engine decodes the USB data stream and writes data to the appropriate end point buffer memory. The status of a completed USB transfer or error condition is indicated via status registers. An interrupt is also generated if enabled. The DMA controller when enabled transfers data between the endpoint buffer and the USB RAM.

Features

- Fully compliant with USB 2.0 full-speed specification.
- 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.
- RAM message buffer size based on endpoint realization and maximum packet size.
- Supports bus-powered capability with low suspend current.
- Supports DMA transfer on all non-control endpoints.
- One duplex DMA channel serves all endpoints.
- Allows dynamic switching between CPU controlled and DMA modes.
- Double buffer implementation for bulk and isochronous endpoints.

1.10.3.2 USB 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.

Features

- OHCI compliant.
- OHCI specifies the operation and interface of the USB host controller and SW driver.
- The host controller has four USB states visible to the SW driver:
  - USBOperational: Process lists and generate SOF tokens.
  - USBReset: Forces reset signaling on the bus, SOF disabled.

- USBSuspend: Monitor USB for wake-up activity.
- USBResume: Forces resume signaling on the bus.

- HCCA register points to interrupt and isochronous descriptors list.
- ControlHeadED and BulkHeadED registers point to control and bulk descriptors list.

1.10.3.3 USB OTG Controller

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.

Features

- Fully compliant with On-The-Go supplement to the USB Specification 2.0 Revision 1.0.
- Supports Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) for dual-role devices under software control. HNP is partially implemented in hardware.
- Provides programmable timers required for HNP and SRP.
- Supports slave mode operation through AHB slave interface.
- Supports the OTG ATX from NXP (ISP 1301) or any external CEA-2011OTG specification compliant ATX.

1.10.4 LCD Controller

The LCD controller provides all of the necessary control signals to interface directly to a variety of color and monochrome LCD panels. Both STN (single and dual panel) and TFT panels can be operated. The display resolution is selectable and can be up to 1024 × 768 pixels. Several color modes are provided, up to a 24-bit true-color non-palettized mode. An on-chip 512-byte color palette allows reducing bus utilization (i.e. memory size of the displayed data) while still supporting a large number of colors.

The LCD interface includes its own DMA controller to allow it to operate independently of the CPU and other system functions. A built-in FIFO acts as a buffer for display data, providing flexibility for system timing. Hardware cursor support can further reduce the amount of CPU time needed to operate the display.

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-bit 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: 320 × 200, 320 × 240, 640 × 200, 640 × 240, 640 × 480, 800 × 600, and 1024 × 768
- Hardware cursor support for single-panel displays
- 15 gray-level monochrome, 3375 color STN, and 32 K 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 128 × 32-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

1.11 System Functions

To enhance the performance of the LPC32x0 incorporates the following System Functions, an Interrupt Controller (INTC), a Watchdog timer, a Millisecond Timer, and several Power Control Features. These functions are described in the following sections

1.11.1 Interrupt controller

The interrupt controller is comprised of three basic interrupt controller blocks, supporting a total of 73 interrupt sources. Each interrupt source can be individually enabled/disabled and configured for high or low level triggering, or rising or falling edge triggering. Each interrupt may also be steered to either the FIQ or IRQ input of the ARM9. Raw interrupt status and masked interrupt status registers allow versatile condition evaluation. In addition to peripheral functions, each of the six general purpose input/output pins and 12 general purpose input pins are connected directly to the interrupt controller.

1.11.2 Watchdog timer

The watchdog timer block is clocked by the main peripheral clock, which clocks a 32-bit counter. A match register is compared to the Timer. When configured for watchdog functionality, a match drives the match output low. The match output is gated with an enable signal that gives the opportunity to generate two type of reset signal: one that only resets chip internally, and another that goes through a programmable pulse generator before it goes to the external pin RESOUT_N and to the internal chip reset.

Features

- Programmable 32-bit timer.
- Internally resets the device if not periodically reloaded.
- Flag to indicate that a watchdog reset has occurred.
- Programmable watchdog pulse output on RESOUT_N pin.
- Can be used as a standard timer if watchdog is not used.
- Pause control to stop counting when core is in debug state.

1.11.3 Millisecond timer

The millisecond timer is clocked by 32 kHz RTC clock, so a pre-scaler is not needed to obtain a lower count rate.

The millisecond timer includes three match registers that are compared to the Timer/Counter value. A match can generate an interrupt and the cause the Timer/Counter either continue to run, stop, or be reset.

Features

- 32-bit Timer/Counter, running from the 32 kHz RTC clock.
- Counter or Timer operation.
- Three 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.
- Pause control to stop counting when core is in debug state.

1.11.4 Clocking and Power Control Features

Clocking

Clocking in the LPC32x0 is designed to be versatile, so that system and peripheral requirements may be met, while allowing optimization of power consumption. Clocks to most functions may be turned off if not needed and some peripherals do this automatically.
The LPC32x0 supports three operational modes, two of which are specifically designed to reduce power consumption. The modes are: RUN mode, Direct RUN mode, and STOP mode.These three operational modes give control over processing speed and power consumption. In addition, clock rates to different functional blocks may be changed by switching clock sources, reconfiguring PLL values, or altering clock divider configurations. This allows a trade-off of power versus processing speed based on application requirements.

Crystal Oscillator

The main oscillator is the basis for the clocks most chip functions use by default. Optionally, many functions can be clocked instead by the output of a PLL (with a fixed 397x rate multiplication) which runs from the RTC oscillator. In this mode, the main oscillator may be turned off unless the USB interface is enabled. If a SYSCLK frequency other than 13 MHz is required in the application, or if the USB block is not used, the main oscillator may be used with a frequency of between 1 MHz and 20 MHz.

PLLs

The LPC32x0 includes three PLLs: The 397x PLL allows boosting the RTC frequency to 13.008896 MHz for use as the primary system clock. The USB PLL provides the 48 MHz clock required by the USB block; and the HCLK PLL provides the basis for the CPU clock, the AHB bus clock, and the main peripheral clock.

Power Control Modes

The LPC32x0 supports three operational modes, two of which are specifically designed to reduce power consumption. The modes are: Run mode, Direct Run mode, and Stop mode.

Run mode is the normal operating mode for applications that require the CPU, AHB bus, or any peripheral function other than the USB block to run faster than the main oscillator frequency. In Run mode, the CPU can run at up to 266 MHz and the AHB bus can run at up to 133 MHz.

Direct Run mode allows reducing the CPU and AHB bus rates in order to save power. Direct Run mode can also be the normal operating mode for applications that do not require the CPU, AHB bus, or any peripheral function other than the USB block to run faster than the main oscillator frequency. Direct Run mode is the default mode following chip reset.

Stop mode causes all CPU and AHB operation to cease, and stops clocks to peripherals other than the USB block.

Reset

Reset is accomplished by an active low signal on the RESET_N input pin. A reset pulse with a minimum width of 10 main oscillator clocks after the oscillator is stable is required to guarantee a valid chip reset. At power-up, 10 milliseconds should be allowed for the oscillator to start up and stabilize after V_{DD} reaches operational voltage. An internal reset with a minimum duration of 10 clock pulses will also be applied if the watchdog timer generates an internal device reset.

The RESET_N pin is located in the RTC power domain. This means that the RTC power must be present for an external reset to have any effect. The RTC power domain nominally runs from 1.2 V, but the RESET_N pin can be driven as high as 1.95 V.

The RESET_N pin is also used to isolate the RTC power domains from other on chip power domains.To minimize power consumption in the RTC power domain while power is removed from other power domains, the RESET_N pin must be held low.

1.12 Serial communication peripherals

In addition to the Ethernet MAC and USB interfaces there many more serial Communication peripheral interfaces on the LPC32x0. Here is a list of the available serial communication interfaces.

- seven UARTs; four Standard UARTs and three 14-clock UARTs
- two SPI Serial I/O Controllers
- two SSP Serial I/O Controllers
- two I2C Serial I/O Controllers
- two I2S Audio Controllers

A short functional description of each of these peripherals is provided in the following sections.

1.12.1 UARTs

The LPC32x0 contains seven UARTs. Four are standard UARTs, and three are 14-clock UARTs.

Standard UARTs — The four standard UARTs are downwards compatible with the INS16Cx50. These UARTs support rates up to 460800 bit/s from a 13 MHz peripheral clock.

Features

- Each standard UART has 64 byte Receive and Transmit FIFOs.
- Receiver FIFO trigger points at 16 B, 32 B, 48 B, and 60 B.
- Transmitter FIFO trigger points at 0 B, 4 B, 8 B, and 16 B.
- Register locations conform to 16C550 industry standard.
- Each standard UART has a fractional rate pre-divider and an internal baud rate generator.
- The standard UARTs support three clocking modes: on, off, and auto-clock. The auto-clock mode shuts off the clock to the UART when it is idle.
- UART 6 includes an IrDA mode to support infrared communication.
- The standard UARTs are designed to support data rates of (2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800) bit/s.
- Each UART includes an internal loopback mode.

14-clock UARTs — The three 14-clock UARTs are designed to support rates up to 921600 bit/s from a 13 MHz peripheral clock, for on-board communication in low noise conditions. This is accomplished by changing the oversampling from 16× to 14×, and altering the rate generation logic.

Features

- Each 14-clock UART has 64 byte Receive and Transmit FIFOs.
- Receiver FIFO trigger points at 1 B, 4 B, 8 B, 16 B, 32 B, and 48 B.
- Transmitter FIFO trigger points at 0 B, 4 B, and 8 B.
- Each 14-clock UART has an internal baud rate generator.
- The 14-clock UARTs are designed to support data rates of (2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600) bit/s.
- Each UART includes an internal loopback mode.

1.12.2 SPI serial I/O controller

The LPC32x0 has two Serial Peripheral Interfaces (SPI). The SPI is a 3-wire serial interface that is able to interface with a large range of serial peripheral or memory devices (SPI mode 0 to 3 compatible slave devices).

Only a single master and a single slave can communicate on the interface during a given data transfer. During a data transfer the master always sends a byte of data to the slave, and the slave always sends a byte of data to the master. The SPI implementation on the LPC32x0 does not support operation as a slave.

Features

- Supports slaves compatible with SPI modes 0 to 3.
- Half duplex synchronous transfers.
- DMA support for data transmit and receive.
- 1-bit to 16-bit word length.
- Choice of LSB or MSB first data transmission.
- 64 × 16-bit input or output FIFO.
- Bit rates up to 52 Mbit/s.
- Busy input function.
- DMA time out interrupt to allow detection of end of reception when using DMA.
- Timed interrupt to facilitate emptying the FIFO at the end of a transmission.
- SPI clock and data pins may be used as general purpose pins if the SPI is not used.

1.12.3 SSP serial I/O Controller

The LPC32x0 contains two SSP controllers. The SSP controller is 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. The SSP supports full duplex transfers, with frames of 4 bits to 16 bits of data flowing from the master to the slave and from the slave to the master. In practice, often only one of these data flows carries meaningful data.

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-bit to 16-bit frame
- Maximum SPI bus data bit rate of one half (Master mode) and one twelfth (Slave mode) of the input clock rate
- DMA transfers supported by GPDMA

1.12.4 I^2C-bus serial I/O controller

There are two I^2C-bus interfaces in the LPC32x0. The blocks for the I^2C-bus are a master only implementation supporting the 400 kHz I^2C-bus mode and lower rates, with 7-bit slave addressing. Each has a four word FIFO for both transmit and receive. An interrupt signal is available from each block.

Features

- The two I^2C-bus blocks are standard I^2C-bus compliant interfaces that may be used in Single Master mode only.
- Programmable clock to allow adjustment of I^2C-bus transfer rates.
- Bidirectional data transfer.

- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.

1.12.5 I²S Audio Controller

The I²S-bus provides a standard communication interface for digital audio applications The I²S-bus specification defines a 3-wire serial bus using one data line, one clock line, and one word select signal. Each I²S connection can act as a master or a slave. The master connection determines the frequency of the clock line and all other slaves are driven by this clock source. The two I²S interfaces on the LPC32x0 provides a separate transmit and receive channel, providing a total of two transmit channels and two receive channels. Each I²S channel supports monaural or stereo formatted data.

Features

- The interface has separate input/output channels each of which can operate in master or slave mode
- Capable of handling 8-bit, 16-bit, and 32-bit word sizes
- Mono and stereo audio data supported
- Supports standard pcm sampling frequencies (16, 22.05, 32, 44.1,48, 96) kHz
- Configurable word select period in master mode (separately for I²S input and output)
- Two 8 word FIFO data buffers are provided, one for transmit and one for receive
- Generates interrupt requests when buffer levels cross a programmable boundary
- Two DMA requests, controlled by programmable buffer levels. These are connected to the GPDMA block
- Controls include reset, stop and mute options separately for I²S input and I²S output

1.13 General purpose I/O

Some device pins that are not dedicated to a specific peripheral function have been designed to be general purpose inputs, outputs, or I/Os. Also, some pins may be configured either as a specific peripheral function or a general purpose input, output, or I/O. A total of 87 pins can potentially be used as general purpose input/outputs, general purpose outputs, and general purpose inputs.

GPIO pins may be dynamically configured as inputs or outputs. Separate registers allow setting or clearing any number of GPIO and GPO outputs controlled by that register simultaneously. The value of the output register for standard GPIOs and GPO pins may be read back, as well as the current actual state of the port pins.

There are 12 GPI, 24 GPO, and 38 GPIO pins. When the SDRAM bus is configured for 16 data bits, 13 of the remaining SDRAM data pins may be used as GPIOs.

Features

- Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.

- A single register selects direction for pins that support both input and output modes.
- Direction control of individual bits.
- For input/output pins, both the programmed output state and the actual pin state can be read.
- There are a total of 12 general purpose inputs, 24 general purpose outputs, and 38 general purpose input/outputs.
- Additionally, 13 SDRAM data lines may be used as GPIOs if a 16-bit SDRAM interface is used (rather than a 32-bit interface).

1.14 Other Peripherals

In addition to the Serial Communication peripherals and GPIO there are many general purpose peripherals available in the LPC32x0. Here is a list of the remaining peripherals on the LPC32x0.

- Keyboard Scanner
- 10-Bit Analog-to-Digital-Converter
- Real-Time Clock
- Millisecond Timer
- A High-speed Timer
- 4 General Purpose 32-Bit Timer/External Event Counters
- 2 simple Pulse-Width Modulators
- 2 Versatile Pulse-Width Modulators

A short functional description of each of these peripherals is provided in the following sections.

1.14.1 Keyboard Scanner

The keyboard scanner function can automatically scan a keyboard of up to 64 keys in an 8×8 matrix. In operation, the keyboard scanner's internal state machine will normally be in an idle state, with all KEY_ROW[n] pins set high, waiting for a change in the column inputs to indicate that one or more keys have been pressed.

When a keypress is detected, the matrix is scanned by setting one output pin high at a time and reading the column inputs. After de-bouncing, the keypad state is stored and an interrupt is generated. The keypad is then continuously scanned waiting for 'extra key pressed' or 'key released'. Any new keypad state is scanned and stored into the matrix registers followed by a new interrupt request to the interrupt controller. It is possible to detect and separate up to 64 multiple keys pressed.

Features

- Supports up to 64 keys in 8×8 matrix.
- Programmable debounce period.
- A key press can wake up the CPU from Stop mode.

1.14.2 10-bit ADC

The ADC is a three channel, 10-bit successive approximation ADC. The ADC may be configured to produce results with a resolution anywhere from 10 bits to 3 bits. When high resolution is not needed, lowering the resolution can substantially reduce conversion time.

The analog portion of the ADC has its own power supply to enhance the low noise characteristics of the converter. This voltage is only supplied internally when the core has voltage. However, the ADC block is not affected by any difference in ramp-up time for VDD_AD and VDD_CORE voltage supplies.

Features

- Measurement range of 0 V to VDD_AD (nominally 3 V).
- Low noise ADC.
- Maximum 10-bit resolution, resolution can be reduced to any amount down to 3 bits for faster conversion.
- Three input channels.
- Uses 32 kHz RTC clock

1.14.3 RTC

The RTC runs at 32768 Hz using a very low power oscillator. The RTC counts seconds and can generate alarm interrupts that can wake up the device from Stop mode. The RTCCLK can also clock the 397x PLL, the Millisecond Timer, the ADC, the Keyboard Scanner and the PWMs. The RTC up-counter value represents a number of seconds elapsed since second 0, which is an application determined time. The RTC counter will reach maximum value after about 136 years. The RTC down-counter is initiated with all 1's.

The Real-Time Clock has a special power domain independent of the rest of the chip and is designed both to be very low power as well as to work down to low voltage levels (follow a discharging battery). It has 32 words of SRAM that can be used to hold data between microcontroller power cycles. The RTC generates a one-second tick from a dedicated 32 kHz crystal oscillator and uses 32-bit registers which should never need resetting because it takes 136 years to reach the maximum register count.

Two 32-bit match registers are readable and writable by the processor. A match will result in an interrupt provided that the interrupt is enabled. The ONSW output pin can also be triggered by a match event, and cause an external power supply to turn on all of the operating voltages, as a way to startup after power has been removed.

The RTC block is implemented in a separate voltage domain. The block is supplied via a separate supply pin from a battery or other power source.

The RTC block also contains 32 words (128 B) of very low voltage SRAM. This SRAM is able to hold its contents down to the minimum RTC operating voltage.

Features

- Measures the passage of time in seconds.
- 32-bit up and down seconds counters.
- Ultra low power design to support battery powered systems.

- Dedicated 32 kHz oscillator.
- An output pin is included to assist in waking up when the chip has had power removed to all functions except the RTC.
- Two 32-bit match registers with interrupt option.
- 32 words (128 B) of very low voltage SRAM.
- The RTC and battery RAM power have an independent power domain and dedicated supply pins, which can be powered from a battery or power supply.

1.14.4 High-speed timer

The high-speed timer block is clocked by the main peripheral clock. The clock is first divided down in a 16-bit programmable prescale counter which clocks a 32-bit Timer/Counter.

The high-speed timer includes three match registers that are compared to the Timer/Counter value. A match can generate an interrupt and cause the Timer/Counter to either continue to run, stop, or be reset. The high-speed timer also includes two capture registers that can take a snapshot of the Timer/Counter value when an input signal transitions. A capture event may also generate an interrupt.

Features

- 32-bit Timer/Counter with programmable 16-bit prescaler.
- Counter or Timer operation.
- Two 32-bit capture registers.
- Three 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.
- Pause control to stop counting when core is in debug state.

1.14.5 Millisecond timer

The Millisecond timer is clocked by the 32 kHz RTC clock. The registers are accessed on a different clock domain while the counter is counting on. This solution speeds up accesses to the Millisecond timer. Reads and writes to registers in the Millisecond timer are clocked by the HCLK. It takes a maximum of three HCLK before the writes are performed to the register.

Features

- 32-bit Timer/Counters.
- Counter or Timer operation.
- Two 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.
- Pause control to stop counting when core is in debug state.

1.14.6 General purpose 32-bit timers/external event counters

In addition to the High Speed Timers the LPC32x0 includes four 32-bit Timer/Counters. The Timer/Counter is designed to count cycles of the system derived clock or an externally-supplied clock. It can optionally generate interrupts or perform other actions at specified timer values, based on four match registers. The Timer/Counter also includes four capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

Features

- A 32-bit Timer/Counter with a programmable 32-bit prescaler
- Counter or Timer operation
- Up to four 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

1.14.7 8-bit Pulse width modulators

The LPC32x0 provides two 8-bit PWMs. They are clocked separately by either the main peripheral clock or the 32 kHz RTC clock. Both PWMs have a duty cycle programmable in 255 steps.

Features

- Clocked by the main peripheral clock or the 32 kHz RTC clock.
- Programmable 4-bit prescaler.
- Duty cycle programmable in 255 steps.
- Output frequency up to 50 kHz when using a 13 MHz peripheral clock.

1.14.8 Motor control pulse width modulator

The Motor Control PWM (MCPWM) provides a set of features for three-phase AC and DC motor control applications in a single peripheral. At the same time, the MCPWM is highly configurable for other generalized timing, counting, capture, and compare applications.

Features

- a 32-bit timer (TIM)

- a 32-bit period register (PER)
- a 32-bit pulse-width (match) register (PW)
- a 10-bit dead-time register (DT) and an associated 10-bit dead-time counter (DTIM)
- a 32-bit capture register
- two PWM (match) outputs (MCnA and MCnB) with opposite polarities
- a period interrupt, a pulse-width interrupt, and a capture interrupt

1.14.9 Timer/counters and Real-Time Clock (RTC)

A 32-bit Watchdog timer can generate internal as well as external resets. This capability allows synchronizing hardware system resets with internal ones. It serves to notify the system that a Watchdog Reset took place as well, which can be a sign that a software or hardware malfunction has occurred.

1.15 System control and analog blocks

These are the blocks responsible for Clock generation, Reset, Power-up and Booting of the chip. There is also a 10-bit A/D converter on-chip that can run at up to 400 kHz sampling frequency.

Clock generation: There are 2 main clock sources. The main crystal oscillator which shares the power domain of the microcontroller and a 32 kHz oscillator which drives the RTC and has its own power domain to keep the RTC alive. The main clock is multiplied up by a PLL to generate the high-speed CPU clock (max. 266 MHz). The RTC 32 kHz clock can also be multiplied up and used as an input to the main PLL so that the entire microcontroller can run from the RTC oscillator clock. Note that having two PLLs in series increases the jitter. Therefore the main crystal oscillator must be used to generate the USB clock. All other sub-systems can use the up-multiplied RTC clock.

Boot-up on Powering on the chip is handled by an on-chip bootloader in ROM that looks at the state of a pin to see if it should attempt to download a program over a serial link (UART) or download a program from NAND Flash and then branch to it. Since the bootloader is in ROM it takes no user memory space.

1.16 Debug and trace blocks

The microcontroller uses the standard ARM Enhanced JTAG Debug interface and, therefore, works with all standard ARM Development tools and hardware. Evaluation and development boards will be available from third party vendors. Software development and debugging tools and compilers from many vendors including ARM are available now and are well proven and mature.

Embedded Trace support is provided through use of an ETB (Embedded Trace Buffer) RAM block which stores Trace information in on-chip RAM to be read out later via the CPU or the E-JTAG interface. This saves over 20 pins and enables true Real-time operation over the Trace window. Both Data and Instruction values can be traced. The trace information is saved to a 45-byte FIFO whose contents are transferred to the ETB in Real-time.

1.17 Architectural power management

Several techniques are employed to allow full user control and customizing of power management as follows:

1. Programmable clock enabling: Each peripheral and AHB matrix has a selectable Clock Enable bit. Thus the user can control which combinations of peripherals he enables for his particular system design and also when to enable them. Further, several peripherals have local power savings means (such as sleep modes) for more power savings.
2. Low-Voltage Operation: The microcontroller is able to operate down to 0.9 Volts which reduces power tremendously (dynamic power is reduced by more than 50 % over 1.2 V operation) but requires operation at lower clock frequencies. This is suitable for very low power standby modes where system operation is required but performance can be compromised. The chip has a pin indicating that the chip is in a low power state so that the software can manage the system accordingly.
3. PLL Clock control: The system has full control over the PLL multiplier and can therefore manage this aspect of power. Note that high-speed DRAM control must be taken into account so that Refresh rates are maintained in low speed modes.
4. STOP mode: In this mode, the AHB matrix clock is disabled and the ARM clock is stopped. This is basically a Static-power-only mode.
5. AHB Bus Matrix Clock Control: The ARM CPU clock can be divided by a factor of 2, 4, or 8 to derive the Peripheral Bus clock. This can be used as a power control mechanism if only low bandwidth transfers are to be handled. Note that the CPU can execute out of its local 64 kB of SRAM and does not need to access the DRAM continuously (of course it also has caches).
6. System operation on RTC Clock: The RTC oscillator uses a 32 kHz crystal and runs at much lower power than the main crystal oscillator. A dedicated PLL is available which multiplies the RTC clock to a frequency where it can be further multiplied by the main PLL to generate the > 200 MHz CPU clock. In the case where the increased jitter caused by cascading PLLs is acceptable, this allows for lower power operation.
7. Power domain switching: The RTC clock is on a separate power domain and also has 32 words of low-power SRAM. To save the maximum amount of power, the microcontroller power can be turned off altogether while keeping the RTC alive with some critical system information that can be saved between power cycles. This also eliminates leakage current power consumption.

Using all the power management techniques allows for very flexible power management and permits power consumption to be tailored to required computational and peripheral operation requirements.

2.1 Bus architecture

2.1.1 Block diagram, AHB matrix view

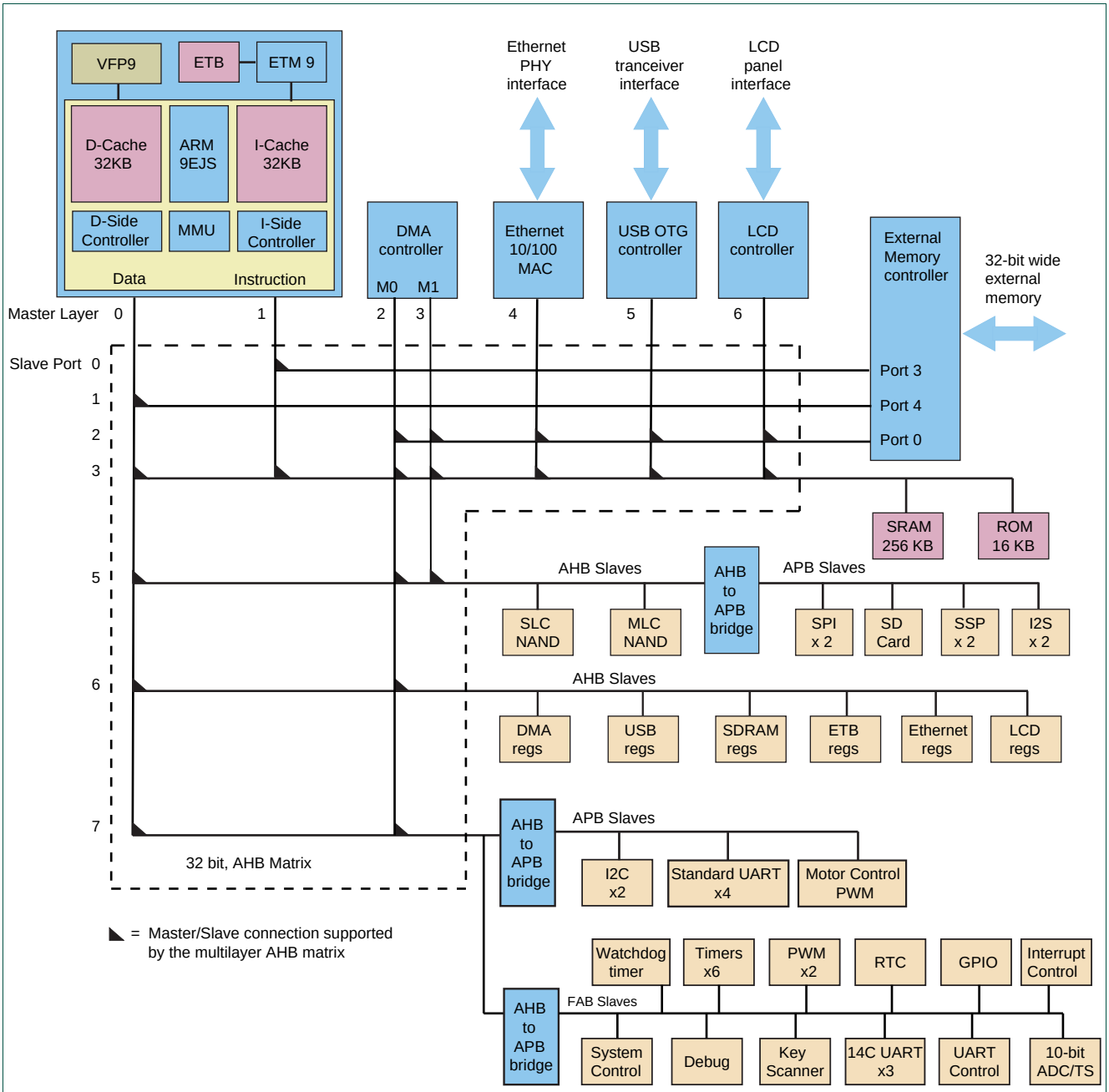


Fig 3. LPC32x0 block diagram, AHB matrix view

The AHB Bus matrix connects AHB Masters to AHB slaves. One benefit of using an AHB bus matrix is parallel AHB transfers. The maximum parallelism occurs when the five masters perform an AHB transfer at the same time. However, the 3 AHB slave ports connected to the EMC controller converge to a single external memory device which limits memory access to one data transfer at a time. The EMC controller takes advantage of this architecture as it can overlap a data transfer with a command for the next transfer.

The AHB matrix implements the AHB Lite bus standard. This means that there is no Bus request / grant arbitration and no split/retry signaling.

2.1.2 Multi-layer AHB Matrix

The AHB bus matrix runs from HCLK and schedules Master requests for each slave port as follows:

1. The master with the highest priority requesting the slave port will receive it regardless of the underlying uniform scheduling algorithm.
2. If all masters have the same priority, then the master selected by the uniform scheduler is given the slave port.
3. If no master is requesting the slave port, the slave port will generate idle cycles with all HSEL signals inactive. No address or data signals will toggle.
4. When a master asserts its LOCK signal, once it has been granted the slave port it will remain granted until the lock is removed.

Whenever a slave port's current master issues a non sequential or idle access, and the LOCK signal is inactive, the slave port is re-arbitrated.

The AHB matrix has the following attributes:

- No memory space access check. (All the 4 GB address range is valid).
- 32 bit wide data busses.
- Master bus access control enabled.
- Each master only has access to the slave ports shown in the block diagram.

The Master bus access control functionality is mainly used for stopping masters from doing AHB transfers when the ARM enters debug mode. The ARM DBG_ACK signal is used to activate the 'disable_req' signals going into the AHB matrix. The AHB matrix allows the current transfer for each master to complete before it inactivates the AHB_GRANT signal to the master, so that no data is lost. It also activates the 'disable_grant' signal for the ARM to read status. Software may also force the AHB matrix to disable AHB_GRANT to the masters.

Note: A Fetch Abort or a Data Abort resulting from an access to any AHB slave is considered a software bug. Software must treat such exceptions as unrecoverable errors.

2.1.3 Bus bridges

2.1.3.1 AHB to FAB bridge

A Fast Access Bus (FAB) bridge interfaces a number of FAB slaves to AHB matrix Slave Port 7. The registers in these slaves are clocked by HCLK.

Write accesses to FAB peripherals take a single HCLK cycle, read accesses take two HCLK cycles. Write accesses are accomplished using write holding registers. Read accesses are done directly from the slave. Logic is included to prevent read-back of registers that have a write in the process of being completed.

FAB slaves are clocked by PERIPH_CLK even though they are connected to a bus running at full HCLK speed.

2.1.3.2 AHB to APB bridges

There are two AHB to APB bridges, one on Slave Port 7 and one on Slave Port 5.

2.1.4 Transfer performance

2.1.4.1 AHB Matrix throughput

Bandwidth is reduced if two or more Masters compete for the same slave layer. This situation is likely to happen on SDRAM accesses even though there are 3 SDRAM slave ports. To maximize CPU performance, one Slave Port is assigned to instruction fetch, and another to data access.

2.1.4.2 SDRAM throughput

The SDRAM controller will have the highest throughput if many AHB slave ports are used because it can buffer single write accesses and it may overlap the end of one transfer with the start of a new one.

2.1.5 Arbitration

If there is more than one master accessing the same slave port, the AHB matrix schedules the accesses using an arbiter. The arbitration scheme used by the AHB matrix is a round robin scheduling. In general this is a good algorithm for avoiding extremely long latencies since no master can occupy any slave port for more than one burst period. The longest burst is 8. Masters with the same priority level will be arbitrated using round robin.

In addition to the AHB matrix arbiter, the SDRAM controller also has an arbiter prioritizing among requests from the four AHB data ports. AHB port 0 has highest priority. In addition, each data port has a time-out counter with programmable time-out. Whenever a time-out occurs, the priority is raised.

2.1.6 Data coherency

2.1.6.1 SDRAM

Data coherency between different AHB data ports is only guaranteed if write buffering is disabled. This is done by programming the E bit to 0 in EMCAHBControl0-4 registers. However this will reduce write performance.

2.1.6.2 ARM CPU

Any cache region defined as copy-back may have a data coherency problem. There is no insurance that other masters will access correct data. A solution for this configuration is to change the cache policy to write-through for the data region or to force cache write-backs using software before allowing access by the other masters.

2.2 Memory map

Figure 4 shows the memory map for the AHB matrix. This view represents all address sources except the USB interface.

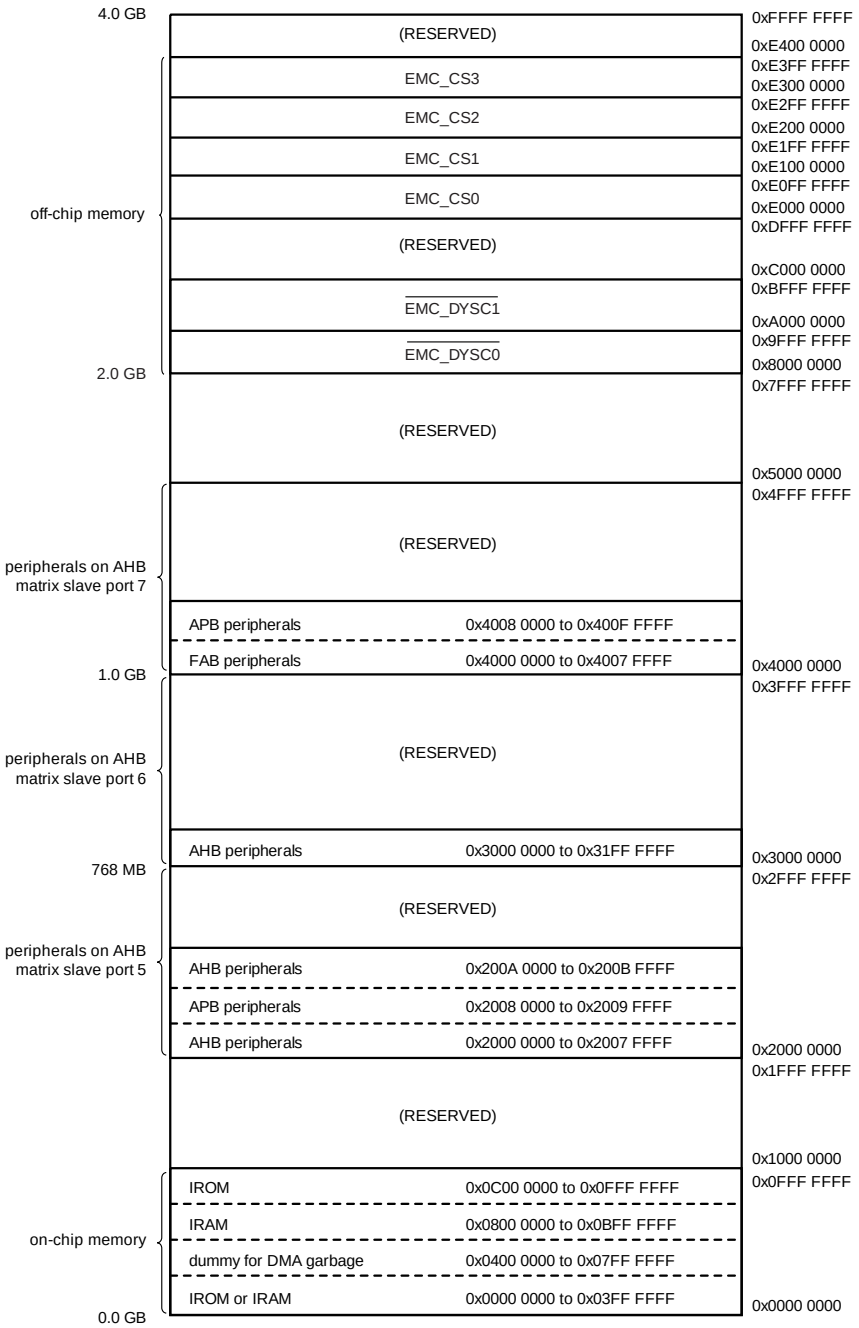


Fig 4. Overall LPC32x0 memory map

2.2.1 CPU memory space

The following table gives the address space for the LPC32x0, as seen by the AHB matrix.

Table 3. Overview of LPC32x0 memory space

Address	Function
0x0000 0000 to 0x0FFF FFFF	On-Chip Memory on AHB matrix slave 3 (see Figure 3): 0x0000 0000 to 0x03FF FFFF = IROM or IRAM. 0x0400 0000 to 0x07FF FFFF = Dummy space for DMA. Reads as all zeroes, write has no effect. 0x0800 0000 to 0x0BFF FFFF = IRAM (128 - 256 kB populated). 0x0C00 0000 to 0x0FFF FFFF = IROM (16 kB populated).
0x1000 0000 to 0x1FFF FFFF	Reserved
0x2000 0000 to 0x2FFF FFFF	Peripherals on AHB matrix slave 5 (see Figure 3): 0x2000 0000 to 0x2007 FFFF = AHB peripherals. 0x2008 0000 to 0x2009 FFFF = APB peripherals. 0x200A 0000 to 0x200B FFFF = AHB peripherals.
0x3000 0000 to 0x3FFF FFFF	Peripherals on AHB matrix slave 6 (AHB peripherals)
0x4000 0000 to 0x4FFF FFFF	Peripherals on AHB matrix slave 7 (see Figure 3): 0x4000 0000 to 0x4007 FFFF = FAB peripherals. 0x4008 0000 to 0x400F FFFF = APB peripherals.
0x5000 0000 to 0x7FFF FFFF	Reserved
0x8000 0000 to 0xFFFF FFFF	Off-Chip Memory Two dynamic memory banks, 512 MB each 0x8000 0000 - 0x9FFF FFFF EMC_DYCS0_N 0xA000 0000 - 0xBFFF FFFF EMC_DYCS1_N Four static memory banks, 16 MB each 0xE000 0000 0xE0FF FFFF EMC_CS0 0xE100 0000 0xE1FF FFFF EMC_CS1 0xE200 0000 0xE2FF FFFF EMC_CS2 0xE300 0000 0xE3FF FFFF EMC_CS3 Accessed as follows: • ARM 9 instruction fetch via AHB matrix slave port 0 to EMC port 3. • ARM 9 data access via AHB matrix slave port 1 to EMC port 4. • DMA controller (both channels) via AHB matrix slave port 2 to EMC port 0.

2.2.2 Peripheral addresses

Table 4 shows the base addresses of the peripheral devices present on the LPC32x0.

Table 4. Peripheral devices on the LPC32x0

Base Address	Peripheral	AHB Slave Port	Peripheral Type
0x2002 0000	SLC NAND Flash controller	5	AHB
0x2008 4000	SSP0	5	APB
0x2008 8000	SPI1	5	APB
0x2008 C000	SSP1	5	APB
0x2009 0000	SPI2	5	APB
0x2009 4000	I2S 0	5	APB

Table 4. Peripheral devices on the LPC32x0

Base Address	Peripheral	AHB Slave Port	Peripheral Type
0x2009 8000	SD card interface	5	APB
0x2009 C000	I2S 1	5	APB
0x200B 0000	MLC NAND Flash controller	5	AHB
0x3100 0000	DMA controller	6	AHB
0x3102 0000	USB interface	6	AHB
0x3104 0000	LCD interface	6	AHB
0x3106 0000	Ethernet interface	6	AHB
0x3108 0000	EMC configuration	6	AHB
0x310C 0000	ETB configuration	6	AHB
0x310E 0000	ETB data	6	AHB
0x4000 4000	System control functions	7	FAB
0x4000 8000	Master interrupt controller (MIC)	7	FAB
0x4000 C000	Slave interrupt controller 1 (SIC1)	7	FAB
0x4001 0000	Slave interrupt controller 2 (SIC2)	7	FAB
0x4001 4000	UART 1	7	FAB
0x4001 8000	UART 2	7	FAB
0x4001 C000	UART 7	7	FAB
0x4002 4000	RTC	7	FAB
0x4002 4080	RTC internal SRAM	7	FAB
0x4002 8000	GPIO	7	FAB
0x4002 C000	Timer 4	7	FAB
0x4003 0000	Timer 5	7	FAB
0x4003 4000	Millisecond timer	7	FAB
0x4003 8000	High speed timer	7	FAB
0x4003 C000	Watchdog timer	7	FAB
0x4004 0000	Debug	7	FAB
0x4004 4000	Timer0	7	FAB
0x4004 8000	ADC (Touch screen controller)	7	FAB
0x4004 C000	Timer1	7	FAB
0x4005 0000	Keyboard Scan	7	FAB
0x4005 4000	UART control register (general UART control)	7	FAB
0x4005 8000	Timer2	7	FAB
0x4005 C000	PWM1 and PWM2	7	FAB
0x4006 0000	Timer3	7	FAB
0x4008 0000	UART 3	7	APB
0x4008 8000	UART 4	7	APB
0x4009 0000	UART 5	7	APB
0x4009 8000	UART 6	7	APB

Table 4. Peripheral devices on the LPC32x0

Base Address	Peripheral	AHB Slave Port	Peripheral Type
0x400A 0000	I2C1	7	APB
0x400A 8000	I2C2	7	APB
0x400E 8000	MCPWM	7	APB

3.1 System control block

The System Control Block includes system features and control registers that are not directly related to specific chip functions. These include chip reset and Boot Map control.

3.1.1 Reset

Reset is accomplished by an active LOW signal on the RESET_N input pin. A reset pulse with a minimum width of 10 oscillator clocks after the oscillator is stable is required to guarantee a valid chip reset. At power-up, 10 milliseconds should be allowed for the oscillator to start up and stabilize after VDD reaches operational voltage.

An internal reset with a minimum duration of 10 clock pulses will also be applied if the watchdog generates an internal device reset. Details of Watchdog Timer operation may be found in the Watchdog Timer chapter.

Most on-chip registers are loaded with a pre-defined value upon occurrence of an internal or external reset. Note that only a few bits in the Real Time Clock are affected by an internal or external reset. Other RTC registers and bits are not modified by reset so that the RTC can continue operation independent of chip reset.

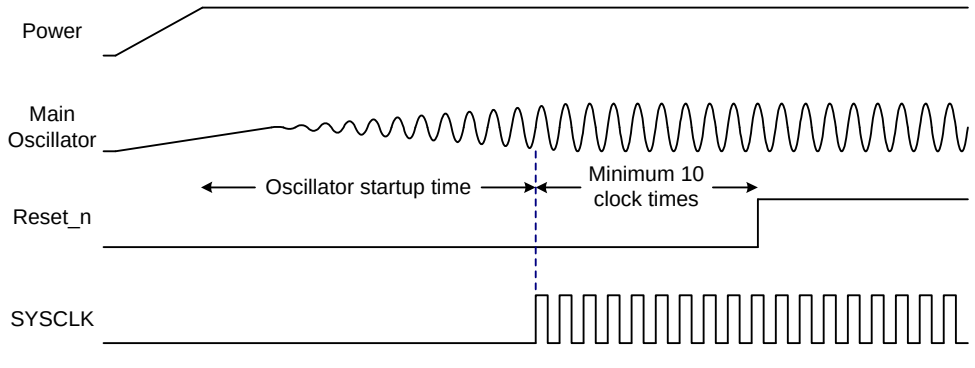


Fig 5. Power-up reset

3.1.2 Boot Map control register (BOOT_MAP - 0x4000 4014)

On reset, the ARM executes code beginning at address 0x0000 0000. The Boot Map control register aliases either Internal ROM or Internal RAM to start at address 0x0000 0000 to provide support options for the ARM vector table. By default this address maps to IROM memory containing instructions from the boot code. Both IROM and IRAM are available at other addresses at all times (see the Bus Architecture and Memory Map chapter). IRAM can be mapped in during the boot process so the application has IRAM for all exception vectors. Code execution must not be within the switched address space when the memory switch takes place.

Table 5. Boot map control register (BOOT_MAP - 0x4000 4014)

Bit	Function	Reset value
0	0 = IROM located at address 0x0000 0000 1 = IRAM located at address 0x0000 0000	0

Additional information about the Boot procedure may be found in the Boot Process chapter.

3.1.3 Unique serial ID registers (SERIAL_ID0 - 0x4000 4130 to SERIAL_ID3 - 0x4000 413C)

Every LPC32x0 chip contains a unique 128-bit serial ID number hard-coded into the four SERIAL_ID registers.

Table 6. Serial ID register (SERIAL_ID0 - 0x4000 4130)

Bit	Access	Function	Reset value
31:0	RO	Serial ID, bits 31 to 0.	part dependent

Table 7. Serial ID register (SERIAL_ID1 - 0x4000 4134)

Bit	Access	Function	Reset value
31:0	RO	Serial ID, bits 63 to 32.	part dependent

Table 8. Serial ID register (SERIAL_ID2 - 0x4000 4138)

Bit	Access	Function	Reset value
31:0	RO	Serial ID, bits 95 to 64.	part dependent

Table 9. Serial ID register (SERIAL_ID3 - 0x4000 413C)

Bit	Access	Function	Reset value
31:0	RO	Serial ID, bits 127 to 96.	part dependent

4.1 Introduction

The LPC32x0 provides detailed control of clock usage among chip functions, allowing fine tuning of power consumption in the target application. Most clocks can be disabled either globally or at an individual peripheral. Clock frequencies can be separately controlled through the use of PLLs, multiplexers, and dividers.

This section describes the generation of the various clocks needed by the LPC32x0 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:

- Clocking and power control
  - Clock identification
  - Default clock settings
- Power reduction modes
  - RUN, Direct RUN, and STOP modes
  - Start controller
  - Autoclocking
- Oscillators
- PLLs
- Clock dividers
- Clock usage in peripheral blocks
- Register description

4.2 Overview

Clocking in the LPC32x0 is versatile, so system and peripheral requirements may be met while still permitting optimization of power consumption. By default, the Main Oscillator is the source for the clocks used in most chip functions. Optionally, many functions can be clocked by the output of a PLL (with a fixed 397x rate multiplication) which is sourced from the Real Time Clock oscillator. In this mode, the Main Oscillator may be turned off unless the USB interface is enabled.

Whichever clock source is selected, a programmable PLL allows the CPU clock to be raised as high as 266 MHz. The AHB bus clock (HCLK) can be derived from that clock and may be as high as 133 MHz.

Clocks to most functions may be individually turned off when those features are not required in the application. In addition, many functions have dedicated clock dividers that may be tuned to provide the required performance without using power unnecessarily.

Another form of power reduction is provided in the form of alternate operational modes. Typically, the CPU is operated from a high frequency clock provided by a PLL. This option is called RUN Mode. At times when the application does not require such performance,

the PLL may be bypassed and the CPU run at a slower rate. This is called Direct RUN Mode. When the CPU has nothing to do but wait for an external event, clocking can be stopped entirely until the event occurs. This is called STOP mode.

Switching between RUN mode and Direct RUN Mode is accomplished entirely under software control. Since the CPU is halted in STOP Mode, hardware must restart clocking when a selected event occurs. This hardware is called the Start Controller.

Details of clocking and power control are found in the following section.

4.3 Clocking and power control

The LPC32x0 includes three operational modes that provide control over processing speed and power consumption. In addition, clock rates to different functional blocks may be controlled by changing clock sources, re-configuring PLL values, or altering clock divider configurations. This allows a trade-off of power versus processing speed based on application requirements.

The LPC32x0 also implements a separate power domain in order to allow turning off power to the bulk of the device while maintaining operation of the Real Time Clock and a small static RAM.

Power consumption is determined primarily by the clock frequencies selected and by which functional blocks are being clocked at any time. Therefore, to minimize power consumption, it is important to turn off clocks to any unused functional blocks. Most functional blocks have a clock enable/disable control contained in a register that is described in this chapter. Some blocks also have more elaborate clock controls.

4.3.1 Clock identification

All clocks in the LPC32x0 are derived from one of two base clock sources. These are OSC_CLK, the output of the main oscillator, and the 13.008896 MHz clock, which is generated by multiplying the 32 kHz RTC clock by 397. This clock is referred to as the 13’ clock.The 13’ MHz clock has a nominal frequency of 13.008896 MHz and has more jitter than the crystal-based OSC_CLK.

Table 10 describes the major clocks in the LPC32x0 and summarizes how they are used. Figure 6 shows the major clock sources and their distribution in the LPC32x0.

Table 10. Clocks and clock usage

Clock Name	Description
OSC_CLK	**Main oscillator clock —** This clock runs from an external crystal in the range of 1 MHz to 20 MHz, typically 13 MHz. Used by: USB PLL, HCLK PLL, SYSCLK.
RTC_CLK	**RTC clock —** Based on 32.768 kHz RTC oscillator. Used by: PLL397, Keyscan, ADC, PWM, MS Timer.
SYSCLK	**System Clock —** Based on the main oscillator frequency (OSC_CLK) or the 13’ MHz PLL397 output. Used by: clock generation logic.

Table 10. Clocks and clock usage

Clock Name	Description
ARM_CLK	**ARM Clock —** Based on the HCLK PLL output, SYSCLK, or PERIPH_CLK. Clock switching and HCLK PLL settings give ARM_CLK a very wide range of potential frequencies Used by: ARM9 CPU, MSSDCLK.
HCLK	**AHB Bus Clock —** Based on PERIPH_CLK, SYSCLK, or HCLK PLL output divided by 1, 2, or 4. The AHB HCLK will typically be ARM_CLK divided by 2 but can run at the same frequency as the ARM or be divided by 4. The AHB HCLK frequency should not be set higher than 133 MHz or lower than SYSCLK. Used by: the AHB matrix and USB AHB, AHB slaves, FAB slaves, and APB slaves.
PERIPH_CLK	**Peripheral Clock —** Based on SYSCLK or HCLK PLL output divided by 1 to 32. The maximum supported frequency of PERIPH_CLK is 20 MHz, typically 13 MHz. Typically, the PERIPH_CLK divider setting is chosen such that the PERIPH_CLK frequency remains the same when switching from Direct RUN mode to RUN mode, taking into account the HCLK PLL settings. This case occurs when the PERIPH_CLK frequency equals the SYSCLK frequency in RUN Mode. Used by: Many peripheral functions.
clk48mhz	**USB 48 MHz clock —** Based on OSC_CLK. The USB interface must be run from a 48 MHz clock. The USB specification has strict requirements for frequency (500 ppm) and jitter (500 ps). For this reason, the crystal-based OSC_CLK is used as the source for this clock, and must be running while the USB is active. OSC_CLK is divided by n before it enters the USB PLL, which must multiply the frequency up to 48 MHz when the USB is to be used. Used by: USB block.
DDRAM_CLK	**DDR SDRAM Clock —** Based on the HCLK PLL output or SYSCLK, divided by 1 or 2. If DDR SDRAM is used, this clock must be programmed to be twice the HCLK frequency. In RUN mode this is typically the same as the ARM_CLK frequency, but there is support for ARM clocking 4 times as fast as HCLK as well. In Direct RUN mode, it is not possible to generate this clock, so DDR SDRAM cannot be accessed in Direct RUN mode. Used by: EMC controller.
MSSDCLK	**SD Card Clock —** Based on ARM_CLK, divided by 1 to 15.

The Main oscillator and the RTC oscillator are shown at the lower left of the diagram. To the right of the Main oscillator may be found the clock mode logic, governed by the Start Controller. To the right of the clock mode logic is the HCLK PLL, clock switching logic and clock dividers, which provide clocks to most of the chip. Certain peripherals that are partly clocked by the RTC clock are shown at the lower right of Figure 6, while the USB block and its special clocking logic are shown at the top.

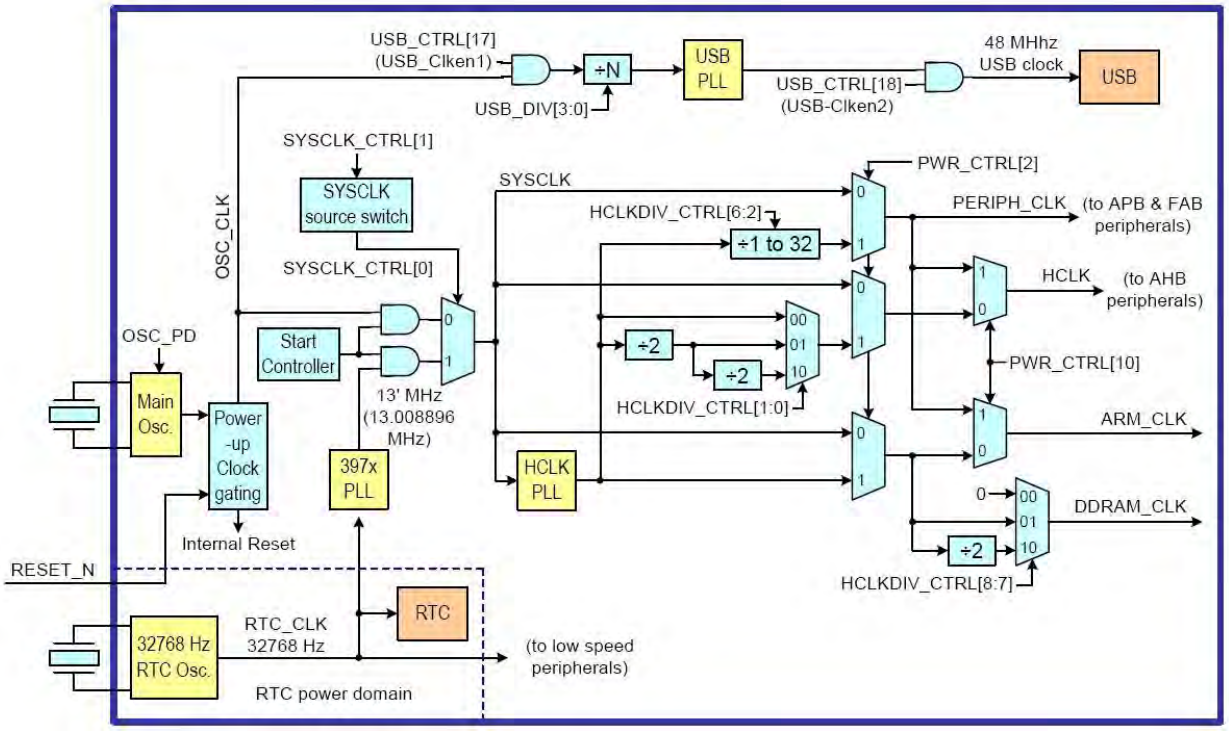


Fig 6. Clock generation for the LPC32x0

Details of how clocks are enabled, switched, and otherwise controlled are contained in the remainder of this chapter.

4.3.2 Default clock settings

At reset, the main oscillator is turned on, providing OSC_CLK, which is routed to SYSCLK. SYSCLK is then routed to all of the clocks that are enabled at reset: ARM_CLK; HCLK; and PERIPH_CLK.

Following is a summary of clock related settings and other information:

- OSC_CLK: Running, frequency determined by external crystal
- RTC_CLK: Running, frequency = 32.768 kHz if the correct external crystal is present
- SYSCLK: Running, frequency = OSC_CLK
- ARM_CLK: Running, frequency = OSC_CLK
- HCLK: Running, frequency = OSC_CLK
- PERIPH_CLK: Running, frequency = OSC_CLK
- clk48mhz: Stopped
- DDRAM_CLK: Stopped
- MSSDCLK: Stopped
- PLL397x: Running, frequency = 13.008896 MHz if RTC_CLK is running and loop control components are present
- HCLK PLL: Powered down, output off

- USB PLL: Powered down, output off

4.4 Operational modes

The LPC32x0 supports three operational modes, two of which are specifically designed to reduce power consumption. The modes are: RUN mode, Direct RUN mode, and STOP mode.

4.4.1 RUN mode

RUN mode is the normal operating mode for applications that require the CPU, AHB bus, or any peripheral function other than the USB block to run faster than the SYSCLK frequency.

- HCLK is running from the HCLK PLL output divided by 1, 2, or 4. The maximum allowed frequency is 133 MHz.
- ARM_CLK is running from the HCLK PLL output. The maximum allowed frequency is 266 MHz.
- Note that the CPU may be placed in the Wait for Interrupt mode while in RUN mode. Details of the Wait for Interrupt mode may be found in ARM architecture documentation, in coprocessor 15 register c7.

4.4.2 Direct RUN mode

Direct RUN mode allows reducing the CPU, AHB, and possibly the PERIPH_CLK rates in order to save power. Direct RUN mode can also be the normal operating mode for applications that do not require the CPU, AHB bus, or any peripheral function other than the USB block to run faster than the SYSCLK frequency. Direct RUN mode is the default mode following chip reset.

- ARM_CLK, HCLK, and PERIPH_CLK are running from SYSCLK: either 13’ MHz or OSC_CLK.
- AHB transfers are allowed.
- The HIGHCORE pin drives low and indicates the need for normal core voltage supply. In this mode the core voltage may be stabilizing. It only needs to be stable at nominal level when going to RUN Mode.

Note: the PERIPH_CLK divider (controlled by register bits HCLK_DIV_CTRL[6:2]) is typically configured to produce the same frequency as SYSCLK, thus allowing peripheral functions to operate at the same speed in both RUN and Direct RUN modes.

4.4.3 STOP mode

STOP mode causes all CPU and AHB operation to cease, and stops clocks to peripherals other than the USB block.

- HCLK is stopped, preventing any AHB communication.
- ARM_CLK is stopped, preventing any instruction execution.
- PERIPH_CLK is stopped, halting most peripheral blocks.
- The HIGHCORE pin drives high to indicates that a lowered core voltage supply is possible.

Note that USB clock generation (from OSC_CLK through clk48mhz, including the main oscillator) is not affected by transitions to RUN mode, Direct RUN mode, or STOP mode. Control of the USB clock is completely separate from these modes.

Figure 7 shows the possible transitions between the power modes.

The STOP mode is entered when the “STOP” clock gating circuitry stops SYSCLK, which is the base clock for the ARM subsystem, including all peripherals except the USB block. STOP mode is entered when software writes a one to PWR_CTRL[0] and the “Start activated” signal is inactive (see Figure 8). STOP mode is exited when one of the active start signals generates the correct edge, which is programmable. This will automatically clear PWR_CTRL[0].

When entering STOP mode, the CPU must run from either the main oscillator or the 13’ MHz clock from PLL397.

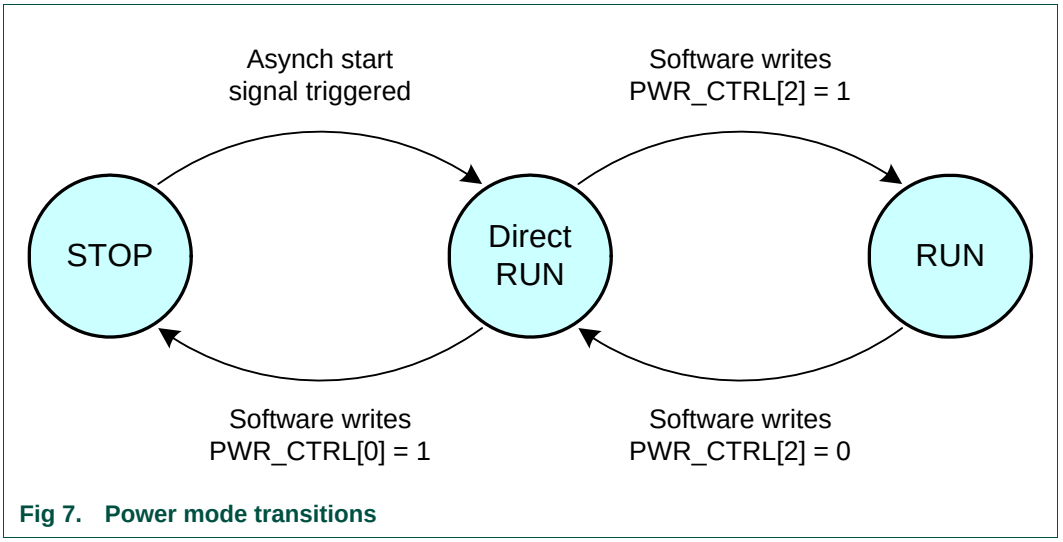


Fig 7. Power mode transitions

4.4.4 Start controller and related functions

4.4.4.1 Start controller

The Start controller provides a means to exit the STOP mode upon occurrence of a number of potential events. These events include interrupts from peripherals that are able to operate without any clock based on SYSCLK, and state changes on selected pins. Each Start source can be individually configured, enabled/disabled, and monitored by software.

The following list summarizes the potential Start sources.

- ADC interrupt
- USB interrupts
- USB_DAT_VP pin
- Millisecond Timer interrupt
- High Speed Timer capture input
- RTC interrupt

- Keyboard scanner interrupt
- GPIO_ through GPIO_5 pins
- UART 2 and 7 HCTS pins (U2_HTCS and U7_HTCS)
- UART 1 through 5, and UART 7 RX pins (Un_RX)
- UART 6 IRRX pin (U6_IRRX)
- SDIO_INT_N (MS_DIO[1]) pin
- MSDIO_START condition (Logical OR of MS_DIO[3:0] pins)
- GPI_0 through GPI_9, GPI_19, and GPI_28 pins
- SYSCLKEN pin
- SPI1 and 2 DATIN pins
- Ethernet MAC
- P0_GPIO and P1_GPIO combined as a single start signal

Figure 8 shows how the Start Controller works and the interaction of the Start feature with other chip functions. Due to the number of potential Start sources, there are two registers for each kind of function related to the Start Controller. One set of registers includes internally generated Start sources, plus some pin sources. The related register names end in '_INT'. The second set of registers includes only pin sources. The related register names end in '_PIN'.

The bottom of Figure 8 shows details of the operation of a single Start source. At the left is the signal that can trigger a Start. Moving to the right, there is a multiplexer that allows selecting which polarity of the signal generates a Start condition. The polarity selection is controlled by a bit in either the START_APR_INT or the START_APR_PIN register. Continuing to the right, there is the flip-flop that records the occurrence of the Start event. The output of this flip-flop provides the raw status of the Start signal (which may be read as a bit in either the START_RSR_INT or the START_RSR_PIN register), prior to masking. Finally, there is the gate that allows enabling or disabling the Start source, as controlled by a bit in either the START_ER_INT or the START_ER_PIN register. The output of this gate represents an event that will actually cause a Start to occur and may be read in either the START_SR_INT or the START_SR_PIN register. Finally, all of the Start sources are combined and used to generate the 'Start activated' signal that causes the device to exit STOP Mode.

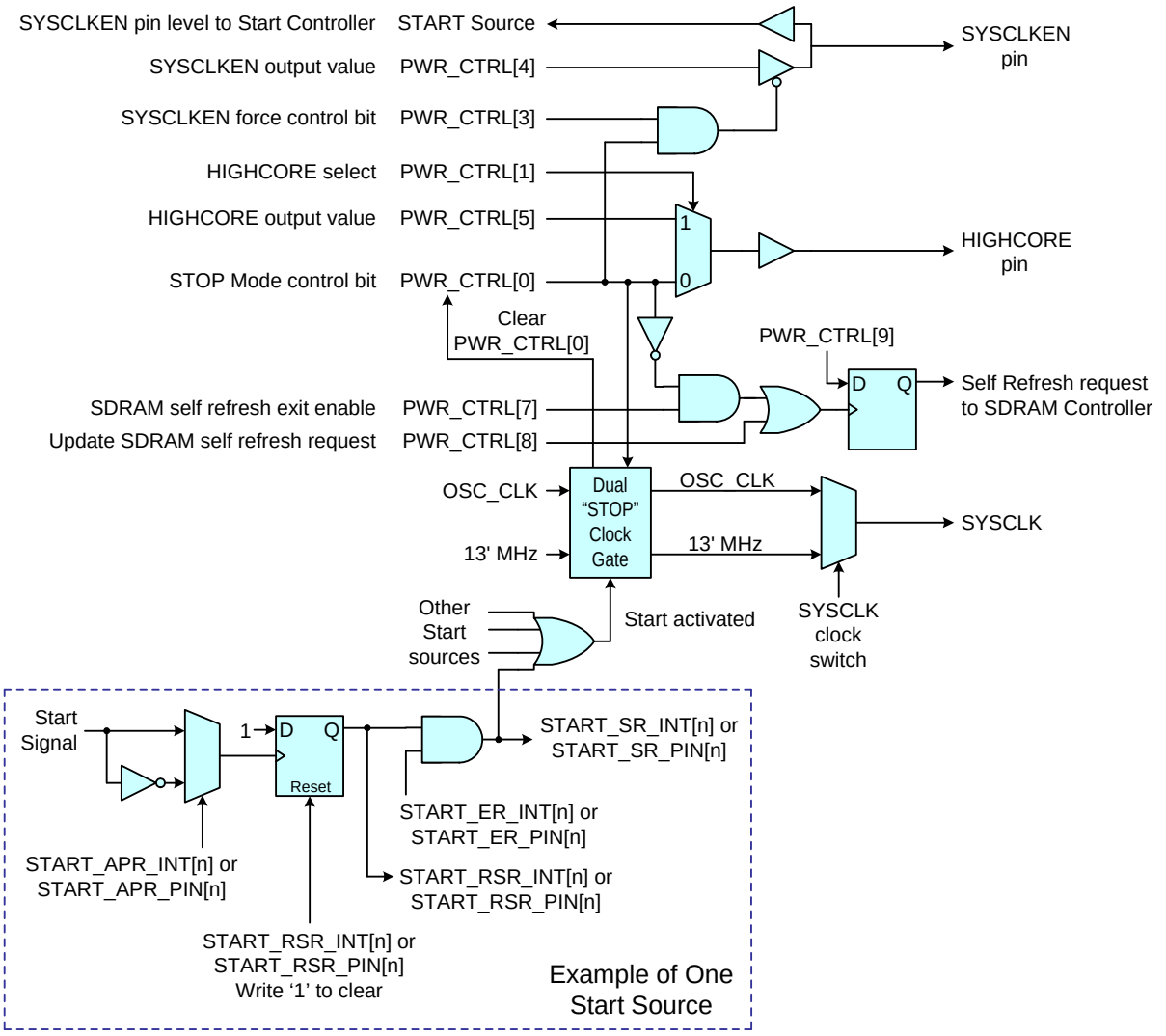


Fig 8. Start controller with core voltage selection and SDRAM self refresh control

The STOP clock gate circuitry prevents any glitches on the output clocks for all timing relationships between start events and clocks. If the "Start activated" signal goes active any time before or at the same time as software writes PWR_CTRL[0] to a 1, STOP mode will not be entered. In this case PWR_CTRL[0] will not be cleared by hardware. Software should always read PWR_CTRL[0] after going out of STOP mode and clear PWR_CTRL[0] if not cleared by hardware.

4.4.4.2 Core voltage selection

The HIGHCORE output pin may be used to save additional power during STOP Mode or low frequency operation, by signaling external circuitry to lower the core supply voltage. If any on-chip clocks are running above 13 MHz, nominal core supply voltage (1.2 V) must be supplied. If all on-chip clocks are running at or below 13 MHz (DIRECT RUN Mode), or during STOP Mode, the core supply voltage may be lowered to 0.9 V (see DC specifications for voltage limits). The logic related to the HIGHCORE pin is shown in Figure 8.

The HIGHCORE output pin is driven low after reset. A low indicates to an external power supply controller that nominal core voltage is needed. If software writes a 0 to the PWR_CTRL[1] bit, the HIGHCORE pin will drive high during STOP mode. The external power supply controller may then cause the core voltage be lowered to 0.9 V if the SYSCLK frequency is not above 13 MHz. After exit from STOP mode, the core voltage needs to stabilize to the nominal voltage before the ARM can change to higher frequency operation, if needed. The power supply must ensure that any over/under swing on the core voltage is within the operating limits. The USB clock cannot be operated in the low core voltage mode. It is important that software reads PWR_CTRL[0] after exiting from STOP mode. If this bit is 1, it needs to be written to a 0 by software in order to guarantee the correct level on the HIGHCORE pin.

In order to lower the operating voltage at low frequencies when not entering STOP Mode, software must control the value of the HIGHCORE pin. This is accomplished by writing a 1 to the PWR_CTRL[1] bit, causing the value of the PWR_CTRL[5] bit to appear on the HIGHCORE pin. When changing power supply voltages, all operating clocks must be at 13 MHz or lower prior to reducing the core supply voltage and remain there until the core supply voltage has stabilized at the nominal voltage. Only then any of the clock speeds can be increased to above 13 MHz.

4.4.4.3 SDRAM self-refresh control

The SDRAM Self Refresh Request signal (see EMCSREFREQ, PWR_CTRL[9]) is an input to the SDRAM controller and takes the SDRAM in and out of self refresh mode. Any external SDRAM devices must be put in self refresh mode before the system enters STOP mode. This is done by software writing first a 1 to PWR_CTRL[9], next a 1 to PWR_CTRL[7], and then writing a 1 and then a 0 to PWR_CTRL[8]. This will assert the SDRAM Self Refresh Request signal. Setting the SDRAM Self Refresh Exit Enable bit (PWR_CTRL[7]) ensures that the SDRAM Self Refresh Request signal is de-asserted as soon as the system exits STOP mode. Software must then wait for the SDRAM controller to indicate that it has put the SDRAM in self refresh mode by polling an SDRAM controller register. Before entering STOP mode, software must program PWR_CTRL[9] to 0. When the system exits STOP mode, the SDRAM Self Refresh Request signal is cleared automatically. The logic controlling the SDRAM Self Refresh Request signal is shown in Figure 8.

4.4.4.4 System clock request

The SYSCLK_EN pin can be used as a method to request external circuitry to provide a clock to the Main oscillator input, SYSX_IN. This allows the possibility of turning off an external clock source when the LPC32x0 is in STOP Mode. This is not necessary if a crystal is connected to the Main oscillator.

When the PWR_CTRL[3] bit = 0 (the default state), SYSCLKEN is driven high when the chip is not in STOP Mode and can be turned off (high impedance) when STOP Mode is entered.

If the SYSCLKEN function is not needed in the system, PWR_CTRL[3] can be used to force the SYSCLKEN pin to always be turned on (not high impedance) and driven to the level defined by PWR_CTRL[4]. This allows SYSCLKEN to be used as a simple General Purpose output pin.

4.4.5 Auto clocking

Some peripherals functions (listed in the AUTOCLK_CTRL register description later in this chapter) have auto clocking functionality. This autoloaded functionality can be disabled. The autoloaded circuitry enables when the device is accessed and disables HCLK automatically when the HCLK is not accessed for a predefined number of cycles. Note that some peripherals also have a software controlled clock gate which can stop all clocks to the autoloaded circuitry.

4.5 Internal SRAM Power off Switches

The Internal SRAM in the LPC32x0 is optimized for low power operation. To lower the current consumption even further, the blocks of internal memory can be switched off. The LPC32x0 does not have a built in power down mode. Instead, the internal SRAM contains several Power Off Switches (POS). Each switch controls a 64 kB block of internal SRAM (IRAM). These power switches are external to the IRAM and remove all power to the memory block. This reduces the power consumption of the memory to a very low value.

Two asynchronous signals (POS0 and POS1) control the power-down behavior of the memory blocks. Both pins have the same functionality (POS) but are used sequentially to reduce peak charge-up currents.

- POS0 enables a small POS transistor with limited current capability. This smaller transistor charges the memory using a low current to avoid having larger peak currents.
- POS1 enables the large POS transistor allowing the IRAM block to enter operational mode.

There is a maximum of 256 kB of internal memory in the LPC32x0 family, one 64 kB block is always powered and the other three 64 kB blocks have power off switches.

There is no time sequencing requirement to powering off a block of memory, we recommend setting POS0 before POS1. To power up a memory block the POS0 switch should be turned on 2 μsec before POS1 is turned on. Violating this timing requirement can result in a current spike in the power supply.

If the power is turned off using the POS switches the contents of the IRAM in the block is lost.

4.6 Oscillators

As shown in Figure 6, there are two crystal oscillators. One is a 32 kHz oscillator that runs the Real Time Clock. This oscillator can be used to run the entire chip (with the exception of the USB block), with SYSCLK equal to 13.008896 MHz through the use of the 397x PLL. The value 13.008896 MHz is referred to as 13’ MHz. The USB block cannot be connected to 13’ MHz because this would not meet the timing requirements set forth in the USB specification.

If a SYSCLK frequency other than 13 MHz is required in the application, or if the USB block is not used, the main oscillator may be used with a frequency of between 1 MHz and 20 MHz. For USB operation, a frequency of at least 13 MHz must be used in order to satisfy the input requirements of the USB PLL. A 13 MHz crystal or an off the shelf crystal of 16 MHz are recommended in a system requiring USB operation.

4.6.1 Main oscillator control

The register bit OSC_CTRL[1] reflects the state of pin DBGEN. The register bit OSC_CTRL[0] is 0 after reset, allowing the main oscillator to run as long as OSC_CTRL[1] is also 0. The external reset input RESET_N must be active until the oscillator outputs a stable clock (typically 2 milliseconds, refer to the data sheet for details). During active reset the output clock of the main oscillator is stopped by the 'Power-up clock gating' block. This prevents any bad clocks from the oscillator during startup to propagate through the device. When RESET_N goes inactive (high), the main oscillator output will be enabled. In addition the Reset_int signal will be held low for another 16 clock cycles before going high. The Reset_int signal is used for internal reset of the device. When RESET_N becomes active, Reset_int will become active immediately.

The CPU begins execution using OSC_CLK. If the 13' MHz clock will be used, software must wait for it to be stable before it can be used as the SYSCLK clock source. In order for switching from OSC_CLK to the 13' MHz clock to function correctly, OSC_CLK must be running at 13 MHz. This is a limitation of the clock switching circuitry.

Note that the main oscillator may use an external clock signal connected to SYSX_IN via a 100 pF series capacitor instead of a crystal. The amplitude of the external clock must be at least 200 mV rms.

The main oscillator has software controllable tuning capacitors. By default, there are 6.4 pF load capacitors added to the SYSX_IN and SYSX_OUT pins. The external load capacitors should be configured to have a value which makes the sum of both capacitors have the nominal value for the crystal. Software can then tune the range down by 6.4 pF, or up by 6.3 pF.

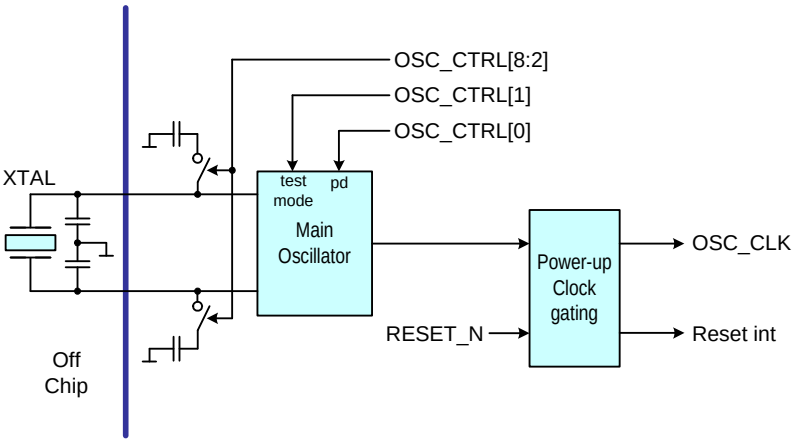


Fig 9. Main oscillator control

4.7 PLLs

The LPC32x0 includes three PLLs: one allows boosting the RTC frequency to 13.008896 MHz for use as SYSCLK; one provides the 48 MHz clock required by the USB block; and one provides the basis for HCLK, ARM_CLK, and PERIPH_CLK. All three PLLs and how they are connected are shown in Figure 6.

The first PLL is a fixed 397x frequency multiplier and is controlled by the register PLL397_CTRL, described in Section 4.11.4.

The other two PLLs, referred to as the HCLK PLL and the USB PLL, are identical in operation. Both are described in the following sections.

4.7.1 PLL397

PLL397 multiplies the 32768 Hz RTC clock up to a 13.008896 MHz clock. The PLL is designed for low power operation and low jitter. PLL397 requires an external low pass loop filter for proper operation. This is shown in Figure 10 and detailed below.

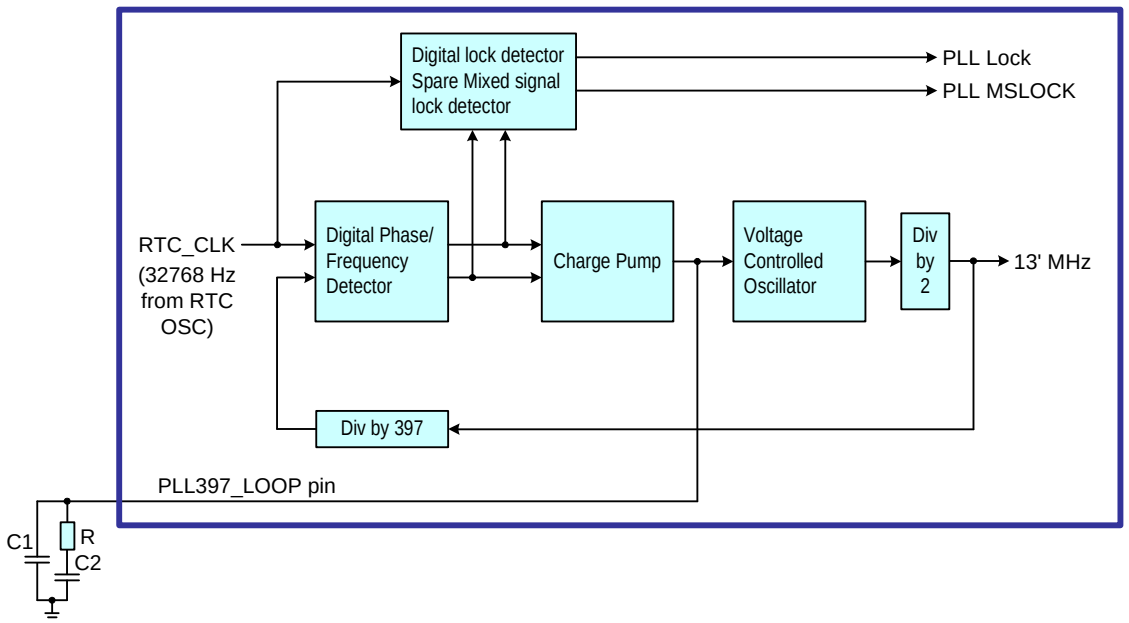


Fig 10. PLL397 and external low pass filtering

Use the following external components for the loop filter.

Table 11. External PLL397 component values

Component	Value - Type - package	Tolerance
R	120 kΩ - 0603	1 %
C1	150 pF - C0G - 0603	5 %
C2	3900 pF - C0G - 0805	5 %

The signals are noise sensitive, so the PCB tracks must be short. Note that package type indicated is the largest one to use. Smaller is better.

4.7.2 HLCK and USB PLL operation

The HCLK and USB PLLs accept an input clock frequency in the range of 1 MHz to 20 MHz. The input frequency is multiplied up to a higher frequency, then divided down to provide the output clock.

The PLL input may initially be divided down by a pre-divider value ‘N’, which may have the values 1, 2, 3, or 4. This pre-divider can allow a greater number of possibilities for the output frequency. Refer to Figure 11 for a block diagram of the PLL.

Note: the pre-divider in the USB PLL must not be confused with the pre-divider that divides the OSC_CLK prior to its arrival at the input to the USB PLL, refer to Figure 6.

Following the PLL input divider is the PLL multiplier. This can multiply the pre-divider output by a value ‘M’, in the range of 1 through 256. The resulting frequency must be in the range of 156 MHz to 320 MHz. The multiplier works by dividing the output of a Current Controlled Oscillator (CCO) by the value of M, then using a phase detector to compare the divided CCO output to the pre-divider output. The error value is used to adjust the CCO frequency.

At the PLL output, there is a post-divider that can be used to bring the CCO frequency down to the desired PLL output frequency. The post-divider value ‘P’ can divide the CCO output by 1, 2, 4, 8, or 16. The post-divider can also be bypassed, allowing the PLL CCO output to be used directly.

An alternative connection allows feeding the PLL output back to the multiplier, rather than using the CCO output directly, although this tends to reduce the PLL output frequency options.

Each PLL is configured by a control register: HCLKPLL_CTRL for the HCLK PLL, and USB_CTRL for the USB PLL. The PLL multiplier, pre-divider, and post-divider values are contained in these registers, as well as other PLL controls and the PLL Lock status.

The PLLs are turned off following a chip Reset and must be enabled by software if they are to be used. Software must fully configure the PLL, wait for the PLL to Lock, then cause the PLL to be connected as a clock source.

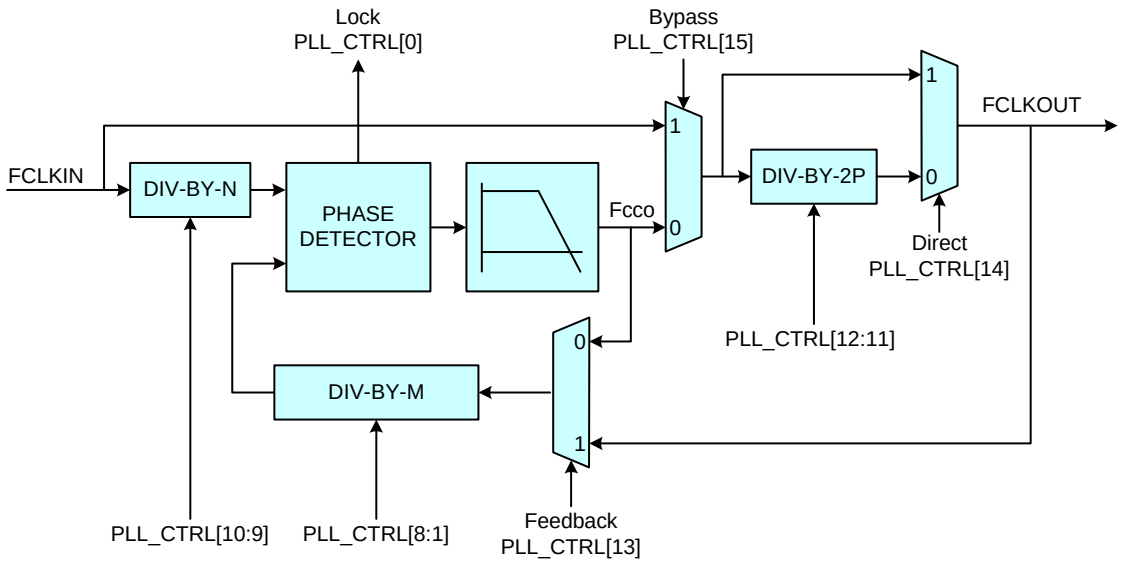


Fig 11. Block diagram of the HCLK and USB PLLs

4.7.3 PLL control bit descriptions

The PLLs are controlled by bits in the HCLKPLL_CTRL and USB_CTRL registers. The USB_CTRL register also contains additional bits to control other USB functions. Refer to Table 12.

Warning: Improper setting of PLL values may result in incorrect operation of any chip function that is dependent on it.

Table 12. PLL control bits

Bit(s)	Description	Access	Reset value
16	PLL Power down. This bit is used to start and stop the PLL. The PLL output must not be used until the PLL is in a Locked state, as indicated by the PLL LOCK bit. 0 = The PLL is in power down mode. 1 = The PLL is in operating mode.	R/W	0
15	Bypass control. Determines whether the PLL multiplier is used. 0 = The CCO output clock is sent to post divider (normal PLL operation). 1 = The PLL input clock bypasses the CCO and is sent directly to the post divider.	R/W	0
14	Direct output control. Determines whether the PLL post-divider is used. 0 = The output of the post-divider is used as output of the PLL. 1 = The output of the PLL is the undivided CCO output, bypassing the post divider.	R/W	0
13	Feedback divider path control. Determines whether the CCO output is fed directly to the PLL feedback divider or whether it goes through the post-divider first. 0 = The feedback divider is clocked by the CCO output. 1 = The feedback divider is clocked by FCLKOUT (the post-divider output).	R/W	0

Table 12. PLL control bits

Bit(s)	Description	Access	Reset value
12:11	PLL post-divider (P) setting. Supplies the value ‘P’ in the PLL frequency calculations. This divider divides down the CCO output. This field is encoded as follows: 00 = divide by 2 (P=1) 01 = divide by 4 (P=2) 10 = divide by 8 (P=4) 11 = divide by 16 (P=8)	R/W	00
10:9	PLL pre-divider (N) setting. Supplies the value ‘N’ in the PLL frequency calculations. The pre-divider reduces the input frequency before it goes to the CCO phase detector. The value stored here is N - 1, giving a range for N of 1 through 4: 00 = 1 01 = 2 10 = 3 11 = 4	R/W	00
8:1	PLL feedback divider (M) setting. Supplies the value ‘M’ in the PLL frequency calculations. The feedback divider divides the output frequency before it is fed back to the CCO phase comparator. The value stored here is M - 1, giving a range for M of 1 through 256: 00000000 = 1 00000001 = 2 …… 11111110 = 255 11111111 = 256	R/W	0x00
0	PLL LOCK status. This bit indicates the status of the PLL. 0 = the PLL is not locked. The PLL output clock must not be used. 1 = the PLL is locked. The PLL output clock is stable and ready to be used.	RO	0

4.7.4 PLL modes and frequency calculation

The PLLs have six basic modes of operation, with different properties and frequency calculations.

The PLL equations in the following mode descriptions use the following parameters:

- FCLKIN, the frequency of the PLL input clock.
- FREF, the frequency of the PLL reference clock, which is the output of the pre-divider.
- FCCO, the frequency of PLL Current Controlled Oscillator.
- FCLKOUT, the output frequency of the PLL.
- N PLL, pre-divider setting based on the bits in the relevant control register. N can have the values 1, 2, 3, or 4.
- M PLL, feedback divider setting based on bits in the relevant control register. M is an integer from 1 through 256.
- P PLL, post-divider setting based on the bits in the relevant control register. P can have the values 1, 2, 4, or 8.

Note: refer to the control register bit field description for information on how to store the values of N, M, and P in the register.

4.7.4.1 Power-down mode

When the PLL power down bit (bit 16 of the PLL_CTRL register) is 0, the analog portion of the PLL is turned off and the output divider is reset. In this mode, the PLL draws very little power. The PLL can pass the input clock to the output if the other control bits are set to enter Direct Bypass mode.

4.7.4.2 Direct mode

In Direct mode, the PLL output divider is not used, causing FCLKOUT to be equal to FCCO. Direct Mode is entered when PLL_CTRL[15] = 0 and PLL_CTRL[14] = 1. The related PLL equation is:

$$FCLKOUT = FCCO = (M \times FCLKIN)/N$$
$$FREF = FCLKIN/N \quad (1)$$

4.7.4.3 Bypass mode

In Bypass mode, the analog portion of the PLL is placed in power down mode and the input clock is routed through the post-divider. Bypass Mode is entered when PLL_CTRL[15] = 1 and PLL_CTRL[14] = 0. The related PLL equation is:

$$FCLKOUT = FCLKIN/(2 \times P) \quad (2)$$

4.7.4.4 Direct Bypass mode

The Direct Bypass mode is a combination of the preceding two mode. The analog portion of the PLL is placed in power down mode, and the input clock is routed to the PLL output. Direct Bypass Mode is entered when PLL_CTRL[15] and PLL_CTRL[14] are both set to 1. The related PLL equation is:

$$FCLKOUT = FCLKIN \quad (3)$$

4.7.4.5 Integer mode

In Integer mode, the PLL CCO output is routed to the post divider, and the PLL feedback loop is driven by FCLKOUT. Integer Mode is entered when PLL_CTRL[15] = 0, PLL_CTRL[14] = 0, and PLL_CTRL[13] = 1. The related PLL equations are:

$$FCLKOUT = M \times (FCLKIN/N)$$
$$FCCO = (FCLKIN/N) \times (M \times 2P) \quad (4)$$
$$FREF = FCLKIN/N$$

4.7.4.6 Non-integer mode

In Non-Integer mode, the PLL CCO output is routed to the post divider, and the PLL feedback loop is driven by the CCO output. Non-Integer Mode is entered when PLL_CTRL[15], PLL_CTRL[14], and PLL_CTRL[13] all = 0. The related PLL equations are:

$$FCLKOUT = (M/(2 \times P)) \times (FCLKIN/N)$$
$$FCCO = M \times (FCLKIN/N) \quad (5)$$
$$FREF = FCLKIN/N$$

4.7.4.7 PLL requirements

In modes where the PLL is active (Integer Mode and Non-Integer Mode), the PLL inputs and settings must meet the following conditions:

- FCLKIN must be in the range of 1 MHz to 20 MHz. Bear in mind that OSC_CLK is divided by n in order to produce FCLKIN to the USB PLL.
- FCCO must be in the range of 156 MHz to 320 MHz.
- FREF must be in the range of 1 MHz to 27 MHz.

Note: selecting a low FCCO frequency will result in lower power consumption.

4.7.4.8 Notes about the USB PLL

There are constraints to the main oscillator selection if the application requires use of the USB interface. USB requires that the F_{CCO} of the USB PLL be 192 MHz. This is because it is the only legitimate value for F_{CCO} that allows the post-divider to produce a 48 MHz output. This also fixes the post-divider at divide by 4 (P=2). The value of the crystal used for the main oscillator then must be selected such that it can support the 192 MHz CCO frequency. Note there is a divide-by-n predivider between OSC_CLK and the USB PLL input. A 13 MHz crystal, or standard 16 MHz crystal are recommended for this purpose.

For a system using the USB interface, the PLL and divider equations can be reduced to:

OSC_CLK = 2496 / M, where M = 104 to 192

OSC_CLK values that can produce a 48 MHz USB clock with no intrinsic rate error are: 13, 15.6, 16, 16.64, 19.2, 19.5, and 19.968 MHz.

4.7.4.9 Example settings for the HCLK PLL

Examples in the table have the following settings in common:

- PLL Power Down: HCLKPLL_CTRL[16] = 1
- Bypass control: HCLKPLL_CTRL[15] = 0
- Direct output control: HCLKPLL_CTRL[14] = 1
- Feedback divider path control: HCLKPLL_CTRL[13] = don’t care (due to HCLKPLL_CTRL[14])
- Pre-divider setting: HCLKPLL_CTRL[10:9] = 00
- Post-divider setting: HCLKPLL_CTRL[12:11] = don’t care (due to HCLKPLL_CTRL[14])

Table 13. HCLK PLL examples

Output clock (MHz)	Source Frequency	Feedback divider (M)	CCO frequency (MHz)
208.14	13.008896 MHz (13’ MHz from RTC and PLL397)	16	208.14
208	13.0000 MHz (from a crystal on the main oscillator)	16	208
200	20.0000 MHz (from a crystal on the main oscillator)	10	200
103.2192	14.7456 MHz (from a crystal on the main oscillator)	7	103.2192

4.8 Clock dividers

Limited clock dividers are provided for PERIPH_CLK, HCLK, and DDRAM_CLK in the clock generation circuitry. Many individual peripheral blocks have their own clock dividers that are used as rate generators, etc. These are described in the chapter for the relevant peripheral.

The divider associated with PERIPH_CLK is provided primarily to allow the PERIPH_CLK rate to remain constant when the device is switched from RUN mode to Direct RUN mode. The PERIPH_CLK divider allows dividing the HCLK PLL output by a value from 1 to 32. This allows for matching PERIPH_CLK to the SYSCLK rate for the maximum HCLK PLL output frequency (266 MHz) with a SYSCLK frequency as low as 6.5 MHz. The PERIPH_CLK divider is controlled by bits 6 through 2 of the HCLKDIV_CTRL register.

The HCLK divider allows selection of the ratio of HCLK to ARM_CLK. HCLK can be the same rate as ARM_CLK, or it can be divided by 2 or 4. The maximum rate for HCLK is 133 MHz. The HCLK divider is controlled by bits 1 through 0 of the HCLKDIV_CTRL register.

If DDR SDRAM is used, DDRAM_CLK must be twice the HCLK rate. Typically, HLCK will run at half the ARM_CLK rate, and DDRAM_CLK will be the same as ARM_CLK. If the HCLK rate is set to one fourth of ARM_CLK, then DDRAM_CLK should be half of ARM_CLK. DDRAM_CLK should be stopped (the default at reset) if DDR SDRAM is not used. The DDRAM_CLK divider is controlled by bits 8 through 7 of the HCLKDIV_CTRL register.

4.9 SYSCLK switching

If the Main Oscillator frequency is 13 MHz, it is possible to switch SYSCLK to the 13’ MHz clock output by PLL397. If the Main Oscillator frequency is not 13 MHz, switching to 13’ MHz should not be attempted.

Figure 12 shows the basics of the SYSCLK clock switching circuitry. Details such as synchronizer flip-flops, reset, clock gating and software triggering are not shown.

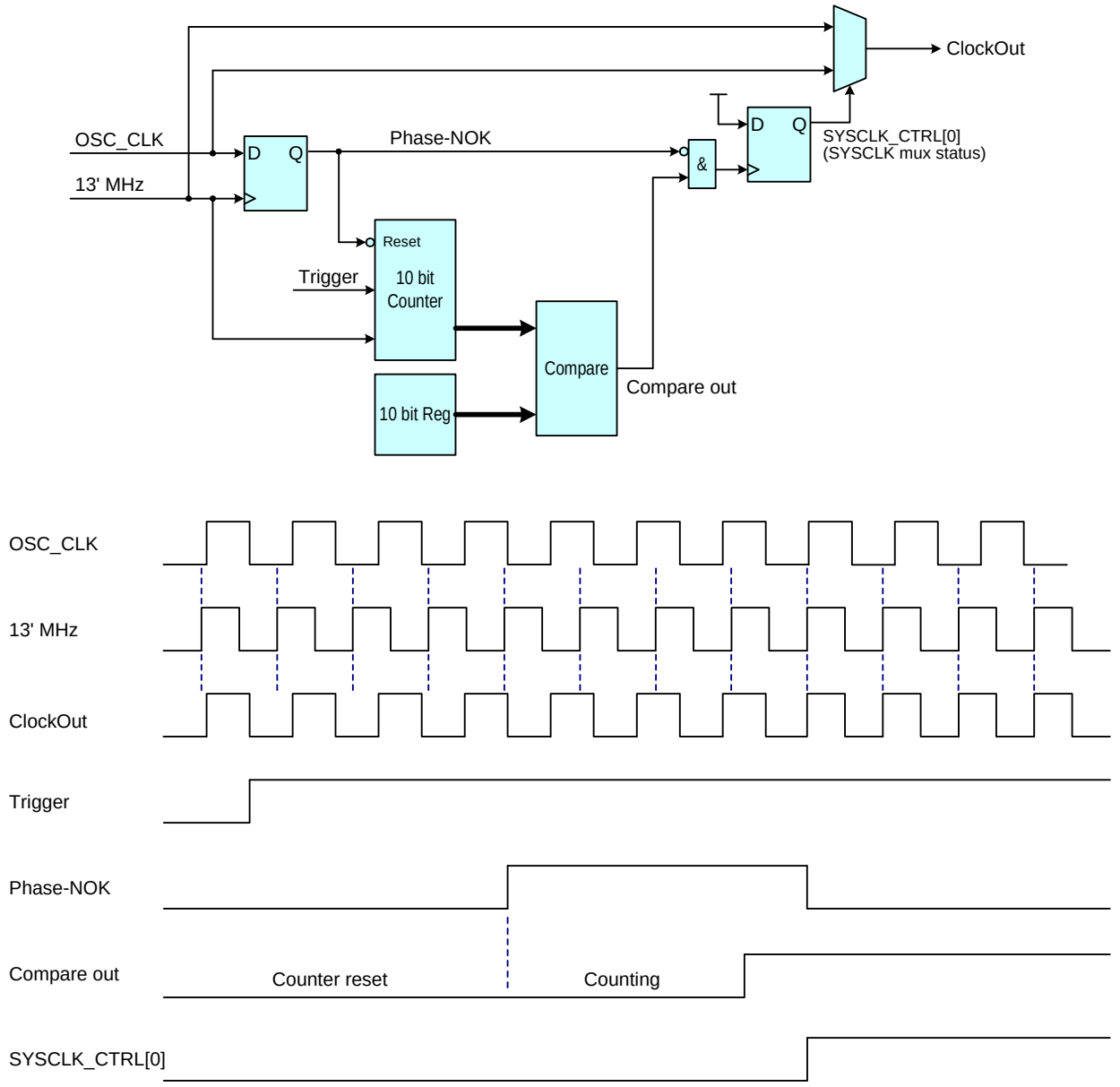


Fig 12. SYSCLK source clock switching

If the system is running from the RTC generated 13’ MHz clock (13.008896 MHz), the main oscillator must be enabled, and the SYSCLK source switched to it when the USB block is to be used.

The clock switcher circuit allows switching SYSCLK from the 13’ MHz to the main oscillator clock (if it is running at 13 MHz) without stopping the PLLs. Clocks are switched at a time when the 13 MHz clock has just slipped behind the 13’ MHz clock in phase. This ensures that the output clock after the clock switching MUX will have a slightly longer high period. The PLL output will go slightly down in frequency for a short period, but the effect

will be limited by the fact that the high clock period during switch over is nearly as short as the normal 13' MHz period. The phase synchronization is controlled by the value in the SYSCLK_CTRL register bits 11 to 2.

If the main oscillator has been started in order to switch to it, the CPU must wait a fixed time in order ensure that the main oscillator clock is stable. The CPU can then write to bit 1 in the SYSCLK_CTRL register to trigger the circuitry shown in Figure 12. When the main oscillator and 13' MHz clocks have slipped to the wanted position, the switching will occur without any further CPU intervention. the CPU must wait until it gets a response that the switch has occurred by polling the bit SYSCLK_CTRL[0] register until it is set before resuming execution at the new clock frequency. The CPU can read the status in the SYSCLK_CTRL[0] bit to determine which clock is being used at any given time.

4.9.1 Clock switching details

The sampling of the 13 MHz clock on 13' MHz edges will check if the 13 MHz clock is low on a rising 13' MHz clock edge. This means that the 13 MHz is in the correct phase. The switching point should be shortly after the 13 MHz clock has slipped behind the 13' MHz clock. First the 10 bit counter with compare will count a number of samples with the wrong phase before outputting a high to the AND gate. On the first sample with correct phase after this, the clock will be switched.

4.10 Clock usage in peripheral blocks

Peripheral blocks use one or more of the clocks produced by the clock generation block. Many peripherals use one clock for the bus interface to the CPU, and another clock for the peripheral function itself. In the case of the USB block, the USB function is operated from the special 48 MHz clock generated by the USB PLL, while the bus interface to the CPU operates from HCLK. The USB clock uses a third clock (PERIPH_CLK) to operate an I^2C-bus interface whose purpose is to communicate with an external USB transceiver.

Table 14 shows clocking for LPC32x0 peripheral functions.

Table 14. Clocks used by various peripheral blocks

Peripheral	AHB Matrix Location	Bus Clock Source	Function Clock Source
DMA controller	Master	HCLK	HCLK
Ethernet MAC	Master	HCLK	HCLK
USB controller	Master	HCLK	clk48mhz, except I^2C clocked by PERIPH_CLK
LCD controller	Master	HCLK	HCLK; external LCDCLKIN (pin)
MLC NAND Flash controller	AHB slave	HCLK	HCLK
SLC NAND Flash interface	AHB slave	HCLK	HCLK
IRAM	AHB slave	HCLK	HCLK
IROM	AHB slave	HCLK	HCLK
External Memory Controller	AHB slave	HCLK	PERIPH_CLK; DDRAM_CLK; RTC_TICK
SD card interface	APB	HCLK	ARM_CLK (divide by N)
I2S0 and I2S1	APB	HCLK	HCLK; HCLK (fractional divider)

Table 14. Clocks used by various peripheral blocks *…continued*

Peripheral	AHB Matrix Location	Bus Clock Source	Function Clock Source
SPI1 and SPI2	APB	HCLK	HCLK
SSP0 and SSP1	APB	HCLK	HCLK
I2C1 and I2C2	APB	HCLK	HCLK
MCPWM	APB	HCLK	HCLK
Standard UARTs	APB	HCLK	PERIPH_CLK or HCLK (selectable)
System Control Functions	FAB	HCLK	HCLK
GPIO	FAB	HCLK	PERIPH_CLK
Interrupt controllers	FAB	HCLK	PERIPH_CLK
Timer[0,1,2,3]	FAB	HCLK	PERIPH_CLK
14-clock UARTs	FAB	HCLK	PERIPH_CLK; PERIPH_CLK (divide by N)
High speed timer	FAB	HCLK	PERIPH_CLK
Watchdog timer	FAB	HCLK	PERIPH_CLK
Keyboard Scan	FAB	HCLK	32 kHz RTC_CLK
Millisecond timer	FAB	HCLK	32 kHz RTC_CLK
PWM1 and PWM2	FAB	HCLK	PERIPH_CLK; PWM clocked by 32 kHz RTC_CLK or PERIPH_CLK (selectable)
ADC / Touch screen	FAB	HCLK	32 kHz RTC_CLK; PERIPH_CLK;
RTC	FAB	HCLK	32 kHz RTC_CLK
Debug	FAB	HCLK	ARM_CLK, JTAG_TCK
ETB	AHB	HCLK	ARM_CLK

4.11 Register description

Table 15 shows the LPC32x0 clocking and power control registers.

Table 15. Clocking and power control registers

Address	Name	Description	Reset State	Access
0x4000 4044	PWR_CTRL	AHB/ARM power control register	0x0000 0012	R/W
0x4000 404C	OSC_CTRL	Main oscillator control register	0x0000 0100	R/W
0x4000 4050	SYSCLK_CTRL	SYSCLK control register	0x0000 0B48	R/W
0x4000 4048	PLL397_CTRL	PLL397 PLL control register	0	R/W
0x4000 4058	HCLKPLL_CTRL	ARM and HCLK PLL control register	0	R/W
0x4000 4040	HCLKDIV_CTRL	HCLK divider settings	0	R/W
0x4000 40A4	TEST_CLK	Clock testing control	0	R/W
0x4000 40EC	AUTOCLK_CTRL	Auto clock control register	0	R/W
0x4000 4030	START_ER_PIN	Start Enable register - pin sources	0	R/W
0x4000 4020	START_ER_INT	Start Enable register - internal sources	0	R/W
0x4000 4018	P0_INTR_ER	Start and Interrupt Enable register; p0 & p1 sources	0	R/W
0x4000 4038	START_SR_PIN	Start status register, pin sources	0	R/-
0x4000 4028	START_SR_INT	Start status register, internal sources	0	R/-

Table 15. Clocking and power control registers *…continued*

Address	Name	Description	Reset State	Access
0x4000 4034	START_RSR_PIN	Start Raw status register, pin sources	0	R/W
0x4000 4024	START_RSR_INT	Start Raw status register, internal sources	0	R/W
0x4000 403C	START_APR_PIN	Start activation Polarity register, pin sources	0	R/W
0x4000 402C	START_APR_INT	Start activation Polarity register, internal sources	0	R/W
0x4000 4064	USB_CTRL	USB PLL and pad control register	0x0008 0000	R/W
0x4000 401C	USBDIV_CTRL	USB PLL pre-divider settings	0x0000 000C	R/W
0x4000 4080	MS_CTRL	SD Card interface clock and pad control	0	R/W
0x4000 40E8	DMACLK_CTRL	DMA clock control register	0x0000 0001	R/W
0x4000 40C8	FLASHCLK_CTRL	Flash clock control	0x0000 0003	R/W
0x4000 4090	MACCLK_CTRL	Ethernet MAC clock control	0	R/W
0x4000 4054	LCDCLK_CTRL	LCD clock control	0	R/W
0x4000 407C	I2S_CTRL	I^2S control register	0	R/W
0x4000 4078	SSP_CTRL	SSP0 and SSP1 clock control	0	R/W
0x4000 40C4	SPI_CTRL	SPI1 and SPI2 clock and pin control	0	R/W
0x4000 40AC	I2CCLK_CTRL	I^2C clock control register	0	R/W
0x4000 40C0	TIMCLK_CTRL1	Timer[5:0] and MCPWM clock control	0	R/W
0x4000 40BC	TIMCLK_CTRL	Timer clock control	0	R/W
0x4000 40B4	ADCLK_CTRL	ADC clock control	0	R/W
0x4000 4060	ADCLK_CTRL1	Second ADC clock control register	0	R/W
0x4000 40B0	KEYCLK_CTRL	Keypad clock control	0	R/W
0x4000 40B8	PWMCLK_CTRL	PWM clock control	0	R/W
0x4000 40E4	UARTCLK_CTRL	General UART clock control register	0x0000 000F	R/W
0x4000 4110	POS0_IRAM_CTRL	Internal Memory power off control register 0	0	R/W
0x4000 4114	POS1_IRAM_CTRL	Internal Memory power off control register 1	0	R/W

4.11.1 Power Control register (PWR_CTRL - 0x4000 4044)

The PWR_CTRL register contains controls for general power related functions.

Table 16. Power Control register (PWR_CTRL - 0x4000 4044)

Bit	Function	Reset value
31:11	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
10	Force HCLK and ARMCLK to run from PERIPH_CLK in order to save power. 0 = Normal mode. 1 = ARM and AHB matrix (HCLK) runs with PERIPH_CLK frequency.	0
9	EMCSREFREQ value. EMCSREFREQ is used by the SDRAM interface, refer to the External Memory Controller chapter for details. This value is not reflected on EMCSREFREQ before either PWR_CTRL[8] is changed from 0 to 1 or the Start Controller brings the system out of STOP mode. 0 = No SDRAM self refresh. 1 = SDRAM self refresh request.	0

Table 16. Power Control register (PWR_CTRL - 0x4000 4044) *…continued*

Bit	Function	Reset value
8	Update EMCSREFREQ (SDRAM self refresh request). 0 = No action. 1 = Update EMCSREFREQ according to PWR_CTRL[9]. Software must clear this bit again.	0
7	SDRAM auto exit self refresh enable. If enabled, the SDRAM will automatically exit self refresh mode when the CPU exits STOP mode. Note: software must always clear this bit after exiting from STOP mode. 0 = Disable auto exit self refresh. 1 = Enable auto exit self refresh.	0
6	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
5	HIGHCORE pin level. Allows the HIGHCORE pin to be used as a GPO if bit 1 in this register is written with a 1. 0 = HIGHCORE will drive low. 1 = HIGHCORE will drive high.	0
4	SYSCLKEN pin level. Can be used if using SYSCLK_EN pin as GPO. Bit 3 in this register should be set to 1 when using the pin as GPO. 0 = SYSCLKEN will drive low. 1 = SYSCLKEN will drive high.	1
3	SYSCLKEN pin drives high when an external input clock on SYSXIN is requested. The pin is in high impedance mode when no external clock is needed. 0 = SYSCLKEN will drive high when not in STOP mode and 3-state in STOP mode. 1 = SYSCLKEN will always drive the level specified by bit 4.	0

Table 16. Power Control register (PWR_CTRL - 0x4000 4044) *…continued*

Bit	Function	Reset value
2	RUN mode control. In Direct RUN mode the ARM, HCLK is clocked directly from the SYSCLK mux. This is the default setting. After the PLL outputs a stable clock, writing a 1 to this register will switch all the above clock sources to the PLL clock or divided versions of the PLL clock. Note: the HCLK PLL clock frequency must be higher than SYSCLK frequency. 0 = Direct RUN mode. 1 = Normal RUN mode. ARM, HCLK is sourced from the PLL output.	0
1	Core voltage supply level signalling control. The output pin HIGHCORE is defined to indicate nominal Core voltage when low and a lowered core voltage when driving high. 0 = HIGHCORE pin will drive high during STOP mode and drive low in all other modes. 1 = HIGHCORE pin is always driving the level as specified in bit 5.	1
0	STOP mode control register. In STOP mode the two clock sources to the AHB/ARM clock mux is stopped. This means that the ARM, the ARM-PLL, and HCLK clocks are stopped. The USB clock is not stopped automatically by the STOP mode hardware. The USB clock may be left running or stopped by software while the system is in STOP mode. Read: 0 = The Device is not in STOP mode. 1 = An active start event has occurred after this bit has been written to a 1, but before STOP mode has actually been entered by the hardware. Software must restore this bit to 0 immediately after exiting STOP mode. Write: 0 = Restore value to 0 if STOP was never entered. 1 = Instruct hardware to enter STOP mode.	0

4.11.2 Main Oscillator Control register (OSC_CTRL - 0x4000 404C)

The OSC_CTRL register controls the operation of the main crystal oscillator.

Table 17. Main Oscillator Control register (OSC_CTRL - 0x4000 404C)

Bit	Function	Reset value
31:9	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8:2	0000000 = Don't add any load capacitance to SYSX_IN and SYSX_OUT. xxxxxxx = Add (xxxxxxx binary × 0.1) pF load capacitance to SYSX_IN and SYSX_OUT. 1000000 = Default setting of 6.4 pF added. In total 12.7 pF (nominal value) can be added to the external load capacitors. Capacitor value on the two pins is always programmed equal. Any difference must be on the external capacitors.	1000000
1	Main oscillator test mode. In test mode the oscillator will not oscillate but pass the external clock supplied at osc_in as the output clock. In typical applications, this bit should be left at the default value. Bit 0 in this register must be 1 (main ocillator disabled) for test mode to work. 0 = Normal to. Either oscillation mode or power down mode. 1 = Test mode.	0
0	Main oscillator enable. 0 = Main oscillator is enabled. 1 = Main oscillator is disabled and in power down mode.	0

4.11.3 SYSCLK Control register (SYSCLK_CTRL - 0x4000 4050)

The SYSCLK_CTRL register controls switching SYSCLK between the main oscillator and PLL397.

Table 18. SYSCLK Control Register (SYSCLK_CTRL - 0x4000 4050)

Bit	Function	Reset value
31:12	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
11:2	The number in this register is used by the clock switching circuitry to decide how long a bad phase must be present before the clock switching is triggered. This register must always be written with a value before the clock switch is used in phase detect mode. The recommended value is 0x50, max value is 0xA9. (Higher values may result in no switching at all)	0x2D2
1	A write access to this bit triggers switching between the 13' MHz clock source and the Main oscillator. Write: 0 = Switch to Main oscillator. 1 = Switch to 13' MHz clock source (PLL397 output). Read: Returns the last written value.	0
0	SYSCLK MUX status Read only: 0 = Main oscillator selected as the clock source. (Default after external reset, not reset by watchdog reset) 1 = 13' MHz PLL397 output selected as the clock source.	0

4.11.4 PLL397 Control register (PLL397_CTRL - 0x4000 4048)

The PLL397_CTRL register controls the 397x PLL that runs from the RTC clock. The output of this PLL can be selected as the source for SYSCLK.

Table 19. PLL397 Control register (PLL397_CTRL - 0x4000 4048)

Bit	Function	Reset value
31:11	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
10	PLL MSLOCK status (Read only) This is a backup lock signal only to be used if the main lock signal in bit 0 is not functional. This lock signal comes from a mixed signal lock detect circuit. 0 = PLL is not locked. 1 = PLL is locked. This means that the PLL output clock is stable.	0
9	PLL397 bypass control. For test only. 0 = No bypass. 1 = Bypass. PLL is bypassed and output clock is the input clock.	0
8:6	PLL397 charge pump bias control. Note that −12.5 % of resistance means +12.5 % of the current. 000 = Normal bias setting. 001 = −12.5 % of resistance. 010 = −25 % of resistance. 011 = −37.5 % of resistance. 100 = +12.5 % of resistance. 101 = +25 % of resistance. 110 = +37.5 % of resistance. 111 = +50 % of resistance.	0
5:2	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
1	PLL397 operational control. Generally, most of the LPC32x0, including the PLLs, will run from the main oscillator. In this case the PLL397 should be stopped to save power. However, it is possible to use the 13’ MHz clock from PLL397 instead. Upon reset, PLL397 is started by default, but it is the main oscillator clock that is used by the system. Note that after power-up or being turned on by software, PLL397 needs time to stabilize and the PLL lock status must go active before the output clock is used. Software can switch over to the PLL397 clock when it is locked. 0 = PLL397 is running. 1 = PLL397 is stopped and is in low power mode.	0
0	PLL LOCK status (Read only) 0 = PLL is not locked. 1 = PLL is locked. This means that the PLL output clock is stable.	0

4.11.5 HCLK PLL Control register (HCLKPLL_CTRL - 0x4000 4058)

The HCLKPLL_CTRL register controls the settings of HCLK PLL (see Figure 6) that supplies the base clock that is normally used for the ARM CPU clock, the AHB HCLK, and the DDR SDRAM clock. It can also be used as the basis for PERIPH_CLK. The input clock to the PLL is SYSCLK. The output can be in the range of 26 MHz to 266 MHz.

Table 20. HCLK PLL Control register (HCLKPLL_CTRL - 0x4000 4058)

Bit	Function	Reset value
31:17	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
16	PLL Power down. This bit is used to start/stop the PLL. Startup time must be respected from when the PLL is started until the output clock is used. Startup time is indicated by PLL LOCK going high. 0 = PLL is in power down mode. 1 = PLL is in operating mode.	0
15	Bypass control 0 = CCO clock is sent to post divider. 1 = PLL input clock bypasses the CCO and is sent directly to the post divider.	0
14	Direct output control 0 = The output of the post-divider is used as output of the PLL 1 = CCO clock is the direct output of the PLL, bypassing the post divider	0
13	Feedback divider path control. 0 = Feedback divider clocked by CCO clock. 1 = Feedback divider clocked by FCLKOUT.	0
12:11	PLL post-divider (P) setting. This divider divides down the output frequency. If 50 % duty cycle is needed, the post-divider should always be active. 00 = divide by 2 (P=1) 01 = divide by 4 (P=2) 10 = divide by 8 (P=4) 11 = divide by 16 (P=8)	0
10:9	PLL pre-divider (N) setting. This divider divides down the input frequency before going to the phase comparator. 00 = 1 01 = 2 10 = 3 11 = 4	0
8:1	PLL feedback divider (M) setting. This divider divides down the output frequency before being fed back to the phase comparator. 00000000 = 1 00000001 = 2 : : 11111110 = 255 11111111 = 256	0
0	PLL LOCK status (Read only) 0 = PLL is not locked. 1 = PLL is locked. This means that the PLL output clock is stable.	0

4.11.6 HCLK Divider Control register (HCLKDIV_CTRL - 0x4000 4040)

The HCLKDIV_CTRL register controls the division factor for some of the clocks that may be based on the HLCK PLL output clock. These clocks are PERIPH_CLK, HCLK, and DDRAM_CLK.

Table 21. HCLK Divider Control register (HCLKDIV_CTRL - 0x4000 4040)

Bit	Function	Reset value
31:8	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8:7	DDRAM_CLK control. Note that the clock architecture does not support using DDR SDRAM in Direct RUN mode. DDR SDRAM can only be accessed when in RUN mode and ARM runs twice or 4 times HCLK frequency. 00 = DDRAM clock stopped. Use this setting if external SDR SDRAM is used. 01 = DDRAM nominal speed. DDRAM clock is same speed as ARM. Software needs to make sure that HCLK is half of this frequency. This is the normal setting for DDRAM. 10 = DDRAM half speed. DDRAM clock is half the frequency of ARM clock. Can be used if ARM runs 4 times HCLK frequency. 11 = Not used.	0
6:2	PERIPH_CLK divider control. PERIPH_CLK is the clock going to APB/FAB slaves. This setting may be programmed once after power up and may not be changed afterwards. This setting does not affect PERIPH_CLK frequency in Direct RUN mode. 00000 = PERIPH_CLK is ARM PLL clock in RUN mode. 00001 = PERIPH_CLK is ARM PLL clock divided by 2 in RUN mode. …… 11110 = PERIPH_CLK is ARM PLL clock divided by 31 in RUN mode. 11111 = PERIPH_CLK is ARM PLL clock divided by 32 in RUN mode.	0
1:0	HCLK divider control. This setting may typically be programmed once after power up and not changed afterwards. This setting do not affect HCLK frequency in Direct RUN mode. HCLK must not be set to a frequency higher than 133 MHz. 00 = HCLK is ARM PLL clock in RUN mode. 01 = HCLK is ARM PLL clock divided by 2 in RUN mode. 10 = HCLK is ARM PLL clock divided by 4 in RUN mode. 11 = Not used.	0

4.11.7 Test Clock Selection register (TEST_CLK - 0x4000 40A4)

For testing purposes, selected internal clocks may be output on the GPO_0 / TST_CLK1 pin or the TST_CLK2 pin. For TST_CLK1, the clocks that may be output are PERIPH_CLK, RTC_CLK, or OSC_CLK. For TST_CLK2, the clocks that may be output are HCLK, PERIPH_CLK, the USB 48 MHz clock, OSC_CLK, or the output of PLL397x.

The TEST_CLK register enables the clock output function and selects the clock that will be output on GPO_0 / TST_CLK1 or TST_CLK2 pins.

Table 22. Test Clock Selection register (TEST_CLK - 0x4000 40A4)

Bit	Function	Reset value
31:7	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
6:5	The selected clock is output on GPO_0 / TEST_CLK1 pin if bit 4 of this register contains a 1. 00 = PERIPH_CLK. This clock stops in STOP mode. 01 = RTC clock, un-synchronized version. Available in STOP mode also (32.768 kHz) 10 = Main oscillator clock. Available in STOP mode as long as the main oscillator is enabled. 11 = Not used.	0
4	0 = GPO_0 / TST_CLK1 output is connected to the GPIO block. 1 = GPO_0 / TST_CLK1 output is the clock selected by register bits [6:5].	0
3:1	The selected clock is output on the TST_CLK2 pin if bit 0 of this register contains a 1. 000 = HCLK. 001 = PERIPH_CLK. 010 = USB clock (48 MHz output from USB PLL). 011 = reserved. 100 = reserved. 101 = Main oscillator clock. Available in STOP mode as long as the main oscillator is enabled. 110 = reserved. 111 = PLL397 output clock (13.008896 MHz).	0
0	0 = TST_CLK2 is turned off 1 = TST_CLK2 outputs the clock selected by register bits [3:1]	0

4.11.8 Autoclock Control register (AUTOCLK_CTRL - 0x4000 40EC)

For power saving purposes, a number of functional blocks default to have their clocks automatically turned off if they have been inactive for a predetermined amount of time. This feature can be disabled on a block-by-block basis by settings bits in the AUTOCLK_CTRL register.

Table 23. Autoclock Control register (AUTOCLK_CTRL - 0x4000 40EC)

Bit	Function	Reset value
31:7	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
6	0 = Autoclock enabled on USB Slave HCLK. Stops clocking after 128 HCLK of inactivity. There is one clock additional latency to access the USB block if the clock has been stopped. 1 = Always clocked.	0

Table 23. Autoclock Control register (AUTOCLK_CTRL - 0x4000 40EC)

Bit	Function	Reset value
5:2	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
1	0 = Autoclock enabled on IRAM. Stops clocking after 16 HCLKs of inactivity. There is one clock additional latency to access the IRAM if the clock has been stopped. 1 = Always clocked.	0
0	0 = Autoclock enabled on IROM. Stops clocking after 8 HCLKs of inactivity. There is one clock additional latency to access the IROM if the clock has been stopped. 1 = Always clocked.	0

4.11.9 Start Enable register for Pin Sources (START_ER_PIN - 0x4000 4030)

The START_ER_PIN register allows individually enabling device pins to start up the chip from STOP mode.

Table 24. Start Enable register for Pin Sources (START_ER_PIN - 0x4000 4030)

Bit	Function	Reset value
31	U7_RX	0
30	U7_HCTS	0
29	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
28	U6_IRRX	0
27	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
26	U5_RX / USB_DAT_VP	0
25	GPI_28	0
24	U3_RX	0
23	U2_HCTS	0
22	U2_RX	0
21	U1_RX	0
20:19	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
18	SDIO_INT_N (MS_DIO[1] pin)	0
17	MSDIO_START. Logical OR of MS_DIO[3:0]	0
16	GPI_6 / HSTIM_CAP	0
15	GPI_5	0
14	GPI_4	0
13	GPI_3	0
12	GPI_2	0
11	GPI_1 / SERVICE_N	0
10	GPI_0 / I2S1RX_SDA	0
9	SYSCLKEN pin	0
8	SPI1_DATIN	0
7	GPI_7	0

Table 24. Start Enable register for Pin Sources (START_ER_PIN - 0x4000 4030) *...continued*

Bit	Function	Reset value
6	SPI2_DATIN	0
5	GPI_19 / U4_RX	0
4	GPI_9	0
3	GPI_8	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.10 Start Enable register for Internal Sources (START_ER_INT - 0x4000 4020)

The START_ER_INT register allows individually enabling internal interrupt sources to start up the chip from STOP mode. It is used in conjunction with the START_ER_PIN, START_RSR_INT, START_RSR_PIN, START_SR_INT, START_SR_PIN, START_APR_INT, and START_APR_PIN registers to control startup from STOP mode. Refer to the Start Controller description in this chapter for more information.

Table 25. Start Enable register for Internal Sources (START_ER_INT - 0x4000 4020)

Bit	Function	Reset value
31	TS_INT (AD_IRQ) Touchscreen/ADC interrupt.	0
30	TS_P Touch screen interface press detect, note: active low	0
29	TS_AUX Touch screen interface auxiliary interrupt	0
28:27	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
26	USB_AHB_NEED_CLK	0
25	MSTIMER_INT	0
24	RTC_INT Interrupt from RTC	0
23	USB_NEED_CLK	0
22	USB_INT	0
21	USB_I2C_INT	0
20	USB_OTG_TIMER_INT	0
19	USB_OTG_ATX_INT_N	0
18:17	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
16	KEY_IRQ: Keyboard scanner interrupt signal	0
15:8	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
7	Ethernet MAC Start request.	0
6	Enable Port 0 and Port 1 start request. This enables start request for all P0_[7:0] and P1_[23:0], signals, which are ORed together. See P0_INTR_ER register for additional details	0
5	GPIO_5. See GPIO_0.	0
4	GPIO_4. See GPIO_0.	0
3	GPIO_3. See GPIO_0.	0

Table 25. Start Enable register for Internal Sources (START_ER_INT - 0x4000 4020)

Bit	Function	Reset value
2	GPIO_2. See GPIO_0.	0
1	GPIO_1. See GPIO_0.	0
0	GPIO_0. 0 = Start signal is disabled. 1 = Start signal is enabled.	0

4.11.11 Port 0 and Port 1 start and interrupt enable register (P0_INTR_ER - 0x4000 4018)

The P0_INTR_ER register enables or disables the ORed start and interrupt signals associated with Ports 0 and 1. Table 26 shows the bit assignment of the P0_INTR_ER control register.

The P0_INTR_ER register allows individually enabling Port 0 and Port 1 I/O pins as interrupt sources and to start up the chip from STOP mode. It is used in conjunction with the START_ER_PIN, START_RSR_INT, START_RSR_PIN, START_SR_INT, START_SR_PIN, START_APR_INT, and START_APR_PIN registers to control startup from STOP mode. Refer to the Start Controller description in this chapter for more information.

Note: There is only a single signal that feeds into bit 6 of the START registers, this means that there is only single bit control in the START_ER_PIN, START_RSR_INT, START_RSR_PIN, START_SR_INT, START_SR_PIN, START_APR_INT, and START_APR_PIN registers for all P0 and P1 signals.

Table 26. Port 0 and 1 interrupt and start register control (P0_INTR_ER - 0x4000 4018)

P0_INTR_ER	Function	Description
31:11	P1.[23:3]	see p0.0 description
10	P1.2	see p0.0 description
9	P1.1	see p0.0 description
8	P1.0	see p0.0 description
7	P0.7	see p0.0 description
6	P0.6	see p0.0 description
5	P0.5	see p0.0 description
4	P0.4	see p0.0 description
3	P0.3	see p0.0 description
2	P0.2	see p0.0 description
1	P0.1	see p0.0 description
0	P0.0	1 = enables both interrupt and start for this pin 0 = disables both interrupt and start for this pin

4.11.12 Start Status Register for Pin Sources (START_SR_PIN - 0x4000 4038)

The START_SR_PIN shows the current state of possible device pin startup sources, after masking by START_ER_PIN.

Table 27. Start Status Register for Pin Sources (START_SR_PIN - 0x4000 4038)

Bit	Function	Reset value
31:3	Same sources as for the START_ER_PIN register. Unused bits in this register read as 0. This allows the ARM to use the “find first bit set” CLZ instruction. Read: 0 = Pin or signal is inactive after masking. 1 = Pin or signal is active after masking.	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.13 Start Status Register for Internal Sources (START_SR_INT - 0x4000 4028)

The START_SR_INT shows the current state of possible internal startup sources, after masking by START_ER_INT.

Table 28. Start Status Register for Internal Sources (START_SR_INT - 0x4000 4028)

Bit	Function	Reset value
31:3	Same sources as for the START_ER_INT register. Unused bits in this register read as 0. This allows the ARM to use the “find first bit set” CLZ instruction. Read: 0 = Pin or signal is inactive after masking. 1 = Pin or signal is active after masking.	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.14 Start Raw Status Register for Pin Sources (START_RSR_PIN - 0x4000 4034)

The START_RSR_PIN shows the current state of possible device pin startup sources, prior to masking.

Table 29. Start Raw Status Register for Pin Sources (START_RSR_PIN - 0x4000 4034)

Bit	Function	Reset value
31:3	Same sources as for the START_ER_PIN register. Read: 0 = Pin or signal is inactive before masking. 1 = Pin or signal is active before masking. Write: 0 = No effect. 1 = The captured state is cleared. Each source can be individually cleared.	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.15 Start Raw Status Register for Internal Sources (START_RSR_INT - 0x4000 4024)

The START_RSR_INT shows the current state of possible internal startup sources, prior to masking.

Table 30. Start Raw Status Register for Internal Sources (START_RSR_INT - 0x4000 4024)

Bit	Function	Reset value
31:3	Same sources as for the START_ER_INT register. Read: 0 = Pin or signal is inactive before masking. 1 = Pin or signal is active before masking. Write: 0 = No effect. 1 = The captured state is cleared. Each source can be individually cleared.	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.16 Start Activation Polarity Register for Pin Sources (START_APR_PIN - 0x4000 403C)

The START_APR_PIN allows selecting the polarity that device pin start signal sources use as a start condition.

Table 31. Start Activation Polarity Register for Pin Sources (START_APR_PIN - 0x4000 403C)

Bit	Function	Reset value
31:3	Same sources as for the START_ER_INT register. 0 = Active state is captured on falling edge of start signal. 1 = Active state is captured on rising edge of start signal.	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.17 Start Activation Polarity Register for Internal Sources (START_APR_INT - 0x4000 402C)

The START_APR_INT allows selecting the polarity that internal start signal sources use as a start condition.

Table 32. Start Activation Polarity Register for Internal Sources (START_APR_INT - 0x4000 402C)

Bit	Function	Reset value
31:3	Same sources as for the START_ER_INT register. 0 = Active state is captured on falling edge of start signal. 1 = Active state is captured on rising edge of start signal.	0
2:0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

4.11.18 USB Control register (USB_CTRL - 0x4000 4064)

The USB_CTRL register provides control of the USB clocks, PLL, and pads.

Table 33. USB Control register (USB_CTRL - 0x4000 4064)

Bit	Function	Reset value
31:25	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
24	USB Slave HCLK control. 0 = Slave HCLK disabled. 1 = Slave HCLK enabled.	0
23	usb_i2c_enable. Control signal for mux. the mux drives a "0" out on USB_OE_TP_N when set. This enables “transparent I^2C mode” for communication with an external USB transceiver. 0 = ip_3506_otg_tx_en_n is fed to OE_TP_N pad. 1 = ’0’ is fed to OE_TP_N pad.	0
22	usb_dev_need_clk_en. During initialization the usb_dev_need_clk should not be fed to the clock switch. After initializing the external USB transceiver, this bit should be programmed to "1". Note that setting this bit to "0" also disables the software request in OTG_CLOCK_CONTROL register. 0 = usb_dev_need_clk is not let into the clock switch. 1 = usb_dev_need_clk is let into clock switch.	0
21	usb_host_need_clk_en. During initialization the usb_host_need_clk_en should not be fed to the clock switch. After initializing the external USB transceiver, this bit should be programmed to "1". Note that setting this bit to "0" also disables the software request in OTG_CLOCK_CONTROL register. 0 = usb_host_need_clk_en is not let into the clock switch. 1 = usb_host_need_clk_en is let into clock switch.	0
20:19	Pad control for USB_DAT_VP and USB_SE0_VM pads. 00 = Pull-up added to pad. 01 = Bus keeper. Retains the last driven value. 10 = No added function. 11 = Pull-down added to pad.	01
18	USB_Clken2 clock control. This bit must be written to a 1 after the PLL indicates stable output clock. 0 = Stop clock going into USB block. 1 = Enable clock going into USB block.	0
17	USB_Clken1 clock control. This bit should be written to a 0 when USB is not active. 0 =Stop clock going into the USB PLL. 1 = Enable clock going into the USB PLL.	0
16	PLL Power down. This bit is used to start/stop the PLL. Startup time must be respected from when the PLL is started until the output clock is used. Startup time is indicated by PLL LOCK going high. 0 = PLL is in power down mode. 1 = PLL is in operating mode.	0
15	Bypass control. 0 = CCO clock is sent to post divider. 1 = PLL input clock bypasses the CCO and is sent directly to the post divider.	0

Table 33. USB Control register (USB_CTRL - 0x4000 4064) *…continued*

Bit	Function	Reset value
14	Direct output control. 0 = The output of the post-divider is used as output of the PLL. 1 = CCO clock is the direct output of the PLL, bypassing the post divider.	0
13	Feedback divider path control. 0 = Feedback divider clocked by CCO clock. 1 = Feedback divider clocked by post FCLKOUT.	0
12:11	PLL post-divider (P) setting. This divider divides down the output frequency. If 50 % duty cycle is needed, the post-divider should always be active. 00 = divide by 2 (P=1) 01 = divide by 4 (P=2) 10 = divide by 8 (P=4) 11 = divide by 16 (P=8)	0
10:9	PLL pre-divider (N) setting. This divider divides down the input frequency before going to the phase comparator. 00 = 1 01 = 2 10 = 3 11 = 4	0
8:1	PLL feedback divider (M) setting. This divider divides down the output frequency before being fed back to the phase comparator. Note: Remember that there is a pre-divide which defaults to 13 in front of this PLL. 00000000 = 1 00000001 = 2 …… 11111110 = 255 11111111 = 256	0
0	PLL LOCK status (Read only, write is don't care) 0 = PLL not locked. 1 = PLL locked. This means that the PLL output clock is stable.	0

4.11.19 USB clock pre-divide register (USB_DIV - 0x4000 401C)

The USB_DIV register controls the clock input to the USB PLL. Table 34 shows the bit assignment of the USB_DIV control register.

Table 34. USB prettified register (USB_DIV - 0x4000 401C)

USB_DIV	Function	Description	Reset value
31:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3:0	USB_RATE	Controls the USB pre-clock divider, setting the input rate to the USB. The USB input rate = OSC_CLK / (USB_RATE+1)	0x0C

4.11.20 Memory Card Control register (MS_CTRL - 0x4000 4080)

The MS_CTRL register selects whether the SD card interface is enabled. It also controls pad pull-up and pull-down and clocks to the related peripheral blocks.

Table 35. Memory Card Control register (MS_CTRL - 0x4000 4080)

Bit	Function	Reset value
31:11	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
10	Disable SD_card pins. If the SD Card interface is not used, this bit (10) and bit 9 should be programmed to 0, and bits 6 through 8 should be programmed to 1. 0 = enable SD_card interface pins. 1 = disable SD_card interface pins and enable Peripheral MUX registers see Table 650.	0
9	Enables clock and pull-ups to MSSDIO pins. If the SD Card interface is not used, this bit should be programmed to 0, and bits 6 through 8 should be programmed to 1. To enable clocking, the SD_card controller requires both bits 5 and 9 be set to 1. 0 = MSSDIO pull-up and clock disabled. 1 = MSSDIO pull-up and clock enable.	0
8	MSSDIO2 and MSSDIO3 pad control. 0 = MSSDIO2 and 3 pad has pull-up enabled. 1 = MSSDIO2 and 3 pad has no pull-up.	0
7	MSSDIO1 pad control. 0 = MSSDIO1 pad has pull-up enabled. 1 = MSSDIO1 pad has no pull-up.	0
6	MSSDIO0/MSBS pad control. 0 = MSSDIO0 pad has pull-up enable. 1 = MSSDIO0 pad has no pull-up.	0
5	SD Card clock control. This bit controls MSSDCLK to the SD Card block. The registers in the peripheral block cannot be accessed if the clock is stopped. To enable clocking, the SD_card controller requires both bits 5 and 9 be set to 1. 0 = Clocks disabled. 1 = Clocks enabled.	0
4	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3:0	These register bits control the divider ratio when generating the clock from the ARM PLL output clock. Software must insure that the maximum clock frequency of the targeted device is not exceeded. 0000 = MSSDCLK stopped. Divider in low power mode. 0001 = MSSDCLK equals ARM PLL output clock divided by 1. …… 1110 = MSSDCLK equals ARM PLL output clock divided by 14. 1111 = MSSDCLK equals ARM PLL output clock divided by 15.	0

4.11.21 DMA Clock Control register (DMACLK_CTRL - 0x4000 40E8)

The DMACLK_CTRL register allows disabling the clock to the DMA controller in order to save power if the DMA controller is not being used.

Table 36. DMA Clock Control register (DMACLK_CTRL - 0x4000 40E8)

Bit	Function	Reset value
31:1	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
0	0 = All clocks to DMA stopped. No accesses to DMA registers are allowed. 1 = All clocks to DMA enabled.	1

4.11.22 NAND Flash Clock Control register (FLASHCLK_CTRL - 0x4000 40C8)

The LPC32x0 incorporates two NAND Flash controllers, one for single-level NAND Flash (the SLC Flash controller), and one for multi-level NAND Flash (the MLC Flash controller). The FLASHCLK_CTRL register controls some aspects of the two NAND Flash memory interfaces: enabling and disabling clocks; selecting one of the controllers to be used; and controlling interrupts and DMA.

Table 37. NAND Flash Clock Control register (FLASHCLK_CTRL - 0x4000 40C8)

Bit	Function	Reset value
31:6	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
5	Determines which NAND Flash controller interrupt is connected to the interrupt controller. 0: enable the SLC (single level) NAND Flash controller interrupt. 1: enable the MLC (multi-level) NAND Flash controller interrupt.	0
4	Enable NAND_DMA_REQ on NAND_RnB. This applies only to the MLC. 0: disable 1: enable	0
3	Enable NAND_DMA_REQ on NAND_INT. This applies only to the MLC. 0: disable 1: enable	0
2	SLC/MLC select. Selects either the single-level (SLC), or multi-level (MLC) NAND Flash controller. 0: Select MLC flash controller. 1: Select SLC flash controller.	0
1	MLC NAND Flash clock enable control. 0: Disable clocks to the block, including the AHB interface. 1: Enable clock.	1
0	SLC NAND Flash clock enable control. 0: Disable clocks to the block, including the AHB interface. 1: Enable clock.	1

4.11.23 Ethernet MAC Clock Control register (MACCLK_CTRL - 0x4000 4090)

The MAC_CLK_CTRL register controls three clocks which are connected to the Ethernet MAC. To enable the ethernet MAC block for proper operation set the Master, Slave and Control bits in this register to 1. To disable the Ethernet MAC set all three of the bits to 0.

See Table 643 and associated text for information using the MII and RMII interface described in bits 3 and 4.

Table 38. Ethernet MAC Clock Control Register (MAC_CLK_CTRL 0x4000 4090)

MAC_CLK_CTRL	Name	Description	Reset value
31:11	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
4:3	HDW_INF_CTRL	Ethernet MAC Hardware interface control 00 = Do not connect Ethernet MAC to Port pins. 01 = Connect Ethernet MAC to Port pins in MII Mode. 10 = Do not connect Ethernet MAC to Port pins. 11 = Connect Ethernet MAC to Port pins in RMII Mode.	00
2	MASTER_CLK	Master Interface Clock 0 = Disabled. 1 = Enabled. note: For proper operation, bits 0, 1, and 2 in this register must be set to 1 to fully enable the Ethernet MAC.	0
1	SLAVE_CLK	Slave Interface Clock 0 = disabled 1 = enabled note: For proper operation, bits 0, 1, and 2 in this register must be set to 1 to fully enable the Ethernet MAC.	0
0	REG_CLK	Control Registers Clock 0 = disabled. 1 = enabled note: For proper operation, bits 0, 1, and 2 in this register must be set to 1 to fully enable the Ethernet MAC.	0

4.11.24 LCD Clock Control register (LCDCLK_CTRL - 0x4000 4054)

The LCD_CFG register controls the selection of output pins needed for different LCD panel configurations, as well as prescaling of the clock used for LCD data generation.

The contents of the LCD_CFG register are described in Table 39.

Table 39. LCD Configuration register (LCD_CFG, RW - 0x4000 4054)

Bits	Name	Function	Reset value
31:9	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8	DISPLAY_TYPE	Sets the Display type: 0 = TFT Display 1 = STN Display	0x0
7:6	MODE_SELECT	Selects output pin group. See Table 199 and Table 200	0x0
5	HCLK_ENABLE	Enables HCLK signal to LCD controller	0x0
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.	0x0

4.11.25 I2S Clock Control register (I2S_CTRL - 0x4000 407C)

The I2S_CTRL register controls some aspects of the two I2S interfaces (I2S0 and I2S1): enabling and disabling clocks, selecting DMA request line, and controlling source clock output or input configuration.

Table 40. I2S Clock Control register (I2S_CTRL - 0x4000 407C)

Bit	Function	Reset value
31:7	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
6	I2S1_CLK_TX_MODE select. 0 = TX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes. 1 = RX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes.	0
5	I2S1_CLK_RX_MODE select. 0 = RX_CLK drives I2S RX timing. (applies to both Master (output) and Slave (input) modes) 1 = TX_CLK drives I2S RX timing. (applies to both Master (output) and Slave (input) modes)	0
4	I2S1 DMA1 connection control 0 = UART7 RX is connected to DMA; (I2S1 DMA 1 is not connected to DMA) 1 = I2S1 DMA 1 is connected to DMA; (UART7 RX is not connected to DMA)	0
3	I2S0_CLK_TX_MODE select. 0 = TX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes. 1 = RX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes.	0
2	I2S0_CLK_RX_MODE select. 0 = RX_CLK drives I2S RX timing. (applies to both Master (output) and Slave (input) modes) 1 = TX_CLK drives I2S RX timing. (applies to both Master (output) and Slave (input) modes)	0
1	I2S1_CLK enable. 0 = Disable clock. 1 = Enable clock.	0
0	I2S0_CLK enable. 0 = Disable clock. 1 = Enable clock.	0

4.11.26 SSP Control register (SSP_CTRL - 0x4000 4078)

The SSP_CTRL register controls some aspects of the two SSP interfaces: enabling and disabling clocks and switching between the interface to the GPDMA Controller

Table 41. SSP Control register (SSP_CTRL - 0x4000 4078)

Bit	Function	Reset value
31:6	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
5	SSP1 RX DMA connection control 0 = SSP1 RX is not connected to DMA and SPI2 is connected to DMA. 1 = SSP1 RX is connected to DMA and SPI2 is not connected to DMA	0
4	SSP1 TX DMA connection control 0 = SSP1 TX is not connected to DMA and SPI1 is connected to DMA. 1 = SSP1 TX is connected to DMA and SPI1 is not connected to DMA.	0
3	SSP0 RX DMA connection control 0 = SSP0 RX is not connected to DMA and SPI3 is connected to DMA. 1 = SSP0 RX is connected to DMA and SPI3 is not connected to DMA.	0
2	SSP0 TX DMA connection control 0 = SSP0 TX is not connected to DMA and SPI4 is connected to DMA. 1 = SSP0 TX is connected to DMA and SPI4 is not connected to DMA.	0
1	SSP1 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
0	SSP0 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

4.11.27 SPI Control register (SPI_CTRL - 0x4000 40C4)

The SPI_CTRL register controls some aspects of the two SPI interfaces: enabling and disabling clocks; connecting the interface to the related pins; and controlling pin output values if the SPI interface is not used (i.e. used as a GPO). The SPI clock enable bit (0 and 1) must be enabled to use the peripheral, this includes the use of the SPI as GPO.

Table 42. SPI Control register (SPI_CTRL - 0x4000 40C4)

Bit	Function	Reset value
31:8	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
7	SPI2_DATIO output level. 0 = The pin drives low (if bit 5 is 0). 1 = The pin drives high (if bit 5 is 0).	0
6	SPI2_CLK output level. 0 = The pin drives low (if bit 5 is 0). 1 = The pin drives high (if bit 5 is 0).	0
5	Output pin control. By default, the SPI2_DATIO and SPI2_CLK pins are driven to the values set in bits 7 and 6. In order to use the SPI2 block, this bit must be written to a 1. 0 = SPI2_DATIO and SPI2_CLK outputs the level set by bit 6 and 7. 1 = SPI2_DATIO and SPI2_CLK are driven by the SPI2 block.	0
4	SPI2 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

Table 42. SPI Control register (SPI_CTRL - 0x4000 40C4) …continued

Bit	Function	Reset value
3	SPI1_DATIO output level. 0 = The pin drives low (if bit 1 is 0). 1 = The pin drives high (if bit 1 is 0).	0
2	SPI1_CLK output level. 0 = The pin drives low (if bit 1 is 0). 1 = The pin drives high (if bit 1 is 0).	0
1	Output pin control. By default, the SPI1_DATIO and SPI1_CLK pins are driven to the values set in bits 3 and 2. In order to use the SPI1 block, this bit must be written to a 1. 0 = SPI1_DATAIO and SPI1_CLK outputs the level set by bit 2 and 3. 1 = SPI1_DATIO and SPI1_CLK are driven by the SPI1 block.	0
0	SPI1 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

4.11.28 I^2C Clock Control register (I2CCLK_CTRL - 0x4000 40AC)

The I2CCLK_CTRL register controls the clocks to the two I^2C interfaces.

Table 43. I^2C Clock Control register (I2CCLK_CTRL - 0x4000 40AC)

Bit	Function	Reset value
31:5	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
4	Driver strength control for USB_I2C_SCL and USB_I2C_SDA. For 1.8 V operation set this bit to 1. 0 = USB I^2C pins operate in low drive mode. 1 = USB I^2C pins operate in high drive mode.	0
3	I2C2_SCL and I2C2_SDA driver strength control. For 1.8 V operation set this bit to 1. 0 = I2C2 pins operate in low drive mode. 1 = I2C2 pins operate in high drive mode.	0
2	I2C1_SCL and I2C1_SDA driver strength control. For 1.8 V operation set this bit to 1. 0 = I2C1 pins operate in low drive mode. 1 = I2C1 pins operate in high drive mode.	0
1	Software must set this bit before using the I2C2 block. It can be cleared if the I2C2 block is not in use. 0 = I2C2 HCLK stopped. No I^2C registers are accessible. 1 = I2C2 HCLK enabled.	0
0	Software must set this bit before using the I2C1 block. It can be cleared if the I2C1 block is not in use. 0 = I2C1 HCLK stopped. No I^2C registers are accessible. 1 = I2C1 HCLK enabled.	0

4.11.29 Timer Clock Control1 and MCPWM register (TIMCLK_CTRL1 - 0x4000 40C0)

The TIMCLK_CTRL1 register allows enabling and disabling the clocks to two peripheral device blocks: The Motor controller PWM and Timer[3,2,1,0].

Table 44. Timer Clock Control register (TIMCLK_CTRL1 - 0x4000 40C0)

Bit	Function	Reset value
6	Motor Control clock enable control. 0 = Disable clock. 1 = Enable clock.	0
5	Timer 3 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
4	Timer 2 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
3	Timer 1 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
2	Timer 0 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
1	Timer 5 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
0	Timer 4 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

4.11.30 Timer Clock Control register (TIMCLK_CTRL - 0x4000 40BC)

The TIMCLK_CTRL register allows enabling and disabling the clocks to the High Speed Timer and the Watchdog Timer.

Table 45. Timer Clock Control register (TIMCLK_CTRL - 0x4000 40BC)

Bit	Function	Reset value
31:3	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
1	HSTimer clock enable control. 0 = Disable clock. 1 = Enable clock.	0
0	Watchdog clock enable control. 0 = Disable clock. 1 = Enable clock.	0

4.11.31 ADC Clock Control register (ADCLK_CTRL - 0x4000 40B4)

The ADCLK_CTRL register allows enabling or disabling the clock to the Analog to Digital Converter and Touch Screen.

Table 46. ADC Clock Control register (ADCLK_CTRL - 0x4000 40B4)

Bit	Function	Reset value
31:1	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
0	0 = Disable 32 kHz clock to ADC block. 1 = Enable clock.	0

4.11.32 ADC Clock Control1 register (ADCLK_CTRL1 - 0x4000 4060)

The ADCLK_CTRL1 register controls switching the source of the ADC/Touch Screen clock. If the PERIPH_CLK is selected, use the clock divider to reduce the input frequency of the peripheral clock to a value less than 400KHz before using the ADC or touch screen.

Table 47. ADC Clock Control register (ADCLK_CTRL1 - 0x4000 4060)

Bit	Function	Reset value
31:9	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8	ADCCLK_SEL; ADC clock select 0 = Clock ADC and touch screen from RTC clock. 1 = Clock ADC and touch screen from PERIPH_CLK clock.	0
7:0	ADC_FREQ. Controls the clock divider for ADC when Peripheral clock (bit 8) is enabled. Value in register is one less than divide value. reg value = (divider -1) 00000000 = 1 00000001 = 2 …… 11111110 = 255 11111111 = 256	0

4.11.33 Keyboard Scan Clock Control register (KEYCLK_CTRL - 0x4000 40B0)

The KEYCLK_CTRL register allows enabling or disabling the clock to the Keyboard Scan peripheral.

Table 48. Keyboard Scan Clock Control register (KEYCLK_CTRL - 0x4000 40B0)

Bit	Function	Reset value
31:1	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
0	0 = Disable clock to Keyboard block. 1 = Enable clock.	0

4.11.34 PWM Clock Control register (PWMCLK_CTRL - 0x4000 40B8)

The PWMCLK_CTRL register controls the clocks to the PWM blocks: enabling or disabling clocks, selecting the clock source, and setting the clock divider for each PWM.

Table 49. PWM Clock Control register (PWMCLK_CTRL - 0x4000 40B8)

Bit	Function	Reset value
31:12	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
11:8	PWM2_FREQ. Controls the clock divider for PWM2. 0000 = PWM2_CLK = off 0001 = PWM2_CLK = CLKin : : 1111 = PWM2_CLK = CLKin / 15	0
7:4	PWM1_FREQ. Controls the clock divider for PWM1. The encoding is the same as for PWM2_CLK above.	0
3	PWM2 clock source selection: 0 = 32 kHz RTC_CLK 1 = PERIPH_CLK	0
2	PWM2 block enable 0 = Disable clock to PWM2 block. 1 = Enable clock to PWM2 block.	0
1	PWM1 clock source selection: 0 = 32 kHz RTC_CLK 1 = PERIPH_CLK	0
0	PWM1 block enable 0 = Disable clock to PWM1 block. 1 = Enable clock to PWM1 block.	0

4.11.35 UART Clock Control register (UARTCLK_CTRL - 0x4000 40E4)

The UARTCLK_CTRL register allows turning off clocks to the standard (not high speed) UARTs in order to save power when they are not used. High speed UARTs always operate in autoclock mode. See the High Speed UART chapter for details.

Table 50. UART Clock Control register (UARTCLK_CTRL - 0x4000 40E4)

Bit	Function	Reset value
31:4	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3	0 = Uart6 HCLK disabled and in low power mode. No accesses to UART registers are allowed. 1 = Uart6 HCLK enabled.	1
2	0 = Uart5 HCLK disabled and in low power mode. No accesses to UART registers are allowed. 1 = Uart5 HCLK enabled.	1
1	0 = Uart4 HCLK disabled and in low power mode. No accesses to UART registers are allowed. 1 = Uart4 HCLK enabled.	1
0	0 = Uart3 HCLK disabled and in low power mode. No accesses to UART registers are allowed. 1 = Uart3 HCLK enabled.	1

4.11.36 Power Off Switch internal memory control registers (POS0_IRAM_CTRL - 0x4000 4110, POS1_IRAM_CTRL - 0x4000 4114)

The POS0_IRAM_CTRL and POS1_IRAM_CTRL registers allow turning off power to 64 kB blocks of internal memory in order to save power when they are not used.

There is a maximum of 256 kB of internal memory in the LPC32x0 family, one 64kB block is always powered and the other three 64 kB blocks have power off switches.

There is no sequencing requirement to powering off a block of memory. To **power up** a memory block, the POS0 switch should be turned on (POS0 = 1) 2 μsec before POS1 is turned on (POS1 =1). Violating this timing requirement can result in a current spike in the power supply

If the power is turned off using the POS switches, the contents of the IRAM in the block is lost.

Table 51. Power Off Switch State

POS0	POS1	Memory state
LOW	LOW	Memory off
HIGH	LOW	Power up
HIGH	HIGH	Operational
LOW	HIGH	Not valid

Table 52. Power Off Switch 0 internal memory control register (POS0_IRAM_CTRL - 0x4000 4110)

Bit	Symbol	Function	Reset value
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3	POS0[1]	POS0 control for sram at address 0x30000 - 0x03FFFF (see Table 51).	1
2	POS0[1]	POS0 control for sram at address 0x20000 - 0x02FFFF (see Table 51).	1
1	POS0	POS0 control for sram at address 0x10000 - 0x01FFFF (see Table 51).	1
0	-	Reserved.	0

[1] Does not apply to LPC3220

Table 53. Power Off Switch 1 internal memory control register (POS1_IRAM_CTRL - 0x4000 4114)

Bit	Symbol	Function	Reset value
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3	POS1[1][2]	POS1 control for sram at address 0x30000 - 0x03FFFF (see Table 51).	1
2	POS1[1][2]	POS1 control for sram at address 0x20000 - 0x02FFFF (see Table 51).	1
1	POS1[1]	POS1 control for sram at address 0x10000 - 0x01FFFF (see Table 51).	1
0	-	Reserved.	0

[1] For correct operation when powering up sram, this bit can only be set 2 μsec after POS0 bit has been set.

[2] Does not apply to LPC3220

5.1 Introduction

The LPC32x0 Interrupt controller is comprised of three copies of a basic interrupt controller block connected so as to from a single larger interrupt controller. Each basic interrupt controller is capable of supporting up to 32 interrupts.

5.2 Features

- Supports 74 interrupt sources.
- FAB bus interface for fast register access.
- Interrupt enable bit for each interrupt.
- Individually selectable interrupt polarity and type allows for high or low level triggered interrupts, as well as rising or falling edge triggered interrupts.
- Each interrupt may be steered to either the IRQ or FIQ input to the CPU.
- Raw interrupt status and masked interrupt status registers are available for versatile condition evaluation.
- Provides a software interrupt with a message register.

5.3 Description

The Interrupt Controller is accessed via the FAB bus and is clocked by PERIPH_CLK clock in all modes except stop mode. The internal connections of the Interrupt Controller are shown in Figure 13. As illustrated, inputs to the Interrupt Controller that are asynchronous are synchronized prior to processing. The output of the Interrupt Controller drives the FIQ and IRQ inputs to the CPU.

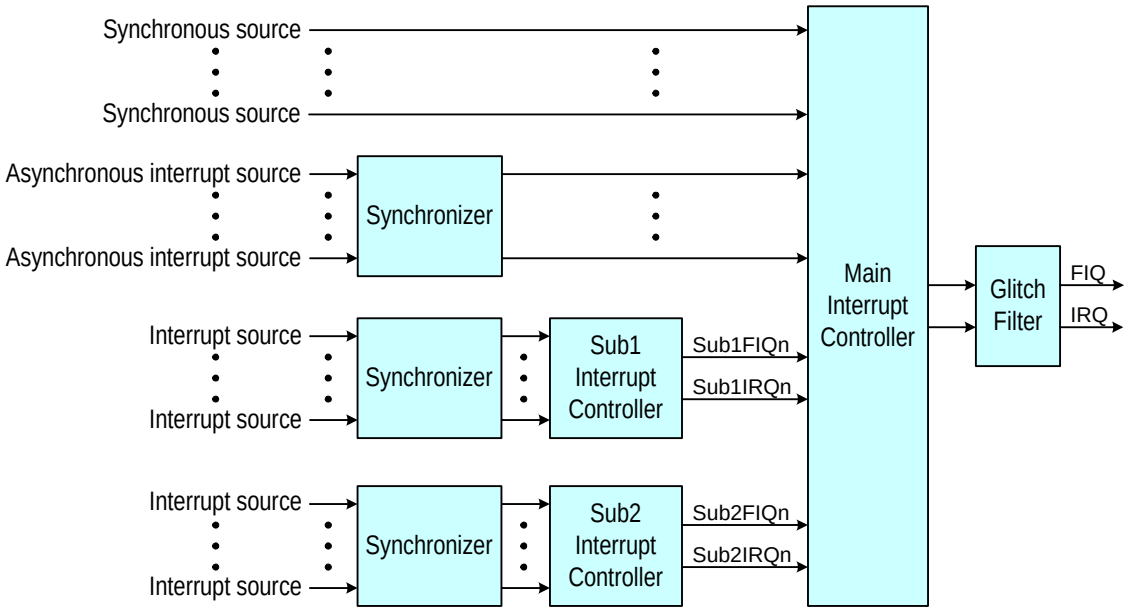


Fig 13. Block diagram of the interrupt controller

All external pin interrupts are connected via synchronizing circuits to the two Sub Interrupt Controllers (SIC1 and SIC2). The four interrupt outputs of the SICs are connected to four interrupt inputs of the Main interrupt controller (MIC). It is advised to always configure these inputs on the MIC to active low level.

All interrupts may be programmed for a specific polarity of either a level or an edge. If set up to trigger on an edge, an active interrupt state is stored until cleared by the host. Each interrupt source can be individually masked and the interrupt status can be read both before and after masking. Each interrupt source is set to generate either an IRQ or a FIQ. The interrupt mode for each interrupt pin must be configured in the SIC registers APR and ATR. A typical bit slice of the interrupt controller is shown in Figure 14.

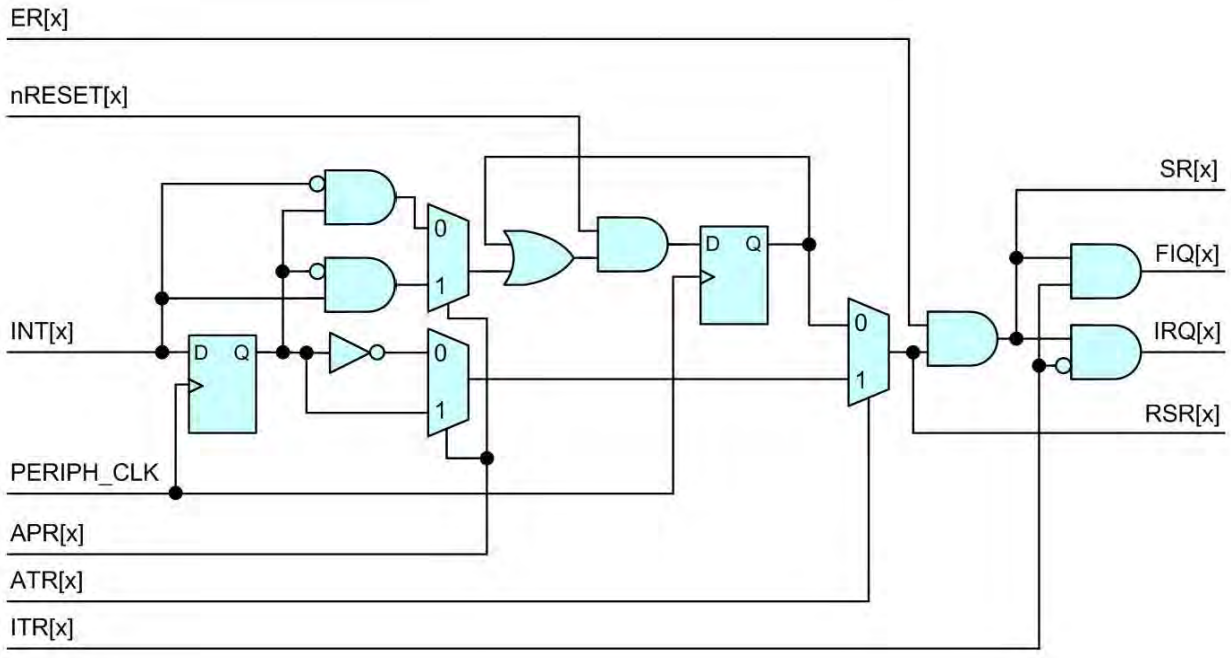


Fig 14. Bit slice of interrupt controller

There are a number of internal interrupt sources which are synchronous. These are input directly to the interrupt controller. All external pin interrupts are assumed to be asynchronous, and these are synchronized prior to arrival at the interrupt controller. In addition, asynchronous internal interrupt sources are synchronized before being connected to the interrupt controller.

The interrupt controller outputs IRQ and FIQ signals to the CPU. Since there could be glitches on these signals, a glitch filter is included at the output of the interrupt controller.

When the CPU responds to an interrupt, software must determine which interrupt service routine to invoke. This may be done by reading the Status Registers and using a software prioritization scheme to single out one interrupt for service if more than one is pending.

An interrupt can wake the device up from the CPU Wait for Interrupt mode (described in ARM CPU documentation, register c7 of coprocessor 15) provided that the peripheral generating the interrupt is clocked. Since the interrupt controller works only when clocked, an interrupt cannot wake up the CPU from stop mode. However, a number of events can wake up the CPU from STOP mode via the Start Controller (described in the Clocking and Power Control chapter). Some events that can cause a wake-up from STOP mode can also be interrupt sources.

If a pin is used both as an active interrupt and a start signal, several configurations can be used:

- A pin configured as a level triggered interrupt requires the pin to retain the active level until the interrupt is processed and cleared.
- A pin configured as an edge triggered interrupt requires the pin to retain the active level until the interrupt controller receives clocks and recognizes the edge.

- For shorter pulses the start signal status can be used to activate a software generated interrupt. The source can be communicated using a global variable.

The software interrupt feature is activated by a register bit in the SW_INT register, and is otherwise handled in the same manner as a hardware interrupt. Seven bits in the SW_INT register are available to provide information to the software interrupt service routine.

At reset, all registers in the interrupt controllers are set to zeros disabling the interrupt controller by default. Software must configure the controller during initialization.

Note that software must enable the SubFIQn and SubIRQn sources in the Main interrupt controller if any interrupts in the Sub Interrupt Controller will be used. These sources are active low.

5.4 Register description

Table 54. Interrupt controller registry summary

Address	Register name	Description	Reset value	Type
0x4000 8000	MIC_ER	Enable Register for the Main Interrupt Controller	0	R/W
0x4000 8004	MIC_RSR	Raw Status Register for the Main Interrupt Controller	x	R/W
0x4000 8008	MIC_SR	Status Register for the Main Interrupt Controller	0	RO
0x4000 800C	MIC_APR	Activation Polarity select Register for the Main Interrupt Controller	0	R/W
0x4000 8010	MIC_ATR	Activation Type select Register for the Main Interrupt Controller	0	R/W
0x4000 8014	MIC_ITR	Interrupt Type select Register for the Main Interrupt Controller	0	R/W
0x4000 C000	SIC1_ER	Enable register for Sub Interrupt Controller 1	0	R/W
0x4000 C004	SIC1_RSR	Raw Status Register for Sub Interrupt Controller 1	-	R/W
0x4000 C008	SIC1_SR	Status Register for Sub Interrupt Controller 1	0	RO
0x4000 C00C	SIC1_APR	Activation Polarity select Register for Sub Interrupt Controller 1	0	R/W
0x4000 C010	SIC1_ATR	Activation Type select Register for Sub Interrupt Controller 1	0	R/W
0x4000 C014	SIC1_ITR	Interrupt Type select Register for Sub Interrupt Controller 1	0	R/W
0x4001 0000	SIC2_ER	Enable register for Sub Interrupt Controller 2	0	R/W
0x4001 0004	SIC2_RSR	Raw Status Register for Sub Interrupt Controller 2	x	R/W
0x4001 0008	SIC2_SR	Status Register for Sub Interrupt Controller 2	0	RO
0x4001 000C	SIC2_APR	Activation Polarity select Register for Sub Interrupt Controller 2	0	R/W
0x4001 0010	SIC2_ATR	Activation Type select Register for Sub Interrupt Controller 2	0	R/W
0x4001 0014	SIC2_ITR	Interrupt Type select Register for Sub Interrupt Controller 2	0	R/W

Note: Internal peripheral interrupt sources are active HIGH unless otherwise noted.

5.4.1 Interrupt Enable Register for the Main Interrupt Controller (MIC_ER - 0x4000 8000)

The MIC_ER register contains bits that allow enabling and disabling individual interrupt sources to the Main Interrupt Controller. For all interrupt enable bits, 0 = interrupt disabled (default at reset) and 1 = interrupt enabled. The upper two bits in MIC_ER control all FIQ interrupts from the Sub Interrupt Controllers, while the lower two bits control all IRQ interrupts from the Sub Interrupt Controllers. Table 55 describes the function of each bit in this register.

Table 55. Interrupt Enable Register for the Main Interrupt Controller (MIC_ER - 0x4000 8000)

Bits	Name	Description	Reset value
31	Sub2FIQn	High priority (IRQ) interrupts from SIC2.	0
30	Sub1FIQn	High priority (IRQ) interrupts from SIC1.	0
29	Ethernet	Ethernet interrupt.	0
28	DMA_INT	General Purpose DMA Controller interrupt.	0
27	MSTIMER_INT	Match interrupt 0 or 1 from the Millisecond Timer.	0
26	IIR1	UART1 interrupt.	0
25	IIR2	UART2 interrupt.	0
24	IIR7	UART7 interrupt.	0
23	I2S1	I2S1 Interrupt	0
22	I2S0	I2S0 interrupt	0
21	SSP1	SSP1 interrupt	0
20	SSP0	SSP0 interrupt	0
19	Timer3	Timer3 interrupt	0
18	Timer2	Timer2 interrupt	0
17	Timer1	Timer1 interrupt	0
16	Timer0	Timer0 interrupt	0
15	SD0_INT	Interrupt 0 from the SD Card interface.	0
14	LCD_INT	Interrupt from the LCD controller.	0
13	SD1_INT	Interrupt 1 from the SD Card interface.	0
12	Reserved	Reserved, do not modify.	0
11	FLASH_INT	Interrupt from the NAND Flash controller.	0
10	IIR6	UART6 interrupt.	0
9	IIR5	UART5 interrupt.	0
8	IIR4	UART4 interrupt.	0
7	IIR3	UART3 interrupt.	0
6	WATCH_INT	Watchdog Timer interrupt.	0
5	HSTIMER_INT	Match interrupt from the High Speed Timer.	0
4	Timer5	Timer5 interrupt	0
3	Timer4/MCPWM	Timer4 or Motor Controller PWM interrupt if TIMCLK_CTRL1[6] = 0 then Timer4. if TIMCLK_CTRL1[6] = 1 then MCPWM.	0
2	Reserved	Reserved, do not modify.	0
1	Sub2IRQn	Low priority (IRQ) interrupts from SIC2.	0
0	Sub1IRQn	Low priority (IRQ) interrupts from SIC1.	0

5.4.2 Interrupt Enable Register for Sub Interrupt Controller 1 (SIC1_ER - 0x4000 C000)

The SIC1_ER register contains bits that allow enabling and disabling individual interrupt sources to Sub Interrupt Controller 1. For all interrupt enable bits, 0 = interrupt disabled (default at reset) and 1 = interrupt enabled. Table 56 describes the function of each bit in this register.

Table 56. Interrupt Enable Register for Sub Interrupt Controller 1 (SIC1_ER - 0x4000 C000)

Bits	Name	Description	Reset value
31	USB_i2c_int	Interrupt from the USB I^2C interface.	0
30	USB_dev_hp_int	USB high priority interrupt.	0
29	USB_dev_lp_int	USB low priority interrupt.	0
28	USB_dev_dma_int	USB DMA interrupt.	0
27	USB_host_int	USB host interrupt.	0
26	USB_otg_atx_int_n	External USB transceiver interrupt. Active LOW.	0
25	USB_otg_timer_int	USB timer interrupt.	0
24	SW_INT	Software interrupt (caused by bit 0 of the SW_INT register).	0
23	SPI1_INT	Interrupt from the SPI1 interface.	0
22	KEY_IRQ	Keyboard scanner interrupt.	0
21	Reserved	Reserved, do not modify.	0
20	RTC_INT	Match interrupt 0 or 1 from the RTC.	0
19	I2C_1_INT	Interrupt from the I2C1 interface. Active LOW.	0
18	I2C_2_INT	Interrupt from the I2C2 interface. Active LOW.	0
17	PLL397_INT	Lock interrupt from the 397x PLL.	0
16:15	Reserved	Reserved, do not modify.	0
14	PLLHCLK_INT	Lock interrupt from the HCLK PLL.	0
13	PLLUSB_INT	Lock interrupt from the USB PLL.	0
12	SPI2_INT	Interrupt from the SPI2 interface.	0
11:9	Reserved	Reserved, do not modify.	0
8	TS_AUX	Touch screen aux interrupt	0
7	TS_IRQ (ADC_INT)	Touch screen irq interrupt	0
6	TS_P	Touch screen pen down interrupt.	0
5	Reserved	Reserved, do not modify.	0
4	GPI_28	Interrupt from the GPI_28 pin.	0
3	Reserved	Reserved, do not modify.	0
2	JTAG_COMM_RX	RX full interrupt from the JTAG Communication Channel.	0
1	JTAG_COMM_TX	TX empty interrupt from the JTAG Communication Channel.	0
0	Reserved	Reserved, do not modify.	0

5.4.3 Interrupt Enable Register for Sub Interrupt Controller 2 (SIC2_ER - 0x4001 0000)

The SIC2_ER register contains bits that allow enabling and disabling individual interrupt sources to Sub Interrupt Controller 2. For all interrupt enable bits, 0 = interrupt disabled (default at reset) and 1 = interrupt enabled. Table 57 describes the function of each bit in this register.

Table 57. Interrupt Enable Register for Sub Interrupt Controller 2 (SIC2_ER - 0x4001 0000)

Bits	Name	Description	Reset value
31	SYSCLK mux	Status of the SYSCLK Mux (SYSCLK_CTRL[0]). May be used to begin operations that require a change to the alternate clock source.	0
30:29	Reserved	Reserved, do not modify.	0
28	GPI_6	Interrupt from the GPI_6 (HSTIM_CAP) pin.	0
27	GPI_5	Interrupt from the GPI_5 pin.	0
26	GPI_4	Interrupt from the GPI_4 (SPI1_BUSY) pin.	0
25	GPI_3	Interrupt from the GPI_3 pin.	0
24	GPI_2	Interrupt from the GPI_2 pin.	0
23	GPI_1	Interrupt from the GPI_1 (SERVICE_N) pin.	0
22	GPI_0	Interrupt from the GPI_0 pin.	0
21	Reserved	Reserved, do not modify.	0
20	SPI1_DATIN	Interrupt from the SPI1_DATIN pin.	0
19	U5_RX	Interrupt from the UART5 RX pin.	0
18	SDIO_INT_N	Interrupt from the MS_DIO1 pin. Active LOW.	0
17:16	Reserved	Reserved, do not modify.	0
15	GPI_7	Interrupt from the GPI_7 pin.	0
14:13	Reserved	Reserved, do not modify.	0
12	U7_HCTS	Interrupt from the UART7 HCTS pin.	0
11	GPI_19	Interrupt from the GPI_19 (U4_RX) pin.	0
10	GPI_9	Interrupt from the GPI_9 (KEY_COL7) pin.	0
9	GPI_8	Interrupt from the GPI_8 (KEY_COL6, SPI2_BUSY) pin.	0
8	Pn_GPIO	ALL Port 0 and Port 1 GPIO pins OR'ed.	0
7	U2_HCTS	Interrupt from the UART2 HCTS pin.	0
6	SPI2_DATIN	Interrupt from the SPI2_DATIN pin.	0
5	GPIO_5	Interrupt from the GPIO_5 pin.	0
4	GPIO_4	Interrupt from the GPIO_4 pin.	0
3	GPIO_3	Interrupt from the GPIO_3 (KEY_ROW7) pin.	0
2	GPIO_2	Interrupt from the GPIO_2 (KEY_ROW6) pin.	0
1	GPIO_1	Interrupt from the GPIO_1 pin.	0
0	GPIO_0	Interrupt from the GPIO_0 pin.	0

5.4.4 Main Interrupt Controller Raw Status Register (MIC_RSR - 0x4000 8004); Sub1 Raw Status Register (SIC1_RSR - 0x4000 C004); Sub2 Raw Status Register (SIC2_RSR - 0x4001 0004)

The Raw Status Registers provide information about the state of interrupt sources before they are potentially masked by the corresponding Enable Register. These registers also allow clearing edge triggered interrupts whether or not they are masked. Table 58 describes the function of bits in this register.

Table 58. Interrupt Controller Raw Status Registers (MIC_RSR, SIC1_RSR, SIC2_RSR)

Bits	Description	Reset value
31:0	Raw Interrupt Status. Reading the RSR shows which interrupt sources are active before being masked by the ER. Writing to the RSR clears the interrupt status from edge triggered sources. Level triggered sources must be cleared at the source. The interrupt to which each bit applies can be found in the table for the related Enable Register (MIC_ER, SIC1_ER, or SIC2_ER). Read: 0 = Source is not generating an interrupt. 1 = Source is generating an interrupt. Write: 0 = No Operation. 1 = Clear the interrupt status of edge triggered sources.	-

5.4.5 Main Interrupt Controller Status Register (MIC_SR - 0x4000 8008); Sub1 Status Register (SIC1_SR - 0x4000 C008); Sub2 Status Register (SIC2_SR - 0x4001 0008)

The Interrupt Status Registers provide information on which interrupts are actually pending, after being masked by the corresponding Enable Register. Table 59 describes the function of bits in this register.

Table 59. Interrupt Status Registers (MIC_SR, SIC1_SR, and SIC2_SR)

Bits	Description	Reset value
31:0	Interrupt status. A high bit indicates that the unmasked interrupt source is generating an interrupt. The interrupt to which each bit applies can be found in the table for the related Enable Register (MIC_ER, SIC1_ER, or SIC2_ER). 0 = No interrupt pending (Default) 1 = Interrupt pending	0

5.4.6 Main Interrupt Controller Activation Polarity Register (MIC_APR - 0x4000 800C; Sub1 Activation Polarity Register (SIC1_APR - 0x4000 C00C); Sub2 Activation Polarity Register (SIC2_APR - 0x4001 000C)

The interrupt Activation Polarity Registers allow selection of the activation polarity of each interrupt. In connection with the Activation Type registers, four basic modes may be chosen: low level, high level, falling edge, or rising edge triggering. Table 60, Table 61, and Table 62 describe the function of bits in these registers.

Table 60. Activation Polarity Registers (MIC_APR - 0x4000 800C)

Bits	Name	Description	Operational value	Reset value
31	Sub2FIQn	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
30	Sub1FIQn	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
29	Ethernet	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
28	DMA_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
27	MSTIMER_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
26	IIR1	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0

Table 60. Activation Polarity Registers (MIC_APR - 0x4000 800C)

Bits	Name	Description	Operational value	Reset value
25	IIR2	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
24	IIR7	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
23	I2S1	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
22	I2S0	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
21	SSP1	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
20	SSP0	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
19	Timer3	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
18	Timer2	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
17	Timer1	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
16	Timer0	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
15	SD0_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
14	LCD_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
13	SD1_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
12	Reserved	Reserved, do not modify.	-	0
11	FLASH_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
10	IIR6	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
9	IIR5	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
8	IIR4	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
7	IIR3	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
6	WATCH_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
5	HSTIMER_INT	Interrupt Polarity. See bit 0 description	Active HIGH (1)	0
4	Timer5	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
3	Timer4 / MCPWM	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
2	Reserved	Reserved, do not modify.	-	0
1	Sub2IRQn	Interrupt Polarity. See bit 0 description	Active LOW (0)	0
0	Sub1IRQn	Interrupt Polarity. 0 = Interrupt is generated on a low level signal or falling edge. 1 = Interrupt is generated on a high level signal or rising edge.	Active LOW (0)	0

Table 61. Activation Polarity Register (SIC1_APR - 0x4000 C00C)

Bits	Name	Description	Operational value	Reset value
31	USB_i2c_int	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
30	USB_dev_hp_int	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
29	USB_dev_lp_int	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
28	USB_dev_dma_int	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
27	USB_host_int	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
26	USB_otg_atx_int_n	Interrupt Polarity select. See bit 1 description	Active LOW (0)	0
25	USB_otg_timer_int	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0

Table 61. Activation Polarity Register (SIC1_APR - 0x4000 C00C) *...continued*

Bits	Name	Description	Operational value	Reset value
24	SW_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
23	SPI1_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
22	KEY_IRQ	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
21	Reserved	Reserved, do not modify.	-	0
20	RTC_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
19	I2C_1_INT	Interrupt Polarity select. See bit 1 description	Active LOW (0)	0
18	I2C_2_INT	Interrupt Polarity select. See bit 1 description	Active LOW (0)	0
17	PLL397_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
16	Reserved	Reserved, do not modify.	-	0
15	Reserved	Reserved, do not modify.	-	0
14	PLLHCLK_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
13	PLLUSB_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
12	SPI2_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
11:9	Reserved	Reserved, do not modify.	-	0
8	TS_AUX	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
7	TS_INT	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
6	TS_P	Interrupt Polarity select. See bit 1 description	user defined	0
5	Reserved	Reserved, do not modify.	-	0
4	GPI_28	Interrupt Polarity select. See bit 1 description	user defined	0
3	Reserved	Reserved, do not modify.	-	0
2	JTAG_COMM_RX	Interrupt Polarity select. See bit 1 description	Active HIGH (1)	0
1	JTAG_COMM_TX	Interrupt Polarity select. 0 = Interrupt is generated on a low level signal or falling edge. 1 = Interrupt is generated on a high level signal or rising edge.	Active LOW (0)	0
0	Reserved	Reserved, do not modify.	-	0

Table 62. Activation Polarity Register (SIC2_APR - 0x4001 000C)

Bits	Name	Description	Operational value	Reset value
31	SYSCLK mux	Interrupt Polarity select. See bit 0 description	Active HIGH (1)	0
30:29	Reserved	Reserved, do not modify.	-	0
28	GPI_6	Interrupt Polarity select. See bit 0 description	user defined	0
27	GPI_5	Interrupt Polarity select. See bit 0 description	user defined	0
26	GPI_4	Interrupt Polarity select. See bit 0 description	user defined	0
25	GPI_3	Interrupt Polarity select. See bit 0 description	user defined	0
24	GPI_2	Interrupt Polarity select. See bit 0 description	user defined	0
23	GPI_1	Interrupt Polarity select. See bit 0 description	user defined	0
22	GPI_0	Interrupt Polarity select. See bit 0 description	user defined	0
21	Reserved	Reserved, do not modify.	-	0

Table 62. Activation Polarity Register (SIC2_APR - 0x4001 000C)

Bits	Name	Description	Operational value	Reset value
20	SPI1_DATIN	Interrupt Polarity select. See bit 0 description	Active HIGH (1)	0
19	U5_RX	Interrupt Polarity select. See bit 0 description	Active HIGH (1)	0
18	SDIO_INT_N	Interrupt Polarity select. See bit 0 description	Active LOW (0)	0
17:16	Reserved	Reserved, do not modify.	-	0
15	GPI_7	Interrupt Polarity select. See bit 0 description	user defined	0
14:13	Reserved	Reserved, do not modify.	-	0
12	U7_HCTS	Interrupt Polarity select. See bit 0 description	Active HIGH (1)	0
11	GPI_19	Interrupt Polarity select. See bit 0 description	user defined	0
10	GPI_9	Interrupt Polarity select. See bit 0 description	user defined	0
9	GPI_8	Interrupt Polarity select. See bit 0 description	user defined	0
8	Pn_GPIO	Interrupt Polarity select. See bit 0 description	user defined	0
7	U2_HCTS	Interrupt Polarity select. See bit 0 description	Active HIGH (1)	0
6	SPI2_DATIN	Interrupt Polarity select. See bit 0 description	Active HIGH (1)	0
5	GPIO_5	Interrupt Polarity select. See bit 0 description	user defined	0
4	GPIO_4	Interrupt Polarity select. See bit 0 description	user defined	0
3	GPIO_3	Interrupt Polarity select. See bit 0 description	user defined	0
2	GPIO_2	Interrupt Polarity select. See bit 0 description	user defined	0
1	GPIO_1	Interrupt Polarity select. See bit 0 description	user defined	0
0	GPIO_0	Interrupt Polarity select. 0 = Interrupt is generated on a low level signal or falling edge. 1 = Interrupt is generated on a high level signal or rising edge.	user defined	0

5.4.7 Main Interrupt Controller Activation Type Register (MIC_ATR - 0x4000 8010); Sub1 Activation Type Register (SIC1_ATR - 0x4000 C010); Sub2 Activation Type Register (SIC2_ATR - 0x4001 0010)

The interrupt Activation Type Registers allow selection of the trigger type of each interrupt. In connection with the Activation Polarity registers, four basic modes may be chosen: low level, high level, falling edge, or rising edge triggering. Table 63, Table 64, and Table 65 describe the function of bits in these registers.

Table 63. Activation Type Register (MIC_ATR - 0x4000 8010)

Bits	Name	Description	Operational value	Reset value
31	Sub2FIQn	Interrupt Activation Type, see bit 0 description	Level (0)	0
30	Sub1FIQn	Interrupt Activation Type, see bit 0 description	Level (0)	0
29	Ethernet	Interrupt Activation Type, see bit 0 description	Level (0)	0
28	DMA_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
27	MSTIMER_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
26	IIR1	Interrupt Activation Type, see bit 0 description	Level (0)	0
25	IIR2	Interrupt Activation Type, see bit 0 description	Level (0)	0

Table 63. Activation Type Register (MIC_ATR - 0x4000 8010)

Bits	Name	Description	Operational value	Reset value
24	IIR7	Interrupt Activation Type, see bit 0 description	Level (0)	0
23	I2S1	Interrupt Activation Type, see bit 0 description	Level (0)	0
22	I2S0	Interrupt Activation Type, see bit 0 description	Level (0)	0
21	SSP1	Interrupt Activation Type, see bit 0 description	Level (0)	0
20	SSP0	Interrupt Activation Type, see bit 0 description	Level (0)	0
19	Timer3	Interrupt Activation Type, see bit 0 description	Level (0)	0
18	Timer2	Interrupt Activation Type, see bit 0 description	Level (0)	0
17	Timer1	Interrupt Activation Type, see bit 0 description	Level (0)	0
16	Timer0	Interrupt Activation Type, see bit 0 description	Level (0)	0
15	SD0_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
14	LCD_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
13	SD1_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
12	Reserved	Reserved, do not modify.		0
11	FLASH_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
10	IIR6	Interrupt Activation Type, see bit 0 description	Level (0)	0
9	IIR5	Interrupt Activation Type, see bit 0 description	Level (0)	0
8	IIR4	Interrupt Activation Type, see bit 0 description	Level (0)	0
7	IIR3	Interrupt Activation Type, see bit 0 description	Level (0)	0
6	WATCH_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
5	HSTIMER_INT	Interrupt Activation Type, see bit 0 description	Level (0)	0
4	Timer5	Interrupt Activation Type, see bit 0 description	Level (0)	0
3	Timer4 / MCPWM	Interrupt Activation Type, see bit 0 description	Level (0)	0
2	Reserved	Reserved, do not modify.		0
1	Sub2IRQn	Interrupt Activation Type, see bit 0 description	Level (0)	0
0	Sub1IRQn	Interrupt Activation Type selection, determines whether each interrupt is level sensitive or edge sensitive. 0 = Interrupt is level sensitive. (Default) 1 = Interrupt is edge sensitive.	Level (0)	0

Table 64. Activation Type Register (SIC1_ATR - 0x4000 C010)

Bits	Name	Description	Operational value	Reset value
31	USB_i2c_int	Interrupt Activation Type, see bit 1 description	level (0)	0
30	USB_dev_hp_int	Interrupt Activation Type, see bit 1 description	level (0)	0
29	USB_dev_lp_int	Interrupt Activation Type, see bit 1 description	level (0)	0
28	USB_dev_dma_int	Interrupt Activation Type, see bit 1 description	level (0)	0
27	USB_host_int	Interrupt Activation Type, see bit 1 description	level (0)	0
26	USB_otg_atx_int_n	Interrupt Activation Type, see bit 1 description	level (0)	0
25	USB_otg_timer_int	Interrupt Activation Type, see bit 1 description	level (0)	0
24	SW_INT	Interrupt Activation Type, see bit 1 description	level (0)	0

Table 64. Activation Type Register (SIC1_ATR - 0x4000 C010) *...continued*

Bits	Name	Description	Operational value	Reset value
23	SPI1_INT	Interrupt Activation Type, see bit 1 description	level (0)	0
22	KEY_IRQ	Interrupt Activation Type, see bit 1 description	level (0)	0
21	Reserved	Reserved, do not modify.	-	0
20	RTC_INT	Interrupt Activation Type, see bit 1 description	level (0)	0
19	I2C_1_INT	Interrupt Activation Type, see bit 1 description	level (0)	0
18	I2C_2_INT	Interrupt Activation Type, see bit 1 description	level (0)	0
17	PLL397_INT	Lock interrupt from the 397x PLL.	edge(1)	0
16:15	Reserved	Reserved, do not modify.	-	0
14	PLLHCLK_INT	Interrupt Activation Type, see bit 1 description	edge(1)	0
13	PLLUSB_INT	Interrupt Activation Type, see bit 1 description	edge(1)	0
12	SPI2_INT	Interrupt Activation Type, see bit 1 description	level (0)	0
11:9	Reserved	Reserved, do not modify.	-	0
8	TS_AUX	Interrupt Activation Type, see bit 1 description	user defined	0
7	TS_INT	Interrupt Activation Type, see bit 1 description	user defined	0
6	TS_P	Interrupt Activation Type, see bit 1 description	user defined	0
5	Reserved	Reserved, do not modify.	-	0
4	GPI_28	Interrupt Activation Type, see bit 1 description	user defined	0
3	Reserved	Reserved, do not modify.		0
2	JTAG_COMM_RX	Interrupt Activation Type, see bit 1 description	level (0)	0
1	JTAG_COMM_TX	Interrupt Activation Type selection, determines whether each interrupt is level sensitive or edge sensitive. 0 = Interrupt is level sensitive. (Default) 1 = Interrupt is edge sensitive.	level (0)	0
0	Reserved	Reserved, do not modify.	-	0

Table 65. Activation Type Register (SIC2_ATR - 0x4001 0010)

Bits	Name	Description	Operational value	Reset value
31	SYSCLK mux	Interrupt Activation Type, see bit 0 description		0
30:29	Reserved	Reserved, do not modify.	-	0
28	GPI_6	Interrupt Activation Type, see bit 0 description	user defined	0
27	GPI_5	Interrupt Activation Type, see bit 0 description	user defined	0
26	GPI_4	Interrupt Activation Type, see bit 0 description	user defined	0
25	GPI_3	Interrupt Activation Type, see bit 0 description	user defined	0
24	GPI_2	Interrupt Activation Type, see bit 0 description	user defined	0
23	GPI_1	Interrupt Activation Type, see bit 0 description	user defined	0
22	GPI_0	Interrupt Activation Type, see bit 0 description	user defined	0
21	Reserved	Reserved, do not modify.	-	0
20	SPI1_DATIN	Interrupt Activation Type, see bit 0 description		0
19	U5_RX	Interrupt Activation Type, see bit 0 description		0

Table 65. Activation Type Register (SIC2_ATR - 0x4001 0010)

Bits	Name	Description	Operational value	Reset value
18	SDIO_INT_N	Interrupt Activation Type, see bit 0 description	Level (0)	0
17:16	Reserved	Reserved, do not modify.	-	0
15	GPI_7	Interrupt Activation Type, see bit 0 description	user defined	0
14:13	Reserved	Reserved, do not modify.	-	0
12	U7_HCTS	Interrupt Activation Type, see bit 0 description		0
11	GPI_19	Interrupt Activation Type, see bit 0 description	user defined	0
10	GPI_9	Interrupt Activation Type, see bit 0 description	user defined	0
9	GPI_8	Interrupt Activation Type, see bit 0 description	user defined	0
8	Pn_GPIO	Interrupt Activation Type, see bit 0 description		0
7	U2_HCTS	Interrupt Activation Type, see bit 0 description		0
6	SPI2_DATIN	Interrupt Activation Type, see bit 0 description		0
5	GPIO_5	Interrupt Activation Type, see bit 0 description	user defined	0
4	GPIO_4	Interrupt Activation Type, see bit 0 description	user defined	0
3	GPIO_3	Interrupt Activation Type, see bit 0 description	user defined	0
2	GPIO_2	Interrupt Activation Type, see bit 0 description	user defined	0
1	GPIO_1	Interrupt Activation Type, see bit 0 description	user defined	0
0	GPIO_0	Interrupt Activation Type, determines whether each interrupt is level sensitive or edge sensitive. 0 = Interrupt is level sensitive. (Default) 1 = Interrupt is edge sensitive.	user defined	0

5.4.8 Main Interrupt Controller Interrupt Type Register (MIC_ITR - 0x4000 8014); Sub1 Interrupt Type Register (SIC1_ITR - 0x4000 C014); Sub2 Interrupt Type Register (SIC2_ITR - 0x4001 0014)

The Interrupt Type Registers allow each interrupt to be reflected to the CPU as either a standard Interrupt Request (IRQ) or a Fast Interrupt Request (FIQ). Table 66 describes the function of bits in this register.

Table 66. Sub1 Interrupt Type Registers (MIC_ITR, SIC1_ITR, and SIC2_ITR)

Bits	Description	Reset value
31:0	Interrupt Type selection, determines whether each interrupt is a standard interrupt request (IRQ), or a Fast Interrupt Request (FIQ). The interrupt to which each bit applies can be found in the table for the related Enable Register (MIC_ER, SIC1_ER, or SIC2_ER). 0 = The interrupt is routed to the IRQ output of the interrupt controller. 1 = The interrupt is routed to the FIQ output of the interrupt controller.	0

5.4.9 Software Interrupt Register (SW_INT - 0x4000 40A8)

The SW_INT register allows software to cause a hardware interrupt specifically reserved for this purpose. Additional bits in the register allow the possibility of passing information about the reason for the software interrupt to the service routine.

Table 67. Software Interrupt Register (SW_INT - 0x4000 40A8)

Bits	Description	Reset value
7:1	Implemented as read/write register bits. Can be used to pass a parameter to the interrupt service routine.	0x00
0	0 = SW_INT source inactive. 1 = SW_INT source active. Software must ensure that this bit is high for more than one SYSCLK period. This can be accomplished by causing foreground software to set SW_INT[0] and the software interrupt service routine to clear the bit.	0

6.1 Introduction

The DMA controller allows 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 bi-directional port requires one stream for transmit and one for receives. The source and destination areas can each be either a memory region or a peripheral.

6.2 Features

- Eight DMA channels. Each channel can support an unidirectional transfer.
- 16 DMA request lines.
- Single DMA and burst DMA request signals. Each peripheral connected to the DMA Controller can assert either a burst DMA request or a single DMA request. The DMA burst size is set by programming the DMA Controller.
- Memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers are 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.
- Two AHB bus masters for transferring data. These interfaces transfer data when a DMA request goes active. Either master can be selected for source or destination on each DMA channel.
- 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, 16, 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.

6.3 Functional description

This section describes the major functional blocks of the DMA Controller.

6.3.1 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 15 shows a block diagram of the DMA Controller.

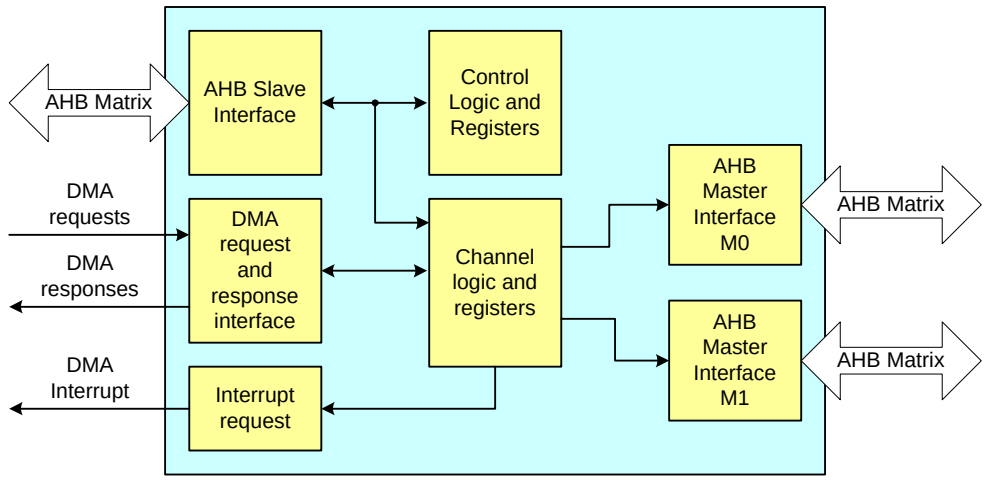


Fig 15. DMA controller block diagram

The functions of the DMA Controller are described in the following sections.

6.3.1.1 AHB slave interface

All transactions to DMA Controller registers on the AHB slave interface are 32 bits wide. Eight bit and 16-bit accesses are not supported and will result in an exception.

6.3.1.2 Control logic and register bank

The register block stores data written or to be read across the AHB interface.

6.3.1.3 DMA request and response interface

See DMA Interface description for information on the DMA request and response interface.

6.3.1.4 Channel logic and channel register bank

The channel logic and channel register bank contains registers and logic required for each DMA channel.

6.3.1.5 Interrupt request

The interrupt request generates the interrupt to the ARM processor.

6.3.1.6 AHB master interface

The DMA Controller contains two AHB master interfaces. Each 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.

6.3.1.6.1 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. Note that the

6.3.1.6.2 Endian behavior

The DMA Controller can cope with both little-endian and big-endian addressing. Software can set the endianness of each AHB master individually.

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 68 shows endian behavior for different source and destination combinations.

Table 68. 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		
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

Table 68. 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	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
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		

Table 68. 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	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		

6.3.1.6.3 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.

6.3.1.7 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.

6.3.1.8 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. In the worst case this is as large as a one quadword.

It is recommended that memory-to-memory transactions use the lowest priority channel. Otherwise other AHB bus masters are prevented from accessing the bus during DMA Controller memory-to-memory transfer.

6.3.1.9 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.

6.3.2 DMA system connections

6.3.2.1 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 and whether the transfer is the last in the data packet. 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 all request types.

Table 69. Peripheral connections to the DMA controller and matching flow control signals.

Peripheral Number	DMA Slave	DMACBREQ	DMACSREQ	DMACLBREQ	DMACLSREQ
15	SSP0 transmit	X	X	-	-
15.0	Reserved	-	-	-	-
14.1	SSP0 receive	X	X	-	-
14.0	Reserved	-	-	-	-
13	I2S0 DMA1	X	-	-	-
12	NAND Flash (same as 1)	X	-	-	-
11.1	SSP1 transmit	X	X	-	-
11.0	SPI1 receive and transmit	X	-	-	-
10.1	I2S1 DMA1	X	-	-	-
10.0	14-clock-Uart7 receive	X	-	-	-
9	14-clock-Uart7 transmit	X	-	-	-
8	14-clock-Uart2 receive	X	-	-	-
7	14-clock-Uart2 transmit	X	-	-	-
6	14-clock-Uart1 receive	X	-	-	-
5	14-clock-Uart1 transmit	X	-	-	-
4	SD Card interface receive and transmit	X	X	X	X
3.1	SSP1 receive	X	X	-	-
3.0	SPI2 receive and transmit	X		-	-
2	I2S1 DMA0	X	-	-	-
1	NAND Flash (same as 12)	X	-	-	-
0	I2S0 DMA0	X	-	-	-

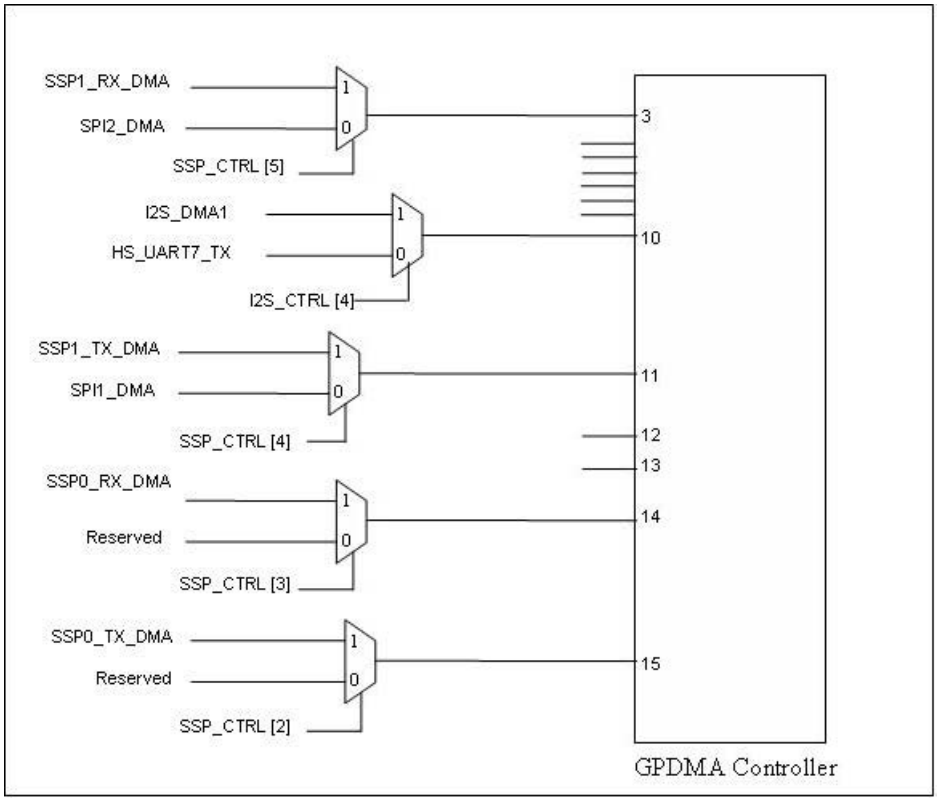


Fig 16. DMA Request line Multiplexing

On some peripherals, the DMACxREQ lines may be multiplexed with other peripherals. This multiplexing is detailed in Table 69 and shown in Figure 16. Note that the procedure for performing a DMA operation requires the additional step of setting the appropriate DMA multiplexing bit in one of two control registers; either SSP_CTRL or I2S_CTRL prior to initiating a transfer.

6.3.2.2 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.

The connection of the DMA Controller to supported peripheral devices is shown in Table 69.

6.4 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 70.

Table 70. Register summary

Address	Name	Description	Reset state	Access
Special registers				
0x4000 4078	SSP_CTRL	SSP Control Register described in another Chapter[1]	0	RW
0x4000 407C	I2S_CTRL	I2S Control Register described in another Chapter[2]	0	RW
General registers				
0x3100 0000	DMACIntStat	DMA Interrupt Status Register	0	RO
0x3100 0004	DMACIntTCStat	DMA Interrupt Terminal Count Request Status Register	0	RO
0x3100 0008	DMACIntTCClear	DMA Interrupt Terminal Count Request Clear Register	-	WO
0x3100 000C	DMACIntErrStat	DMA Interrupt Error Status Register	0	RO
0x3100 0010	DMACIntErrClr	DMA Interrupt Error Clear Register	-	WO
0x3100 0014	DMACRawIntTCStat	DMA Raw Interrupt Terminal Count Status Register	0	RO
0x3100 0018	DMACRawIntErrStat	DMA Raw Error Interrupt Status Register	0	RO
0x3100 001C	DMACEnbldChns	DMA Enabled Channel Register	0	RO
0x3100 0020	DMACSoftBReq	DMA Software Burst Request Register	0	R/W
0x3100 0024	DMACSoftSReq	DMA Software Single Request Register	0	R/W
0x3100 0028	DMACSoftLBReq	DMA Software Last Burst Request Register	0	R/W
0x3100 002C	DMACSoftLSReq	DMA Software Last Single Request Register	0	R/W
0x3100 0030	DMACConfig	DMA Configuration Register	0	R/W
Channel 0 registers				
0x3100 0100	DMACC0SrcAddr	DMA Channel 0 Source Address Register	0	R/W
0x3100 0104	DMACC0DestAddr	DMA Channel 0 Destination Address Register	0	R/W
0x3100 0108	DMACC0LLI	DMA Channel 0 Linked List Item Register	0	R/W
0x3100 010C	DMACC0Control	DMA Channel 0 Control Register	0	R/W
0x3100 0110	DMACC0Config	DMA Channel 0 Configuration Register	0[3]	R/W
Channel 1 registers				
0x3100 0120	DMACC1SrcAddr	DMA Channel 1 Source Address Register	0	R/W
0x3100 0124	DMACC1DestAddr	DMA Channel 1 Destination Address Register	0	R/W
0x3100 0128	DMACC1LLI	DMA Channel 1 Linked List Item Register	0	R/W
0x3100 012C	DMACC1Control	DMA Channel 1 Control Register	0	R/W
0x3100 0130	DMACC1Config	DMA Channel 1 Configuration Register	0[3]	R/W
Channel 2 registers				
0x3100 0140	DMACC2SrcAddr	DMA Channel 2 Source Address Register	0	R/W
0x3100 0144	DMACC2DestAddr	DMA Channel 2 Destination Address Register	0	R/W
0x3100 0148	DMACC2LLI	DMA Channel 2 Linked List Item Register	0	R/W
0x3100 014C	DMACC2Control	DMA Channel 2 Control Register	0	R/W

Table 70. Register summary *…continued*

Address	Name	Description	Reset state	Access
0x3100 0150	DMACC2Config	DMA Channel 2 Configuration Register	0[3]	R/W
Channel 3 registers				
0x3100 0160	DMACC3SrcAddr	DMA Channel 3 Source Address Register	0	R/W
0x3100 0164	DMACC3DestAddr	DMA Channel 3 Destination Address Register	0	R/W
0x3100 0168	DMACC3LLI	DMA Channel 3 Linked List Item Register	0	R/W
0x3100 016C	DMACC3Control	DMA Channel 3 Control Register	0	R/W
0x3100 0170	DMACC3Config	DMA Channel 3 Configuration Register	0[3]	R/W
Channel 4 registers				
0x3100 0180	DMACC4SrcAddr	DMA Channel 4 Source Address Register	0	R/W
0x3100 0184	DMACC4DestAddr	DMA Channel 4 Destination Address Register	0	R/W
0x3100 0188	DMACC4LLI	DMA Channel 4 Linked List Item Register	0	R/W
0x3100 018C	DMACC4Control	DMA Channel 4 Control Register	0	R/W
0x3100 0190	DMACC4Config	DMA Channel 4 Configuration Register	0[3]	R/W
Channel 5 registers				
0x3100 01A0	DMACC5SrcAddr	DMA Channel 5 Source Address Register	0	R/W
0x3100 01A4	DMACC5DestAddr	DMA Channel 5 Destination Address Register	0	R/W
0x3100 01A8	DMACC5LLI	DMA Channel 5 Linked List Item Register	0	R/W
0x3100 01AC	DMACC5Control	DMA Channel 5 Control Register	0	R/W
0x3100 01B0	DMACC5Config	DMA Channel 5 Configuration Register	0[3]	R/W
Channel 6 registers				
0x3100 01C0	DMACC6SrcAddr	DMA Channel 6 Source Address Register	0	R/W
0x3100 01C4	DMACC6DestAddr	DMA Channel 6 Destination Address Register	0	R/W
0x3100 01C8	DMACC6LLI	DMA Channel 6 Linked List Item Register	0	R/W
0x3100 01CC	DMACC6Control	DMA Channel 6 Control Register	0	R/W
0x3100 01D0	DMACC6Config	DMA Channel 6 Configuration Register	0[3]	R/W
Channel 7 registers				
0x3100 01E0	DMACC7SrcAddr	DMA Channel 7 Source Address Register	0	R/W
0x3100 01E4	DMACC7DestAddr	DMA Channel 7 Destination Address Register	0	R/W
0x3100 01E8	DMACC7LLI	DMA Channel 7 Linked List Item Register	0	R/W
0x3100 01EC	DMACC7Control	DMA Channel 7 Control Register	0	R/W
0x3100 01F0	DMACC7Config	DMA Channel 7 Configuration Register	0[3]	R/W

[1] This register is described in either Section 22.5.1 and Section 4.11.26 of this user manual.

[2] This register is described in either Section 24.7.1 and Section 4.11.25 of this user manual.

[3] Bit 17 of this register is a read-only status flag.

6.4.1 DMA Interrupt Status Register (DMACIntStat - 0x3100 0000)

The DMACIntStat Register is read-only and shows the status of the interrupts after masking. A HIGH 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 71 shows the bit assignments of the DMACIntStat Register.

Table 71. DMA Interrupt Status Register (DMACIntStat - 0x3100 0000)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.2 DMA Interrupt Terminal Count Request Status Register (DMACIntTCStat - 0x3100 0004)

The DMACIntTCStat Register is read-only and indicates the status of the terminal count after masking. Table 72 shows the bit assignments of the DMACIntTCStat Register.

Table 72. DMA Interrupt Terminal Count Request Status Register (DMACIntTCStat - 0x3100 0004)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.3 DMA Interrupt Terminal Count Request Clear Register (DMACIntTCClear - 0x3100 0008)

The DMACIntTCClear Register is write-only and clears one or more terminal count interrupt requests. When writing to this register, each data bit that is set HIGH causes the corresponding bit in the status register (DMACIntTCStat) to be cleared. Data bits that are LOW have no effect. Table 73 shows the bit assignments of the DMACIntTCClear Register.

Table 73. DMA Interrupt Terminal Count Request Clear Register (DMACIntTCClear - 0x3100 0008)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.4 DMA Interrupt Error Status Register (DMACIntErrStat - 0x3100 000C)

The DMACIntErrStat Register is read-only and indicates the status of the error request after masking. Table 74 shows the bit assignments of the DMACIntErrStat Register.

Table 74. DMA Interrupt Error Status Register (DMACIntErrStat - 0x3100 000C)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.5 DMA Interrupt Error Clear Register (DMACIntErrClr - 0x3100 0010)

The DMACIntErrClr Register is write-only and clears the error interrupt requests. When writing to this register, each data bit that is HIGH causes the corresponding bit in the status register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register. Table 75 shows the bit assignments of the DMACIntErrClr Register.

Table 75. DMA Interrupt Error Clear Register (DMACIntErrClr - 0x3100 0010)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.6 DMA Raw Interrupt Terminal Count Status Register (DMACRawIntTCStat - 0x3100 0014)

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 HIGH bit indicates that the terminal count interrupt request is active prior to masking. Table 76 shows the bit assignments of the DMACRawIntTCStat Register.

Table 76. DMA Raw Interrupt Terminal Count Status Register (DMACRawIntTCStat - 0x3100 0014)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.7 DMA Raw Error Interrupt Status Register (DMACRawIntErrStat - 0x3100 0018)

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 HIGH bit indicates that the error interrupt request is active prior to masking. Table 77 shows the bit assignments of register of the DMACRawIntErrStat Register.

Table 77. DMA Raw Error Interrupt Status Register (DMACRawIntErrStat - 0x3100 0018)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.8 DMA Enabled Channel Register (DMACEnbldChns - 0x3100 001C)

The DMACEnbldChns Register is read-only and indicates which DMA channels are enabled, as indicated by the Enable bit in the DMACCxConfig Register. A HIGH bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer. Table 78 shows the bit assignments of the DMACEnbldChns Register.

Table 78. DMA Enabled Channel Register (DMACEnbldChns - 0x3100 001C)

Bit	Name	Function
31:8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
7:0	EnabledChannels	Enable status for DMA channels. Each bit represents one channel: 0 - DMA channel is disabled. 1 - DMA channel is enabled.

6.4.9 DMA Software Burst Request Register (DMACSoftBReq - 0x3100 0020)

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 79 shows the bit assignments of the DMACSoftBReq Register.

Table 79. DMA Software Burst Request Register (DMACSoftBReq - 0x3100 0020)

Bit	Name	Function
31:16	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
15:0	SoftBReq	Software burst request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function (refer to Table 69 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.

Note: It is recommended that software and hardware peripheral requests are not used at the same time.

6.4.10 DMA Software Single Request Register (DMACSoftSReq - 0x3100 0024)

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 80 shows the bit assignments of the DMACSoftSReq Register.

Table 80. DMA Software Single Request Register (DMACSoftSReq - 0x3100 0024)

Bit	Name	Function
31:16	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.11 DMA Software Last Burst Request Register (DMACSoftLBReq - 0x3100 0028)

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 81 shows the bit assignments of the DMACSoftLBReq Register.

Table 81. DMA Software Last Burst Request Register (DMACSoftLBReq - 0x3100 0028)

Bit	Name	Function
31:16	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.12 DMA Software Last Single Request Register (DMACSoftLSReq - 0x3100 002C)

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 82 shows the bit assignments of the DMACSoftLSReq Register.

Table 82. DMA Software Last Single Request Register (DMACSoftLSReq - 0x3100 002C)

Bit	Name	Function
31:16	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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.

6.4.13 DMA Configuration Register (DMACConfig - 0x3100 0030)

The DMACConfig 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 83 shows the bit assignments of the DMACConfig Register.

Table 83. DMA Configuration Register (DMACConfig - 0x3100 0030)

Bit	Name	Function
31:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

Table 83. DMA Configuration Register (DMACConfig - 0x3100 0030) *…continued*

Bit	Name	Function
2	M1	AHB Master 1 endianness configuration: 0 = little-endian mode (default). 1 = big-endian mode.
1	M0	AHB Master 0 endianness configuration: 0 = little-endian mode (default). 1 = big-endian mode.
0	E	DMA Controller enable: 0 = disabled (default). Disabling the DMA Controller reduces power consumption. 1 = enabled.

6.4.14 DMA Channel registers

The channel registers are used to program the eight DMA channels. These registers consist of:

- Eight DMACCxSrcAddr Registers.
- Eight DMACCxDestAddr Registers.
- Eight DMACCxLLI Registers.
- Eight DMACCxControl Registers.
- Eight DMACCxConfig Registers.

When performing scatter/gather DMA, the first four of these are automatically updated.

6.4.15 DMA Channel Source Address Registers (DMACCxSrcAddr - 0x3100 01x0)

The eight read/write DMACCxSrcAddr Registers (DMACC0SrcAddr to DMACC7SrcAddr) 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 increments.
- 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 84 shows the bit assignments of the DMACCxSrcAddr Registers.

Table 84. DMA Channel Source Address Registers (DMACCxSrcAddr - 0x3100 01x0)

Bit	Name	Function
31:0	SrcAddr	DMA source address. Reading this register will return the current source address.

6.4.16 DMA Channel Destination Address registers (DMACCxDestAddr - 0x3100 01x4)

The eight read/write DMACCxDestAddr Registers (DMACC0DestAddr to DMACC7DestAddr) 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 increments 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 85 shows the bit assignments of the DMACCxDestAddr Register.

Table 85. DMA Channel Destination Address registers (DMACCxDestAddr - 0x3100 01x4)

Bit	Name	Function
31:0	DestAddr	DMA Destination address. Reading this register will return the current destination address.

6.4.17 DMA Channel Linked List Item registers (DMACCxLLI - 0x3100 01x8)

The eight read/write DMACCxLLI Registers (DMACC0LLI to DMACC7LLI) 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 86 shows the bit assignments of the DMACCxLLI Register.

Table 86. DMA Channel Linked List Item registers (DMACCxLLI - 0x3100 01x8)

Bit	Name	Function
31:2	LLI	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.
1	R	Reserved, and must be written as 0, masked on read.
0	LM	AHB master select for loading the next LLI: 0 - AHB Master 0. 1 - AHB Master 1.

6.4.18 DMA channel control registers (DMACCxControl - 0x3100 01xC)

The eight read/write DMACCxControl Registers (DMACC0Control to DMACC7Control) 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 87 shows the bit assignments of the DMACCxControl Register.

Table 87. DMA channel control registers (DMACCxControl - 0x3100 01xC)

Bit	Name	Function
31	I	Terminal count interrupt enable bit. 0 - the terminal count interrupt is disabled. 1 - the terminal count interrupt is enabled.
30:28	R	Reserved, and must be written as 0, masked on read.
27	DI	Destination increment: 0 - the destination address is not incremented after each transfer 1 - the destination address is incremented after each transfer.
26	SI	Source increment: 0 - the source address is not incremented after each transfer. 1 - the source address is incremented after each transfer.
25	D	Destination AHB master select: 0 - AHB Master 0 selected for destination transfer. 1 - AHB Master 1 selected for destination transfer.
24	S	Source AHB master select: 0 - AHB Master 0 selected for source transfer. 1 - AHB Master 1 selected for source transfer.
23:21	DWidth	Destination transfer width. Transfers wider than the AHB master bus width are not supported. 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
20:18	SWidth	Source transfer width. Transfers wider than the AHB master bus width are illegal. 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

Table 87. DMA channel control registers (DMACCxControl - 0x3100 01xC) *...continued*

Bit	Name	Function
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
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
11:0	TransferSize	Transfer size. A write to this field sets the size of the transfer when the DMA Controller is the flow controller. The transfer size value must be set before the channel is enabled. Transfer size is updated as data transfers are completed. The transfers size is the number of transfers for: size = (current defined source width). For 32-bit transfers, up to 16K bytes can be transferred. For 8-bit transfers, the limit is 4K bytes. note: Transfer bytes = (transfer size * source width in bytes) 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 the DMA Controller is not the flow controller.

6.4.19 Channel Configuration registers (DMACCxConfig - 0x3100 01x0)

The eight DMACCxConfig Registers (DMACC0Config to DMACC7Config) are read/write with the exception of bit[17] which is read-only. Use these registers to configure the specific DMA channel. The registers are not updated when a new LLI is requested. Table 88 shows the bit assignments of the DMACCxConfig Registers.

Table 88. Channel Configuration registers (DMACCxConfig - 0x3100 01x0)

Bit	Name	Function
31:19	Reserved	Reserved, do not modify, masked on read.
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.
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.
16	L	Lock. When set, this bit enables locked transfers.
15	ITC	Terminal count interrupt mask. When cleared, this bit masks out the terminal count interrupt of the relevant channel.
14	IE	Interrupt error mask. When cleared, this bit masks out the error interrupt of the relevant channel.
13:11	FlowCntrl	Flow control and transfer type. This value indicates the flow controller and transfer type. The flow controller can be the DMA Controller, the source peripheral, or the destination peripheral. The transfer type can be memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral. Refer to Table 89 for the encoding of this field.
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 69 for peripheral identification.
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 69 for peripheral identification.
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.

6.4.19.1 Lock control

The lock control may set the lock bit by writing a 1 to bit 16 of the DMACCxConfig Register. When a burst occurs, the AHB arbiter will not de-grant the master during the burst until the lock is deasserted. 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.

6.4.19.2 Flow control and transfer type

Table 89 lists the bit values of the three flow control and transfer type bits identified in Table 88.

Table 89. Flow control and transfer type bits

Bit value	Transfer type	Controller
000	Memory to memory	DMA
001	Memory to peripheral	DMA
010	Peripheral to memory	DMA
011	Source peripheral to destination peripheral	DMA
100	Source peripheral to destination peripheral	Destination peripheral
101	Memory to peripheral	Peripheral
110	Peripheral to memory	Peripheral
111	Source peripheral to destination peripheral	Source peripheral

6.5 Using the DMA controller

6.5.1 DMA efficiency

To optimize performance of the SDRAM when setting up an SDRAM transfer, it must be ensured that:

- The read/write width to/from SDRAM is the maximum possible. In general, word (32 bit) width is recommended.
- Both source and destination are accessed on the same master port. This ensures that the DMA will pump data in AHB bursts of 4 words or 8 halfwords whenever possible.

Note: Because of a limitation in the SDRAM controller, use of byte read/write is not recommended to or from SDRAM. The reason is that the DMA controller does not support an INC16 burst access, which can occur on the DMA master ports when using byte read/write to or from SDRAM.

6.5.2 Programming the DMA controller

All accesses to the DMA Controller internal register must be word (32-bit) reads and writes.

6.5.2.1 Enabling the DMA peripheral

On some peripherals, the DMACxREQ lines are multiplexed. To enable the DMA controller request line for a specific peripheral set the Enable bit in either the SSP_CTRL or I2S_CTRL for the peripheral that will be used, see Figure 16.

6.5.2.2 Disabling the DMA controller

To disable the DMA controller:

- Read the DMACEnbldChns 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 DMACConfig register.

6.5.2.3 Enabling the DMA controller

To enable the DMA controller set the Enable bit in the DMACConfig register.

6.5.2.4 Disabling the DMA controller

To disable the DMA controller:

- Read the DMACEnbldChns 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 DMACConfig register.

6.5.2.5 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.

6.5.2.6 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.

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

6.5.2.7 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 DMACEnbldChns controller register and find out which channels are inactive.
2. Choose an inactive channel that has the required priority.
3. Program the DMA controller

6.5.2.8 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.

6.5.2.9 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 DMACIntTCClear and DMACIntErrClear register. The previous channel operation might have left interrupt active.
3. Write the source address into the DMACCxSrcAddr register.
4. Write the destination address into the DMACCxDestAddr register.
5. Write the address of the next LLI into the DMACCxLLI 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 DMACCxControl register.
7. Write the channel configuration information into the DMACCxConfig register. If the enable bit is set then the DMA channel is automatically enabled.

6.5.3 Flow control

The peripheral 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. If the packet length is unknown when the DMA channel is enabled, either the source or destination peripherals can be used as the flow controller.

For simple or low-performance peripherals that know the packet length (that is, when the peripheral is the flow controller), a simple way to indicate that a transaction has completed is for the peripheral to generate an interrupt and enable the processor to reprogram the DMA channel.

The transfer size value (in the DMACCxControl register) is ignored if a peripheral is configured as the flow controller.

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 can have either the peripheral or the DMA Controller as the flow controller so there are eight possible control scenarios.

Table 90 indicates the request signals used for each type of transfer.

Table 90. DMA request signal usage

Transfer direction	Request generator	Flow controller
Memory-to-peripheral	Peripheral	DMA Controller
Memory-to-peripheral	Peripheral	Peripheral
Peripheral-to-memory	Peripheral	DMA Controller
Peripheral-to-memory	Peripheral	Peripheral
Memory-to-memory	DMA Controller	DMA Controller
Source peripheral to destination peripheral	Source peripheral and destination peripheral	Source peripheral
Source peripheral to destination peripheral	Source peripheral and destination peripheral	Destination peripheral
Source peripheral to destination peripheral	Source peripheral and destination peripheral	DMA Controller

6.5.3.1 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 if the DMA Controller is performing the flow control.
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 DMACCxLLI Register is not 0, then reload the DMACCxSrcAddr, DMACCxDestAddr, DMACCxLLI, and DMACCxControl registers and go to back to step 2. However, if DMACCxLLI is 0, the DMA stream is disabled and the flow sequence ends.

6.5.3.2 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 if the DMA Controller is performing the flow control.
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 performing flow control, or by the 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 DMACCxLLI Register is not 0, then reload the DMACCxSrcAddr, DMACCxDestAddr, DMACCxLLI, and DMACCxControl Registers and go to back to step 2. However, if DMACCxLLI is 0, the DMA stream is disabled and the flow sequence ends.

6.5.3.3 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 DMACCxLLI Register is not 0, then reload the DMACCxSrcAddr, DMACCxDestAddr, DMACCxLLI, and DMACCxControl Registers and go to back to step 2. However, if DMACCxLLI 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.

6.5.4 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 DMACCxControl and DMACCxConfig Channel Registers. Interrupt status registers are provided which group the interrupt requests from all the DMA channels prior to interrupt masking (DMACRawIntTCStat and DMACRawIntErrStat), and after interrupt masking (DMACIntTCStat and DMACIntErrStat). The DMACIntStat Register combines both the DMACIntTCStat and DMACIntErrStat requests into a single register to enable the source of an interrupt to be quickly found. Writing to the DMACIntTCClear or the DMACIntErrClr Registers with a bit set HIGH enables selective clearing of interrupts.

6.5.4.1 Hardware interrupt sequence flow

When a DMA interrupt request occurs, the Interrupt Service Routine needs to:

1. Read the DMACIntTCStat Register to determine whether the interrupt was generated due to the end of the transfer (terminal count). A HIGH 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 DMACIntErrStat Register to determine whether the interrupt was generated due to an error occurring. A HIGH bit indicates that an error occurred.
3. Service the interrupt request.
4. For a terminal count interrupt, write a 1 to the relevant bit of the DMACIntTCClr Register. For an error interrupt write a 1 to the relevant bit of the DMACIntErrClr Register to clear the interrupt request.

6.5.5 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, which 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.

Note: When transferring data to or from the SDRAM, the SDRAM access must always be programmed to 32 bit accesses. The SDRAM memory controller does not support AHB-INCR4 or INCR8 bursts using halfword or byte transfer-size. Start address in SDRAM should always be aligned to a burst boundary address.

6.5.5.1 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 0x0C000024, the next boundary (calculated from the burst size and transfer width) is 0x0C000040.

The transfer will be split into two AHB transactions:

- a 7-transfer burst starting at address 0x0C000024
- a 9-transfer burst starting at address 0x0C000040.

6.5.6 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 DMACCxLLI 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 a LLI (referred to as the packet of data) usually requires one or more DMA bursts (to each of the source and destination).

6.5.6.1 Linked list items

A Linked List Item (LLI) consists of four words. These words are organized in the following order:

1. DMACCxSrcAddr.
2. DMACCxDestAddr.
3. DMACCxLLI.
4. DMACCxControl.

Note: The DMACCxConfig DMA channel Configuration Register is not part of the linked list item.

6.5.6.1.1 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 DMACCxControl 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 DMACIntTCClear Register must be set to clear the interrupt.

6.5.6.1.2 Example of scatter/gather DMA

See Figure 17 for an example of an LLI. A rectangle of memory has to be transferred to a peripheral. The addresses of each line of data are given, in hexadecimal, at the left-hand side of the figure. The LLIs describing the transfer are to be stored contiguously from address 0x20000.

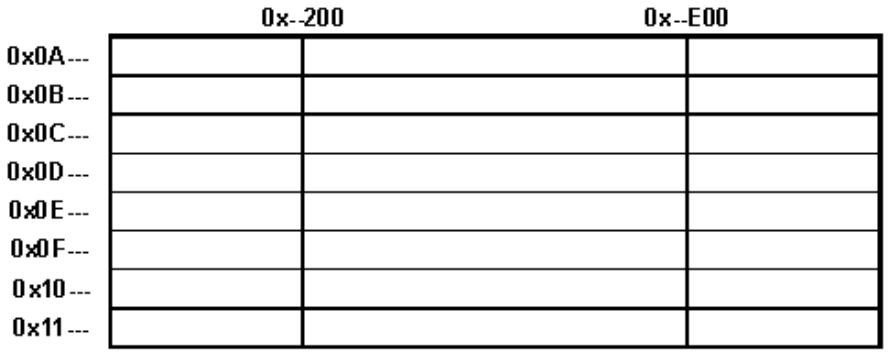


Fig 17. LLI example

The first LLI, stored at 0x20000, defines the first block of data to be transferred, which is the data stored between addresses 0x0A200 and 0x0AE00:

- Source start address 0x0A200.
- 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, 0x20010.

The second LLI, stored at 0x20010, describes the next block of data to be transferred:

- Source start address 0x0B200.
- 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, 0x20020.

A chain of descriptors is built up, each one pointing to the next in the series. To initialize the DMA stream, the first LLI, 0x20000, 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 0x20070 and contains:

- Source start address 0x11200.
- 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.

7.1 Introduction

The LPC32x0 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(SDR) and Double Data Rate(DDR) SDRAM. The EMC is an Advanced Microcontroller Bus Architecture (AMBA) compliant peripheral

7.2 Features of the EMC

- Dynamic memory interface support including Single Data Rate and Double Data Rate SDRAM.
- Supports mobile SDRAM devices with 1.8 V I/O interface.
- 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 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
- Power-saving modes dynamically control clock and clock enable to SDRAMs.
- Dynamic memory self-refresh mode controlled by software.
- The EMC supports 2 k, 4 k, and 8 k row address synchronous memory devices, which are typically 512 Mbit, 256 Mbit, and 128 Mbit devices, with 4, 8, 16, or 32 data bits per device.
- Separate reset domains allow the for auto-refresh through a chip reset if desired.
- Four chip selects for static memory devices.
- Two chip selects for synchronous memory devices.

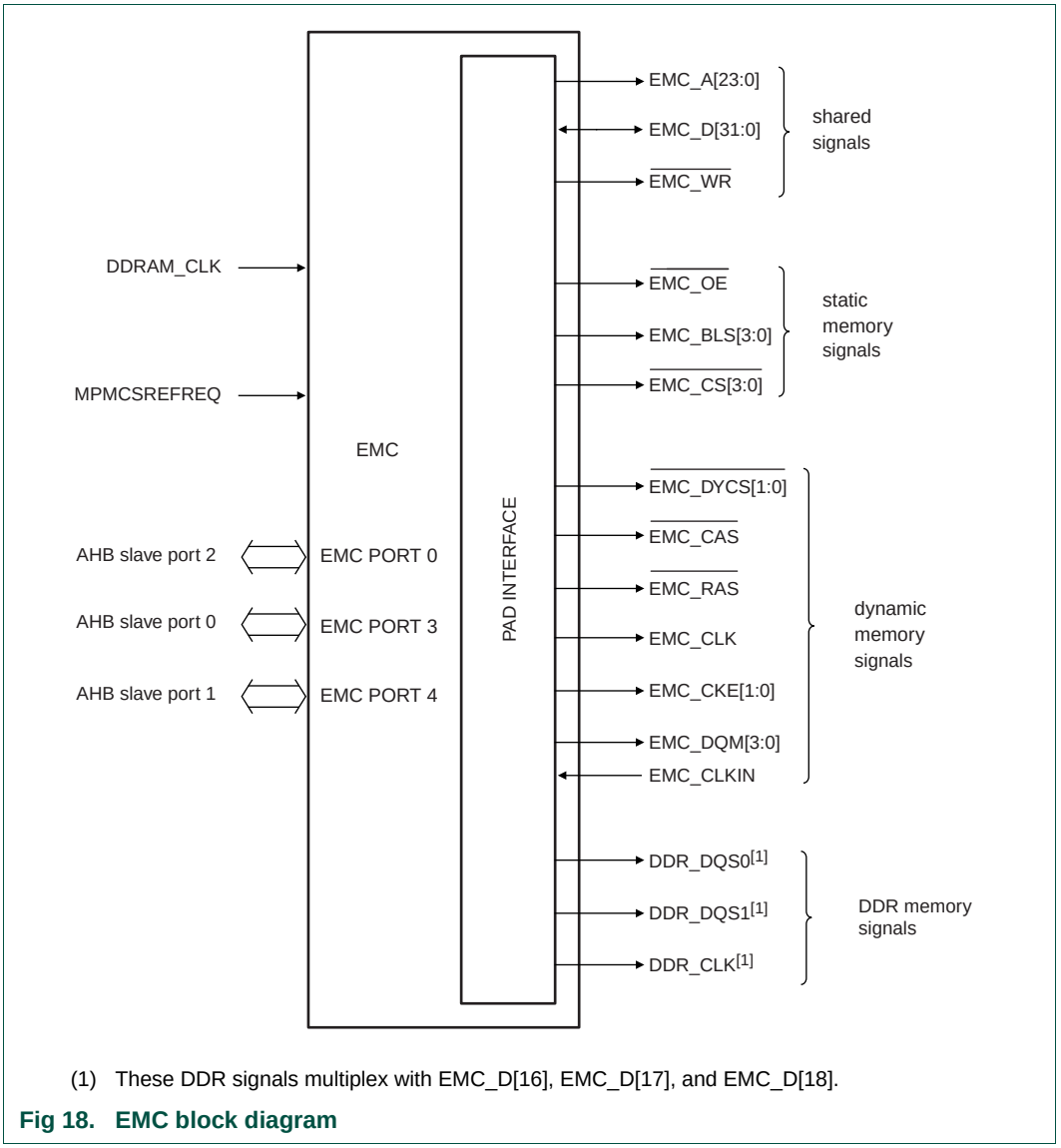
Note: Synchronous static memory devices (synchronous burst mode) are not supported.

7.3 EMC pins

The EMC supports an SDR SDRAM memory bus up to 32-bits wide or a 16-bit DDR SDRAM bus. Additional signals are required for DDR SDRAM, which are brought out on the same pins as EMC_D[16:18]. In DDR mode, EMC_D bits 19 through 31 may be used as additional parallel I/O pins P2[12:0]. EMC pins are shown in both Table 91 and Figure 18.

Table 91. EMC pins in SRAM, SDR and DDR operating modes

EMC interface pin(s)	Static RAM function	SDR SDRAM function	DDR SDRAM function
EMC_A[00] - EMC_A[12]	Address bus, bits 0 to 12	Address bus, bits 0 to 12	Address bus, bits 0 to 12
EMC_A[13]	Address bus, bit 13	Address bus, BA 0	Address bus, BA 0
EMC_A[14]	Address bus, bit 14	Address bus, BA 1	Address bus, BA 1
EMC_A[15] - EMC_A[23]	Address bus, bit 15 to 23	-	-
EMC_D[00] - EMC_D[15]	Data bus, bits 0 to 15	Data bus, bits 0 to 15	Data bus, bits 0 to 15
EMC_D[16] / EMC_DQS0	Data bus, bit 16	Data bus, bit 16	Data strobe, lower byte
EMC_D[17] / EMC_DQS1	Data bus, bit 17	Data bus, bit 17	Data strobe, upper byte
EMC_D[18] / EMC_CLK_N	Data bus, bit 18	Data bus, bit 18	Inverted SDRAM clock
EMC_D[19] - EMC_D[31]	Data bus bits 19 through 31	Data bus, bits 19 through 31	P2[12:0]
EMC_OE_N	SRAM Output Enable		
EMC_BLS[0] - EMC_BLS[3]	SRAM byte lane select 0-3		
EMC_CS[0]_N - EMC_CS[3]_N	SRAM chip select 0-3		
EMC_WR_N	SRAM write strobe	SDRAM write strobe	SDRAM write strobe
EMC_CLK		SDRAM clock	SDRAM clock
EMC_CLKIN		SDRAM clock feedback	
EMC_CKE		SDRAM clock enable	SDRAM clock enable
EMC_DYCS[0]_N, EMC_DYCS[1]_N		SDRAM chip select 0-1	SDRAM chip select 0-1
EMC_CAS_N		SDRAM column address strobe	SDRAM column address strobe
EMC_RAS_N		SDRAM row address strobe	SDRAM row address strobe
EMC_DQM[0] - EMC_DQM[3]		SDRAM byte write mask 0 through 3	EMC_DQM0 = LDM EMC_DQM1 = UDM



(1) These DDR signals multiplex with EMC_D[16], EMC_D[17], and EMC_D[18].

Fig 18. EMC block diagram

7.4 Bus hold circuits

In SDR SDRAM mode, all data bus pins (EMC_D[31:0] are configured to have bus hold circuits. These cause the pins to retain the last logic level that was driven. In DDR SDRAM mode, the bus hold configuration remains the same except that the inverted clock output (the EMC_D[18] / EMC_CLK_N pin) has the bus hold circuit turned off.

Table 92 shows the overall configuration of bus hold circuits when the entire data bus is configured for SDRAM operation via the EMC_D_SEL bit. The EMC_D_SEL control bit may be read as bit 3 of P2_MUX_STATE register, described in the GPIO chapter. The value of EMC_D_SEL is controlled by the P2_MUX_SET and P2_MUX_CLR registers. The DDR_SEL control bit is bit 1 of the SDRAMCLK_CTRL register, described elsewhere in this chapter.

When the upper data bus (EMC_D[31:19]) is configured for GPIO operation, the bus hold circuits for those pins are disabled.

Table 92. Bus hold configuration for EMC_D[31:0] when EMC_D_SEL = '0'

Bus pin(s)	Bus hold when DDR_SEL = '0'	Bus hold when DDR_SEL = '1'
EMC_D[31:19]	On	On
EMC_D[18]	On	Off
EMC_D[17:0]	On	On

7.5 Supported memory devices

The EMC supports a wide variety of SDRAM configurations. However, the 1.8 V interface levels are primarily supported by 'Mobile' SDRAMs. This section provides examples of dynamic memory devices that are supported by the EMC. Table 93 and Table 94 show SDR and DDR SDRAM devices respectively.

Table 93. Examples of compatible SDR SDRAM devices[1][2]

Manufacturer	Part number	Size	Organization
Micron	MT48H4M16LF	64 Mb	4M x 16
Samsung	K4M64163PH	64 Mb	4M x 16
Micron	MT48H8M16LF	128 Mb	8M x 16
Samsung	K4M28163PF	128 Mb	8M x 16
Infineon	HYB18L128160	128 Mb	8M x 16
Infineon	HYE18L128160	128 Mb	8M x 16
Micron	MT48H8M32LF	256 Mb	8M x 32
Micron	MT48H16M16LF	256 Mb	16M x 16
Samsung	K4S56163PF	256 Mb	16M x 16
Infineon	HYB18L256160	256 Mb	16M x 16
Infineon	HYE18L256160	256 Mb	16M x 16
Hynix	HY5S5B6ELF	256 Mb	16M x 16
Micron	MT48H32M16LF	512 Mb	32M x 16
Samsung	K4S51163PF	512 Mb	32M x 16

[1] This table is not intended to be an exhaustive list of supported devices.

[2] Devices listed in this table have been selected by comparing manufacturer data sheet specifications to the features of the EMC, and have not been tested in a system.

Table 94. Examples of compatible DDR SDRAM devices[1][2]

Manufacturer	Part number	Size	Organization
Micron	MT46H8M16LF	128 Mb	8M x 16
Micron	MT46H16M16LF	256 Mb	16M x 16
Hynix	HY5MS5B6LF	256 Mb	16M x 16
Micron	MT46H32M16LF	512 Mb	32M x 16

[1] This table is not intended to be an exhaustive list of supported devices.

[2] Devices listed in this table have been selected by comparing manufacturer data sheet specifications to the features of the EMC, and have not been tested in a system.

7.6 SDRAM self-refresh mode

The SDRAM has logic to determine when it should go in and out of self-refresh mode. This is described in the PWR_CTRL register description in the Clocking and Power Control chapter. The SR bit in the EMCDynamicControl register must always be written to '0', note that the default value is '1'.

7.7 Register description

This section describes the EMC registers and provides information needed to program it to interface the controller and external memory. The EMC registers are shown in Table 95.

Table 95. EMC register summary

Address	Register name	Description	Reset value	Type
0x4000 4068	SDRAMCLK_CTRL	Controls various SDRAM configuration details.	0	R/W
0x3108 0000	EMCControl	Controls operation of the EMC.	0x3	R/W
0x3108 0004	EMCStatus	Provides EMC status information.	0x5	RO
0x3108 0008	EMCConfig	Configures operation of the EMC.	0	R/W
0x3108 0020	EMCDynamicControl	Controls dynamic memory operation.	0x006	R/W
0x3108 0024	EMCDynamicRefresh	Configures dynamic memory refresh operation.	0	R/W
0x3108 0028	EMCDynamicReadConfig	Configures the dynamic memory read strategy.	0	R/W
0x3108 0030	EMCDynamictRP	Selects the precharge command period.	0x0F	R/W
0x3108 0034	EMCDynamictRAS	Selects the active to precharge command period.	0xF	R/W
0x3108 0038	EMCDynamictSREX	Selects the self-refresh exit time.	0xF	R/W
0x3108 0044	EMCDynamictWR	Selects the write recovery time.	0xF	R/W
0x3108 0048	EMCDynamictRC	Selects the active to active command period.	0x1F	R/W
0x3108 004C	EMCDynamictRFC	Selects the auto-refresh period.	0x1F	R/W
0x3108 0050	EMCDynamictXSR	Selects the exit self-refresh to active command time	0x1F	R/W
0x3108 0054	EMCDynamictRRD	Selects the active bank A to active bank B latency	0xF	R/W
0x3108 0058	EMCDynamictMRD	Selects the load mode register to active command time	0xF	R/W
0x3108 005C	EMCDynamictCDLR	Selects the last data in to read command time.	0xF	R/W
0x3108 0080	EMCStaticExtendedWait	Selects time for long static memory read and write transfers.	0xF	R/W
0x3108 0100	EMCDynamicConfig0	Selects the configuration information for the SDRAM.	0	R/W
0x3108 0104	EMCDynamicRasCas0	Selects the RAS and CAS latencies for the SDRAM.	0x303	R/W
0x3108 0120	EMCDynamicConfig1	Selects the configuration information for the SDRAM.*	0	R/W
0x3108 0124	EMCDynamicRasCas1	Selects the RAS and CAS latencies for the SDRAM.*	0x303	R/W
0x3108 0200	EMCStaticConfig0	Selects the memory configuration for static chip select 0.	0x2[1]	R/W
0x3108 0204	EMCStaticWaitWen0	Selects the delay from chip select 0 to write enable.	0x0	R/W
0x3108 0208	EMCStaticWaitOen0	Selects the delay from chip select 0 or address change, whichever is later, to output enable.	0x0	R/W
0x3108 020C	EMCStaticWaitRd0	Selects the delay from chip select 0 to a read access.	0x1F	R/W
0x3108 0210	EMCStaticWaitPage0	Selects the delay for asynchronous page mode sequential accesses for chip select 0.	0x1F	R/W
0x3108 0214	EMCStaticWaitWr0	Selects the delay from chip select 0 to a write access.	0x1F	R/W

Table 95. EMC register summary *…continued*

Address	Register name	Description	Reset value	Type
0x3108 0218	EMCStaticWaitTurn0	Selects the number of bus turnaround cycles for chip select 0.	0xF	R/W
0x3108 0220	EMCStaticConfig1	Selects the memory configuration for static chip select 1.	0x2	R/W
0x3108 0224	EMCStaticWaitWen1	Selects the delay from chip select 1 to write enable.	0x0	R/W
0x3108 0228	EMCStaticWaitOen1	Selects the delay from chip select 1 or address change, whichever is later, to output enable.	0x0	R/W
0x3108 022C	EMCStaticWaitRd1	Selects the delay from chip select 1 to a read access.	0x1F	R/W
0x3108 0230	EMCStaticWaitPage1	Selects the delay for asynchronous page mode sequential accesses for chip select 1.	0x1F	R/W
0x3108 0234	EMCStaticWaitWr1	Selects the delay from chip select 1 to a write access.	0x1F	R/W
0x3108 0238	EMCStaticWaitTurn1	Selects the number of bus turnaround cycles for chip select 1.	0xF	R/W
0x3108 0240	EMCStaticConfig2	Selects the memory configuration for static chip select 2.	0x2	R/W
0x3108 0244	EMCStaticWaitWen2	Selects the delay from chip select 2 to write enable.	0x0	R/W
0x3108 0248	EMCStaticWaitOen2	Selects the delay from chip select 2 or address change, whichever is later, to output enable.	0x0	R/W
0x3108 024C	EMCStaticWaitRd2	Selects the delay from chip select 2 to a read access.	0x1F	R/W
0x3108 0250	EMCStaticWaitPage2	Selects the delay for asynchronous page mode sequential accesses for chip select 2.	0x1F	R/W
0x3108 0254	EMCStaticWaitWr2	Selects the delay from chip select 2 to a write access.	0x1F	R/W
0x3108 0258	EMCStaticWaitTurn2	Selects the number of bus turnaround cycles for chip select 2.	0xF	R/W
0x3108 0260	EMCStaticConfig3	Selects the memory configuration for static chip select 3.	0x2	R/W
0x3108 0264	EMCStaticWaitWen3	Selects the delay from chip select 3 to write enable.	0x0	R/W
0x3108 0268	EMCStaticWaitOen3	Selects the delay from chip select 3 or address change, whichever is later, to output enable.	0x0	R/W
0x3108 026C	EMCStaticWaitRd3	Selects the delay from chip select 3 to a read access.	0x1F	R/W
0x3108 0270	EMCStaticWaitPage3	Selects the delay for asynchronous page mode sequential accesses for chip select 3.	0x1F	R/W
0x3108 0274	EMCStaticWaitWr3	Selects the delay from chip select 3 to a write access.	0x1F	R/W
0x3108 0278	EMCStaticWaitTurn3	Selects the number of bus turnaround cycles for chip select 3.	0xF	R/W
0x3108 0400	EMCAHBControl0	Control register for AHB port 0.	0	R/W
0x3108 0404	EMCAHBStatus0	Status register for AHB port 0.	0	R/W
0x3108 0408	EMCAHBTimeOut0	Timeout register for AHB port 0.	0	R/W
0x3108 0460	EMCAHBControl3	Control register for AHB port 3.	0	R/W
0x3108 0464	EMCAHBStatus3	Status register for AHB port 3.	0	R/W
0x3108 0468	EMCAHBTimeOut3	Timeout register for AHB port 3.	0	R/W
0x3108 0480	EMCAHBControl4	Control register for AHB port 4.	0	R/W
0x3108 0484	EMCAHBStatus4	Status register for AHB port 4.	0	R/W
0x3108 0488	EMCAHBTimeOut4	Timeout register for AHB port 4.	0	R/W
0x4000 406C	DDR_LAP_NOM	Contains the nominal value for DDR DQS input delay.	0	R/W
0x4000 4070	DDR_LAP_COUNT	Value of the DDR SDRAM ring oscillator counter.	0	RO
0x4000 4074	DDR_CAL_DELAY	Current calibrated value of the DDR DQS input delay.	0	RO

[1] This value may be changed to 0x82 by the bootloader if it runs EMC boot procedure, see Section 35.2.2.1

7.7.1 SDRAM Clock Control Register (SDRAMCLK_CTRL - 0x4000 4068)

The SDRAMCLK_CTRL register controls the enable, reset, and timing of the SDRAM interface.

Table 96. SDRAM Clock Control Register (SDRAMCLK_CTRL - 0x4000 4068)

Bit	Function	Reset value
22	SDRAM_PIN_SPEED3. This signal controls the slew rate of the pin SDRAM pin EMC_CLK. See bit 20 for details. 0 = Fast slew rate. 1 = Slower slew rate. also, see Notes for bit 20.	0
21	SDRAM_PIN_SPEED2. This signal controls the slew rate of the pins SDRAM pads EMC_A[14:0], EMC_CKE, EMC_CS_N, EMC_RAS_N, EMC_CAS_N, and EMC_WR_N. 0 = Fast slew rate. 1 = Slower slew rate. also, see Notes for bit 20.	0
20	SDRAM_PIN_SPEED1. This signal controls the slew rate of the pins SDRAM pads EMC_D[31:0], and EMC_DQM[3:0]. This control bit slows the output driver down to reduce switching noise. Normally fast slew rate is used. 0 = Fast slew rate. 1 = Slower slew rate. Notes: When using a 1.8V I/O supply for the memory interface set this control signal LOW (0). When using a 2.5/3.3 V I/O supply for the memory interface set this control signal HIGH (1). Setting the control signal HIGH while operating the bus using a 1.8V I/O supply voltage yields a low-noise, low-speed operating mode for the interface.	0
19	SW_DDR_RESET. writing a 1 applies a reset is to the EMC, and it must be set back to 0. This may be used when the EMC is in DDR mode and the clocks are not properly synchronized when starting and stopping clocks. Note: DDRAM_CLK must not be running while resetting the EMC (HCLKDIV_CTRL[8:7] must be [00]) 0 = No EMC reset. 1 = Active EMC reset.	0
18:14	COMMAND_DELAY. The register is used to delay command, data and address signals to SDRAM relative to EMC_CLK. Note that all timing is for nominal process, temperature, voltage. The value of COMMAND_DELAY is set by software using the Ring oscillator to determine the current DDR_LAP_COUNT and this equation. $\text{COMMAND\_DELAY} = \frac{\text{DDR\_LAP\_COUNT}}{\text{DDR\_LAP\_NOM} \times 0.25} \times \text{DESIRED\_DELAY}$ DESIRED_DELAY is desired delay in 0.25 ns increments. Note: All bit combinations can be used. Max delay is 7.75 ns	0
13	Delay circuitry Adder status. Reading a 1 here means that a value too close to min/max has been programmed in DDR_CAL_DELAY or the sensitivity has been programmed too high in SDRAMCLK_CTRL[12:10] 0 = No overflow or sign bit. 1 = Last calibration produced either an overflow or a negative number (underflow).	0

Table 96. SDRAM Clock Control Register (SDRAMCLK_CTRL - 0x4000 4068) *…continued*

Bit	Function	Reset value
12:10	SENSITIVITY_FACTOR for DDR SDRAM calibration. This value controls how much the error value is shifted down. More shifting means less sensitivity of the calibration. 000 = No right shift. …. 111 = Shift right with 7.	0
9	CAL_DELAY. 0 = Use un-calibrated delay settings for DDR SDRAM. 1 = Use calibrated delay settings for DDR SDRAM.	0
8	SW_DDR_CAL. When writing from 0 to 1 a DDR calibration is performed. Must be set back to 0. 0 = No manual DDR delay calibration. 1 = Perform a DDR delay calibration.	0
7	RTC_TICK_EN 0 = No automatic DDR delay calibration. 1 = Enable automatic DDR delay calibration on each RTC TICK.	0
6:2	DDR_DQSIN_DELAY. These bits control the delay of the DQS input from the DDR SDRAM device. The DQS signal is used to capture read data from SDRAM. Note that all timing is for nominal process, temperature, voltage. The timing must be calibrated by software using the Ring Oscillator. Refer to DDR DQS_DELAY[6] calibration for details. Note: All bit combinations can be used. Max delay is 7.75 ns.	0
1	DDR_SEL. This affects the pin multiplexing as described elsewhere in this chapter. 0 = SDR SDRAM is used. 1 = DDR SDRAM is used. In this mode, the DQS delay circuitry is also enabled.	0
0	0 = SDRAM HCLK and Inverted HCLK enabled. 1 = All Clocks to EMC block disabled. Note that no masters can access the EMC in this mode.	0

7.7.2 EMC Control Register (EMCControl - 0x3108 0000)

The EMCControl register is a read/write register that controls operation of the memory controller. This register must only be written while the EMC is in the idle state. Table 97 shows the bit assignments for the EMCControl register.

Table 97. EMC Control Register (EMCControl - 0x3108 0000)

Bits	Name	Type	Function
31:3	Reserved	-	Reserved, read undefined, do not modify.

Table 97. EMC Control Register (EMCControl - 0x3108 0000) *…continued*

Bits	Name	Type	Function
2	Low-power mode (L)	R/W	Indicates normal, or low-power mode: 0 = normal. 1 = low-power mode. Entering low-power mode reduces EMC power consumption. Dynamic memory is refreshed as necessary. The EMC returns to normal functional mode by clearing the low-power mode bit (L), or by Reset. This bit must only be modified when the EMC is in idle state.[1]
1	Reserved	-	Reserved, read undefined, do not modify.
0	EMC Enable (E)	R/W	Indicates if the EMC is enabled or disabled: 0 = disabled. 1 = enabled. Disabling the EMC reduces power consumption. When the EMCr is disabled the memory is not refreshed. The EMCr 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] 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.

7.7.3 EMC Status Register (EMCStatus - 0x3108 0004)

The read-only EMCStatus register provides EMC status information. Table 98 shows the bit assignments for the EMCStatus register.

Table 98. EMC Status Register (EMCStatus - 0x3108 0004)

Bits	Name	Type	Description
31:3	Reserved		Reserved, read undefined.
2	Self-refresh acknowledge (SA)	RO	This bit indicates the operating mode of the EMC: 0 = normal mode 1 = self-refresh mode.
1	Reserved	-	Reserved, read undefined, do not modify.
0	Busy (B)	RO	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: 0 = EMC is idle. 1 = EMC is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode.

7.7.4 EMC Configuration Register (EMCConfig - 0x3108 0008)

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 99 shows the bit assignments for the EMCConfig register.

Table 99. EMC Configuration Register (EMCConfig - 0x3108 0008)

Bits	Name	Type	Description
31:1	Reserved	-	Reserved, read undefined, do not modify.
0	Endian mode (N)	R/W	Endian mode: 0 = little-endian mode. 1 = big-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.

7.7.5 Dynamic Memory Control Register (EMCDynamicControl - 0x3108 0020)

The EMCDynamicControl register controls dynamic memory operation. The control bits can be altered during normal operation. Table 100 shows the bit assignments for the EMCDynamicControl register.

Table 100. Dynamic Memory Control Register (EMCDynamicControl - 0x3108 0020)

Bits	Name	Type	Description
31:14	Reserved	-	Reserved, read undefined, do not modify.
13	Low-power SDRAM deep-sleep mode (DP)	R/W	0 = normal operation. 1 = enter deep power down mode.[4]
12:9	Reserved	-	Reserved, read undefined, do not modify.
8:7	SDRAM initialization (I)	R/W	00 = issue SDRAM NORMAL operation command. 01 = issue SDRAM MODE command.[1] 10 = issue SDRAM PALL (precharge all) command. 11 = issue SDRAM NOP (no operation) command).
6	Reserved	-	Reserved, read undefined, do not modify.
5	Memory clock control (MMC)	R/W	0 = EMC_CLK enabled (POR reset value). 1 = EMC_CLK disabled.[2]
4	Inverted Memory Clock Control (IMCC)	R/W	0 = EMC_CLK_N enabled. 1 = DDR_CLK_N disabled.
3	Self-Refresh Clock Control (SRMCC)	R/W	0 = EMC_CLK and EMC_CLK_N run continuously during self-refresh mode. 1 = EMC_CLK and EMC_CLK_N are stopped during self-refresh mode.
2	Self-refresh request, EMCSREFREQ (SR)	R/W	0 = normal mode. Note: this bit must be 0 for correct operation, note default is ‘1’.
1	Dynamic memory clock control (CS)	R/W	0 = EMC_CLK stops when all SDRAMs are idle and during self-refresh mode. 1 = EMC_CLK runs continuously.[4] When clock control is LOW the output clock EMC_CLK is stopped when there are no SDRAM transactions. The clock is also stopped during self-refresh mode.
0	Dynamic memory clock enable (CE)	R/W	0 = clock enable of idle devices are deasserted to save power. 1 = all clock enables are driven HIGH continuously.[3][4]

[1] For SDRAM chip selects that are configured for 32-bit wide transfers, single SDRAM bursts are used. When SDRAM chip selects are configured for 16-bit wide transfers, a burst length of 2 is used. Mode registers in related SDRAM devices must be programmed accordingly.

[2] Disabling EMC_CLK or EMC_CLK_N can be performed if there are no SDRAM memory transactions in progress. When enabled this bit can be used in conjunction with the dynamic memory clock control (CS) field.

[3] Clock enable must be HIGH during SDRAM initialization.

[4] To put low-power SDRAM device(s) into deep power down mode, the Low-power SDRAM deep-sleep mode (DP) bit, the Dynamic memory clock control (CS) bit, and the Dynamic memory clock enable (CE) bit must be set.

7.7.6 Dynamic Memory Refresh Timer Register (EMCDynamicRefresh - 0x3108 0024)

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 101 shows the bit assignments for the EMCDynamicRefresh register.

Table 101. Dynamic Memory Refresh Timer Register (EMCDynamicRefresh - 0x3108 0024)

Bits	Name	Type	Description
31:11	Reserved	-	Reserved, read undefined, do not modify.
10:0	Refresh timer (REFRESH)	R/W	Indicates the multiple of 16 clocks between SDRAM refresh cycles. 0x0 = refresh disabled. 0x1 - 0x7FF = n × 16 = 16n clocks between SDRAM refresh cycles. For example: 0x1 = 1 × 16 = 16 clocks between SDRAM refresh cycles. 0x8 = 8 × 16 = 128 clocks between SDRAM refresh cycles.

For example, for the refresh period of 16 μs, and a clock frequency of 50 MHz, the following value must be programmed into this register:

$(16 \times 10^{-6} \times 50 \times 10^{6}) / 16 = 50$ or 0x32

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.

7.7.7 Dynamic Memory Read Configuration Register (EMCDynamicReadConfig - 0x3108 0028)

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. Therefore the worst case value for all of the chip selects must be programmed. Table 102 shows the bit assignments for the EMCDynamicReadConfig register.

Table 102. Dynamic Memory Read Configuration Register (EMCDynamicReadConfig - 0x3108 0028)

Bits	Name	Type	Description
31:13	Reserved	-	Reserved, read undefined, do not modify.
12	DDR_DRP	R/W	DDR SDRAM read data capture polarity 0 = data captured on the negative edge of HCLK. 1 = data captured on the positive edge of HCLK.
11:10	Reserved	-	Reserved, read undefined, do not modify.
9:8	DDR_DRD	R/W	DDR SDRAM read data strategy 00 = reserved, do not use. 01 = command delayed by COMMAND_DELAY time. 10 = reserved, do not use. 11 = reserved, do not use.
7:5	Reserved	-	Reserved, read undefined, do not modify.
4	SDR_SRP	R/W	SDR SDRAM read data capture polarity 0 = data captured on the negative edge of HCLK. 1 = data captured on the positive edge of HCLK.
3:2	Reserved	-	Reserved, read undefined, do not modify.
1:0	SDR_SRD	R/W	SDR SDRAM read data strategy 00 = reserved, do not use. 01 = command delayed by COMMAND_DELAY time. 10 = reserved, do not use. 11 = reserved, do not use.

7.7.8 Dynamic Memory Precharge Command Period Register (EMCDynamictRP - 0x3108 0030)

The EMCDynamictRP register enables programming of the precharge command period, tRP. 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 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 103 shows the bit assignments for the EMCDynamictRP register.

Table 103. Dynamic Memory Precharge Command Period Register (EMCDynamictRP - 0x3108 0030)

Bits	Name	Type	Description
31:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	Precharge command period (tRP)	R/W	0x0 - 0xE = n + 1 clock cycles. 0xF = 16 clock cycles.

7.7.9 Dynamic Memory Active to Precharge Command Period Register (EMCDynamictRAS - 0x3108 0034)

The EMCDynamictRAS register enables programming of 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 104 shows the bit assignments for the EMCDynamictRAS register.

Table 104. Dynamic Memory Active to Precharge Command Period Register (EMCDynamictRAS - 0x3108 0034)

Bits	Name	Type	Description
31:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	Active to precharge command period (tRAS)	R/W	0x0 - 0xE = n + 1 clock cycles. 0xF = 16 clock cycles.

7.7.10 Dynamic Memory Self-refresh Exit Time Register (EMCDynamictSREX - 0x3108 0038)

The EMCDynamictSREX register enables programming of the self-refresh exit time, tSREX. It is recommended that software modify this register 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. For some DDR-SDRAM data sheets, this parameter is known as tXSNR. 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 105 shows the bit assignments for the EMCDynamictSREX register.

Table 105. Dynamic Memory Self-refresh Exit Time Register (EMCDynamictSREX - 0x3108 0038)

Bits	Name	Type	Description
31:7	Reserved	-	Reserved, read undefined, do not modify.
6:0	Self-refresh exit time (tSREX)	R/W	0x0 - 0x7E = n + 1 clock cycles. 0x7F = 128 clock cycles.

7.7.11 Dynamic Memory Write Recovery Time Register (EMCDynamictWR - 0x3108 0044)

The EMCDynamictWR register enables programming of 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 106 shows the bit assignments for the EMCDynamictWR register.

Table 106. Dynamic Memory Write Recovery Time Register (EMCDynamictWR - 0x3108 0044)

Bits	Name	Type	Description
31:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	Write recovery time (t_{WR})	R/W	0x0 - 0xE = n + 1 clock cycles. 0xF = 16 clock cycles.

7.7.12 Dynamic Memory Active To Active Command Period Register (EMCDynamictRC - 0x3108 0048)

The EMCDynamictRC register enables programming of 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 107 shows the bit assignments for the EMCDynamictRC register.

Table 107. Dynamic Memory Active To Active Command Period Register (EMCDynamictRC - 0x3108 0048)

Bits	Name	Type	Description
31:5	Reserved	-	Reserved, read undefined, do not modify.
4:0	Active to active command period (t_{RC})	R/W	0x0 - 0x1E = n + 1 clock cycles. 0x1F = 32 clock cycles.

7.7.13 Dynamic Memory Auto-refresh Period Register (EMCDynamictRFC - 0x3108 004C)

The EMCDynamictRFC register enables programming of 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 108 shows the bit assignments for the EMCDynamictRFC register.

Table 108. Dynamic Memory Auto-refresh Period Register (EMCDynamictRFC - 0x3108 004C)

Bits	Name	Type	Description
31:5	Reserved	-	Reserved, read undefined, do not modify.
4:0	Auto-refresh period and auto-refresh to active command period (t_{RFC})	R/W	0x0 - 0x1E = n + 1 clock cycles. 0x1F = 32 clock cycles.

7.7.14 Dynamic Memory Exit Self-refresh Register (EMCDynamictXSR - 0x3108 0050)

The EMCDynamictXSR register enables programming of 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, but is sometimes called tXSNR in some DDR SDRAM data sheets. 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 109 shows the bit assignments for the EMCDynamictXSR register.

Table 109. Dynamic Memory Exit Self-refresh Register (EMCDynamictXSR - 0x3108 0050)

Bits	Name	Type	Description
31:8	Reserved	-	Reserved, read undefined, do not modify.
7:0	Exit self-refresh to active command time (t_{XSR})	R/W	0x0 - 0xFE = n + 1 clock cycles. 0xFF = 256 clock cycles.

7.7.15 Dynamic Memory Active Bank A to Active Bank B Time Register (EMCDynamictRRD - 0x3108 0054)

The EMCDynamictRRD register enables programming of 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 110 shows the bit assignments for the EMCDynamictRRD register.

Table 110. Dynamic Memory Active Bank A to Active Bank B Time Register (EMCDynamictRRD - 0x3108 0054)

Bits	Name	Type	Description
31:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	Active bank A to active bank B latency (t_{RRD})	R/W	0x0 - 0xE = n + 1 clock cycles. 0xF = 16 clock cycles.

7.7.16 Dynamic Memory Load Mode Register To Active Command Time (EMCDynamictMRD - 0x3108 0058)

The EMCDynamictMRD register enables setting 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 111 shows the bit assignments for the EMCDynamictMRD register.

Table 111. Dynamic Memory Load Mode Register To Active Command Time (EMCDynamictMRD - 0x3108 0058)

Bits	Name	Type	Description
31:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	Load mode register to active command time (t_{MRD})	R/W	0x0 - 0xE = n + 1 clock cycles. 0xF = 16 clock cycles.

7.7.17 Dynamic Memory Last Data In to Read Command Time (EMCDynamicCDLR - 0x3108 005C)

The EMCDynamicCDLR register enables setting the last data in to read command time, tCDLR. 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 tCDLR. This register is accessed with one wait state.

Table 112 shows the bit assignments for the EMCDynamictCDLR register.

Table 112. Dynamic Memory Last Data In to Read Command Time (EMCDynamicCDLR - 0x3108 005C)

Bits	Name	Type	Description
31:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	Last data in to read command time (t_{CDLR})	R/W	0x0 - 0xE = n + 1 clock cycles. 0xF = 16 clock cycles.

7.7.18 Static Memory Extended Wait register (EMCStaticExtendedWait - 0x3108 0080)

When the EW bit of the EMCStatic Config Registers is enabled, the 10-bit, read/write, EMCStaticExtendedWait Register is used to time long static memory read and write transfers that are longer than can be supported by the EMCStaticWaitRd0-3 or EMC StaticWaitWr0-3 Registers. There is only a single EMCStaticExtendedWait Register.

This is used by the relevant static memory chip select if the appropriate ExtendedWait (EW) bit in the EMCStaticConfig0-3 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 113 shows the bit assignments for the EMCStaticExtendedWait registers.

Table 113. Static Memory Extended Wait register (EMCStaticExtendedWait - address 0x3108 0080) bit description

Bit	Symbol	Type	Description	Reset Value
31:10	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9:0	Extended wait time out (EXTENDEDWAIT)	R/W	0x0 = 16 clock cycles (POR reset value). 0x1 = (n+1) x16 clock cycles. The delay is in HCLK cycles.	

For example, for a static memory read/write transfer time of 16 µs, and a HCLK frequency of 50 MHz, the following value must be programmed into this register: $((16 \times 10^{-6}) \times (50 \times 10^{6}) / 16)) - 1 = 49$

7.7.19 Dynamic Memory Configuration Registers (EMCDynamicConfig0 - 0x3108 0100 and EMCDynamicConfig1 - 0x3108 0120)

The EMCDynamicConfig0 and EMCDynamicConfig1 registers enable programming of configuration information for the relevant dynamic memory chip select. This register is normally only modified during system initialization. This register is accessed with one wait state.

Table 114 shows the bit assignments for the EMCDynamicConfig0-1 register.

If the write protect bit (P) is set, a data abort is generated on a write to a relevant memory location.

Table 114. Dynamic Memory Configuration Registers (EMCDynamicConfig0 - 0x3108 0100 and EMCDynamicConfig1 - 0x3108 0120)[1][2]

Bits	Name	Type	Description
31:21	Reserved	-	Reserved, read undefined, do not modify.
20	Write protect (P)	R/W	0 = writes not protected. 1 = write protected.
19:15	Reserved	-	Reserved, read undefined, do not modify.
14:7	Address mapping (AM)	R/W	See Table 115.
6:3	Reserved	-	Reserved, read undefined, do not modify.
2:0	Memory device (MD)	R/W	000 = SDR SDRAM. 001 = reserved. 010 = low power SDR SDRAM. 011 = reserved. 100 = DDR SDRAM. 101 = reserved. 110 = low power DDR SDRAM. 111 = reserved.

[1] The buffers must be disabled during SDRAM initialization, and be enabled during normal operation.

[2] The SDRAM column and row width and number of banks are computed automatically from the address mapping.

Table 115 shows the various address mapping possibilities. By setting the appropriate address mapping bits, the user can select between high-performance mode (fast access and higher power consumption) or low-power mode (slower access and low power-consumption). Address mappings that are not shown in Table 115 are reserved.

Table 115. Address mapping

[14]	[13:12]	[11:9]	[8:7]	Description
16-bit external bus high-performance address mapping (Row, Bank, Column)				
0	00	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	00	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	00	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	00	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	00	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	00	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	00	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	00	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	00	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	00	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
16-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
0	01	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	01	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	01	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	01	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	01	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	01	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	01	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	01	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	01	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	01	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
32-bit external bus high-performance address mapping (Row, Bank, Column)				
1	00	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
1	00	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
1	00	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
1	00	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
1	00	001	10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
1	00	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
1	00	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
1	00	010	10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
1	00	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
1	00	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
1	00	011	10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
1	00	010	01	256 Mb (8Mx32), 4 banks, row length = 12, column length = 9
1	00	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
1	00	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10

Table 115. Address mapping *...continued*

[14]	[13:12]	[11:9]	[8:7]	Description
1	00	011	01	512Mb (16Mx32), 4 banks, row length = 13, column length = 9
1	00	100	01	1Gb (32Mx32), 4 banks, row length = 13, column length = 10
32-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
1	01	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
1	01	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
1	01	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
1	01	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
1	01	001	10	64Mb (2Mx32), 4 banks, row length = 11, column length = 8
1	01	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
1	01	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
1	01	010	10	128Mb (4Mx32), 4 banks, row length = 12, column length = 8
1	01	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
1	01	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
1	01	011	10	256Mb (8Mx32), 4 banks, row length = 13, column length = 8
1	01	010	01	256 Mb (8Mx32), 4 banks, row length = 12, column length = 9
1	01	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
1	01	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
1	01	011	01	512Mb (16Mx32), 4 banks, row length = 13, column length = 9
1	01	100	01	1Gb (32Mx32), 4 banks, row length = 13, column length = 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.
- 4 x 8-bit wide memory devices, choose a 32-bit wide address mapping.
- 2 x 8-bit wide memory devices, choose a 16-bit wide address mapping.

7.7.20 Dynamic Memory RAS and CAS Delay Register (EMCDynamicRasCas0 - 0x3108 0104 and EMCDynamicRasCas1 - 0x3108 0104)

The EMCDynamicRasCas0 and EMCDynamicRasCas0 registers enable programming of RAS and CAS latencies for the relevant dynamic memory. These registers must only be modified during system initialization. These registers are accessed with one wait state.

Note: The values programmed into this register must be consistent with the values used to initialize the SDRAM memory device.

Table 116 shows the bit assignments for the EMCDynamicRasCas0 register.

Table 116. Dynamic Memory RAS and CAS Delay Register (EMCDynamicRasCas0 - 0x3108 0104)

Bits	Name	Type	Description
31:11	Reserved	-	Reserved, read undefined, do not modify.
10:7	CAS latency (CAS)	R/W	0000 = reserved. 0001 = one half clock cycle. 0010 = one clock cycle. 0011 = one and a half clock cycles. 0100 = two clock cycles. 0101 = two and a half clock cycles. 0110 = three clock cycles. 0111 = three and a half clock cycles. 1000 = four clock cycles. 1001 = four and a half clock cycles. 1010 = five clock cycles. 1011 = five and a half clock cycles. 1100 = six clock cycles. 1101 = six and a half clock cycles. 1110 = seven clock cycles. 1111 = seven and a half clock cycles.
6:4	Reserved	-	Reserved, read undefined, do not modify.
3:0	RAS latency (active to read/write delay) (RAS)	R/W	0000 = reserved. 0001 to 1110 = n clock cycles. 1111 = 15 clock cycles.

7.7.21 Static Memory Configuration registers (EMCStaticConfig0-3 - 0x3108 0200, 0220, 0240, 0260)

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

If the write protect bit (P) is set, a data abort is generated on a write to a relevant memory location.

Table 117. Static Memory Configuration registers (EMCStaticConfig0-3 - address 0x3108 0200, 0x3108 0220, 0x3108 0240, 0x3108 0260) bit description

Bit	Name	Type	Description
31:21	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
20	Write protect (P)	R/W	00 = Writes not protected (POR reset value). 01 = Write protected.
19:9	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

Table 117. Static Memory Configuration registers (EMCStaticConfig0-3 - address 0x3108 0200, 0x3108 0220, 0x3108 0240, 0x3108 0260) bit description

Bit	Name	Type	Description
8	Extended wait (EW)	R/W	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 = Extended wait disabled (POR reset value). 1 = Extended wait enabled.
7	Byte lane state (PB)	R/W	This bit affects the behavior of the EMC_BLSn[3:0] and EMC_WR_N signals on the External Memory Interface. When the byte lane state bit, PB, is set to 0, the EMC_BLSn[3:0] signals behave as byte lane write strobes and will only be active (LOW) during static memory writes. The EMC_WR_N signal never goes active when PB is 0. (POR reset value) When the byte lane state bit, PB, is set to 1, the EMC_BLSn[3:0] signals behave as byte lane enable strobes and will be active (LOW) for both static memory read and write access, and signal EMC_WR_N will be LOW for writes. This is used when interfacing to a static memory with multiple byte lane strobe pins and a separate write strobe pin. This mode is typically used with 16-bit and 32-bit static memory chips. Writes: PB = 0: The active bits in EMC_BLSn[3:0] are LOW; EMC_WR_N is **not** active. PB = 1: The active bits in EMC_BLSn[3:0] are LOW; EMC_WR_N is active. Reads: PB = 0: All the bits in EMC_BLSn[3:0] are HIGH. PB = 1: The active bits in EMC_BLSn[3:0] are LOW.
6	Chip select polarity (PC)	R/W	The value of the chip select polarity on power-on reset is 0. 0 = Active LOW chip select. 1 = Active HIGH chip select.
5:4	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

Table 117. Static Memory Configuration registers (EMCStaticConfig0-3 - address 0x3108 0200, 0x3108 0220, 0x3108 0240, 0x3108 0260) bit description

Bit	Name	Type	Description
3	Page mode (PM)	R/W	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 = Disabled (POR reset value). 1 = Async page mode enabled (page length four)
2	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
1:0	Memory width (MW)	R/W	Memory Width When using a system that implements 8-bit static memory, address signals A[23:0] are mapped in a 1:1 correspondence with pins A[23:0], since transactions occur on byte boundaries. This is done automatically by hardware in the EMC controller when the value of MW is set to 0x0. When using a system that implements 16-bit static memory, address signals A[23:1] are right-justified by one bit and output on the physical address pins A[22:0]. Address signal A0 is not necessary because all transactions occur on half-word boundaries. Address pin A23 is not used because the address space available for each CSn is 16 MB. This is done automatically by hardware in the EMC controller when the value of MW is set to 0x1. When using a system that implements 32-bit static memory, address signals A[23:2] are right-justified by two bits and output on the physical address pins A[21:0]. Address signals A0 and A1 are not necessary because all transactions occur on word boundaries. Address pins A23 and A22 are not used because the address space available for each CSn is 16 MB. This is done automatically by hardware in the EMC controller when the value of MW is set to 0x2. 00 = 8 bit. 01 = 16 bit. 10 = 32 bit (POR reset value). 11 = Reserved.

[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.

7.7.22 Static Memory Write Enable Delay registers (EMCStaticWaitWen0-3 - 0x3108 0204, 0224, 0244, 0264)

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 118 shows the bit assignments for the EMCStaticWaitWen0-3 registers.

Table 118. Static Memory Write Enable Delay registers (EMCStaticWaitWen0-3 - address 0x3108 0204,0x3108 0224, 0x3108 0244, 0x3108 0264) bit description

Bit	Symbol	Type	Description
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
3:0	Wait write enable (WAITWEN)		Delay from chip select assertion to write enable. 0000 = One HCLK cycle delay between assertion of chip select and write enable (POR reset value). 0001-1111 = (n + 1) HCLK cycle delay. The delay is (WAITWEN +1) x HCLK.

7.7.23 Static Memory Output Enable Delay registers (EMCStaticWaitOen0-3 - 0x3108 0208, 0228, 0248, 0268)

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 119 shows the bit assignments for the EMCStaticWaitOen0-3 registers.

Table 119. Static Memory Output Enable delay registers (EMCStaticWaitOen03 - address 0x3108 0208, 0x3108 0228, 0x3108 0248, 0x3108 0268) bit description

Bit	Symbol	Type	Description
31:4	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
3:0	Wait output enable (WAITOEN)	R/W	Delay from chip select assertion to output enable. 0000 = No delay (POR reset value). 0001-1111 = n cycle delay. The delay is WAITOEN x t_{HCLK}.

7.7.24 Static Memory Read Delay registers (EMCStaticWaitRd0-3 - 0x3108 020C, 022C, 024C, 026C)

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 120 shows the bit assignments for the EMCStaticWaitRd0-3 registers.

Table 120. Static Memory Read Delay registers (EMCStaticWaitRd0-3 - address 0x3108 020C, 0x3108 022C, 0x3108 024C, 0x3108 026C) bit description

Bit	Symbol	Type	Description
31:5	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
4:0	Non-page mode read wait states or asynchronous page mode readfirst access wait state (WAITRD)	R/W	Non-page mode read or asynchronous page mode read, first read only: 00000 - 11110 = (n + 1) HCLK cycles for read accesses. 11111 = 32 HCLK cycles for read accesses For non-sequential reads, the wait state time is (WAITRD + 1) x tHCLK.

7.7.25 Static Memory Page Mode Read Delay registers (EMCStaticwaitPage0-3 - 0x3108 0210, 0230, 0250, 0270)

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 121 shows the bit assignments for the EMCStaticWaitPage0-3 registers.

Table 121. Static Memory Page Mode Read Delay registers0-3 (EMCStaticWaitPage0-3 - address 0x3108 0210, 0x3108 0230, 0x3108 0250, 0x3108 0270) bit description

Bit	Symbol	Type	Description
31:5	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
4:0	Asynchronous page mode read after the first read wait states (WAITPAGE)	R/W	Number of wait states for asynchronous page mode read accesses after the first read: 00000 -11110 = (n+ 1) HCLK cycle read access time. note: For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE + 1) x tHCLK. 11111 = 32 HCLK cycle read access time

7.7.26 Static Memory Write Delay registers (EMCStaticWaitwr0-3 - 0x3108 0214, 0234, 0254, 0274)

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 122 shows the bit assignments for the EMCStaticWaitWr0-3 registers.

Table 122. Static Memory Write Delay registers0-3 (EMCStaticWaitWr - address 0x3108 0214, 0x3108 0234, 0x3108 0254, 0x3108 0274) bit description

Bit	Symbol	Type	Description
31:5	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
4:0	Write wait states (WAITWR)	R/W	SRAM wait state time for write accesses after the first read: 00000-11110 = (n + 2) HCLK cycle write access time. note: The wait state time for write accesses after the first read is WAITWR (n + 2) x tHCLK. 11111 = 33 HCLK cycle write access time (POR reset value).

7.7.27 Static Memory Turn Round Delay registers (EMCStaticWaitTurn0-3 - 0x3108 0218, 0238, 0258, 0278)

The EMCStaticWaitTurn0-3 registers enable you to program the minimum 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 123 shows the bit assignments for the EMCStaticWaitTurn0-3 registers.

Table 123. Static Memory Turn Round Delay registers0-3 (EMCStaticWaitTurn0-3 - address 0x3108 0218, 0x3108 0238, 0x3108 0258, 0x3108 0278) bit description

Bit	Symbol	Type	Description
31:4	Reserved	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
3:0	Bus turnaround cycles (WAITTURN)	R/W	0000 -1110 = (n + 1) HCLK turnaround cycles. note: Bus turnaround time is (WAITTURN + 1) x tHCLK. 1111 = 16 HCLK turnaround cycles.

To prevent bus contention on the external memory data bus, the WAITTURN field ensures a minimum number of bus turnaround cycles between EMC static memory read access and subsequent external bus accesses. The WAITTURN field also ensures a minimum number of turnaround cycles between static memory read and subsequent dynamic memory access.

The WAITTURN field affects:

- read-to-read on the same or different EMC static chip select
- read-to-write on the same or different EMC static chip select
- read static chip select to dynamic chip select read or write

Remark: The WAITTURN only ensures a minimum turn-around time. Some lower values will have no effect when the inherent turn-around time for successive access to the static memory chip select are longer than the WAITTURN delay. The WAITTURN will not ensure a wait between successive external memory access when using the ARM LDM instruction. To ensure there is no burst access for successive external memory reads use the ARM LDR instruction.

7.7.28 EMC AHB Control Registers (EMCAHBControl0 - 0x3108 0400, EMCAHBControl3 - 0x3108 0460, EMCAHBControl4 - 0x3108 0480)

The EMCAHBControl0, EMCAHBControl3, EMCAHBControl4 registers are used to control operation of the AHB interfaces to the EMC. These registers can be altered during normal operation.

Table 124 shows the bit assignments for the EMCAHBControl0, EMCAHBControl3, EMCAHBControl4 registers.

Table 124. EMC AHB Control Registers (EMCAHBControl0 - 0x3108 0400, EMCAHBControl3 - 0x3108 0460, EMCAHBControl4 - 0x3108 0480)

Bits	Name	Type	Description
31:1	Reserved	-	Reserved, read undefined, do not modify.
0	AHB Port Buffer Enable (E)	R/W	0 = disable buffer. 1 = enable buffer.

The HPROT[2] signal is used (bufferable signals) to determine if merging takes place, and the E bit is used to determine if the AHB write buffer is used to hold the transfer. Table 125 lists the possible transfer types.

Table 125. Transfer Types

E bit	HPROT[2]	Transfer[1]
0	0	Nonbuffered, no data merging
0	1	Nonbuffered, data merging allowed
1	0	Buffered, no data merging
1	1	Buffered, data merging allowed

[1] The terms used in the transfer description column are defined here:

Buffered — means HREADY is returned immediately for first or last transfer.

Nonbuffered — means HREADY is held off until transfer is placed in memory.

Merging allowed — means non word burst transfers are converted into word transfers (INCR4 BYTE is passed as 1 word write)

No data merging — means each transfer of the burst is passed through as is (INCR4 BYTE is passed as 4 byte writes).

7.7.29 EMC AHB Status Registers (EMCAHBStatus0 - 0x3108 0404, EMCAHBStatus3 - 0x3108 0464, EMCAHBStatus4 - 0x3108 0484)

The EMCAHBStatus0, EMCAHBStatus3, EMCAHBStatus4 registers status information on the AHB interface. Table 126 shows the bit assignments for the EMCAHBStatus0, EMCAHBStatus3, EMCAHBStatus4 registers.

Table 126. EMC AHB Status Registers (EMCAHBStatus0 - 0x3108 0404, EMCAHBStatus3 - 0x3108 0464, EMCAHBStatus4 - 0x3108 0484)

Bits	Name	Type	Description
31:2	Reserved	-	Reserved, read undefined, do not modify.
1	AHB Port Buffer Status (S)	RO	0 = buffer empty. 1 = buffer contains data.
0	Reserved	-	Reserved, read undefined, do not modify.

7.7.30 EMC AHB Timeout Registers (EMCAHBTimeOut0 - 0x3108 0408, EMCAHBTimeOut3 - 0x3108 0468, EMCAHBTimeOut4 - 0x3108 0488)

The EMCAHBTimeOut0, EMCAHBTimeOut3, EMCAHBTimeOut4 registers are used to ensure that each AHB port is serviced within a specified number of cycles. When a request goes active, the values in the EMCAHBTimeOut0, EMCAHBTimeOut3, EMCAHBTimeOut4 registers are loaded into a down counter. If the transfer is not processed before the counter reaches zero, the priority of the AHB port is increased until the request is serviced.

These registers therefore enable the transaction latency, and indirectly the bandwidth, for a particular port to be defined. The value programmed into these registers depends on the latency required for the particular port. These registers can be altered during normal operation.

Table 127 shows the bit assignments for the EMCAHBTimeOut0, EMCAHBTimeOut3, EMCAHBTimeOut4 registers.

Table 127. EMC AHB Timeout Registers (EMCAHBTimeOut0 - 0x3108 0408, EMCAHBTimeOut3 - 0x3108 0468, EMCAHBTimeOut4 - 0x3108 0488)

Bits	Name	Type	Description
31:10	Reserved	-	Reserved, read undefined, do not modify.
9:0	AHB Timeout (AHBTIMEOUT)	R/W	0x0 = timeout disabled. 0x001 - 0x1FF = number of AHB cycles before timeout is reached.

7.7.31 DDR Calibration Nominal Value (DDR_LAP_NOM - 0x4000 406C)

This register is part of the mechanism for calibrating the DQS input timing if DDR SDRAMs are used. Refer to the section on DDR DQS_DELAY calibration for details.

Table 128. DDR Calibration Nominal Value (DDR_LAP_NOM - 0x4000 406C)

Bits	Function	Nominal value[1]	Reset value
31:0	A nominal count value corresponding to typical process, voltage, and temperature conditions must be written here by software.	0x20	0x00

[1] Nominal Value is for PERIPH_CLK = 13 MHz.

7.7.32 DDR Calibration Measured Value (DDR_LAP_COUNT - 0x4000 4070)

This register is part of the mechanism for calibrating the DQS input timing if DDR SDRAMs are used. DDR_LAP_COUNT is a Read-Only register. Refer to the section on DDR DQS delay calibration for details.

Table 129. DDR Calibration Measured Value (DDR_LAP_COUNT - 0x4000 4070)

Bits	Function	Reset value
31:0	Value of DDR SDRAM ring oscillator counter.	0

7.7.33 DDR Calibration Delay Value (DDR_CAL_DELAY - 0x4000 4074)

Table 130. DDR Calibration Delay Value (DDR_CAL_DELAY - 0x4000 4074)

Bits	Function	Reset value
31:5	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
4:0	The current calibrated delay setting can be read out here. This value can change for every calibration performed.	0

7.8 DDR DQS delay calibration

The DQS calibration circuitry is only needed when DDR SDRAM is used. DDR SDRAM devices output two DQS signals aligned with read data, each DQS is applied to 8 data bits. The EMC uses a delayed version of the DQS signals for sampling the read data. The calibration circuitry makes it possible for software to program a nominal fixed delay for DQS, which will be compensated for varying temperature, voltage and process. Note that the arithmetic done in hardware to accomplish this uses signed numbers.

The delay calibration circuit does an automated calibration on the positive edge of RTC_TICK or manually calibrates on request by the SW_DDR_CAL signal (SDRAMCLK_CTRL[8]). The ring oscillator and cycle counter runs for one period of PERIPH_CLK when calibrating. When RTC_TICK calibration is enabled, there will be a new calibration every second as long as the CPU is not in stop mode. When the CPU is in stop mode the calibration circuitry is automatically disabled in order to keep power consumption low.

When the ring oscillator has run for one PERIPH_CLK period, the counter will have a value reflecting the speed of the ring oscillator. The speed of the ring oscillator represents the speed in the entire device under existing environmental conditions. The counter value is readable by software in the DDR_LAP_COUNT register. Software programs the DDR_LAP_NOM register with a value corresponding to the nominal count value for typical process, voltage and temperature conditions see Table 128 for the nominal value. The nominal value represents the point where an increase of one value in the DDR_DQSIN_DELAY register changes the delay by exactly 0.25 nanosec.

The difference between these two registers represents the deviation from nominal circuit speed. The sensitivity of the circuit is controlled by shifting this value down by the number of bits given by SDRAMCLK_CTRL[12:10]. A large shift causes little compensation to the DDR_DQSIN_DELAY value. Equation 6 shows the resulting DQS_DELAY calculation.

$$DQS\_DELAY = DDR\_CAL\_DELAY \times CAL\_FACTOR \quad (6)$$

$$DDR\_CAL\_DELAY = DQSIN\_DELAY + \left(\frac{LAP\_COUNT - LAP\_NOM}{2^{SENSITIVITY\_FACTOR}}\right) \quad (7)$$

$$CAL\_FACTOR = \left(\frac{LAP\_NOM \times 0.25}{LAP\_COUNT}\right) \quad (8)$$

Definitions for variables used in Equation 6, Equation 7 and Equation 8 and are described in Table 131.

Table 131. DQS Delay Calibration Definitions

Variable	Definition
DQS_DELAY	The resulting desired DQS Delay in ns. This value is calculated automatically in hardware for a given DDR_LAP_NOM and desired DQS_DELAY selected from the DQS Delay Sensitivity Factor Table[132].
DDR_CAL_DELAY	The value read from the DDR_CAL_DELAY register. For reference, multiplying the value in this register by the CAL_FACTOR provides the current DQS_DELAY in ns.
DQSIN_DELAY	The value written into the DDR_DQS_DELAY bitfield, (SDRAMCLK_CTRL[6:2]). This value is obtained from the DQS Delay Sensitivity Factor Table[132] for a given desired delay value.
LAP_COUNT	The value read from the DDR_LAP_COUNT register.
LAP_NOM	The constant value written into the DDR_LAP_NOM register.
SENSITIVITY_FACTOR	The value written into the SENSITIVITY DOWNSHIFT value bitfield, (SDRAMCLK_CTRL[12:10]). This value is obtained from the DQS Delay Sensitivity Factor Table[132] for a given desired delay value.

For the desired DQS delay use Table 132 to select the correct DDR_DQSIN_DELAY(SDRAMCLK_CTRL[6:2]) and SENSITIVITY_FACTOR(SDRAMCLK_CTRL[12:10]). The DDR_DQSIN_DELAY and DDR_LAP_NOM values determine the sensitivity downshift value needed to maintain a constant delay for variations in process, voltage and temperature

The adjusted deviation value (which is a signed number), is added to the software programmed nominal delay in DDR_DQSIN_DELAY. The output of the adder is the value used to program the delay network. On an overflow/underflow in the adder, the maximum or minimum values are used. The adder status can be read in SDRAMCLK_CTRL[13]. In order to avoid adjusting the delay network during an ongoing DDR SDRAM access, the internal bus request signal is used to update the value.

Table 132. DQS Delay Sensitivity Factor values for DDR_DQSIN_DELAY value

DQS_DELAY Desired Delay in ns	**DDR_DQSIN_DELAY Value** SDRAMCLK_CTRL[6:2]	**SENSITIVITY_FACTOR[1]** SDRAMCLK_CTRL[12:10]
0.00	0x00 (0b00000)	7
0.25	0x01 (0b00001)	5
0.50	0x02 (0b00010)	4
0.75	0x03 (0b00011)	4 or 3
1.00	0x04 (0b00100)	3
1.25	0x05 (0b00101)	3
1.50	0x06 (0b00110)	3 or 2
1.75	0x07 (0b00111)	2
2.00	0x08 (0b01000)	2
2.25	0x09 (0b01001)	2
2.50	0x0A (0b01010)	2
2.75	0x0B (0b01011)	2
3.00	0x0C (0b01100)	2 or 1
3.25	0x0D (0b01101)	1
3.50	0x0E (0b01110)	1

Table 132. DQS Delay Sensitivity Factor values for DDR_DQSIN_DELAY value

DQS_DELAY Desired Delay in ns	**DDR_DQSIN_DELAY Value** SDRAMCLK_CTRL[6:2]	**SENSITIVITY_FACTOR**[1] SDRAMCLK_CTRL[12:10]
3.75	0x0F (0b01111)	1
4.00	0x10 (0b10000)	1
4.25	0x11 (0b10001)	1
4.50	0x12 (0b10010)	1
4.75	0x13 (0b10011)	1
5.00	0x14 (0b00100)	1
5.25	0x15 (0b00101)	1
5.50	0x16 (0b00110)	1
5.75	0x17 (0b10111)	1
6.00	0x18 (0b11000)	1 or 0
6.25	0x19 (0b11001)	0
6.50	0x1A (0b11010)	0
6.75	0x1B (0b11011)	0
7.00	0x1C (0b11100)	0
7.25	0x1D (0b11101)	0
7.50	0x1E (0b11110)	0
7.75	0x1F (0b11111)	0

[1] SENSITIVITY_FACTOR is provided for a DDR_LAP_NOM = 0x20 and PERIPH_CLK = 13 MHz.

Non-calibrated delays can be used by programming the CAL_DELAY signal to 0 in SDRAMCLK_CTRL[9]. In this configuration, DQS_DELAY from Equation 6 reduces to Equation 9

$$\text{DQS\_DELAY} = \text{DDR\_DQSIN\_DELAY} \times \frac{\text{LAP\_NOM} \times 0.25}{\text{LAP\_COUNT}} \quad (9)$$

The delay circuitry is only clocked when DDR SDRAM is selected in SDRAMCLK_CTRL[1].

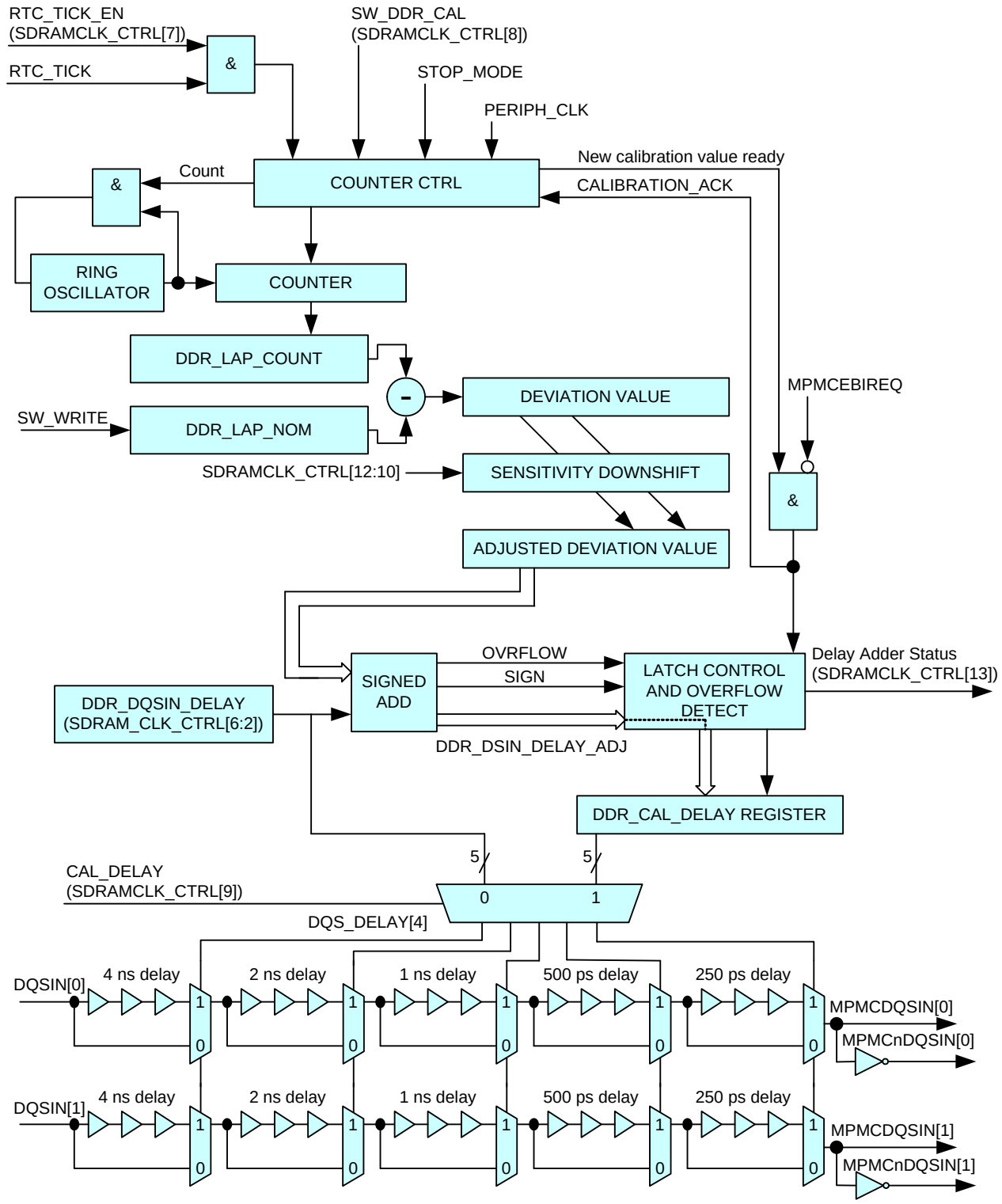


Fig 19. DDR DQS delay calibration

8.1 Introduction

Note: The LPC32x0 has two NAND Flash controllers, one for multi level NAND Flash devices and one for single level NAND Flash devices. The two NAND Flash controllers use the same pins to interface to external NAND Flash devices, so only one interface may be active at a time. The NAND Flash controllers can be disabled by bits in the FLASHCLK_CTRL register in order to save power when they are not used.

The Multi Level Cell MLC NAND Flash controller interfaces to multi-level NAND Flash devices. An external NAND Flash device (of either multi-level or single-level type) may be used to allow the bootloader to automatically load application code into internal RAM for execution, see the UM10326 user manual for details.

8.2 Features

- Supports small (528 bytes) and large (2114 bytes) page.
- Supports single and multi-level NAND flash memory.
- Programmable NAND timing parameters.
- Reed-Solomon (R/S) encoder/decoder (10 bit symbols).
- Automatic error detection, with hardware correction for up to 4 symbols (4-40 bit).
- Auto encode/decode cycles using built-in serial data buffer.
- 528-bytes serial data buffer.
- Supports DMA.

8.3 Pin descriptions

Table 133. NAND-Flash memory controller pins

Pin name	Type	NAND Flash Signal	Function
FLASH_CE_N	output	CEn	Chip select, active LOW.
FLASH_WR_N	output	WEn	Write enable, active LOW.
FLASH_RD_N	output	REn	Read Enable, active LOW.
FLASH_ALE	output	ALE	Address Latch Enable.
FLASH_CLE	output	CLE	Command Latch Enable.
FLASH_RDY	input	RDY	Active HIGH Ready signal.
FLASH_IO[7:0]	input/output	D_IO	I/O pins, commands, address and data.

8.3.1 Interrupt signals from NAND flash controllers

The interrupt from the MLC NAND Flash controller is masked with NAND_INT_E and ORed with the interrupt signal from the SLC NAND Flash controller before it goes to the interrupt controller. The connections of the interrupts of the MLC and SLC NAND Flash controllers are shown in Figure 20.

8.3.2 DMA request signals from flash controllers

The dma_breq(0), dma_sreq(0), and dma_sreq(1) are ORed together and connected to the DMA controller as the burst request signal from the SLC Flash controller (DMA controller peripheral number 1). In order to be able to use a peripheral to peripheral DMA transfer to the SLC NAND Flash controller, this burst request signal is also connected to DMA controller peripheral number 12 when the SLC Flash controller is selected.

When the MLC NAND Flash controller is selected, the burst request signal from the MLC Flash controller is connected to DMA controller peripheral number 12.

The connections of the DMA signals of the MLC and SLC NAND Flash controllers are shown in Figure 20.

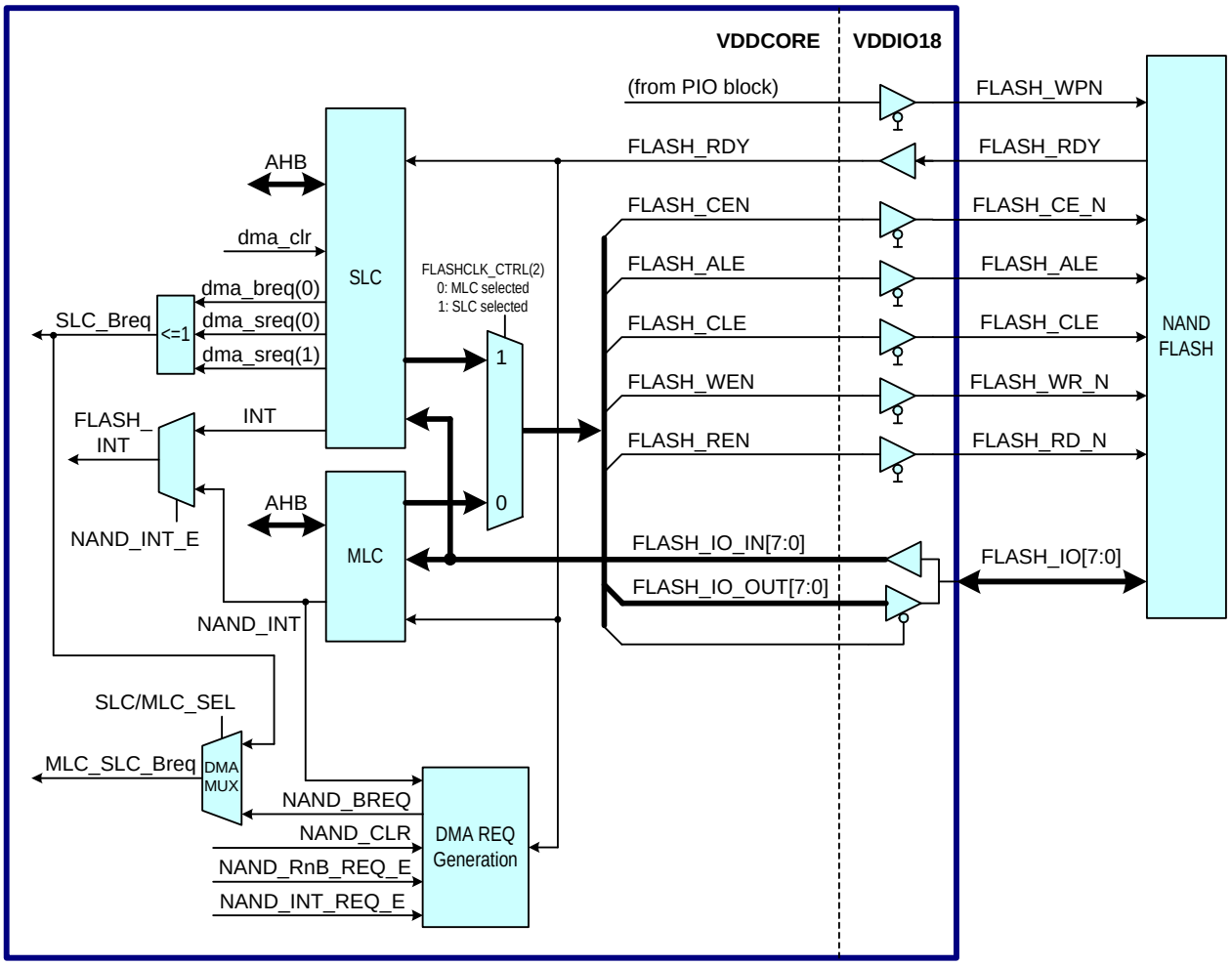


Fig 20. NAND flash controllers

8.4 MLC NAND flash controller functional description

Serial data transfers to/from the NAND flash can be performed directly by the CPU. The CPU can transfer data directly from the NAND flash while the controller simultaneously performs the R/S decoding. The CPU only needs to transfer data from the controller's Data Buffer when an error occurs. Since the expected error rate is small, this will result in minimal impact. Alternately, the CPU can force the controller to transfer the data to the serial Data Buffer where the CPU can then read it. The CPU can transfer data directly to the NAND flash while the controller simultaneously performs the R/S encoding to calculate the ECC codes. Alternately the CPU can write the data to controller's serial Data Buffer and then force the controller to independently transfer the data to the NAND flash. The transfer speed will be limited by either the NAND flash throughput or the system's AHB bus clock (HCLK).

The basic block diagram of the MLC NAND Flash controller is shown in Figure 21.

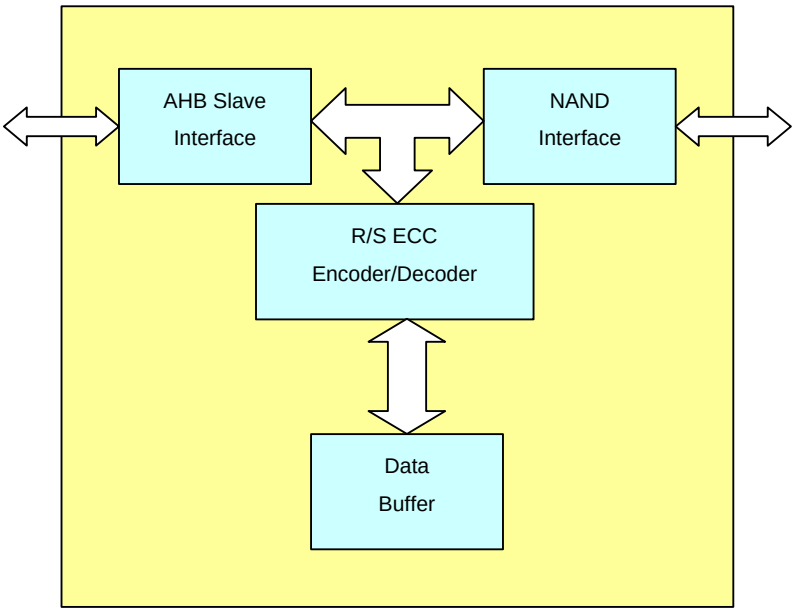


Fig 21. MLC NAND flash controller

8.4.1 Reed-Solomon encoder/decoder

The Reed-Solomon (R/S) encoder and decoder allow the controller to perform error detection and correction using redundant data stored in the overhead area of each page.

The R/S encoder and decoder use the combined User and Overhead area of each NAND flash page (528) bytes as the data stream. This data is converted to 10-bit symbols as required by the R/S algorithm. The R/S encoder generates 8 symbols (10 bytes) of redundant data (ECC codes). This data is stored along with 518 bytes of user data in each NAND flash page. The R/S decoder uses the redundant data (10 bytes) to perform error detection. If an error is detected, the controller attempts to perform correction. After correction is performed, the CPU can read the corrected data from the controller's serial Data Buffer.

Since the R/S algorithm requires a fixed length data stream, error detection/correction can only be performed in discrete blocks of data. This restricts data storage to a minimum size. This minimum size is the length of a standard 528-byte NAND flash page (518 bytes of user data).

The consequence of this restriction is that certain NAND functionality that involves partial page access cannot be supported by the controller while providing error detection and correction. For example, certain applications use the overhead area of each page to store management data. The NAND flash can then be scanned by merely reading the overhead area of each page. Since the R/S algorithm requires the entire data stream, this type of operation cannot be supported (this can, however, still be performed but without error detection and correction).

8.4.1.1 Large block NAND flash support

Large block NAND flash devices use 2112-byte pages. This is four times the standard page size. The R/S Encoder/Decoder directly supports 518-byte pages only. In order to function with 2112-byte pages, these large pages are divided into four sections of equal length. Each section then contains the equivalent of a 528-byte page. Care must be taken not to lose the manufactures format for bad block information in devices that are formatted with a 2112 block size (2048-byte user). Software must read this information and preserve it prior to using the 4 x 528-byte block large page format provided by this controller.

8.4.1.2 Erased page detection support

The R/S Decoder includes a feature to detect when all data in a page (528-bytes) is 0xFF. This indicates that the page has been erased. When this condition is detected, the decoder does not perform ECC processing and indicates to the CPU that no errors were detected. This feature prevents the R/S decoder from attempting to perform error correction processing on erased pages.

8.4.2 Serial data buffer

The serial Data Buffer is a 528-byte buffer primarily used by the R/S algorithm to perform error correction. Data can be transferred by the CPU directly to/from the serial Data Buffer. Access to the buffer, however, is restricted to sequential access. This is a consequence of the R/S algorithm requiring 10-bit code-words. Any data access to the buffer must undergo appropriate translation that allows sequential access only.

Any NAND serial data access is also performed on the serial Data Buffer by the controller. NAND serial data read accesses cause the data read from the NAND flash to be written to the Data Buffer by the controller. NAND serial data write accesses cause the data to be written to the Data Buffer as well as the NAND flash by the controller.

Data can be read directly from the serial Data Buffer. This is normally performed by the CPU when an error has been detected by the R/S decoder. The corrected data must be read by the CPU from the serial Data Buffer.

Data can be read from the NAND flash device to the Data Buffer by the controller without any CPU intervention.

Data can be written to the Data Buffer by the CPU without writing the data to the NAND flash. The controller can then independently write the data to the NAND flash.

8.4.3 Operation

Due to the addition of ECC error correction, certain changes to the NAND flash protocol are required in some instances. Also, certain commands and/or command sequences cannot be supported.

Because ECC error correction is performed over the entire usable page data (518 bytes), this data becomes the minimum size data block that can be transferred to/from the individual NAND flash pages. This restricts commands that can normally specify a page address to use 00h as the first byte of the address (A0-A7). Also commands such as Read Mode (2) 0x01 that are specifically used for partial page access cannot be used. Furthermore, any command sequences (such as 0x50+0x80) that perform partial page access cannot be used. Using these unsupported features will result in unexpected operation and/or loss of data.

The exception to some of these restrictions is the Read Mode (3) (0x50) command. This command can be used by the CPU when only the overhead data is required. In reality, the controller will translate the command to a Read Mode (1) (00h) command before sending to the NAND flash and therefore the time required to perform a page read and transfer of the entire 528 bytes to the controller will be incurred. The controller will read 528 bytes into the User and Overhead Buffer regions and automatically set the Buffer pointer to the Overhead region. The CPU can then read the overhead data directly from the controller's serial Data Buffer.

Note that due to the fact that the entire 528 bytes must be read by the controller, using this command might cause timing problems. The only advantage to using the Read Mode (3) command over the Read Mode (1) command is that the CPU need not read all 518 bytes. This transfer is instead performed by the controller at possibly a higher transfer rate. However, for CPUs with very high speed access to the controller, it may be advantageous to read all 518 bytes using the Read Mode (1) command if there is a possibility that the extra data may become useful at a later date.

From an operation point of view, the NAND programmer's model uses address spaces to communicate with the NAND flash. Commands and addresses are written to specific addresses (registers) within the controller's address space. Data is read/written from/to the NAND flash using unique address ranges that lie within the controller's address space.

8.4.3.1 Page format

Standard NAND devices include two sections per page. The first section (User Data Area) is typically used to store general data. The second section (Overhead Data Area) is typically used to store overhead information such as status and ECC parity data. The controller requires 10 bytes of ECC parity data for the R/S ECC processing. The placement of this data differs between small and large block devices as describes in the following sections.

8.4.3.1.1 Small block NAND flash devices

For small block devices (528-byte page size) the user and overhead areas of each page (512 + 16 bytes) are combined to form a 528-byte area which is then divided into three sections:

1. User data.
2. Overhead data.
3. ECC Parity data.

Figure 22 illustrates how each page is partitioned to accommodate the 10 bytes of ECC parity data. Note that the Overhead area available for CPU usage is deduced to 6 bytes such that the total CPU usable data is 518 bytes (512+6).

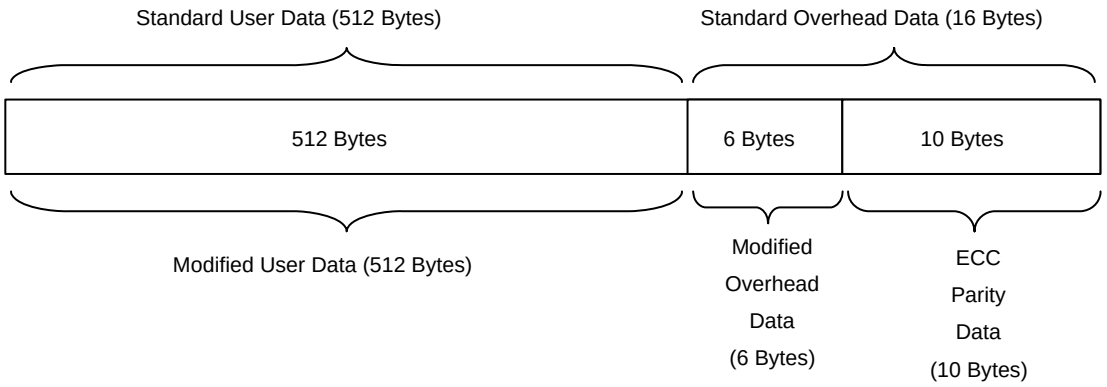


Fig 22. Small page partitioning to accommodate the 10 bytes of ECC parity data

8.4.3.1.2 Large block NAND flash devices

For large block devices (2112-byte page size) the user and overhead areas of each page (2048 + 64 bytes) are combined to form a 2112-byte area which is then divided into four sections. Each of these sections is then further subdivided into three sections as follows:

1. User data.
2. Overhead data.
3. ECC Parity data.

Figure 23 illustrates how each subsection is partitioned to accommodate the 10 bytes of ECC parity data. Note that the Overhead area available for CPU usage is 6 bytes such that the CPU usable data is 518 bytes (512+6) per section and 2072 bytes (4*518) per page.

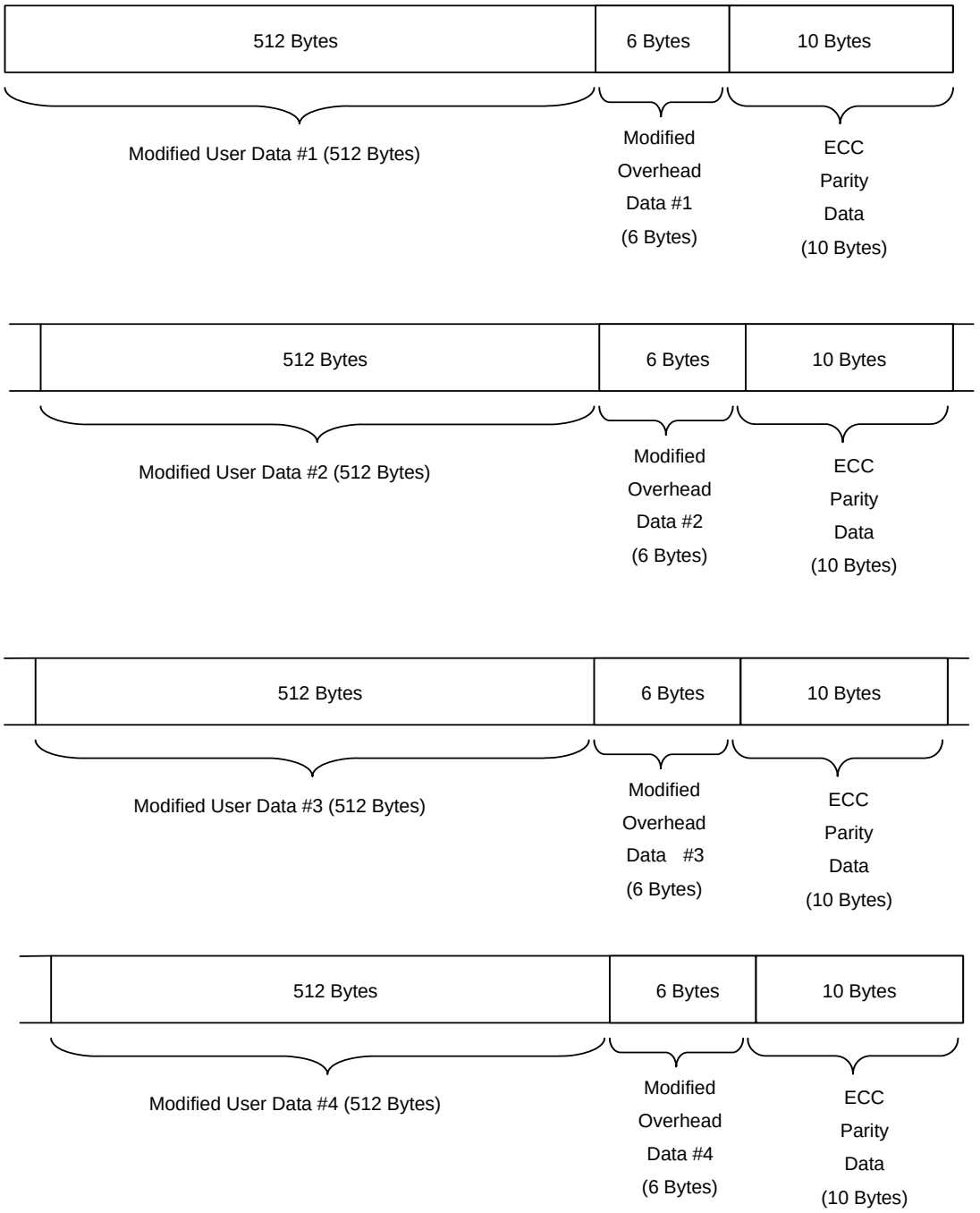


Fig 23. Large page partitioning to accommodate the 10 bytes of ECC parity data

8.4.3.2 Supported commands

Due to the addition of ECC error correction, certain changes to the NAND protocol are required in some instances. Also, certain commands and/or command sequences cannot be supported. Table 134 lists all supported, unsupported, and supported-with-restrictions commands. Subsequent sections describe the restrictions for each restricted command.

Table 134. NAND flash commands

Command Name	First cycle	Second cycle	Supported
Serial Data Input	0x80	-	Restriction
Random Serial Data Input	0x85	-	No
Read Mode (1)	0x00	-	Restriction
Read Mode (2)	0x01	-	No
Read Mode (3)	0x50	-	Restriction
Read Start	0x30	-	Yes
Read Start With Data Cache	0x31	-	Yes
Read Start Page Copy	0x35	-	No
Read Start With Data Cache Last Page	0x3F	-	Yes
Reset	0xFF	-	Yes
Auto Program (true)	0x10	-	Restriction
Auto Program (dummy)	0x11	-	Restriction
Auto Program (cache)	0x15	-	Restriction
Auto Block Erase	0x60	0xD0	Yes
Status Read (1)	0x70	-	Yes
Status Read (2)	0x71	-	Yes
ID Read (1)	0x90	-	Yes
ID Read (2)	0x91	-	Yes

Command usage is restricted during NAND busy periods and also during controller busy periods. Table 135 shows these restrictions.

Table 135. NAND flash commands

Command Name	NAND Busy Access	Controller busy Access
Serial Data Input	No	No
Random Serial Data Input	No	No
Read Mode (1)	No	No
Read Mode (2)	No	No
Read Mode (3)	No	No
Read Start	No	No
Read Start With Data Cache	No	No
Read Start Page Copy	No	No
Read Start With Data Cache Last Page	No	No
Reset	Yes	Yes
Auto Program (true)	No	No
Auto Program (dummy)	No	No
Auto Program (cache)	No	No
Auto Block Erase	No	No
Status Read (1)	Yes	No
Status Read (2)	Yes	No
ID Read (1)	No	No
ID Read (2)	No	No

8.4.3.2.1 Serial data input command

Operation with this command is subject to the following restrictions:

1. Address A0-A7 (A0-A11 for large block devices) must be 0x00.
2. 50h + 80h command sequence (partial page serial input) not allowed.
3. CPU transfers exactly 518 bytes/page.
4. Controller transfers exactly 10 bytes/page.

8.4.3.2.2 Read mode (1)

Operation with this command is subject the following restrictions:

1. Address A0-A7 (A0-A11 for large block devices) must be 0x00.
2. CPU transfers exactly 518/528 bytes/page.

8.4.3.2.3 Read mode (3)

Operation with this command is subject to the following restrictions:

1. Address A0-A7 (A0-A11 for large block devices) must be 00h.
2. CPU reads overhead data from controller's serial Data Buffer.

This command can be used by the CPU when only the overhead data in the NAND page is required. Since the entire page data is required to perform the R/S ECC processing, the controller will translate the command to a Read Mode (1) (00h) command before sending to the NAND flash and therefore the time required to perform a page read and transfer the entire 528 bytes to the controller will be incurred.

The controller will read 528 bytes into the controller's serial Data Buffer and automatically set the Buffer pointer to the Overhead region. The CPU can then read the overhead data directly from the controller's serial Data Buffer. The CPU can also read the 518-byte user data from the serial Data Buffer (the Reset User Buffer Pointer register must first be written).

Note that due to the fact that the entire 528 bytes must be read by the controller, using this command may not be prudent from a timing perspective. The only advantage to using the Read Mode(3) command over the Read Mode (1) command is that the CPU need not read all 518 bytes. This transfer is instead performed by the controller at a possibly higher transfer rate. However, for CPUs with high speed access to the controller, it may be prudent to read all 518 bytes using the Read Mode (1) command if there is a possibility that the extra data may become useful at a later date.

8.4.3.2.4 Auto program commands

These commands are supported only if they are used following the Serial Data Input (0x80) command sequence.

8.4.3.2.5 Status Read commands

Operation with the Status Read (1) (70h) and Status Read (2) (71h) commands is subject to the following restriction(s):

1) CPU must not use these commands unless the controller's controller Ready bit of the controller's Status register is set.

8.4.3.2.6 Software configurable block write protection

The software configurable block protection provides a mechanism for preventing Auto Program (0x10, 0x11, 0x15) and Erase Start (0xD0) commands from being forwarded from the controller to the NAND flash under certain conditions when requested by the CPU. The intent of this feature is to allow write protection of a software configurable section of the NAND flash. This feature is enabled by the CPU software. The software configurable block write protection address range is programmable by the CPU software. A lockout mechanism is provided to prevent rogue tasks from unintentionally modifying the software block protection configuration.

If the software block protection feature is enabled, the Auto Program (0x10, 0x11, 0x15) commands are blocked when any of the following conditions are true:

1. If the address (A9-A24) of the previous Serial Data Input (0x80) lies within the software block write protection address range.
2. If the previous command was not a Serial Data Input (0x80) command.
3. If the previous Serial Data Input (0x80) command was not followed by a complete address sequence.

For any of these conditions, a Reset (0xFF) command, if forwarded in lieu of the Auto Program command, will reset the NAND flash and effectively abort the operation.

If the software block protection feature is enabled, the Erase Start (0xD0) command is blocked when any of the following conditions are true:

1. If the address (A9-A24) of the previous Auto Block Erase (0x60) command lies within the software block write protection address range.
2. If the previous command was not a Auto Block Erase (0x60) command.
3. If the previous Auto Block Erase (0x60) command was not followed by a complete address sequence.

For any of these conditions, a Reset (0xFF) command, if forwarded in lieu of the Start Erase command, will reset the NAND flash and effectively abort the operation.

It is the CPU software's responsibility to detect and manage the Block Write Protection fault events. The CPU controller provides interrupt and status capabilities that allow the CPU software to be notified when a Block Write Protection fault occurs.

8.5 Register description

Table 136 shows the registers associated with the MLC NAND Flash controller and a summary of their functions. Following the table are details for each register.

Table 136. MLC NAND flash registers

Address	Name	Description	Reset value	Access
0x200A 8000	MLC_BUFF	MLC NAND Data Buffer.	-	R/W
0x200B 0000	MLC_DATA	Start of MLC data buffer	-	R/W
0x200B 8004	MLC_ADDR	MLC NAND Flash Address Register.	0x0	WO
0x200B 8000	MLC_CMD	MLC NAND Flash Command Register.	0x0	WO
0x200B 8008	MLC_ECC_ENC_REG	MLC NAND ECC Encode Register.	0x0	WO

Table 136. MLC NAND flash registers

Address	Name	Description	Reset value	Access
0x200B 800C	MLC_ECC_DEC_REG	MLC NAND ECC Decode Register.	0x0	WO
0x200B 8010	MLC_ECC_AUTO_ENC_REG	MLC NAND ECC Auto Encode Register.	0x0	WO
0x200B 8014	MLC_ECC_AUTO_DEC_REG	MLC NAND ECC Auto Decode Register.	0x0	WO
0x200B 8018	MLC_RPR	MLC NAND Read Parity Register.	0x0	WO
0x200B 801C	MLC_WPR	MLC NAND Write Parity Register.	0x0	WO
0x200B 8020	MLC_RUBP	MLC NAND Reset User Buffer Pointer Register.	0x0	WO
0x200B 8024	MLC_ROBP	MLC NAND Reset Overhead Buffer Pointer Register.	0x0	WO
0x200B 8028	MLC_SW_WP_ADD_LOW	MLC NAND Software Write Protection Address Low Register.	0x0	WO
0x200B 802C	MLC_SW_WP_ADD_HIG	MLC NAND Software Write Protection Address High Register.	0x0	WO
0x200B 8030	MLC_ICR	MLC NAND controller Configuration Register.	0x0	WO
0x200B 8034	MLC_TIME_REG	MLC NAND Timing Register.	0x37	WO
0x200B 8038	MLC_IRQ_MR	MLC NAND Interrupt Mask Register.	0x0	WO
0x200B 803C	MLC_IRQ_SR	MLC NAND Interrupt Status Register.	0x0	RO
0x200B 8044	MLC_LOCK_PR	MLC NAND Lock Protection Register.	0x0	WO
0x200B 8048	MLC_ISR	MLC NAND Status Register.	0x0	RO
0x200B 804C	MLC_CEH	MLC NAND Chip-Enable Host Control Register.	0x0	WO

8.5.1 MLC NAND flash Command register (MLC_CMD, RW - 0x200B 8000)

The Command register is an 8-bit write only register used to send commands to the NAND flash (CLE asserted). The software must write all commands to this register. Commands written to the register will be written to the NAND flash with the following exceptions:

1. Read Mode (3) 0x50 (substituted with Read Mode (1) 0x00)
2. Commands blocked by the Software write protection features (if enabled).
3. Commands written while the controller is not ready (except the Reset (0xFF) command which resets the controller).

Note that the normal NAND protocol rules regarding busy access apply when writing commands to this register. The software should access the controller's Status register to determine if commands can be sent to the NAND flash. Commands should not be sent if either the controller Ready or NAND Ready bits of the controller's Status register are not set. The exception is the Reset (0xFF) command which forces both the NAND flash and the controller to abort the current operation. Commands written to this register are forwarded to the NAND flash by the controller. These commands may cause unexpected operation and/or loss of data if the NAND flash is not ready.

Table 137. MLC NAND Flash Command Register (MLC_CMD, RW - 0x200B 8000)

Bits	Description	Reset value
7:0	Command Code	0x0

8.5.2 MLC NAND flash Address register (MLC_ADDR, WO - 0x200B 8004)

The address register is used to send address data to the NAND flash (ALE asserted). Data written to this register will be written to the NAND flash with the following exceptions:

1. Data written while controller is not ready is discarded.

Addresses written to this register are forwarded to the NAND flash by the controller. This may cause unexpected operation and/or loss of data if the NAND flash is not ready.

Table 138. MLC NAND Flash Address Register (MLC_ADDR, WO - 0x200B 8004)

Bits	Description	Reset value
7:0	Address	0x0

8.5.3 MLC NAND ECC Encode Register (MLC_ECC_ENC_REG, WO - 0x200B 8008)

Writing to this register starts a data encode cycle. Any data written to the NAND data address space or the controller's serial Data Buffer space thereafter is encoded by the R/S ECC encoder. Writing to this register will terminate any ongoing encode or decode cycle in order to begin the new encode cycle. Note that it is the CPU's responsibility to ensure that this register is written to prior to sending any page data if error detection/correction is desired. Writing to this register clears the following flags in the controller's Status register:

1. ECC Ready.
2. Errors Detected.
3. Decoder Failure.

Table 139. MLC NAND ECC Encode Register (MLC_ECC_ENC_REG, WO - 0x200B 8008)

Bits	Description	Reset value
7:0	Writing to this register starts a data encode cycle.	0x0

8.5.4 MLC NAND ECC Decode Register (MLC_ECC_DEC_REG, WO - 0x200B 800C)

Writing any data to this register starts a data decode cycle. Any data read from the NAND data address space thereafter is decoded by the R/S ECC decoder. Writing to this register will terminate any ongoing encode or decode cycle in order to begin the new decode cycle. Note that it is the CPU's responsibility to ensure that this register is written to prior to reading any page data if error detection/correction for that page is desired. Writing to this register clears the following flags in the controller's Status register:

1. ECC Ready.
2. Errors Detected.
3. Decoder Failure.

Table 140. MLC NAND ECC Decode Register (MLC_ECC_DEC_REG, WO - 0x200B 800C)

Bits	Description	Reset value
7:0	Writing to this register starts a data decode cycle.	0x0

8.5.5 MLC NAND ECC Auto Encode Register (MLC_ECC_AUTO_ENC_REG, WO - 0x200B 8010)

Writing this register starts an automatic encode cycle. The controller automatically sends 528 bytes to the NAND flash after performing the R/S ECC encoding to the data in the controller's serial Data Buffer. The CPU must first write the Encode register and then write 518 bytes of data to the controller's serial Data Buffer. If the CPU requires the controller to automatically send the Auto-Program command to the NAND flash after sending the parity data, then the CPU should write the desired Auto-Program command (0x10, 0x11, 0x15) to this register. This operation is validated with bit 8 of the register. The controller sends the Auto-program command to the NAND flash after sending the parity data. The controller will then wait for the NAND flash's Ready/nBusy signal (indicating that the NAND flash has completed the Auto-program operation) before indicating the controller Ready status and interrupt update. Note that the CPU must allow the controller to complete the cycle. The CPU should read the controller's Status register to determine the completion of the cycle (or use the interrupt feature of the controller). Any commands and/or data sent to the NAND flash during this time are discarded by the controller (except the Reset (0xFF) command). Writing to this register will terminate any ongoing encode or decode cycle in order to begin the new encode cycle. Writing the Reset (0xFF) command to the Command register will terminate the ongoing cycle.

The intended operation using this register without the automatic Auto-program command is as follows:

1. Write Serial Input command (80h) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes to serial Data Buffer.
5. Write Auto Encode register with Bit 8=0.
6. Read Status register.[1]
7. Wait for controller Ready status bit set.
8. Write Auto-program command.

The intended operation using this register with the automatic Auto-program command is as follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes to serial Data Buffer.
5. Write Auto Encode register with Auto-Program command, Bit 8=1.
6. Read Status register.[1]
7. Wait controller Ready status bit set.

Note that the Reset (0xFF) command may be written to the command register at any time during the above sequence to reset both the controller and the NAND flash.

1. The controller will generate an controller Ready interrupt (if enabled). Failure to follow the above sequences may result in unexpected behavior and/or loss of data.

Table 141. MLC NAND ECC Auto Encode Register (MLC_ECC_AUTO_ENC_REG, WO - 0x200B 8010)

Bits	Description	Reset value
31:9	Reserved.	0x0
8	0: Auto-program command disabled. 1: Auto-program command enabled.	0
7:0	Auto-program command.	0x0

8.5.6 MLC NAND ECC Auto Decode Register (MLC_ECC_AUTO_DEC_REG, WO - 0x200B 8014)

Writing this register starts an automatic decode cycle. The controller automatically reads 528 bytes from the NAND flash and performs the R/S ECC decoding. The CPU can then retrieve the data from the controller's serial Data Buffer after reading the error status and severity from the controller's Status register. The Data Buffer pointer is automatically set to the overhead region if the last command was Read Mode(3) (0x50), otherwise the pointer is set to the user region of the serial Data Buffer. The Reset User Buffer Pointer and Reset Overhead Buffer Pointer registers can be used to re-position the Buffer pointer to the desired region. Note that the CPU must allow the controller to complete the cycle. The CPU should read the controller's Status register to determine the completion of the cycle (or use the interrupt feature of the controller). Any commands and/or data sent to the NAND flash during this time are discarded by the controller (except the Reset (0xFF) command). Writing to this register will terminate any ongoing encode or decode cycle in order to begin the new decode cycle. Writing the Reset (0xFF) command to the Command register will terminate the ongoing cycle. Note that it is the CPU's responsibility to perform the necessary NAND read operation. The intended operation using this register is as follows:

1. CPU sends Read Mode(1)/Read Mode(3) command to NAND flash.
2. CPU sends appropriate address data for the desired page.
3. CPU writes to Auto Decode register.
4. CPU reads Status register until the controller Ready flag is set.[2]
5. CPU reads Status register to determine error status/severity.[3]
6. CPU reads page data from the controller's Data Buffer.

Failure to follow this sequence may result in unexpected behavior and/or loss of data.

Note that the Reset (0xFF) command may be written to the command register at any time during the above sequence to reset both the controller and the NAND flash.

Table 142. MLC NAND ECC Auto Decode Register (MLC_ECC_AUTO_DEC_REG, WO - 0x200B 8014)

Bits	Description	Reset value
7:0	Writing any data to this register starts an automatic decode cycle.	0x0

2. The controller will generate an controller Ready interrupt (if enabled).
3. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled).

8.5.7 MLC NAND Read Parity Register (MLC_RPR, WO - 0x200B 8018)

Writing any data to this register forces the controller to read 10 bytes from the NAND flash device. This data is the parity data used by the R/S ECC decoder. This feature is useful if the CPU has no use for this data and can therefore allow the controller to automatically read it with no CPU intervention. This feature is intended to be used after the CPU has started a decode cycle and read all user data (518 bytes) from the NAND device. Accessing this register will then force the controller to read the parity data (10-bytes) required to complete the ECC decode cycle. Note that the use of this register is optional. The CPU itself can read the parity data directly. Note that the CPU must allow the controller to complete the read sequence. The CPU should read the controller's Status register to determine the completion of the sequence (or use the interrupt feature of the controller). Any commands and/or data sent to the NAND during this time are discarded by the controller.

Table 143. MLC NAND Read Parity Register (MLC_RPR, WO - 0x200B 8018)

Bits	Description	Reset value
7:0	Writing any data to this register force the controller to read 10 byte parity data from the NAND flash device.	0x0

8.5.8 MLC NAND Write Parity Register (MLC_WPR, WO - 0x200B 801C)

Writing any data to this register forces the controller to write the parity data (10 bytes) to the NAND device. This data is the parity data calculated by the R/S ECC encoder. This feature is intended to be used after the CPU has started an encode cycle and written all user data (518 bytes) to the NAND flash device. Accessing this register will then force the controller to write the parity data (10-bytes) as calculated by the R/S ECC encoder. The CPU should read the controller's Status register to determine the completion of the sequence (or use the interrupt feature of the controller). Any commands and/or data sent to the NAND during this time are discarded by the controller.

Table 144. MLC NAND Write Parity Register (MLC_WPR, WO - 0x200B 801C)

Bits	Description	Reset value
7:0	Writing any data to this register force the controller to write 10 byte parity data to the NAND flash device.	0x0

8.5.9 MLC NAND Reset User Buffer Pointer register (MLC_RUBP, WO - 0x200B 8020)

The Reset User Buffer Pointer register is a write only register used to force the serial Data Buffer pointer to the start of the user data region. Access to this buffer is sequential such that if the CPU must start reading data at the beginning of this buffer, this register must be written with any value.

Table 145. MLC NAND Reset User Buffer Pointer Register (MLC_RUBP, WO - 0x200B 8020)

Bits	Description	Reset value
7:0	Writing any data to this register force the serial Data Buffer pointer to the start of the user data region.	0x0

8.5.10 MLC NAND Reset Overhead Buffer Pointer register (MLC_ROBP, WO - 0x200B 8024)

The Reset User Buffer pointer register is a write only register used to force the serial Data Buffer pointer to the start of the overhead data region. Access to this buffer is sequential such that if the CPU must start reading overhead data, this register must be written with any value.

Table 146. MLC NAND Reset Overhead Buffer Pointer Register (MLC_ROBP, WO - 0x200B 8024)

Bits	Description	Reset value
0	Writing any data to this register force the serial Data Buffer pointer to THE start of the overhead data region.	0x0

8.5.11 MLC NAND Software Write Protection Address Low register (MLC_SW_WP_ADD_LOW, WO - 0x200B 8028)

The Software Write Protection address registers are 24-bit write only registers. They are used to store the address range used for the software write protect feature. The address low register contains the lower bound for the write protected area. The address high register contains the upper bound for the write protected area. These registers are compared with address bytes 2,3,4 (2,3 for three byte address devices) that follow the Serial Data input (0x80) and Auto Block Erase command (0x60). Note: On large page devices these registers are compared with address bytes 3,4,5 (3, 4 for four byte address devices). If this address falls within the address range specified by the registers then the Reset (0xFF) command is sent to the NAND device in lieu of the Auto Program (0x10, 0x11, 0x15) or the Auto Block Erase Second Cycle (0xD0) sent by the host. This will effectively abort the operation.

Note that in order to modify these registers, the Lock Protect register must first be written with the appropriate value.

Table 147. MLC NAND Software Write Protection Address Low Register (MLC_SW_WP_ADD_LOW, WO - 0x200B 8028)

Bits	Description	Reset value
23:0	The lower bound for the write protected area.	0x0

8.5.12 MLC NAND Software Write Protection Address High register (MLC_SW_WP_ADD_HIG, WO - 0x200B 802C)

Table 148. MLC NAND Software Write Protection Address High Register (MLC_SW_WP_ADD_HIG, WO - 0x200B 802C)

Bits	Description	Reset value
23:0	The upper bound for the write protected area.	0x0

8.5.13 MLC NAND Controller Configuration register (MLC_ICR, WO - 0x200B 8030)

This register is used to configure the controller as shown below. Note that in order to modify this register, the Lock Protect register must first be written with the appropriate value.

Table 149. MLC NAND Controller Configuration Register (MLC_ICR, WO - 0x200B 8030)

Bits	Description	Reset value
31:4	Reserved	0x0
3	0: Software Write protection disabled. 1: Software Write protection enabled.	0
2	0: small block flash device (512 +16 byte pages). 1: large block flash device (2k + 64 byte pages).	0
1	0: NAND flash address word count 3 for small page device, 4 for large page device. 1: NAND flash address word count 4 for small page device, 5 for large page device.	0
0	0: NAND flash I/O bus with 8-bit. 1: NAND flash I/O bus with 16-bit (Not supported).	0

8.5.14 MLC NAND Timing Register (MLC_TIME_REG, WO - 0x200B 8034)

These values should be configured to match the NAND device timing requirements. The values represent multiples of HCLK. Note that in order to modify this register, the Lock Protect register must first be written with the appropriate value.

Table 150. MLC NAND Timing Register MLC_TIME_REG, (WO - 0x200B 8034)

Bits	Bitfield Name	Description	Reset value
31:26	-	Reserved	0x0
25:24	TCEA_DELAY	TCEA_DELAY (nCE low to dout valid, tCEA). The parameter should be configured as follows: ((TCEA_DELAY)/HCLK) >= (tCEA - tREA), where tCEA is the Chip Enable Low to Output Valid parameter of the NAND device, and tREA is the Read Enable Low to Output Valid parameter of the NAND device.	0x0
23:19	BUSY_DELAY	BUSY_DELAY (Read/Write high to busy, tWB/tRB). The parameter should be configured as follows: ((BUSY_DELAY)/HCLK) >= max(tWB, tRB), where tWB is the Write Enable High to Ready/Busy Low parameter of the NAND device, and tRB is the Read Enable High to Ready/Busy Low parameter of the NAND device.	0x0
18:16	NAND_TA	NAND_TA (Read high to high impedance, tRHZ). The parameter should be configured as follows: ((RD_HIGH - RD_LOW)/HCLK) + (NAND_TA/HCLK) >= tRHZ, where tRHZ is the Read Enable High to Output Hi-Z parameter of the NAND device.	0x0
12:15	RD_HIGH	RD_HIGH (Read high hold time, tREH) The parameter should be configured as follows: ((RD_HIGH - RD_LOW)/HCLK) >= tREH and ((RD_LOW + 1)/HCLK) + ((RD_HIGH - RD_LOW)/HCLK) >= tRC, where tREH is the Read Enable High to Read Enable Low parameter of the NAND device, and tRC is the Read Enable Low to Read Enable Low parameter of the NAND device.	0x0

Table 150. MLC NAND Timing Register MLC_TIME_REG, (WO - 0x200B 8034)

Bits	Bitfield Name	Description	Reset value
11:8	RD_LOW	RD_LOW (Read pulse width, tRP) the parameter should be configured as follows: ((RD_LOW + 1)/HCLK) >= tRP and ((RD_LOW + 1)/HCLK) >= tREA + tSU Where tRP is the Read Enable Low to Read Enable High parameter of the NAND device, tREA is the Read Enable Low to Output Valid parameter of the NAND device, and tSU is the Interface's setup time requirement for the NAND_IO bus.	0x0
7:4	WR_HIGH	WR_HIGH (Write high hold time, tWH) The parameter should be configured as follows: ((WR_HIGH - WR_LOW)/HCLK) >= tWH and ((WR_LOW+1)/HCLK) + ((WR_HIGH - WR_LOW)/HCLK) >= tWC, Where tWH is the Write Enable High to Write Enable Low parameter of the NAND device, and tWC is the Write Enable Low to Write Enable Low parameter of the NAND device.	0x0
3:0	WR_LOW	WR_LOW (Write pulse width, tWP) the parameter should be configured as follows: ((WR_LOW+1)/HCLK) >= tWP Where tWP is the Write Enable Low to Write Enable High parameter of the NAND device.	0x7

8.5.15 MLC NAND Interrupt Mask Register (MLC_IRQ_MR, WO - 0x200B 8038)

Setting each bit in this register enables the corresponding interrupt. At reset, all interrupts are masked. Each mask bit is logically ANDed with the corresponding interrupt and the results from all the interrupts are logically ORed to create the controller's interrupt signal.

Table 151. MLC NAND Interrupt Mask Register (MLC_IRQ_MR, WO - 0x200B 8038)

Bits	Description	Reset value
7:6	Reserved.	0x0
5	NAND Ready (0: Disabled, 1: Enabled) This interrupt occurs when the NAND flash's Ready/nBusy signal transitions from the Busy state to the Ready state. This interrupt is delayed by the NAND flash's tWB/tRB parameters.	0
4	Controller Ready (0: Disabled, 1: Enabled) This interrupt indicates that the controller has completed one of the following actions: 1) Parity read complete 2) Parity write complete 3) Auto decode complete 4) Auto encode complete	0
3	Decode failure (0: Disabled, 1: Enabled) This interrupt indicates that the R/S ECC decoder has detected errors present in the last decode cycle that cannot be properly corrected (this indicates that the severity of the error exceeds the correction capability of the decoder).	0

Table 151. MLC NAND Interrupt Mask Register (MLC_IRQ_MR, WO - 0x200B 8038)

Bits	Description	Reset value
2	Decode error detected (0: Disabled, 1: Enabled) This interrupt indicates that the R/S ECC decoder has detected (and possibly corrected) errors present in the last decode cycle. The CPU should read the controller's Status register to determine the severity of the error. The CPU should also discard the data and read the corrected data from the controller's serial Data Buffer.	0
1	ECC Encode/Decode ready (0: Disabled, 1: Enabled) This interrupt indicates that the ECC Encoder or Decoder has completed the encoding or decoding process. For an encode cycle this interrupt occurs after the following actions: 1) Host begins encoding cycle by accessing the ECC Encode register, 2) Host writes 518 bytes of NAND data, and 3) R/S ECC encoding completes. For a decode cycle this interrupt occurs after the following actions: 1) Host begins decoding cycle by accessing the ECC Decode register, 2) Host reads 518/528 bytes of NAND data, and 3) R/S ECC decoding completes.	0
0	Software write protection fault (0: Disabled, 1: Enabled) This interrupt indicates that the last NAND write operation was aborted due to a write protection fault. This interrupt can occur after the Erase Start (0x60) command or any Auto Program (0x10, 0x11, 0x15) command is written to the NAND after the previous address data following the Serial Input (0x80) or Auto Erase (0x60) commands falls within the software protection address range and software write protection is enabled.	0

8.5.16 MLC NAND Interrupt Status Register (MLC_IRQ_SR, RO - 0x200 803C)

The interrupt status register is used for polling interrupt source information. A set bit indicates that the corresponding interrupt has occurred. The entire register contents are cleared once the register is read such that there is no need to clear this register to reset the interrupts. Note that this registers reflects the interrupts regardless of the Interrupt Mask register. Each interrupt bit is logically ANDed with the corresponding interrupt mask bit and the results from all the interrupts are logically ORed to create the controller's interrupt signal.

Table 152. MLC NAND Interrupt Status Register (MLC_IRQ_SR, RO - 0x200B 803C)

Bits	Description	Reset value
7:6	Reserved.	0x0
5	NAND Ready (0: Inactive, 1: Active) This interrupt occurs when the NAND flash's Ready/nBusy signal transitions from the Busy state to the Ready state. This interrupt is delayed by the NAND flash's tWB/tRB parameters.	0
4	controller Ready (0: Inactive, 1: Active) This interrupt indicates that the controller has completed one of the following actions: 1) Parity read complete 2) Parity write complete 3) Auto decode complete 4) Auto encode complete	0
3	Decode failure (0: Inactive, 1: Active) This interrupt indicates that the R/S ECC decoder has detected errors present in the last decode cycle that cannot be properly corrected (this indicates that the severity of the error exceeds the correction capability of the decoder).	0

Table 152. MLC NAND Interrupt Status Register (MLC_IRQ_SR, RO - 0x200B 803C)

Bits	Description	Reset value
2	Decode error detected (0: Inactive, 1: Active) This interrupt indicates that the R/S ECC decoder has detected (and possibly corrected) errors present in the last decode cycle. The CPU should read the controller's Status register to determine the severity of the error. The CPU should also discard the data and read the corrected data from the controller's serial Data Buffer.	0
1	ECC Encode/Decode ready (0: Inactive, 1: Active) This interrupt indicates that the ECC Encoder or Decoder has completed the encoding or decoding process. For an encode cycle this interrupt occurs after the following actions: 1) Host begins encoding cycle by accessing the ECC Encode register, 2) Host writes 518 bytes of NAND data, and 3) R/S ECC encoding completes. For a decode cycle this interrupt occurs after the following actions: 1) Host begins decoding cycle by accessing the ECC Decode register, 2) Host reads 518/528 bytes of NAND data, and 3) R/S ECC decoding completes.	0
0	Software write protection fault (0: Inactive, 1: Active) This interrupt indicates that the last NAND write operation was aborted due to a write protection fault. This interrupt can occur after the Erase Start (0x60) command or any Auto Program (0x10, 0x11, 0x15) command is written to the NAND after the previous address data following the Serial Input (0x80) or Auto Erase (0x60) commands falls within the software protection address range and software write protection is enabled.	0

8.5.17 MLC NAND Lock Protection Register (MLC_LOCK_PR, WO - 0x200B 8044)

The Lock Protect register is used to provide a lockout feature to prevent certain registers from being inadvertently written. Writing a value of 0xA25E to this register unlocks the access to these registers. Access becomes locked immediately after any of these registers are accessed. The registers affected by this feature are:

1. Software Write Protection Address Low.
2. Software Write Protection Address High.
3. Controller configuration.
4. NAND Timing.

Table 153. MLC NAND Lock Protection Register (MLC_LOCK_PR, WO - 0x200B 8044)

Bits	Description	Reset value
15:0	Writing a value of 0xA25E to this register unlocks the access to, MLC_SW_WP_ADD_LOW, MLC_SW_WP_ADD_HIG, MLC_ICR, MLC_WP_REG and MLC_TIME_REG. Access becomes locked immediately after any of these registers are accessed.	0x0

8.5.18 MLC NAND Status Register (MLC_ISR, RO - 0x200B 8048)

The Status register indicates the status of the last R/S ECC encode/decode cycle as well as the status of the Ready/nBusy NAND flash signal.

Table 154. MLC NAND Status Register (MLC_ISR, RO - 0x200B 8048)

Bits	Description	Reset value
7	Reserved	0x0
6	Decoder Failure This flag indicates that the last R/S Decoding cycle was unsuccessful at correcting errors present in the data. This indicates that the number of errors in the data exceeds the decoder's correction ability (more than 4 symbols). The host should inspect this flag prior to validating the data read during the last decoding cycle.	0
5:4	Number of R/S symbols errors This 2-bit field indicates the number of symbol errors detected by the last R/S decoding cycle. Note that this field is only valid when both the following conditions are met: 1) Errors Detected flag is set and 2) Decoder Failure flag is clear. 00: One symbol-error detected. 01: Two symbol-error detected. 10: Three symbol-error detected. 11: Four symbol-error detected.	0
3	ERRORS DETECTED This flag indicates that the last R/S Decode cycle has detected errors in the page data. This flag does not indicate error severity but merely indicates that errors have been detected.	0
2	ECC READY This flag indicates the R/S ECC encoding/decoding process has been completed The Host must check this flag prior to using data read during a decode cycle. The CPU can also check the status of an encode cycle prior to accessing the Write Parity register (this in not necessary since the controller ensures that the R/S encoding has completed before writing any data)	0
1	Controller READY This flag indicates that the controller has completed any of the following: 1) Read parity cycle 2), Write parity cycle, 3) Auto Encode cycle and 4) Auto Decode cycle. The flag is cleared when any of the above operations are started. The flag must be checked by the CPU prior to attempting an access to the corresponding NAND flash device. Failure to perform the check may result in unexpected operation and/or data loss.	0
0	NAND READY This flag reflects the status of the NAND flash's Ready/nBusy signal. Note that the CPU need not consider the NAND flash's tWB, tRB timing parameters. The controller delays the update of the NAND ready flag when data, address, or commands are sent to the NAND flash. This ensures that the NAND ready flag remains clear until the tWB, tRB time has passed and the true status of the NAND flash's Ready/nBusy signal can be reported.	0

8.5.19 MLC NAND Chip-Enable Host Control register (MLC_CEH, WO - 0x200B 804C)

This register allows the CPU to force the NAND flash's Chip-Enable control signal (nCE) to remain asserted. This type of operation allows the use of NAND flash devices that require nCE to remain asserted throughout transfers (devices that do not support "CE

don't care" operation). When this type of operation is required, the CPU must first write to this register to assert nCE prior to performing any transfers (including controller initiated transfers). The CPU can then perform the required transfers. For power consumption reasons, the host should then allow nCE to be de-asserted when the required transfers are complete. To allow nCE to be de-asserted, the host must again write to this register. Note that nCE may not be de-asserted immediately, but instead, nCE operation will revert back to normal operation, ensuring that nCE is de-asserted by the controller.

Table 155. MLC NAND Chip-Enable Host Control Register (MLC_CEH, WO - 0x200B 804C)

Bits	Description	Reset value
31:1	Reserved	0x0
0	0: Force nCE assert	0x0
	1: Normal nCE operation (nCE controlled by controller)	

8.5.20 MLC NAND Data register (MLC_DATA, R/W - starting at address 0x200B 0000)

NAND flash devices use the CLE and ALE hardware signals to allow three types of access: command write; address write; and data read/write.

To perform a command write access, command data must be written to the MLC_CMD register. The MLC interface forwards the command to the NAND device by driving it on the NAND I/O bus while asserting the CLE signal.

To perform an Address write access, address data must be written to the MLC_ADDR register. The MLC interface forwards the address to the NAND device by driving it on the NAND I/O bus while asserting the ALE signal.

In order to accomplish data read or write to a NAND Flash device, accessed must be done sequentially by reads from or writes to any address within the Data address space. The address space supports 8, 16, and 32-bit read accesses. All three widths of access are supported when using 8-bit wide NAND devices. The interface splits transfers of greater width than the NAND device. This allows data to be moved more efficiently across 32-bit buses.

Note that caution must be used when mixing different word length accesses due to the nature of the sequential access. In order to read 518 bytes (all user data), 32-bit word length access is not suitable for the entire process. The recommended method is to read 516 bytes using 32-bit accesses and the remaining 2 bytes using 8-bit or 16-bit accesses.

The address range of the data area allows data to be burst read from or written to memory because the lower address bits have no effect on the access. It is not necessary to prevent the address from incrementing.

Table 156. MLC NAND Data Register (MLC_DATA, R/W - starting at address 0x200B 0000)

Bits	Description	Reset value
31:0	NAND Flash data access	-

8.5.21 MLC NAND Buffer register (MLC_BUFF, R/W - starting at address 0x200A 8000)

The serial Data Buffer is used by the R/S ECC Encoder/Decoder to store page data. This buffer is 528-bytes in length and can only be accessed in a sequential manner. A buffer pointer is used to access the two main regions of the buffer. These regions are referred to as User and Overhead regions. The User region contains 512 bytes of user data. The Overhead region contains 6 bytes of user data and 10 bytes of parity data. To read/write data from/to the data buffer, the Data Buffer pointer should first be set to the desired region using the Reset User Buffer Pointer and Reset Overhead Buffer Pointer registers (some operations automatically update the pointer). Data can then be sequentially read from or written to that region.

The Data Buffer is accessed sequentially by reads from or writes to any address within the Buffer address space. The address space supports 8, 16, and 32-bit read accesses. All three widths of access are supported when using 8-bit wide NAND devices. The interface splits transfers of greater width than the NAND device. This allows data to be moved more efficiently across 32-bit buses.

Note that caution must be used when mixing different word length accesses due to the nature of the sequential access. In order to read 518 bytes (all user data), 32-bit word length access is not suitable for the entire process. The recommended method is to read 516 bytes using 32-bit accesses and the remaining 2 bytes using 8-bit or 16-bit accesses.

The address range of the data buffer area allows data to be burst read from, or written to memory because the lower address bits have no effect on the access. It is not necessary to prevent the address from incrementing.

Table 157. MLC NAND Buffer Register (MLC_BUFF, R/W - starting at address 0x200A 8000)

Bits	Description	Reset value
31:0	NAND Flash data buffer	-

8.6 MLC NAND controller usage

This section shows examples of the command sequences and the controller interactions necessary to perform the typical NAND page read and write operations.

8.6.1 Small block page read operation

The typical NAND page read operation involves the following steps performed by the CPU:

1. Write page read command.
2. Write page address data.
3. Wait until NAND device indicates ready.
4. Read page data.

This sequence must be modified as described in the following sections

8.6.1.1 Read Mode (1)

To perform a page read operation using the Read Mode (1) (00h) command, the CPU can choose two methods. The first is referred to as Normal Decode and the second as Automatic Decode. The difference in these methods is that in the Normal Decode operation the data is transferred directly from the NAND to the CPU as requested by the CPU. This data, however, may contain errors. The CPU must therefore check the controller before making use of the data (by reading the controller's Status register). If the controller indicates that the data contains errors as determined by the R/S ECC processing, the CPU must discard the data and retrieve the error free data from the controller's serial Data Buffer.

In the Auto Decode operation, the CPU forces the controller to read the NAND data into its Data Buffer first. The CPU then reads the error free data from the controller's serial Data Buffer. If the error occurrence is expected to be low, then the Normal Decode operation can yield higher performance.

Normal decode

1. Write Read Mode (1) command (0x00) to Command register.
2. Write address data to Address register.
3. Read controller's Status register.
4. Wait until NAND Ready status bit set.
5. Write Start Decode register.
6. Read 518 NAND data bytes.
7. Write Read Parity register.
8. Read Status register.[4]
9. Wait until ECC Ready status bit set.
10. Check error detection/correction status.[5]
11. If error was detected, read 518/528 bytes from serial Data Buffer.

Step 7 may be omitted if 528 bytes are read in step 6 rather than 518 bytes.

Auto decode

1. Write Read Mode (1) command (0x00) to Command register
2. Write address data to Address register.
3. Write Start Auto Decode register.
4. Read Status Register.[6]
5. Wait until controller Ready status bit set.
6. Check error detection/correction status.[7]
7. Read 518/528 bytes from the serial Data Buffer.

4. The controller will generate an ECC Ready interrupt (if enabled).
5. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled).
6. The controller will generate an controller Ready interrupt (if enabled).
7. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled).

8.6.1.2 Read Mode (3)

To perform a page read operation using the Read Mode (3) command, the CPU must perform the following steps:

1. Write Read Mode (3) command (0x50) to Command register.
2. Write address data to Address register.
3. Write Start Auto Decode Register.
4. Read Status register.[8]
5. Wait until controller Ready status bit set.
6. Check error detection/correction status.[9]
7. Read 6/16 bytes from the serial Data Buffer.

Note that the CPU writes a Read Mode (3) command but the controller automatically substitutes this command with a Read Mode (0) command. This is necessary because the entire page data is necessary for the R/S ECC processing.

8.6.2 Large block page read operation

The typical NAND page read operation involves the following steps performed by the CPU:

1. Write page read command.
2. Write page address data.
3. Wait until NAND device indicates ready.
4. Read page data.

This sequence must be modified as described in the following sections.

8.6.2.1 Read Mode (1)

To perform a page read operation of a large block flash device using the Read Mode (1) (0x00) command, the CPU follows a procedure similar to that of the small block flash device. The difference is that the CPU must perform four decode cycles to read the entire page data.

8. The controller will generate an controller Ready interrupt (if enabled).
9. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled).

Normal decode

1. Write Read Mode (1) command (0x00) to Command register.
2. Write Read Start command (0x30) to Command register.
3. Write address data to Address register.
4. Read controller's Status register.
5. Wait until NAND Ready status bit set.
6. Write Start Decode register.
7. Read 518 NAND data bytes.
8. Write Read Parity register.
9. Read Status register.[10]
10. Wait until ECC Ready status bit set.
11. Check error detection/correction status.[11]
12. If error was detected, read 518/528 bytes from serial Data Buffer.
13. Repeat steps 6 to 12 for 2nd quarter page.
14. Repeat steps 6 to 12 for 3rd quarter page.
15. Repeat steps 6 to 12 for 4th quarter page.

Step 7 may be omitted if 528 bytes are read in step 6 rather than 518 bytes.

Auto decode

1. Write Read Mode (1) command (0x00) to Command register.
2. Write Read Start command (0x30h) to Command register.
3. Write address data to Address register.
4. Write Start Auto Decode register.
5. Read Status Register.[12]
6. Wait for controller Ready status bit set.
7. Check error detection/correction status.[13]
8. Read 518/528 bytes from the serial Data Buffer.
9. Repeat 4-8 for 2nd quarter page.
10. Repeat 4-8 for 3rd quarter page.
11. Repeat 4-8 for 4th quarter page.

8.6.2.2 Read Mode (3)

To perform a page read operation using the Read Mode (3) command, the CPU must perform the following steps:

1. Write Read Mode (3) command (0x50) to Command register.
2. Write address data to Address register.

10. The controller will generate an ECC Ready interrupt (if enabled).
11. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled).
12. The controller will generate an controller Ready interrupt (if enabled).
13. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled)

3. Write Start Auto Decode Register.
4. Read Status register.[14]
5. Wait controller Ready status bit set.
6. Check error detection/correction status.[15]
7. Read 6/16 bytes from the serial Data Buffer.
8. Repeat 3 to 7 for 2nd quarter of overhead data.
9. Repeat 3 to 7 for 3rd quarter of overhead data.
10. Repeat 3 to 7 for 4th quarter of overhead data.

Note that the CPU writes a Read Mode (3) command but the controller automatically substitutes this command with a Read Mode (0) command. This is necessary because the entire page data is necessary for the R/S ECC processing.

8.6.3 Small block page write operation

The typical NAND page write operation involves the following steps performed by the CPU:

1. Write serial input command.
2. Write page address data.
3. Write page data.
4. Write Auto Program command.

Normal encode

The sequence must be modified follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes of NAND data.
5. Write MLC NAND Write Parity register.
6. Read Status register.[16]
7. Wait controller Ready status bit set.
8. Write Auto Program command to Command register.

Auto encode

The sequence must be modified follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes to serial Data Buffer.

14. The controller will generate an controller Ready interrupt (if enabled).
15. The controller will generate an Error Detected or Decoder Failure interrupt (if enabled).
16. The controller will generate an controller Ready interrupt (if enabled).

5. Write Auto Encode register with Bit 8 = 0.
6. Read Status register.[17]
7. Wait controller Ready status bit set.
8. Write Auto Program command to Command register.

Alternately, if the CPU requires the controller to automatically send the Auto-Program command to the NAND flash after sending the parity data the sequence must be modified follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes to serial Data Buffer.
5. Write Auto Encode register with Auto-Program command, Bit 8 = 1.
6. Read Status register.[17]
7. Wait controller Ready status bit set.

8.6.4 Large block page write operation

The typical NAND page write operation involves the following steps performed by the CPU:

1. Write serial input command.
2. Write page address data.
3. Write page data.
4. Write Auto Program command.

Normal encode

The sequence must be modified follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes of NAND data.
5. Write MLC NAND Write Parity register.
6. Read Status register.[18]
7. Wait controller Ready status bit set.
8. Repeat 3 to 7 for 2nd quarter page.
9. Repeat 3 to 7 for 3rd quarter page.
10. 10) Repeat 3 to 7 for 4th quarter page.
11. Write Auto Program command to Command register.

17. The controller will generate an controller Ready interrupt (if enabled).
18. The controller will generate an controller Ready interrupt (if enabled).

Auto encode

The sequence must be modified follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes to serial Data Buffer.
5. Write Auto Encode register with Bit 8 = 0.
6. Read Status register.[19]
7. Wait controller Ready status bit set.
8. Repeat 3 to 7 for 2nd quarter page
9. Repeat 3 to 7 for 3rd quarter page
10. Repeat 3 to 7 for 4th quarter page
11. Write Auto Program command to Command register.

Alternately, if the CPU requires the controller to automatically send the Auto-Program command to the NAND flash after sending the parity data the sequence must be modified follows:

1. Write Serial Input command (0x80) to Command register.
2. Write page address data to Address register.
3. Write Start Encode register.
4. Write 518 bytes to serial Data Buffer (first quarter page).
5. Write Auto Encode register with Bit 8=0.
6. Read Status register.[19]
7. Wait controller Ready status bit set.
8. Repeat 3 to 7 for 2nd quarter page.
9. Repeat 3 to 7 for 3rd quarter page.
10. Write Start Encode register.
11. Write 518 bytes to serial Data Buffer (last quarter page).
12. Write Auto Encode register with Auto-Program command, Bit 8=1.
13. Read Status register.[19]
14. Wait controller Ready status bit set.

8.6.5 Block erase operation

The typical NAND block erase operation involves the following steps performed by the CPU:

1. Write Auto Block Erase command (0x60) to Command register.
2. Write block address data to Address register.
3. Write Erase Start command (0xD0) to Command register.

19. The controller will generate an controller Ready interrupt (if enabled).

This sequence remains unchanged. Note, however, that the controller's Hardware and Software Write protection features may interfere with this command sequence. These features are accomplished by the controller by withholding the Erase Start command (D0) written by the CPU under certain conditions. Also note that the controller withholds the command if the CPU attempts to write the Erase Start command (D0) without first writing the Auto Block Erase command and the appropriate block address data. The controller withholds the command and substitutes it with a Reset (0xFF) command to restore the NAND flash to a known state.

8.6.6 Other operations

All other operations remain identical to the standard NAND operation. Note that due to the controller's requirement to access the NAND device on its own, the CPU must first read the controller's Status register to ensure that the controller is not currently accessing the NAND device to complete active Encode/Decode operations. Failure to perform this operation may result in unexpected operation and/or lost data.

9.1 Introduction

Note: The LPC32x0 has two NAND flash controllers, one for multi level NAND flash devices and one for single-level NAND flash devices. The two NAND flash controllers use the same pins to interface to external NAND flash devices, so only one interface may be active at a time. The NAND flash controllers can be disabled by bits in the FLASHCLK_CTRL register in order to save power when they are not used.

The Single Level Cell SLC NAND flash controller interfaces to single-level NAND flash devices. An external NAND flash device (of either single-level or multi-level type) may be used to allow the bootloader to automatically load application code into internal RAM for execution.

9.2 Features

- 8 bit wide NAND flashes.
- DMA page transfers.
- 20 byte DMA read and write FIFO, 8 byte command FIFO.
- Hardware support for ECC (Error Checking and Correction) on the main data area. If an error is detected, software must correct it. Two bit error detection and one bit error correction is supported. Error detection on the spare area must be done in software.
- Support for 4 and 5 address cycle NAND devices.

9.3 Pin descriptions

Table 158. NAND flash controller pins

Pin name	Type	NAND flash signal	Function
FLASH_CE_N	output	CEn	Chip select, active LOW.
FLASH_WR_N	output	WEn	Write enable, active LOW.
FLASH_RD_N	output	REn	Read Enable, active LOW.
FLASH_ALE	output	ALE	Address Latch Enable.
FLASH_CLE	output	CLE	Command Latch Enable.
FLASH_RDY	input	RDY	Active HIGH Ready signal.
FLASH_IO[7:0]	input/output	D_IO	I/O pins, commands, address and data.

9.3.1 Interrupt signals from NAND flash controllers

The interrupt from the MLC NAND flash controller is masked with NAND_INT_E and ORed with the interrupt signal from the SLC NAND flash controller before it goes to the interrupt controller. The connections of the interrupts of the MLC and SLC NAND flash controllers are shown in Figure 24.

9.3.2 DMA request signals from flash controllers

The dma_breq(0), dma_sreq(0), and dma_sreq(1) are ORed together and connected to the DMA controller as the burst request signal from the SLC flash controller (DMA controller peripheral number 1). To be able to use a peripheral to peripheral DMA transfer to the SLC NAND flash controller, this burst request signal is also connected to DMA controller peripheral number 12 when the SLC flash controller is selected.

When the MLC NAND flash controller is selected, the burst request signal from the MLC flash controller is connected to DMA controller peripheral number 12.

The connections of the DMA signals of the MLC and SLC NAND flash controllers are shown in Figure 24.

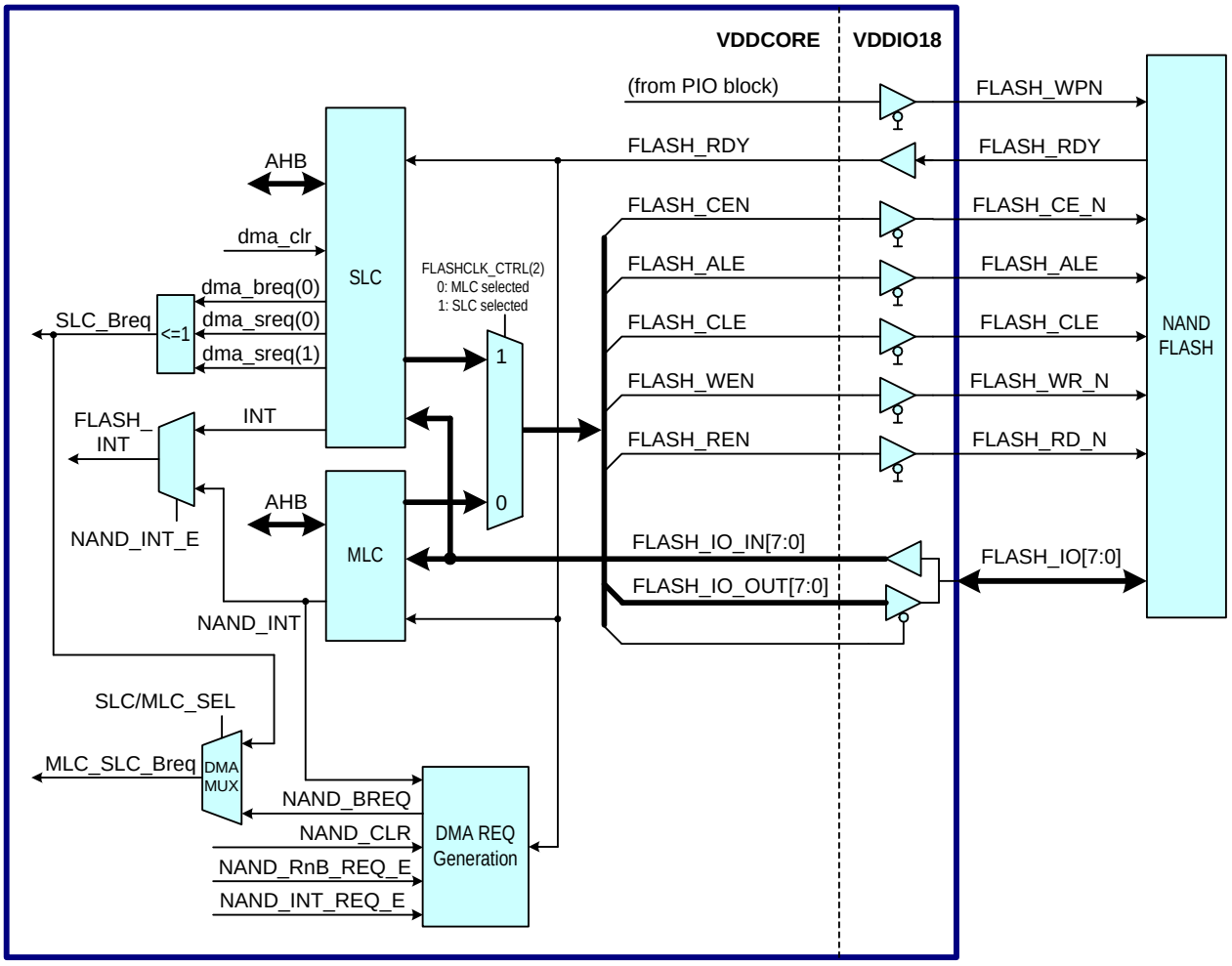


Fig 24. NAND flash connections

9.4 SLC NAND flash controller description

A block diagram of the SLC NAND flash controller is shown in Figure 25.

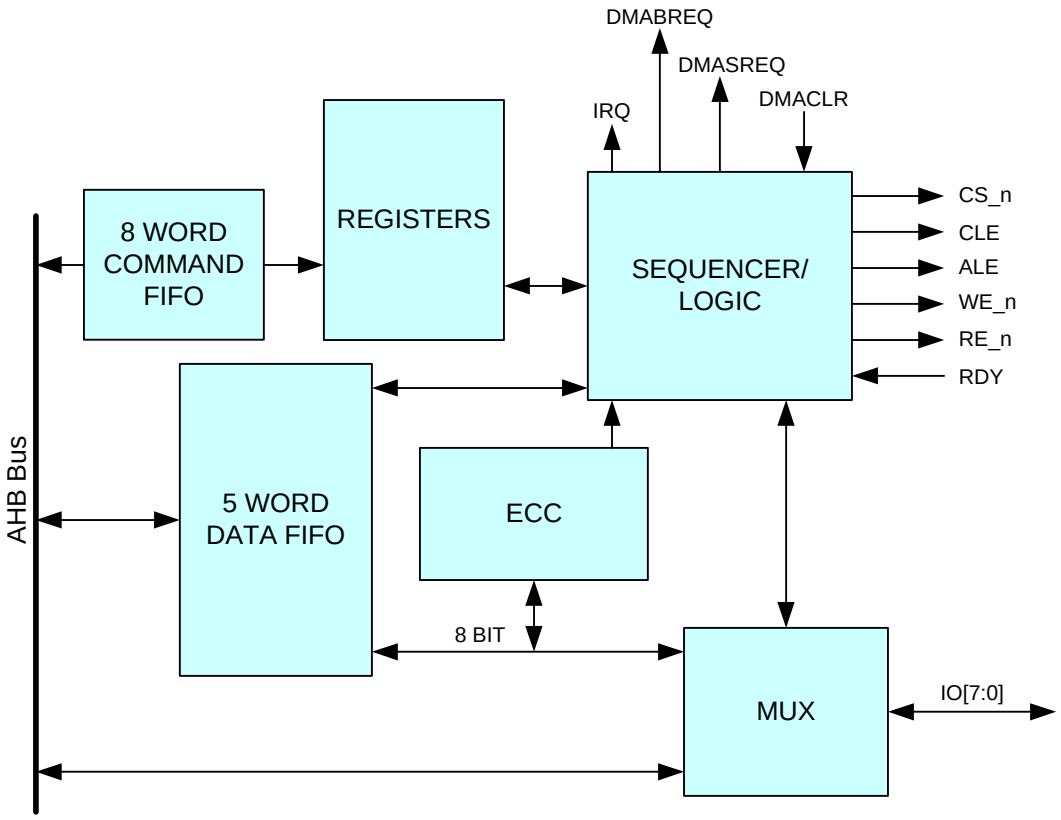


Fig 25. Block diagram of the SLC NAND flash controller

9.5 DMA interface

The following DMA signals are used on the SLC NAND flash controller. Only one request signal can be active at a time. The request signal remains asserted until the DMA controller asserts the DMACLR signal.

9.5.1 DMASREQ

Single word DMA request.

9.5.1.1 DMABREQ

Burst DMA request. The DMABREQ signal is used in the data transfer phase. When reading, it is asserted if the data FIFO has 4 words or more to transfer. When writing, it is asserted if there are less than 4 words in the data FIFO.

Note: the SLC controller produces the signals dma_breq(0), dma_sreq(0), and dma_sreq(1), which are ORed together and connected to the DMA controller as the burst request signal from the SLC flash controller (as peripheral number 1). To be able to use a peripheral to peripheral DMA transfer to the SLC NAND flash controller, this burst request signal is also connected to DMA controller peripheral number 12 when the SLC flash controller is enabled. Refer to the DMA Controller chapter for details of DMA operation.

9.5.1.2 DMACLR

DMA request clear input. The DMA controller asserts this signal during the transfer of the last word in a burst transfer.

9.5.2 Data FIFO

There is only one Data FIFO. The Data FIFO is configured either in Read or in Write mode.

1. When the Data FIFO is configured in Read mode, the sequencer reads data from the NAND flash, and stores the data in the Data FIFO. The FIFO is then emptied by 32-bit reads on the AHB bus from either the ARM or the DMA.
2. When the Data FIFO is configured in Write mode, the ARM or the DMA writes data to the FIFO with 32-bit AHB bus writes. The sequencer then takes data out of the FIFO 8 bits at a time, and writes data to the NAND flash.

9.6 Register description

Table 159 shows the registers associated with the single-level NAND flash controller and a summary of their functions. Following the table are details for each register.

Table 159. Single-level NAND flash controller registers

Address	Name	Description	Reset value	Access
0x2002 0000	SLC_DATA	SLC NAND flash Data Register	-	R/W
0x2002 0004	SLC_ADDR	SLC NAND flash Address Register	-	W
0x2002 0008	SLC_CMD	SLC NAND flash Command Register	-	W
0x2002 000C	SLC_STOP	SLC NAND flash STOP Register	-	W
0x2002 0010	SLC_CTRL	SLC NAND flash Control Register	0x00	R/W
0x2002 0014	SLC_CFG	SLC NAND flash Configuration Register	0x00	R/W
0x2002 0018	SLC_STAT	SLC NAND flash Status Register	00X binary	R
0x2002 001C	SLC_INT_STAT	SLC NAND flash Interrupt Status Register	0x00	R
0x2002 0020	SLC_IEN	SLC NAND flash Interrupt enable register	0x00	R/W
0x2002 0024	SLC_ISR	SLC NAND flash Interrupt set register	0x00	W
0x2002 0028	SLC_ICR	SLC NAND flash Interrupt clear register	0x00	W
0x2002 002C	SLC_TAC	SLC NAND flash Read Timing Arcs Configuration Register	0x00	R/W
0x2002 0030	SLC_TC	SLC NAND flash Transfer Count Register	0x00	R/W
0x2002 0034	SLC_ECC	SLC NAND flash Parity bits	0x00	R
0x2002 0038	SLC_DMA_DATA	SLC NAND flash DMA DATA	-	R/W

9.6.1 SLC NAND flash Data register (SLC_DATA - 0x2002 0000)

SLC_DATA is a 16-bit wide register providing direct access to the NAND flash. The function of bits in SLC_DATA are shown in Table 160. Write data is buffered before being transferred to flash memory. SLC_DATA must be accessed as a word register, although only 8 bits of data are used during a write or provided during a read.

Any read of SLC_DATA are translated to read cycles to flash memory and the bus transaction is extended until requested data are available.

Table 160. SLC NAND flash Data register (SLC_DATA - 0x2002 0000)

Bits	Description	Reset value
15:8	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
7:0	NAND flash read or write data.	-

9.6.2 SLC NAND flash Address register (SLC_ADDR - 0x2002 0004)

SLC_ADDR is an 8-bit wide register providing direct access to the NAND flash address register. The function of bits in SLC_ADDR are shown in Table 161. The ALE output is driven after a write to SLC_ADDR, and the register content is sent to the NAND flash data lines D_OUT[7:0]. Multiple writes to SLC_ADDR may be used to increase the total width of the transmitted address. Writes to SLC_ADDR are stored in an internal FIFO that is also used for writes to SLC_CMD and SLC_STOP. This FIFO allows internal operations to continue while external NAND flash operations are completed.

Table 161. SLC NAND flash Address Register (SLC_ADDR - 0x2002 0004)

Bits	Description	Reset value
7:0	NAND flash read or write address.	-

9.6.3 SLC NAND flash Command register (SLC_CMD - 0x2002 0008)

SLC_CMD is an 8 bit wide register providing direct access to the NAND flash command register. The CLE output is driven after a write to SLC_CMD and the register content is sent to the NAND flash data lines. Writes to SLC_CMD are stored in an internal FIFO that is also used for writes to SLC_ADDR and SLC_STOP. This FIFO allows internal operations to continue while external NAND flash operations are completed.

Table 162. SLC NAND flash Command register (SLC_CMD - 0x2002 0008)

Bits	Description	Reset value
7:0	NAND flash command.	-

9.6.4 SLC NAND flash STOP register (SLC_STOP - 0x2002 000C)

A write to the SLC_STOP register causes the flash controller to suspend all command/address sequences. The function of bits in SLC_STOP are shown in Table 163. The stop command is cleared at the end of a DMA access when the Transfer Count TC = 0. Writes to SLC_STOP are stored in an internal FIFO that is also used for writes to SLC_ADDR and SLC_CMD. This FIFO allows internal operations to continue while external NAND flash operations are completed.

Table 163. SLC NAND flash STOP register (SLC_STOP - 0x2002 000C)

Bits	Description	Reset value
7:0	A write to this register causes the SLC flash controller to suspend all command/address sequences.	-

9.6.5 SLC NAND flash Control register (SLC_CTRL - 0x2002 0010)

The SLC_CTRL register provides basic controls for the NAND flash controller. These include resetting the interface, clearing ECC generation, and starting the DMA function. The function of bits in SLC_CTRL are shown in Table 164.

Table 164. SLC NAND flash Control register (SLC_CTRL - 0x2002 0010)

Bits	Description	Reset value
31:3	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
2	SW_RESET Writing 1 to this bit causes a reset of the SLC NAND flash controller	0
1	ECC_CLEAR Writing 1 to this bit clears ECC parity bits and reset the counter for ECC accumulation	0
0	DMA_START Writing 1 starts DMA data channel	0

9.6.6 SLC NAND flash Configuration register (SLC_CFG - 0x2002 0014)

The SLC_CFG register selects certain configuration options for the SLC NAND flash interface. The function of bits in SLC_CFG are shown in Table 165.

Table 165. SLC NAND flash Configuration register (SLC_CFG - 0x2002 0014)

Bits	Description	Reset value
31:6	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
5	CE_LOW Writing 1 forces CEn always low, otherwise CEn is low only when SLC is accessed	0
4	DMA_ECC 0: DMA ECC channel disabled 1: DMA ECC channel enabled	0
3	ECC_EN 0: ECC disabled 1:ECC enabled	0
2	DMA_BURST 0: burst disabled, use dmasreq0 signal only 1: burst enabled, data channel use DMA_BREQ signal	0
1	DMA_DIR 0 : DMA write to SLC 1 : DMA read from SLC	0
0	WIDTH: external device width select 0: 8-bit device 1: not used	0

9.6.7 SLC NAND flash Status register (SLC_STAT - 0x2002 0018)

The read-only SLC_STAT register indicates the status of the FIFOs and external flash device. The function of bits in SLC_STAT are shown in Table 166.

Table 166. SLC NAND flash Status register (SLC_STAT - 0x2002 0018)

Bits	Description	Reset value
31:3	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
2	DMA_ACTIVE: DMA_FIFO status 0: no data in the DMA_FIFO 1: the DMA_FIFO contains data	0
1	SLC_ACTIVE: SLC_FIFO status 0: no data in the SLC_FIFO 1: the SLC_FIFO contains data	0
0	READY: NAND flash device ready signal 0: device busy 1: device ready	Un-defined

9.6.8 SLC NAND flash Interrupt Status register (SLC_INT_STAT - 0x2002 001C)

The read-only SLC_INT_STAT register reflects the interrupt flags provided by the SLC NAND flash Interface. The function of bits in SLC_INT_STAT are shown in Table 167.

Table 167. SLC NAND flash Interrupt Status register (SLC_INT_STAT - 0x2002 001C)

Bits	Description	Reset value
31:2	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
1	INT_TC_STAT: Terminal Count interrupt status 0: Interrupt is not pending (after masking by SLC_IEN) 1: Interrupt is pending (after masking by SLC_IEN)	0
0	INT_RDY_STAT: Device ready interrupt status 0: Interrupt is not pending (after masking by SLC_IEN) 1: Interrupt is pending (after masking by SLC_IEN)	0

9.6.9 SLC NAND flash Interrupt Enable register (SLC_IEN - 0x2002 0020)

The write-only SLC_IEN register contains the interrupt enable flags for the SLC NAND flash Interface. The function of bits in SLC_IEN are shown in Table 168.

Table 168. SLC NAND flash Interrupt Enable register (SLC_IEN - 0x2002 0020)

Bits	Description	Reset value
31:2	Reserved, user software should not write ones to reserved bits.	-
1	INT_TC_EN 0: disable TC interrupt 1: enable interrupt when TC has reached 0	-
0	INT_RDY_EN 0: disable RDY interrupt 1: enable interrupt when RDY asserted	-

9.6.10 SLC NAND flash Interrupt Set Register (SLC_ISR - 0x2002 0024)

The write-only SLC_ISR register allows software to set the NAND flash interrupt flags. The function of bits in SLC_ISR are shown in Table 169.

Table 169. SLC NAND flash Interrupt Set Register (SLC_ISR - 0x2002 0024)

Bits	Description	Reset value
31:2	Reserved, user software should not write ones to reserved bits.	-
1	INT_TC_SET 0: writing 0 has no effect 1: writing 1 sets the TC interrupt	-
0	INT_RDY_SET 0: writing 0 has no effect 1: writing 1 sets the RDY interrupt	-

9.6.11 SLC NAND flash Interrupt Clear Register (SLC_ICR - 0x2002 0028)

The write-only SLC_ICR register allows software to clear the NAND flash interrupt flags. The function of bits in SLC_ICR are shown in Table 170.

Table 170. SLC NAND flash Interrupt Clear Register (SLC_ICR - 0x2002 0028)

Bits	Description	Reset value
31:2	Reserved, user software should not write ones to reserved bits.	-
1	INT_TC_CLR 0: writing 0 has no effect 1: writing 1 clears TC interrupt	-
0	INT_RDY_CLR 0: writing 0 has no effect 1: writing 1 clears RDY interrupt	-

9.6.12 SLC NAND flash Timing Arcs configuration register (SLC_TAC - 0x2002 002C)

The SLC_TAC register gives control of NAND flash bus timing. The function of bits in SLC_TAC are shown in Table 171.

Table 171. SLC NAND flash Timing Arcs configuration Register (SLC_TAC - 0x2002 002C)

Bits	Description	Reset value
31:28	W_RDY[3:0] The time before the signal RDY is tested in terms of 2 * clock cycles. After these 2*W_RDY[2:0] clocks, RDY is sampled by the interface. If RDY = 0, the bus sequencer stops. RDY is sampled on each clock until it equals 1, then the bus sequencer continues.	0
27:24	W_WIDTH[3:0] Write pulse width in clock cycles. Programmable from 1 to 16 clocks.	0
23:20	W_HOLD[3:0] Write hold time of ALE, CLE, CEn, and Data in clock cycles. Programmable from 1 to 16 clocks.	0
19:16	W_SETUP[3:0] Write setup time of ALE, CLE, CEn, and Data in clock cycles. Programmable from 1 to 16 clocks.	0
15:12	R_RDY[3:0] Time before the signal RDY is tested in terms of 2 * clock cycles. After these 2*R_RDY[2:0] cycles, RDY is sampled by the interface. If RDY = 0, the bus sequencer stops. RDY is sampled on each clock until it equals 1, then the bus sequencer continues.	0
11:8	R_WIDTH[3:0] Read pulse in clock cycles. Programmable from 1 to 16 clocks.	0
7:4	R_HOLD[3:0] Read hold time of ALE, CLE, and CEn in clock cycles. Programmable from 1 to 16 clocks.	0
3:0	R_SETUP[3:0] Read setup time of ALE, CLE, and CEn in clock cycles. Programmable from 1 to 16 clocks.	0

9.6.13 SLC NAND flash Transfer Count register (SLC_TC - 0x2002 0030)

The SLC_TC register indicates the number of DMA transfers remaining before DMA completion. SLC_TC is decremented at the completion of each DMA transfer, and must be re-initialized prior to any subsequent DMA transfer. The value written to the SLC_TC register must be a multiple of 4. The function of bits in SLC_TC are shown in Table 172.

Table 172. SLC NAND flash Transfer Count Register (SLC_TC - 0x2002 0030)

Bits	Description	Reset value
15:0	T_C Number of remaining bytes to be transferred to or from NAND flash memory during DMA.	0

9.6.14 SLC NAND flash Error Correction Code register (SLC_ECC - 0x2002 0034)

The read-only SLC_ECC register contains parity information that is calculated for NAND flash data. See the ECC section of this chapter for details. The function of bits in SLC_ECC are shown in Table 173.

Table 173. SLC NAND flash Error Correction Code Register (SLC_ECC - 0x2002 0034)

Bits	Description	Reset value
31:22	Reserved, user software should not write ones to reserved bits.	0
21:6	LP[15:0] - Line parity	0
5:0	CP[5:0] - Column parity	0

Note: ECC is computed on data blocks of 256-byte size, and they are calculated on data read from (or written to) the NAND flash memory when the DMA is enabled in burst mode. If support of larger pages is needed, multiple ECC generation must be done. ECC is automatically reset before the each packet, so parity bits would have to be saved immediately for later error detection and correction by software. This can be supported without CPU intervention by using DMA controller scatter/gather mode through linked list. In this mode, read access to SLC_ECC will be under control of DMA.

9.6.15 SLC NAND flash DMA Data Register (SLC_DMA_DATA - 0x2002 0038)

The SLC_DMA_DATA register is intended to be accessed by DMA only. All reads and writes to this register are 32 bits wide, each containing 4 data bytes. The lower 8 bits is the first byte transferred, etc. If needed, the endianess of the data can be altered by the DMA controller. The function of bits in SLC_DMA_DATA are shown in Table 174.

Table 174. SLC NAND flash DMA Data Register (SLC_DMA_DATA - 0x2002 0038)

Bits	Description	Reset value
31:24	Last byte transferred, corresponds to upper address.	-
23:16	Third byte transferred.	-
15:8	Second byte transferred.	-
7:0	First byte transferred, corresponds to lower address.	-

SLC_DMA_DATA is buffered in a 5 word FIFO with trigger level on 4 word. The 5th word allows the SLC flash controller to continue reading the flash without a break after each 4 bytes.

9.7 SLC NAND flash read/write sequences

Scatter/gather-linked lists in the DMA are used to initialize the type of transfer and to transfer the data and the ECC.

Access to NAND flash blocks larger than 256 bytes requires additional software handling of ECC information. In the following discussion, the term ECCM1 refers to an ECC value calculated by hardware on the first of two 256 byte data blocks, ECCM2 refers to an ECC value calculated by hardware on the second of two 256 byte data blocks, and ECCS refers to an ECC value calculated by software on the spare area.

9.7.1 Sequence to read a 528 byte page with scatter/gather DMA from SLC NAND flash

1. Set up the SLC NAND flash controller from the CPU (SLC_CFG = 0x1E, SLC_IEN = 0x1, SLC_TAC = is clock rate dependent). This is only needed for setting up the SLC NAND flash controller after reset.

2. Configure the DMA controller and channel setting from the CPU, it should use the scatter gather mode with a linked list.
3. Send a page Read command from the CPU by writing to SLC_CMD (SLC_CMD = 0x00). The SLC_CMD register is buffered, so the AHB bus does not wait on this write.
4. Send the read address command from the CPU by writing to SLC_ADDR (four address writes) (SLC_ADDR = Column_add, SLC_ADDR = Row_Addr_1, SLC_ADDR = Row_Addr_2, SLC_ADDR = Row_Addr_3). The SLC_ADDR register is buffered. The AHB will cause the CPU to wait if the buffer fills up.
5. Write the transfer count to SLC_TC from the CPU to trigger the read (SLC_TC = 0x210).
6. The SLC NAND flash controller samples RDY to wait for the NAND flash to be ready before reading successive data and requesting the DMA controller to transfer data and the calculated ECC to memory.
7. When the page read is done, the DMA controller sends an interrupt to the CPU.
8. The CPU computes the ECCS from the spare area data.
9. The CPU compares ECCM1, ECCM2 and ECCS with the ECC values read from spare area.

9.7.1.1 DMA functions

Table 175. Functions of the Scatter/Gather DMA during a 512 byte read of NAND flash

Function	Main area 512 byte				Spare area 16 byte
Data to transfer	Data block 1	ECCM1	Data block 2	ECCM2	Spare area data
Linked List element	LLI 1	LLI 2	LLI 3	LLI 4	LLI 5
Request signals used	BREQ×16	SREQ	BREQ×16	SREQ	BREQ×1

LLI 1 transfers Data block 1 to memory.

LLI 2 transfers the ECC for Data block 1 from SLC_ECC to memory.

LLI 3 transfers Data block 2 to memory.

LLI 4 transfers the ECC for Data block 2 from SLC_ECC to memory.

LLI 5 transfers the spare area to memory, and gives an interrupt to the CPU when finished.

9.7.2 Sequence to program a 528 byte page with scatter/gather DMA from SLC NAND flash

1. The CPU computes the ECCS for the spare area.
2. Set up the SLC NAND flash controller from the CPU.
3. Configure the DMA controller and channel setting from the CPU, it should use the scatter gather mode with a linked list.
4. Send a page write command from the CPU by writing to SLC_CMD.
5. Send the write address command from the CPU writing to SLC_ADDR (four address writes).

6. Write the transfer count to SLC_TC from the CPU to trigger the write.
7. Write the Program Command to SLC_CMD.
8. The SLC NAND flash controller samples RDY and waits until the NAND flash is ready. After this it asserts a request for DMA. The DMA controller will step in the linked list and another DMA_BREQ is asserted to request a DMA read and save ECCM1. After stepping in the linked list, the transfer will continue to Data block 2. The last request signal in this block is DMA_BREQ, after this the DMA controller will step in the linked list and another DMA_BREQ is asserted to request a DMA read and save ECCM2. After this the DMA controller will step in the linked list and complete the transfer of the spare area.
9. When the page is done, the SLC NAND flash controller sends the Program Command to the NAND flash.
10. The SLC NAND flash controller sends an interrupt when the program sequence in the NAND flash is completed (RDY goes high).
11. The CPU writes the Read Status Command to SLC_CMD and checks the result of the program sequence.

9.7.2.1 DMA functions

Table 176. Functions of the Scatter/Gather DMA during a 512 byte write to NAND flash

Function	Main area 512 byte				Spare area 16 byte	
Data to transfer	Data block 1	ECCM1	Data block 2	ECCM2	ECCS	Spare area data
Linked List element	LLI 1	LLI 2	LLI 3	LLI 4	LLI 5	
Request signals used	BREQ×16	SREQ	BREQ×16	SREQ	BREQ×1	

LLI 1 transfers Data block 1 to the SLC NAND flash controller.

LLI 2 transfers the ECC for Data block 1 from SLC_ECC to memory.

LLI 3 transfers Data block 2 to the SLC NAND flash controller.

LLI 4 transfers the ECC for Data block 2 from SLC_ECC to memory.

LLI 5 transfers the rest of the spare area to memory.

9.8 Error checking and correction

ECC generation of the main area is done by hardware and is based on data blocks of 256 bytes. To be able to detect and correct one bit error in a data block of 256 bytes, 6 Column Parity bits and 8 Line Parity bits are needed.

During main area writes, the ECC hardware calculates the Line Parity (LP0-LP7) and column Parity (CP0-CP5) on the data stream between the FIFO and the flash. The ECC for each 256 byte data block must be read back and stored in the right place in the spare data structure to be programmed later in to the spare area.

During main area reads, new ECC Line Parity (LP0'-LP7') and column Parity (CP0'-CP5') are generated for each data block of 256 bytes and stored in memory. Software must check these against the ECC located in the spare area for the page currently read. If an error is detected, software must handle data correction or other response.

ECC generation for data in the spare area is not done automatically.

The whole spare area must first be built in memory before it is programmed to the NAND flash. Software generates the ECC for the spare area and stores the ECC in the memory at the correct location in the spare data structure. When the Data for the spare area is complete (including ECC for the main and spare area) in memory, data is programmed into the NAND flash with ECC generation off.

During reads of the spare area, data is read out and software computes the ECC.

In the following discussion, the term ECCM1 refers to an ECC value calculated by hardware on the first of two 256 byte data blocks, ECCM2 refers to an ECC value calculated by hardware on the second of two 256 byte data blocks, and ECCS refers to an ECC value calculated by software on the spare area.

9.8.1 How an ECC Code is generated on a 256 byte data block

Figure 26 shows an overview of the ECC generation hardware. Figure 27 gives a graphical view of how Line and Column Parity are calculated.

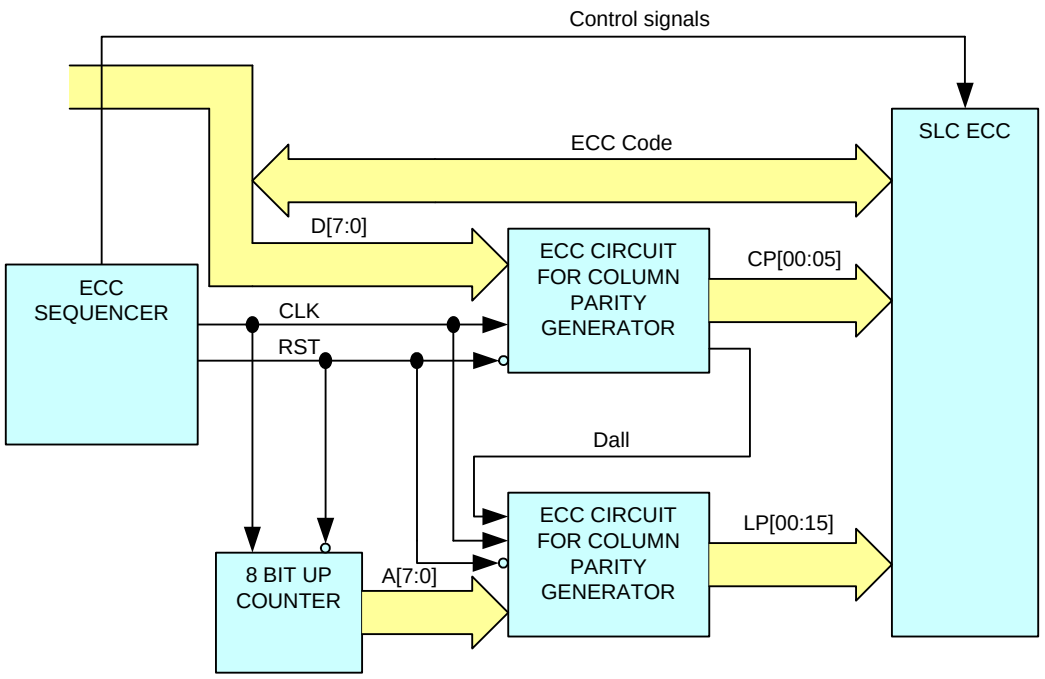


Fig 26. Block diagram of ECC generation

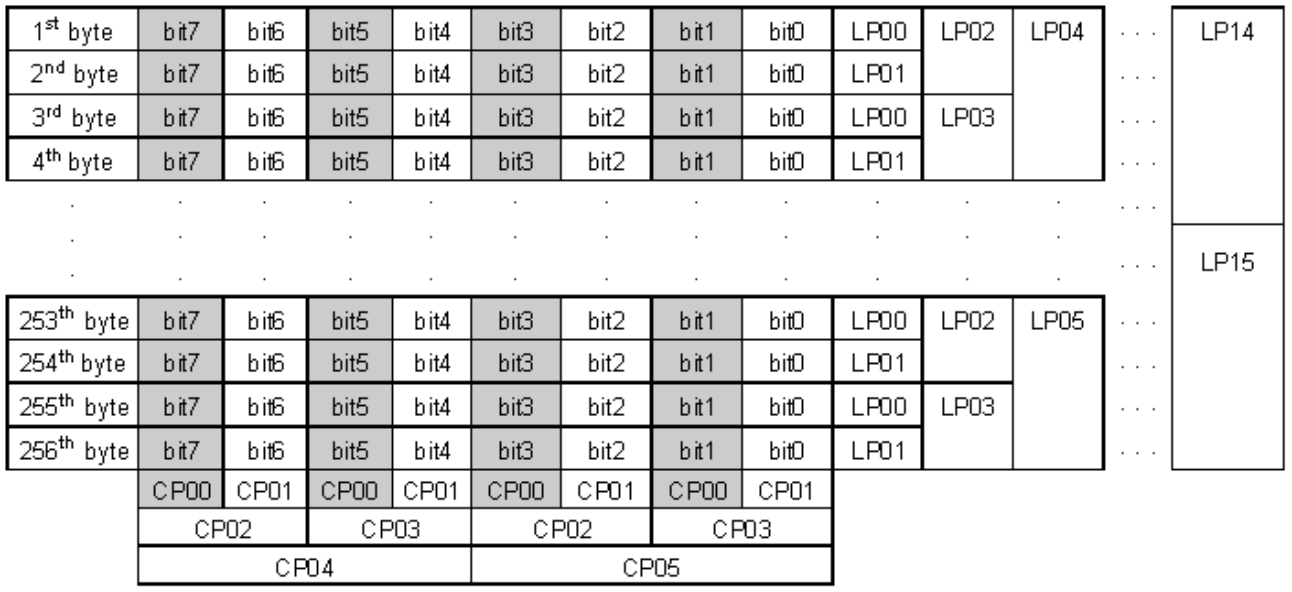


Fig 27. Graphical view of column and line parity

Column Parity is calculated over the entire data block as each data byte is processed. Selected bits of each data byte are added to the previous value of each Column Parity bit.

The equations for the Column Parity bits are:

CP00 = bit7 EXOR bit5 EXOR bit3 EXOR bit1 EXOR CP00

CP01 = bit6 EXOR bit4 EXOR bit2 EXOR bit0 EXOR CP01

CP02 = bit7 EXOR bit6 EXOR bit3 EXOR bit2 EXOR CP02

CP03 = bit5 EXOR bit4 EXOR bit1 EXOR bit0 EXOR CP03

CP04 = bit7 EXOR bit6 EXOR bit5 EXOR bit4 EXOR CP04

CP05 = bit3 EXOR bit2 EXOR bit1 EXOR bit0 EXOR CP05

Line parity is calculated over the entire data block as each data byte is processed. If the sum of the bits in one byte is 0, the line parity does not change when it is recalculated. The sum of the bits in 1 byte of data is:

Dall = bit7 EXOR bit6 EXOR bit5 EXOR bit4 EXOR bit3 EXOR bit2 EXOR bit1 EXOR bit0

Sixteen line parity bits (LP15-LP00) are computed from 256 bytes of data. An 8 bit counter counts data bytes, bits of this counter are used as a mask for Line Parity bits. The counter increments by 1 for each new byte of data. Line Parity is computed by initializing all line parity bits to zero, reading in each byte, computing the byte sum (Dall), and adding Dall to the line parity bits when they are enabled by the appropriate counter bits.

The equations for the Line Parity bits are:

LP00 = LP00 EXOR (Dall AND $\overline{\text{Counter\_bit0}}$)

LP01 = LP01 EXOR (Dall AND Counter_bit0)

LP02 = LP02 EXOR (Dall AND $\overline{\text{Counter\_bit1}}$)

LP03 = LP03 EXOR (Dall AND Counter_bit1)

LP04 = LP04 EXOR (Dall AND $\overline{\text{Counter\_bit2}}$)

LP05 = LP05 EXOR (Dall AND Counter_bit2)

LP06 = LP06 EXOR (Dall AND $\overline{\text{Counter\_bit3}}$)

LP07 = LP07 EXOR (Dall AND Counter_bit3)

LP08 = LP08 EXOR (Dall AND $\overline{\text{Counter\_bit4}}$)

LP09 = LP09 EXOR (Dall AND Counter_bit4)

LP10 = LP10 EXOR (Dall AND $\overline{\text{Counter\_bit5}}$)

LP11 = LP11 EXOR (Dall AND Counter_bit5)

LP12 = LP12 EXOR (Dall AND $\overline{\text{Counter\_bit6}}$)

LP13 = LP13 EXOR (Dall AND Counter_bit6)

LP14 = LP14 EXOR (Dall AND $\overline{\text{Counter\_bit7}}$)

LP15 = LP15 EXOR (Dall AND Counter_bit7)

9.8.1.1 How to detect errors

The combination of Column Parity and Line Parity bits allows detection of two or more erroneous data bits and correction of a single erroneous data bit. Table 177 shows the cases that can occur when calculated ECC data is compared to stored ECC data. Following the table are descriptions of each case.

Table 177. Error detection cases

LP 15	LP 14	LP 13	LP 12	LP 11	LP 10	LP 09	LP 08	LP 07	LP 06	LP 05	LP 04	LP 03	LP 02	LP 01	LP 00	CP 05	CP 04	CP 03	CP 02	CP 01	CP 00	Code stored in flash
																						Comparison (EXOR)
LP 15’	LP 14’	LP 13’	LP 12’	LP 11’	LP 10’	LP 09’	LP 08’	LP 07’	LP 06’	LP 05’	LP 04’	LP 03’	LP 02’	LP 01’	LP 00’	CP 05’	CP 04’	CP 03’	CP 02’	CP 01’	CP 00’	Code generated at read
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	No Error
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	Correctable
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	Uncorrectable
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Code Error

No error

Since there is no difference between the code stored in the flash and the one generated after the read, it is assumed that there is no error in this case.

Correctable error

Since all parity bit pairs (CP00 and CP01),.....,(LP014 and LP15) have one error and one match in them as the result of the comparison between the code stored in flash and the one generated after the read, this case is considered to be a correctable error.

Uncorrectable error

In this case, both CP00 and CP01 are in error as the results of the comparison between the code stored in flash and the one generated after the read. This represents a multiple bit error, and is therefore uncorrectable.

ECC code area error

When only one bit (LP13) is erroneous (the result of the comparison between the code stored in flash and the one generated after the read), it is assumed that the error occurred in the ECC area and not in the data area. This is because a single erroneous data bit should cause a difference in half of the Line Parity bits (by changing Dall, which affects half of the Line Parity bits based on the current counter value), and half of the Column Parity bits (based on the equations for the Column Parity bits, which each include half of the data bits).

9.8.1.2 Finding the location of correctable errors

The error location can be found by XORing the ECC parity bits stored in the flash with ECC bits calculated from the data read out of the flash.

The error location is assembled from XORing the following stored and computed line parity bits:

(LP15,LP13,LP11,LP09,LP07,LP05,LP03,LP01) - this gives the byte address.

($\overline{CP05}$,$\overline{CP03}$,$\overline{CP01}$) - this gives the bit number.

9.8.2 How to generate ECC on pages greater than 256 bytes

The SLC NAND flash controller is able to support single-level NAND flash with pages of (512 +16) byte and (2 K + 64) byte. This is accomplished by splitting those pages up into 256 byte blocks, generating ECC on each block separately, and storing the result in to the spare areas.

9.8.2.1 Example for (512 + 16) byte pages

Table 178. ECC generation for 512 + 16 byte pages

Main area 512 byte		Spare area 16 byte
first 256 byte block	second 256 byte block	
ECCM1 3 byte (hardware generated)	ECCM2 3 byte (hardware generated)	ECCS (generated by software)

The SLC NAND flash controller has only one ECC register: The ECCM1 for the first data block in the main area must be read out and stored in memory before the next data block of 256 byte is transferred, and the ECCM2 for this block must be read out before the spare area is transferred.

On reads, the software compares the ECCM1, ECCM2 and ECCS with the ECC from the spare area of the page read.

Table 179. ECC checking for 512 + 16 byte pages

ECC on from data read	Operation	ECC from flash	Result
ECCM1	XOR	ECCM1’	If not = 0, an error has occurred.
ECCM2	XOR	ECCM2’	If not = 0, an error has occurred.
ECCS	XOR	ECCS’	If not = 0, an error has occurred.

10.1 Introduction

The LCD controller provides all of the necessary control signals to interface directly to a variety of color and monochrome LCD panels.

Remark: The LCD controller is available on LPC3230 and LPC3250 parts only.

10.2 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.

10.3 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

10.4 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 75KB 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.

10.5 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.

10.6 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.

10.7 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.

10.8 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.

10.9 Pin descriptions

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 180, only the LCD related portion of the pin name is shown.

Remark: To enable the LCD controller, see Table 198 and Table 203.

Table 180. LCD controller pins

Pin name	Type	Function
LCDPWR	output	LCD panel power enable.
LCDDCLK	output	LCD panel clock.
LCDENA/LCDM	output	STN AC bias drive or TFT data enable output.
LCDFP	output	Frame pulse (STN). Vertical synchronization pulse (TFT)
LCDLE	output	Line end signal
LCDLP	output	Line synchronization pulse (STN). Horizontal synchronization pulse (TFT)
LCDVD[23:0]	output	LCD panel data. Bits used depend on the panel configuration.
LCDCLKIN	input	Optional clock input.

10.9.1 Signal usage

The signals that are used for various display types are identified in the following sections. To configure the pin groupings see Table 198.

10.9.1.1 Signals used for single panel STN displays

The signals used for single panel STN displays are shown in Table 181. UD refers to upper panel data.

Table 181. Pins used for single panel STN displays

Pin name	4-bit Monochrome (10 pins)	8-bit Monochrome (14 pins)	Color (14 pins)
LCDPWR	Y	Y	Y
LCDDCLK	Y	Y	Y
LCDENAB/ LCDM	Y	Y	Y
LCDFP	Y	Y	Y
LCDLE	Y	Y	Y
LCDLP	Y	Y	Y
LCDVD[3:0]	UD[3:0]	UD[3:0]	UD[3:0]
LCDVD[7:4]	-	UD[7:4]	UD[7:4]
LCDVD[23:8]	-	-	-

10.9.1.2 Signals used for dual panel STN displays

The signals used for dual panel STN displays are shown in Table 182. UD refers to upper panel data, and LD refers to lower panel data.

Table 182. Pins used for dual panel STN displays

Pin name	4-bit Monochrome (14 pins)	8-bit Monochrome (22 pins)	Color (22 pins)
LCDPWR	Y	Y	Y
LCDDCLK	Y	Y	Y
LCDENAB/ LCDM	Y	Y	Y
LCDFP	Y	Y	Y
LCDLE	Y	Y	Y
LCDLP	Y	Y	Y
LCDVD[3:0]	UD[3:0]	UD[3:0]	UD[3:0]
LCDVD[7:4]	-	UD[7:4]	UD[7:4]
LCDVD[11:8]	LD[3:0]	LD[3:0]	LD[3:0]
LCDVD[15:12]	-	LD[7:4]	LD[7:4]
LCDVD[23:16]	-	-	-

10.9.1.3 Signals used for TFT displays

The signals used for TFT displays are shown in Table 183.

Table 183. 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)
LCDPWR	Y	Y	Y	Y
LCDDCLK	Y	Y	Y	Y
LCDENAB/ LCDM	Y	Y	Y	Y
LCDFP	Y	Y	Y	Y
LCDLE	Y	Y	Y	Y
LCDLP	Y	Y	Y	Y
LCDVD[1:0]	-	-	-	RED[1:0]
LCDVD[2]	-	-	Intensity	RED[2]
LCDVD[3]	-	RED[0]	RED[0]	RED[3]
LCDVD[7:4]	RED[3:0]	RED[4:1]	RED[4:1]	RED[7:4]
LCDVD[9:8]	-	-	-	GREEN[1:0]
LCDVD[10]	-	GREEN[0]	Intensity	GREEN[2]
LCDVD[11]	-	GREEN[1]	GREEN[0]	GREEN[3]
LCDVD[15:12]	GREEN[3:0]	GREEN[5:2]	GREEN[4:1]	GREEN[7:4]
LCDVD[17:16]	-	-	-	BLUE[1:0]
LCDVD[18]	-	-	Intensity	BLUE[2]
LCDVD[19]	-	BLUE[0]	BLUE[0]	BLUE[3]
LCDVD[23:20]	BLUE[3:0]	BLUE[4:1]	BLUE[4:1]	BLUE[7:4]

10.10 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 28 shows a simplified block diagram of the LCD controller.

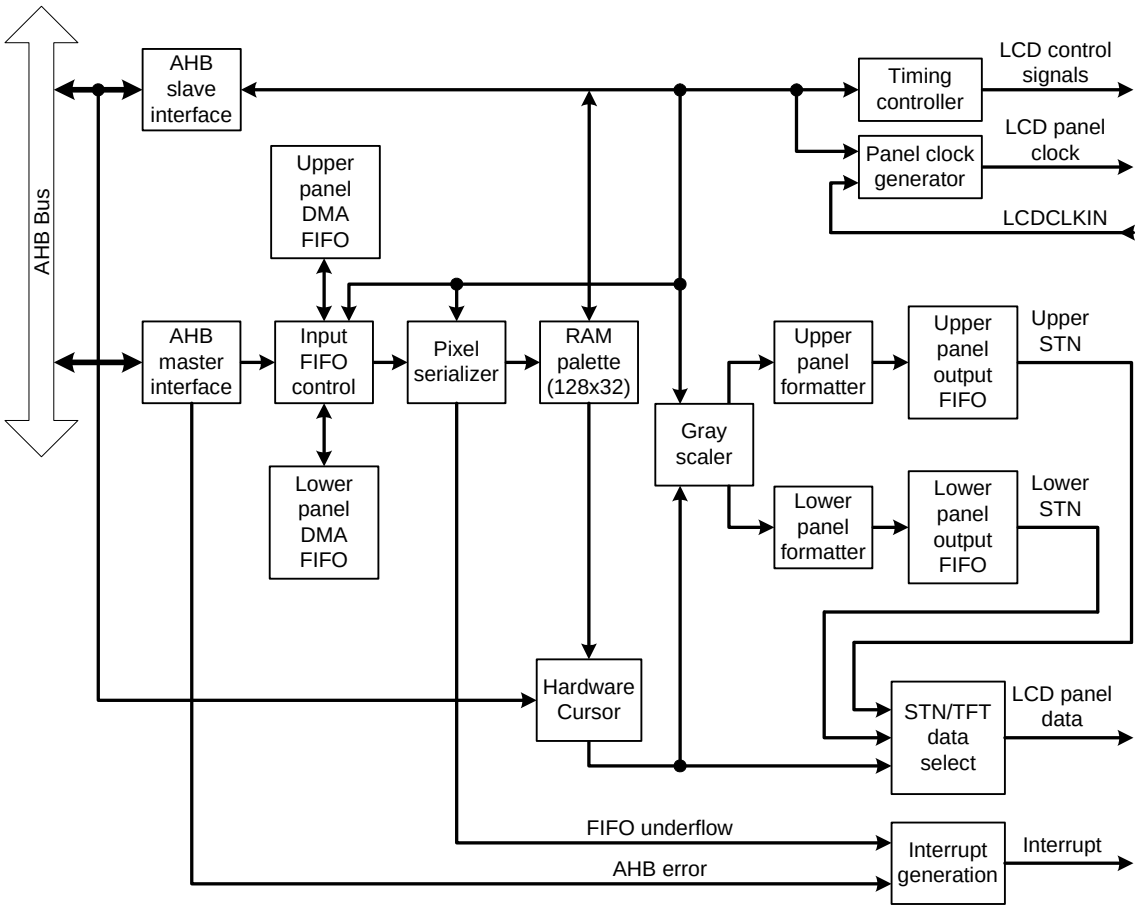


Fig 28. LCD controller block diagram

10.10.1 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.

10.10.1.1 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.

10.10.1.2 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 on-chip SRAM, 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 1KB 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.

10.10.2 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).

10.10.3 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 184 through Table 186 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 184. FIFO bits for Little-endian Byte, Little-endian Pixel order

FIFO bit	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
31	p31	p15	p7	p3	p1	
30	p30					
29	p29	p14				
28	p28					
27	p27	p13	p6			
26	p26					
25	p25	p12				
24	p24					
23	p23	p11	p5	p2		p0
22	p22					
21	p21	p10				
20	p20					
19	p19	p9	p4			
18	p18					
17	p17	p8				
16	p16					
15	p15	p7	p3	p1	p0	
14	p14					
13	p13	p6				
12	p12					
11	p11	p5	p2			
10	p10					
9	p9	p4				
8	p8					
7	p7	p3	p1	p0		
6	p6					
5	p5	p2				
4	p4					
3	p3	p1	p0			
2	p2					
1	p1	p0				
0	p0					

Table 185. FIFO bits for Big-endian Byte, Big-endian Pixel order

FIFO bit	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
31	p0	p0	p0	p0	p0	
30	p1					
29	p2	p1				
28	p3					
27	p4	p2	p1			
26	p5					
25	p6	p3				
24	p7					
23	p8	p4	p2	p1		p0
22	p9					
21	p10	p5				
20	p11					
19	p12	p6	p3			
18	p13					
17	p14	p7				
16	p15					
15	p16	p8	p4	p2	p1	
14	p17					
13	p18	p9				
12	p19					
11	p20	p10	p5			
10	p21					
9	p22	p11				
8	p23					
7	p24	p12	p6	p3		
6	p25					
5	p26	p13				
4	p27					
3	p28	p14	p7			
2	p29					
1	p30	p15				
0	p31					

Table 186. FIFO bits for Little-endian Byte, Big-endian Pixel order

FIFO bit	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
31	p24	p12	p6	p3	p1	
30	p25					
29	p26	p13				
28	p27					
27	p28	p14	p7			
26	p29					
25	p30	p15				
24	p31					
23	p16	p8	p4	p2		p0
22	p17					
21	p18	p9				
20	p19					
19	p20	p10	p5			
18	p21					
17	p22	p11				
16	p23					
15	p8	p4	p2	p1	p0	
14	p9					
13	p10	p5				
12	p11					
11	p12	p6	p3			
10	p13					
9	p14	p7				
8	p15					
7	p0	p0	p0	p0		
6	p1					
5	p2	p1				
4	p3					
3	p4	p2	p1			
2	p5					
1	p6	p3				
0	p7					

Table 187 shows the structure of the data in each DMA FIFO word in RGB mode.

Table 187. 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)
31	-	p1 intensity bit	p1, Blue 4	-
30	-	p1, Blue 4	p1, Blue 3	-
29	-	p1, Blue 3	p1, Blue 2	-
28	-	p1, Blue 2	p1, Blue 1	-
27	-	p1, Blue 1	p1, Blue 0	p1, Blue 3
26	-	p1, Blue 0	p1, Green 5	p1, Blue 2
25	-	p1, Green 4	p1, Green 4	p1, Blue 1
24	-	p1, Green 3	p1, Green 3	p1, Blue 0
23	p0, Blue 7	p1, Green 2	p1, Green 2	p1, Green 3
22	p0, Blue 6	p1, Green 1	p1, Green 1	p1, Green 2
21	p0, Blue 5	p1, Green 0	p1, Green 0	p1, Green 1
20	p0, Blue 4	p1, Red 4	p1, Red 4	p1, Green 0
19	p0, Blue 3	p1, Red 3	p1, Red 3	p1, Red 3
18	p0, Blue 2	p1, Red 2	p1, Red 2	p1, Red 2
17	p0, Blue 1	p1, Red 1	p1, Red 1	p1, Red 1
16	p0, Blue 0	p1, Red 0	p1, Red 0	p1, Red 0
15	p0, Green 7	p0 intensity bit	p0, Blue 4	-
14	p0, Green 6	p0, Blue 4	p0, Blue 3	-
13	p0, Green 5	p0, Blue 3	p0, Blue 2	-
12	p0, Green 4	p0, Blue 2	p0, Blue 1	-
11	p0, Green 3	p0, Blue 1	p0, Blue 0	p0, Blue 3
10	p0, Green 2	p0, Blue 0	p0, Green 5	p0, Blue 2
9	p0, Green 1	p0, Green 4	p0, Green 4	p0, Blue 1
8	p0, Green 0	p0, Green 3	p0, Green 3	p0, Blue 0
7	p0, Red 7	p0, Green 2	p0, Green 2	p0, Green 3
6	p0, Red 6	p0, Green 1	p0, Green 1	p0, Green 2
5	p0, Red 5	p0, Green 0	p0, Green 0	p0, Green 1
4	p0, Red 4	p0, Red 4	p0, Red 4	p0, Green 0
3	p0, Red 3	p0, Red 3	p0, Red 3	p0, Red 3
2	p0, Red 2	p0, Red 2	p0, Red 2	p0, Red 2
1	p0, Red 1	p0, Red 1	p0, Red 1	p0, Red 1
0	p0, Red 0	p0, Red 0	p0, Red 0	p0, Red 0

10.10.4 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 188 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 188. Palette data storage for TFT modes.

Bit(s)	Name (RGB format)	Description (RGB format)	Name (BGR format)	Description (BGR format)
31	I	Intensity / unused	I	Intensity / unused
30:26	B[4:0]	Blue palette data	R[4:0]	Red palette data
25:21	G[4:0]	Green palette data	G[4:0]	Green palette data
20:16	R[4:0]	Red palette data	B[4:0]	Blue palette data
15	I	Intensity / unused	I	Intensity / unused
14:10	B[4:0]	Blue palette data	R[4:0]	Red palette data
9:5	G[4:0]	Green palette data	G[4:0]	Green palette data
4:0	R[4:0]	Red palette data	B[4:0]	Blue palette data

The red and blue pixel data can be swapped to support BGR data format using a control register bit (bit 8 = BGR). See the LCD_CTRL register description for more information. Table 189 shows the bit representation of the palette data for the STN color modes.

Table 189. Palette data storage for STN color modes.

Bit(s)	Name (RGB format)	Description (RGB format)	Name (BGR format)	Description (BGR format)
31	-	Unused	-	Unused
30:27	B[3:0]	Blue palette data	R[3:0]	Red palette data
26	-	Unused	-	Unused
25:22	G[3:0]	Green palette data	G[3:0]	Green palette data
21	-	Unused	-	Unused
20:17	R[3:0]	Red palette data	B[3:0]	Blue palette data
16	-	Unused	-	Unused
15	I	Unused	I	Unused
14:11	B[4:1]	Blue palette data	R[4:1]	Red palette data
10	B[0]	Unused	R[0]	Unused
9:6	G[4:1]	Green palette data	G[4:1]	Green palette data

Table 189. Palette data storage for STN color modes.

Bit(s)	Name (RGB format)	Description (RGB format)	Name (BGR format)	Description (BGR format)
5	G[0]	Unused	G[0]	Unused
4:1	R[4:1]	Red palette data	B[4:1]	Blue palette data
0	R[0]	Unused	B[0]	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 190 shows the bit representation of the palette data for the STN monochrome mode.

Table 190. Palette data storage for STN monochrome mode.

Bit(s)	Name	Description
31	-	Unused
30:27	-	Unused
26	-	Unused
25:22	-	Unused
21	-	Unused
20:17	Y[3:0]	Intensity data
16	-	Unused
15	-	Unused
14:11	-	Unused
10	-	Unused
9:6	-	Unused
5	-	Unused
4:1	Y[3:0]	Intensity data
0	-	Unused

10.10.5 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.

Remark: The cursor Image RAM is does not support burst reads on the CPU side. It does support word reads, word writes and burst writes on the CPU side. Read access is performed by hardware on the LCD side of the Image RAM.

10.10.5.1 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.

10.10.5.2 Cursor sizes

Two cursor sizes are supported, as shown in Table 191.

Table 191. 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

10.10.5.3 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 29 shows how each pixel coordinate is assumed to be the top left corner of the pixel.

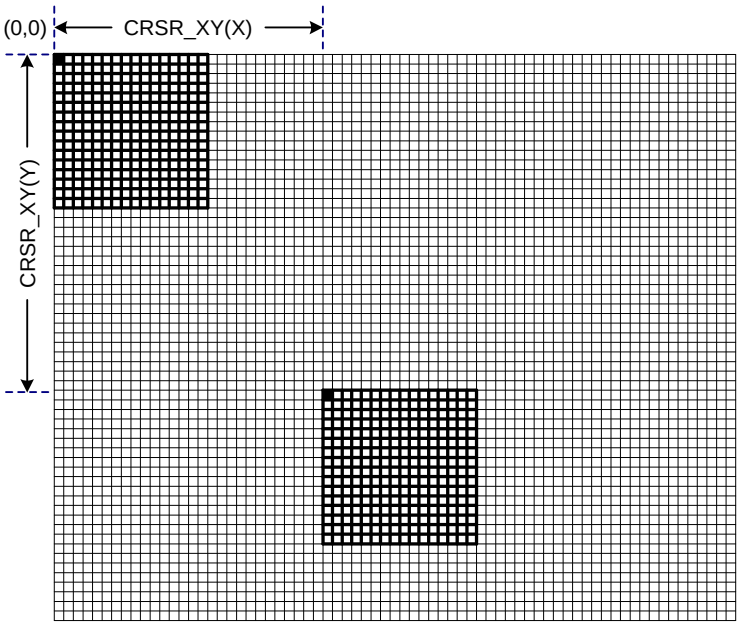


Fig 29. Cursor movement

10.10.5.4 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.

10.10.5.5 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 30). 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.

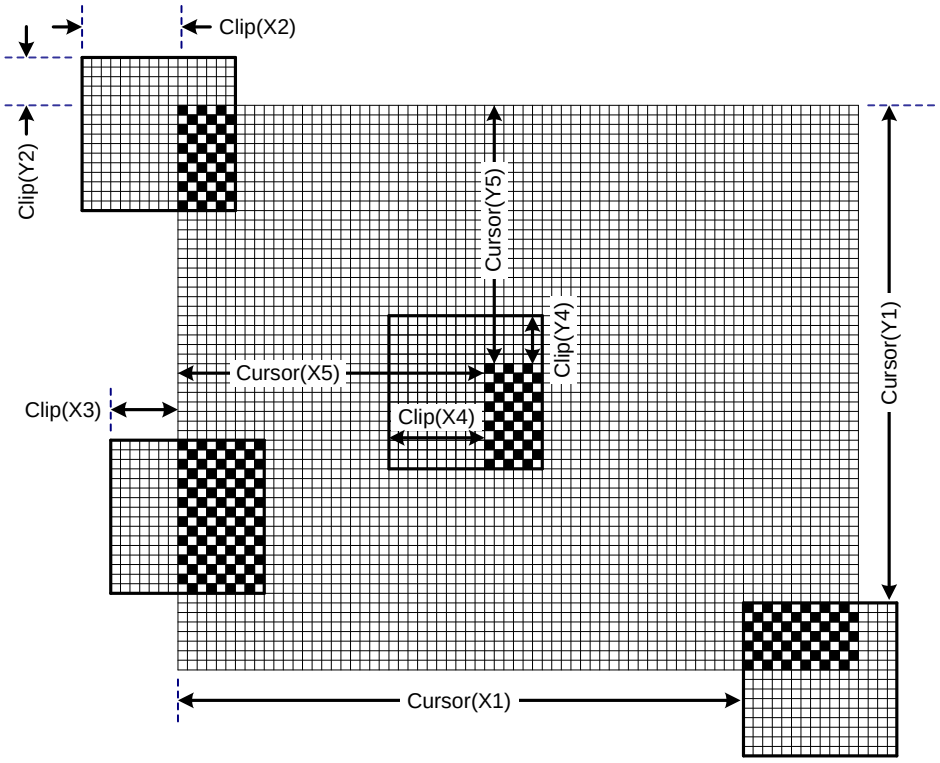


Fig 30. Cursor clipping

10.10.5.6 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 cursor 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 31.

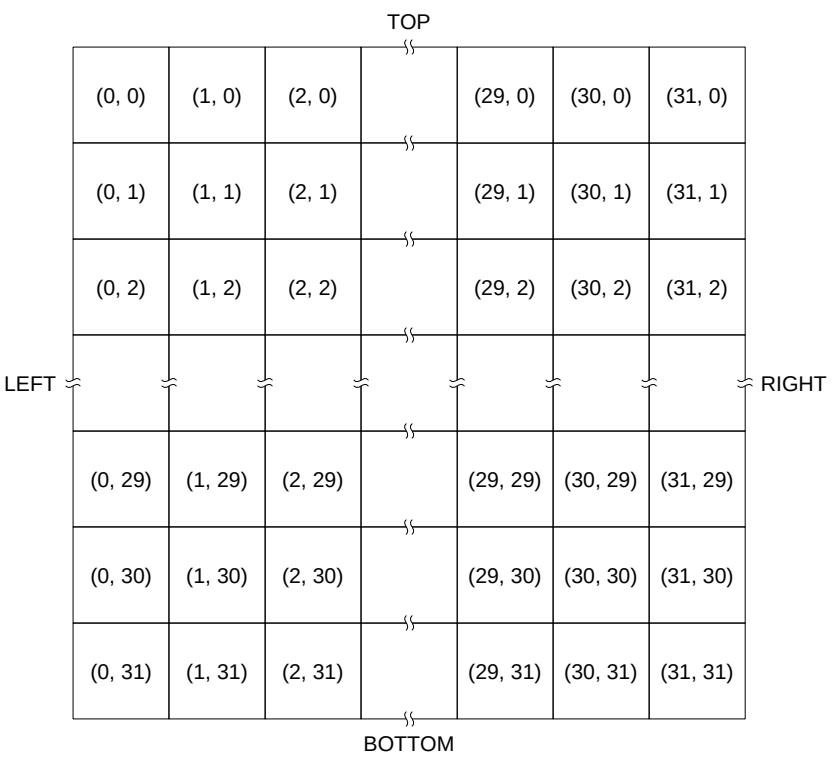


Fig 31. Cursor image format

32 by 32 pixel format

Four cursors are held in memory, each with the same pixel format. Table 192 lists the base addresses for the four cursors.

Table 192. Addresses for 32 x 32 cursors

Address	Description
0x3104 0800	Cursor 0 start address.
0x3104 0900	Cursor 1 start address.
0x3104 0A00	Cursor 2 start address.
0x3104 0B00	Cursor 3 start address.

Table 193 shows the buffer to pixel mapping for Cursor 0.

Table 193. 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
31:30	(12, 0)	(28, 0)	(12, y)	(28, y)	(12, 31)	(28,31)
29:28	(13, 0)	(29, 0)	(13, y)	(29, y)	(13, 31)	(29, 31)
27:26	(14, 0)	(30, 0)	(14, y)	(30, y)	(14, 31)	(30, 31)
25:24	(15, 0)	(31, 0)	(15, y)	(31, y)	(15, 31)	(31, 31)

Table 193. 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
23:22	(8, 0)	(24, 0)	(8, y)	(24, y)	(8, 31)	(24, 31)
21:20	(9, 0)	(25, 0)	(9, y)	(25, y)	(9, 31)	(25, 31)
19:18	(10, 0)	(26, 0)	(10, y)	(26, y)	(10, 31)	(26, 31)
17:16	(11, 0)	(27, 0)	(11, y)	(27, y)	(11, 31)	(27, 31)
15:14	(4, 0)	(20, 0)	(4, y)	(20, y)	(4, 31)	(20, 31)
13:12	(5, 0)	(21, 0)	(5, y)	(21, y)	(5, 31)	(21, 31)
11:10	(6, 0)	(22, 0)	(6, y)	(22, y)	(6, 31)	(22, 31)
9:8	(7, 0)	(23, 0)	(7, y)	(23, y)	(7, 31)	(23, 31)
7:6	(0, 0)	(16, 0)	(0, y)	(16, y)	(0, 31)	(16, 31)
5:4	(1, 0)	(17, 0)	(1, y)	(17, y)	(1, 31)	(17, 31)
3:2	(2, 0)	(18, 0)	(2, y)	(18, y)	(2, 31)	(18, 31)
1:0	(3, 0)	(19, 0)	(3, y)	(19, y)	(3, 31)	(19, 31)

64 by 64 pixel format

Only one cursor fits in the memory space in 64 x 64 mode. Table 194 shows the 64 x 64 cursor format.

Table 194. 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
31:30	(12, 0)	(28, 0)	(44, 0)	(60, 0)	(12, y)	(28, y)	(44, y)	(60, y)	(60, 63)
29:28	(13, 0)	(29, 0)	(45, 0)	(61, 0)	(13, y)	(29, y)	(45, y)	(61, y)	(61, 63)
27:26	(14, 0)	(30, 0)	(46, 0)	(62, 0)	(14, y)	(30, y)	(46, y)	(62, y)	(62, 63)
25:24	(15, 0)	(31, 0)	(47, 0)	(63, 0)	(15, y)	(31, y)	(47, y)	(63, y)	(63, 63)
23:22	(8, 0)	(24, 0)	(40, 0)	(56, 0)	(8, y)	(24, y)	(40, y)	(56, y)	(56, 63)
21:20	(9, 0)	(25, 0)	(41, 0)	(57, 0)	(9, y)	(25, y)	(41, y)	(57, y)	(57, 63)
19:18	(10, 0)	(26, 0)	(42, 0)	(58, 0)	(10, y)	(26, y)	(42, y)	(58, y)	(58, 63)
17:16	(11, 0)	(27, 0)	(43, 0)	(59, 0)	(11, y)	(27, y)	(43, y)	(59, y)	(59, 63)
15:14	(4, 0)	(20, 0)	(36, 0)	(52, 0)	(4, y)	(20, y)	(36, y)	(52, y)	(52, 63)
13:12	(5, 0)	(21, 0)	(37, 0)	(53, 0)	(5, y)	(21, y)	(37, y)	(53, y)	(53, 63)
11:10	(6, 0)	(22, 0)	(38, 0)	(54, 0)	(6, y)	(22, y)	(38, y)	(54, y)	(54, 63)
9:8	(7, 0)	(23, 0)	(39, 0)	(55, 0)	(7, y)	(23, y)	(39, y)	(55, y)	(55, 63)
7:6	(0, 0)	(16, 0)	(32, 0)	(48, 0)	(0, y)	(16, y)	(32, y)	(48, y)	(48, 63)
5:4	(1, 0)	(17, 0)	(33, 0)	(49, 0)	(1, y)	(17, y)	(33, y)	(49, y)	(49, 63)
3:2	(2, 0)	(18, 0)	(34, 0)	(50, 0)	(2, y)	(18, y)	(34, y)	(50, y)	(50, 63)
1:0	(3, 0)	(19, 0)	(35, 0)	(51, 0)	(3, y)	(19, y)	(35, y)	(51, y)	(51, 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 195 shows the pixel encoding bit assignments.

Table 195. 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.

10.10.6 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.

10.10.7 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 196 shows a color display driven with 2 2/3 pixels worth of data in a repeating sequence.

Table 196. 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.

10.10.8 Panel clock generator

The output of the panel clock generator block is the panel clock, pin LCDDCLK. The panel clock can be based on either the peripheral clock for the LCD block or the external clock input for the LCD, pin LCDCLKIN. 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 HCLK or the LCDCLKIN pin.

10.10.9 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.

10.10.10 STN and TFT data select

Support is provided for passive Super Twisted Nematic (STN) and active Thin Film Transistor (TFT) LCD display types:

10.10.10.1 STN displays

STN display panels require algorithmic pixel pattern generation to provide pseudo gray scaling on monochrome displays, or color creation on color displays.

10.10.10.2 TFT displays

TFT display panels require the digital color value of each pixel to be applied to the display data inputs.

10.10.11 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.

10.10.11.1 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.

10.10.11.2 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.

10.10.11.2.1 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

10.10.11.2.2 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.

10.10.12 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:

- LCDLP
- LCDDCLK

- LCDFP
- LCDENAB/ LCDM
- LCDVD[23:0]
- LCDLE

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:

- LCDLP
- LCDDCLK
- LCDFP
- LCDENAB/ LCDM
- LCDLE

The LCDV[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 LCDPWR signal high and enables the LCDV[23:0] signals into their active states. The LCDPWR 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 32 shows the power-up and power-down sequences.

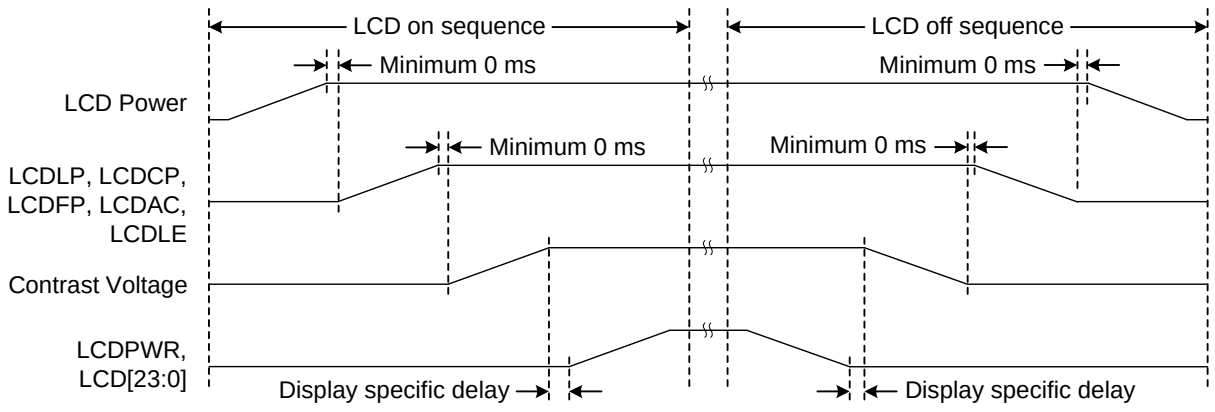


Fig 32. Power up and power down sequences

10.11 Register description

Table 197 shows the registers associated with the LCD controller and a summary of their functions. Following the table are details for each register.

Table 197. LCD controller registers

Address	Name	Description	Reset value	Access
0x4000 4054	LCD_CFG	LCD Configuration and clocking control	0x0	R/W
0x3104 0000	LCD_TIMH	Horizontal Timing Control register	0x0	R/W
0x3104 0004	LCD_TIMV	Vertical Timing Control register	0x0	R/W
0x3104 0008	LCD_POL	Clock and Signal Polarity Control register	0x0	R/W
0x3104 000C	LCD_LE	Line End Control register	0x0	R/W
0x3104 0010	LCD_UPBASE	Upper Panel Frame Base Address register	0x0	R/W
0x3104 0014	LCD_LPBASE	Lower Panel Frame Base Address register	0x0	R/W
0x3104 0018	LCD_CTRL	LCD Control register	0x0	R/W
0x3104 001C	LCD_INTMSK	Interrupt Mask register	0x0	R/W
0x3104 0020	LCD_INTRAW	Raw Interrupt Status register	0x0	RO
0x3104 0024	LCD_INTSTAT	Masked Interrupt Status register	0x0	RO
0x3104 0028	LCD_INTCLR	Interrupt Clear register	0x0	WO
0x3104 002C	LCD_UPCURR	Upper Panel Current Address Value register	0x0	RO
0x3104 0030	LCD_LPCURR	Lower Panel Current Address Value register	0x0	RO
0x3104 0200 - 0x3104 03FC	LCD_PAL	256x16-bit Color Palette registers	0x0	R/W
0x3104 0800 - 0x3104 0BFC	CRSR_IMG	Cursor Image registers	0x0	R/W
0x3104 0C00	CRSR_CTRL	Cursor Control register	0x0	R/W
0x3104 0C04	CRSR_CFG	Cursor Configuration register	0x0	R/W
0x3104 0C08	CRSR_PAL0	Cursor Palette register 0	0x0	R/W
0x3104 0C0C	CRSR_PAL1	Cursor Palette register 1	0x0	R/W
0x3104 0C10	CRSR_XY	Cursor XY Position register	0x0	R/W
0x3104 0C14	CRSR_CLIP	Cursor Clip Position register	0x0	R/W
0x3104 0C20	CRSR_INTMSK	Cursor Interrupt Mask register	0x0	R/W
0x3104 0C24	CRSR_INTCLR	Cursor Interrupt Clear register	0x0	WO
0x3104 0C28	CRSR_INTRAW	Cursor Raw Interrupt Status register	0x0	RO
0x3104 0C2C	CRSR_INTSTAT	Cursor Masked Interrupt Status register	0x0	RO

10.11.1 LCD Configuration register (LCD_CFG, RW - 0x4000 4054)

The LCD_CFG register controls the selection of output pins needed for different LCD panel configurations, as well as prescaling of the clock used for LCD data generation.

The contents of the LCD_CFG register are described in Table 198.

Table 198. LCD Configuration register (LCD_CFG, RW - 0x4000 4054)

Bits	Function	Description	Reset value
31:9	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8	DISPLAY_TYPE	Sets the Display type: 0 = TFT Display 1 = STN Display	0x0
7:6	MODE_SELECT	Selects output pin group. See Table 199 and Table 200	0x0
5	HCLK_ENABLE	Enables HCLK signal to LCD controller	0x0
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.	0x0

Table 199. Mode Select Bits for TFT Display Type (bit 8 =0)

Screens	Description	Value of Bits 7:6	Number of external pins
Single Panel	12-bit (4:4:4)	0x0	18[1]
Single Panel	16-bit (5:6:5)	0x1	22[1]
Single Panel	16-bit (1:5:5:5)	0x2	24[1]
Single Panel	24-bit	0x3	30[1]

[1] See Table 183 for complete pin listing

Table 200. Mode Select Bits for STN Display Type (bit 8 =1)

Screens	Description	Value of Bits 7:6	Number of external pins
Single Panel	4-bit Monochrome	0x0	10[1]
Single Panel	8-bit Monochrome	0x1	14[1]
Single Panel	Color	0x1	14[1]
Dual Panel	4-bit Monochrome	0x2	14[2]
Dual Panel	8-bit Monochrome	0x3	22[2]
Dual Panel	Color	0x3	22[2]

[1] See Table 181 for complete pin listing

[2] See Table 182 for complete pin listing

10.11.2 Horizontal Timing register (LCD_TIMH, RW - 0x3104 0000)

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).

The contents of the LCD_TIMH register are described in Table 201.

Table 201. Horizontal Timing register (LCD_TIMH, RW - 0x3104 0000)

Bits	Function	Description	Reset value
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.	0x0
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.	0x0
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.	0x0
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.	0x0
1:0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.2.1 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
- HBP = 5
- HFP = 5
- Panel Clock Divisor (PCD) = 1 (LCDCLK / 3)

Dual panel mode:

- HSW = 3
- HBP = 5
- HFP = 5

- 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.

10.11.3 Vertical Timing register (LCD_TIMV, RW - 0x3104 0004)

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).

The contents of the LCD_TIMV register are described in Table 202.

Table 202. Vertical Timing register (LCD_TIMV, RW - 0x3104 0004)

Bits	Function	Description	Reset value
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 0–255 extra line clock cycles. Program to zero on passive displays for improved contrast.	0x0

Table 202. Vertical Timing register (LCD_TIMV, RW - 0x3104 0004)

Bits	Function	Description	Reset value
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, LCDFP, 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.	0x0
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.	0x0
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.	0x0

10.11.4 Clock and Signal Polarity register (LCD_POL, RW - 0x3104 0008)

The LCD_POL register controls various details of clock timing and signal polarity.

The contents of the LCD_POL register are described in Table 203.

Table 203. Clock and Signal Polarity register (LCD_POL, RW - 0x3104 0008)

Bits	Function	Description	Reset value
31:27	PCD_HI	Upper five bits of panel clock divisor. See description for PCD_LO, in bits [4:0] of this register.	0x0
26	BCD	Bypass pixel clock divider. Setting this to 1 bypasses the pixel clock divider logic. This is mainly used for TFT displays.	0x0
25:16	CPL	Clocks per line. This field specifies the number of actual LCDDCLK 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.	0x0
15	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 203. Clock and Signal Polarity register (LCD_POL, RW - 0x3104 0008)

Bits	Function	Description	Reset value
14	IOE	Invert output enable. This bit selects the active polarity of the output enable signal in TFT mode. In this mode, the LCDENAB 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 LCDDCLK when LCDENAB is in its active state. 0 = LCDENAB output pin is active HIGH in TFT mode. 1 = LCDENAB output pin is active LOW in TFT mode.	0x0
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 LCDDCLK. 1 = Data is driven on the LCD data lines on the falling edge of LCDDCLK.	0x0
12	IHS	Invert horizontal synchronization. The IHS bit inverts the polarity of the LCDLP signal. 0 = LCDLP pin is active HIGH and inactive LOW. 1 = LCDLP pin is active LOW and inactive HIGH.	0x0
11	IVS	Invert vertical synchronization. The IVS bit inverts the polarity of the LCDFP signal. 0 = LCDFP pin is active HIGH and inactive LOW. 1 = LCDFP pin is active LOW and inactive HIGH.	0x0

Table 203. Clock and Signal Polarity register (LCD_POL, RW - 0x3104 0008)

Bits	Function	Description	Reset value
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, LCDENAB. This field has no effect if the LCD is operating in TFT mode, when the LCDENAB pin is used as a data enable signal.	0x0
5	CLKSEL	Clock Select. This bit controls the selection of the source for LCDCLK. 0 = the clock source for the LCD block is HCLK. 1 = the clock source for the LCD block is LCDCLKIN.	0x0
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 LCDDCLK from the input clock, LCDDCLK = 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 LCDDCLK 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 (LCDDCLK = LCDCLK/3). Dual panel color mode, PCD = 4 (LCDDCLK = LCDCLK/6). Single panel monochrome 4-bit interface mode, PCD = 2(LCDDCLK = LCDCLK/4). Dual panel monochrome 4-bit interface mode and single panel monochrome 8-bit interface mode, PCD = 6(LCDDCLK = LCDCLK/8). Dual panel monochrome 8-bit interface mode, PCD = 14(LCDDCLK = LCDCLK/16).	0x0

10.11.5 Line End Control register (LCD_LE, RW - 0x3104 000C)

The LCD_LE register controls the enabling of line-end signal LCDLE. When enabled, a positive pulse, four LCDCLK periods wide, is output on LCDLE 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.

The contents of the LCD_LE register are described in Table 204.

Table 204. Line End Control register (LCD_LE, RW - 0x3104 000C)

Bits	Function	Description	Reset value
31:17	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
16	LEE	LCD Line end enable. 0 = LCDLE disabled (held LOW). 1 = LCDLE signal active.	0x0
15:7	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
6:0	LED	Line-end delay. Controls Line-end signal delay from the rising-edge of the last panel clock, LCDDCLK. Program with number of LCDCLK clock periods minus 1.	0x0

10.11.6 Upper Panel Frame Base Address register (LCD_UPBASE, RW - 0x3104 0010)

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.

The contents of the LCD_UPBASE register are described in Table 205.

Table 205. Upper Panel Frame Base register (LCD_UPBASE, RW - 0x3104 0010)

Bits	Function	Description	Reset value
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.	0x0
2:0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.7 Lower Panel Frame Base Address register (LCD_LPBASE, RW - 0x3104 0014)

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 206.

Table 206. Lower Panel Frame Base register (LCD_LPBASE, RW - 0x3104 0014)

Bits	Function	Description	Reset value
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.	0x0
2:0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.8 LCD Control register (LCD_CTRL, RW - 0x3104 0018)

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 207.

Table 207. LCD Control register (LCD_CTRL, RW - 0x3104 0018)

Bits	Function	Description	Reset value
31:17	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
15:14	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
11	LcdPwr	LCD power enable. 0 = power not gated through to LCD panel and LCDV[23:0] signals disabled, (held LOW). 1 = power gated through to LCD panel and LCDV[23:0] signals enabled, (active). See LCD power up and power down sequence for details on LCD power sequencing.	0x0

Table 207. LCD Control register (LCD_CTRL, RW - 0x3104 0018)

Bits	Function	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.	0x0
9	BEBO	Big-endian Byte Order. Controls byte ordering in memory: 0 = little-endian byte order. 1 = big-endian byte order.	0x0
8	BGR	Color format selection. 0 = RGB: normal output. 1 = BGR: red and blue swapped.	0x0
7	LcdDual	Single or Dual LCD panel selection. STN LCD interface is: 0 = single-panel. 1 = dual-panel.	0x0
6	LcdMono8	Monochrome LCD interface width. This bit 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.	0x0
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.	0x0

Table 207. LCD Control register (LCD_CTRL, RW - 0x3104 0018)

Bits	Function	Description	Reset value
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.	0x0
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.	0x0
0	LcdEn	LCD enable control bit. 0 = LCD disabled. Signals LCDLP, LCDDCLK, LCDFP, LCDENAB, and LCDLE are low. 1 = LCD enabled. Signals LCDLP, LCDDCLK, LCDFP, LCDENAB, and LCDLE are high. See LCD power up and power down sequence for details on LCD power sequencing.	0x0

10.11.9 Interrupt Mask register (LCD_INTMSK, RW - 0x3104 001C)

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 208.

Table 208. Interrupt Mask register (LCD_INTMSK, RW - 0x3104 001C)

Bits	Function	Description	Reset value
31:5	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
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.	0x0

Table 208. Interrupt Mask register (LCD_INTMSK, RW - 0x3104 001C)

Bits	Function	Description	Reset value
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.	0x0
1	FUFIM	FIFO underflow interrupt enable. 0: The FIFO underflow interrupt is disabled. 1: Interrupt will be generated when the FIFO underflows.	0x0
0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.10 Raw Interrupt Status register (LCD_INTRAW, RO - 0x3104 0020)

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.

The contents of LCD_INTRAW register are described in Table 209.

Table 209. Raw Interrupt Status register (LCD_INTRAW, RO - 0x3104 0020)

Bits	Function	Description	Reset value
31:5	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
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.	0x0
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.	0x0
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.	
0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.11 Masked Interrupt Status register (LCD_INTSTAT, RW - 0x3104 0024)

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.

The contents of LCD_INTSTAT register are described in Table 210.

Table 210. Masked Interrupt Status register (LCD_INTSTAT, RW - 0x3104 0024)

Bits	Function	Description	Reset value
31:5	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
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.	0x0
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.	0x0
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.	0x0
0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.12 Interrupt Clear register (LCD_INTCLR, RW - 0x3104 0028)

The LCD_INTCLR register is Write-Only. Writing a logic 1 to the relevant bit clears the corresponding interrupt.

The contents of the LCD_INTCLR register are described in Table 211.

Table 211. Interrupt Clear register (LCD_INTCLR, RW - 0x3104 0028)

Bits	Function	Description	Reset value
31:5	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
4	BERIC	AHB master error interrupt clear. Writing a 1 to this bit clears the AHB master error interrupt.	0x0
3	VCompIC	Vertical compare interrupt clear. Writing a 1 to this bit clears the vertical compare interrupt.	0x0

Table 211. Interrupt Clear register (LCD_INTCLR, RW - 0x3104 0028)

Bits	Function	Description	Reset value
2	LNBUIC	LCD next address base update interrupt clear. Writing a 1 to this bit clears the LCD next address base update interrupt.	0x0
1	FUFIC	FIFO underflow interrupt clear. Writing a 1 to this bit clears the FIFO underflow interrupt.	0x0
0	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

10.11.13 Upper Panel Current Address register (LCD_UPCURR, RW - 0x3104 002C)

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.

The contents of the LCD_UPCURR register are described in Table 212.

Table 212. Upper Panel Current Address register (LCD_UPCURR, RW - 0x3104 002C)

Bits	Function	Description	Reset value
31:0	LCDUPCURR	LCD Upper Panel Current Address. Contains the current LCD upper panel data DMA address.	0x0

10.11.14 Lower Panel Current Address register (LCD_LPCURR, RW - 0x3104 0030)

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.

The contents of the LCD_LPCURR are described in Table 213.

Table 213. Lower Panel Current Address register (LCD_LPCURR, RW - 0x3104 0030)

Bits	Function	Description	Reset value
31:0	LCDLPCURR	LCD Lower Panel Current Address. Contains the current LCD lower panel data DMA address.	0x0

10.11.15 Color Palette registers (LCD_PAL, RW - 0x3104 0200 to 0x3104 03FC)

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.

The contents of the LCD_PAL register are described in Table 214.

Table 214. Color Palette registers (LCD_PAL, RW - 0x3104 0200 to 0x3104 03FC)

Bits	Function	Description	Reset value
31	I	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.	0x0
30:26	B[4:0]	Blue palette data.	0x0
25:21	G[4:0]	Green palette data.	0x0
20:16	R[4:0]	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.	0x0
15	I	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.	0x0
14:10	B[4:0]	Blue palette data.	0x0
9:5	G[4:0]	Green palette data.	0x0
4:0	R[4:0]	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.	0x0

10.11.16 Cursor Image registers (CRSR_IMG, RW - 0x3104 0800 to 0x3104 0BFC)

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 Image format on page 2-19. 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).

The contents of the CRSR_IMG register are described in Table 215.

Table 215. Cursor Image registers (CRSR_IMG, RW - 0x3104 0800 to 0x3104 0BFC)

Bits	Function	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.	0x0

10.11.17 Cursor Control register (CRSR_CTRL, RW - 0x3104 0C00)

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.

The contents of the CRSR_CTRL register are described in Table 216.

Table 216. Cursor Control register (CRSR_CTRL, RW - 0x3104 0C00)

Bits	Function	Description	Reset value
31:6	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
5:4	CrsrNum[1:0]	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.	0x0
3:1	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
0	CrsrOn	Cursor enable. 0 = Cursor is not displayed. 1 = Cursor is displayed.	0x0

10.11.18 Cursor Configuration register (CRSR_CFG, RW - 0x3104 0C04)

The CRSR_CFG register provides overall configuration information for the hardware cursor.

The contents of the CRSR_CFG register are described in Table 217.

Table 217. Cursor Configuration register (CRSR_CFG, RW - 0x3104 0C04)

Bits	Function	Description	Reset value
31:2	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
1	FrameSync	Cursor frame synchronization type. 0 = Cursor coordinates are asynchronous. 1 = Cursor coordinates are synchronized to the frame synchronization pulse.	0x0
0	CrsrSize	Cursor size selection. 0 = 32x32 pixel cursor. Allows for 4 defined cursors. 1 = 64x64 pixel cursor.	0x0

10.11.19 Cursor Palette register 0 (CRSR_PAL0, RW - 0x3104 0C08)

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.

The contents of the CRSR_PAL0 register are described in Table 218.

Table 218. Cursor Palette register 0 (CRSR_PAL0, RW - 0x3104 0C08)

Bits	Function	Description	Reset value
31:24	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
23:16	Blue	Blue color component.	0x0
15:8	Green	Green color component	0x0
7:0	Red	Red color component	0x0

10.11.20 Cursor Palette register 1 (CRSR_PAL1, RW - 0x3104 0C0C)

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.

The contents of the CRSR_PAL1 register are described in Table 219.

Table 219. Cursor Palette register 1 (CRSR_PAL1, RW - 0x3104 0C0C)

Bits	Function	Description	Reset value
31:24	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
23:16	Blue	Blue color component.	0x0
15:8	Green	Green color component	0x0
7:0	Red	Red color component	0x0

10.11.21 Cursor XY Position register (CRSR_XY, RW - 0x3104 0C10)

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.

The contents of the CRSR_XY register are described in Table 220.

Table 220. Cursor XY Position register (CRSR_XY, RW - 0x3104 0C10)

Bits	Function	Description	Reset value
31:26	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
15:10	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0

10.11.22 Cursor Clip Position register (CRSR_CLIP, RW - 0x3104 0C14)

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 221.

Table 221. Cursor Clip Position register (CRSR_CLIP, RW - 0x3104 0C14)

Bits	Function	Description	Reset value
31:14	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0
7:6	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0

10.11.23 Cursor Interrupt Mask register (CRSR_INTMSK, RW - 0x3104 0C20)

The CRSR_INTMSK register is used to enable or disable the cursor from interrupting the processor.

The contents of the CRSR_INTMSK register are described in Table 222.

Table 222. Cursor Interrupt Mask register (CRSR_INTMSK, RW - 0x3104 0C20)

Bits	Function	Description	Reset value
31:1	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0

10.11.24 Cursor Interrupt Clear register (CRSR_INTCLR, RW - 0x3104 0C24)

The CRSR_INTCLR register is used by software to clear the cursor interrupt status and the cursor interrupt signal to the processor.

The contents of the CRSR_INTCLR register are described in Table 223.

Table 223. Cursor Interrupt Clear register (CRSR_INTCLR, RW - 0x3104 0C24)

Bits	Function	Description	Reset value
31:1	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0

10.11.25 Cursor Raw Interrupt Status register (CRSR_INTRAW, RW - 0x3104 0C28)

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.

The contents of the CRSR_INTRAW register are described in Table 224.

Table 224. Cursor Raw Interrupt Status register (CRSR_INTRAW, RW - 0x3104 0C28)

Bits	Function	Description	Reset value
31:1	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0

10.11.26 Cursor Masked Interrupt Status register (CRSR_INTSTAT, RW - 0x3104 0C2C)

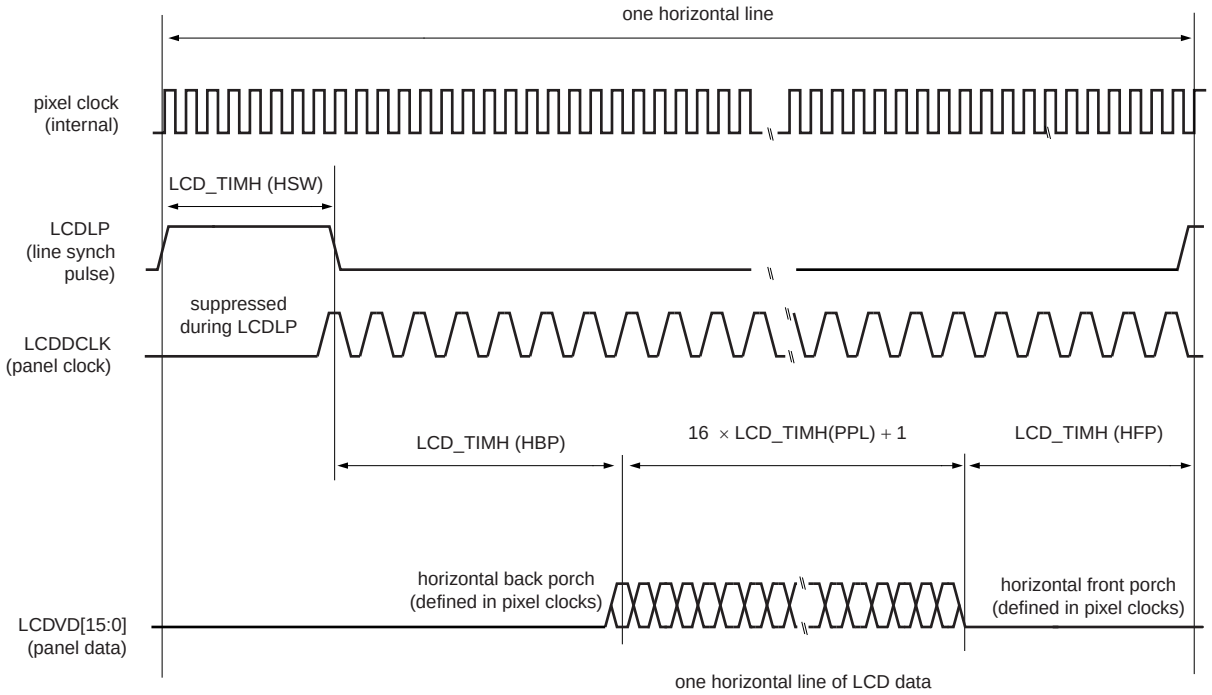
The CRSR_INTSTAT register is set to indicate a cursor interrupt providing that the interrupt is not masked in the CRSR_INTMSK register.

The contents of the CRSR_INTSTAT register are described in Table 225.

Table 225. Cursor Masked Interrupt Status register (CRSR_INTSTAT, RW - 0x3104 0C2C)

Bits	Function	Description	Reset value
31:1	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
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.	0x0

10.12 LCD timing diagrams



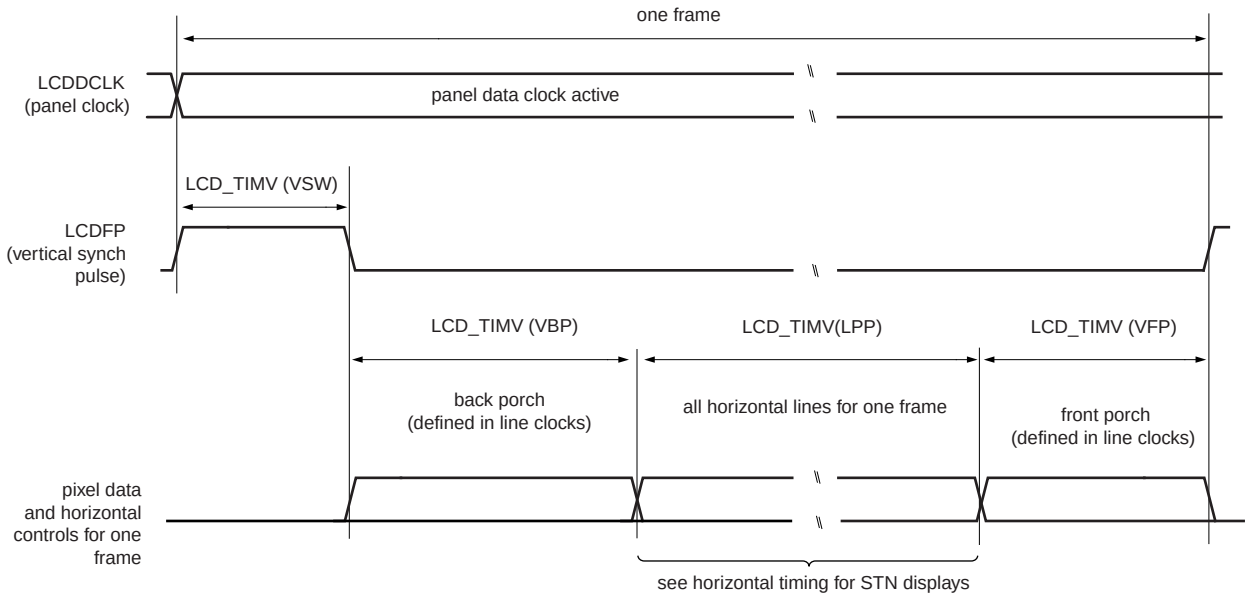
(1) The active data lines will vary with the type of STN panel (4-bit, 8-bit, color, mono) and with single or dual frames.

(2) The LCD panel clock is selected and scaled by the LCD controller and used to produce LCDCLK.

(3) The duration of the LCDLP signal is controlled by the HSW field in the LCD_TIMH register.

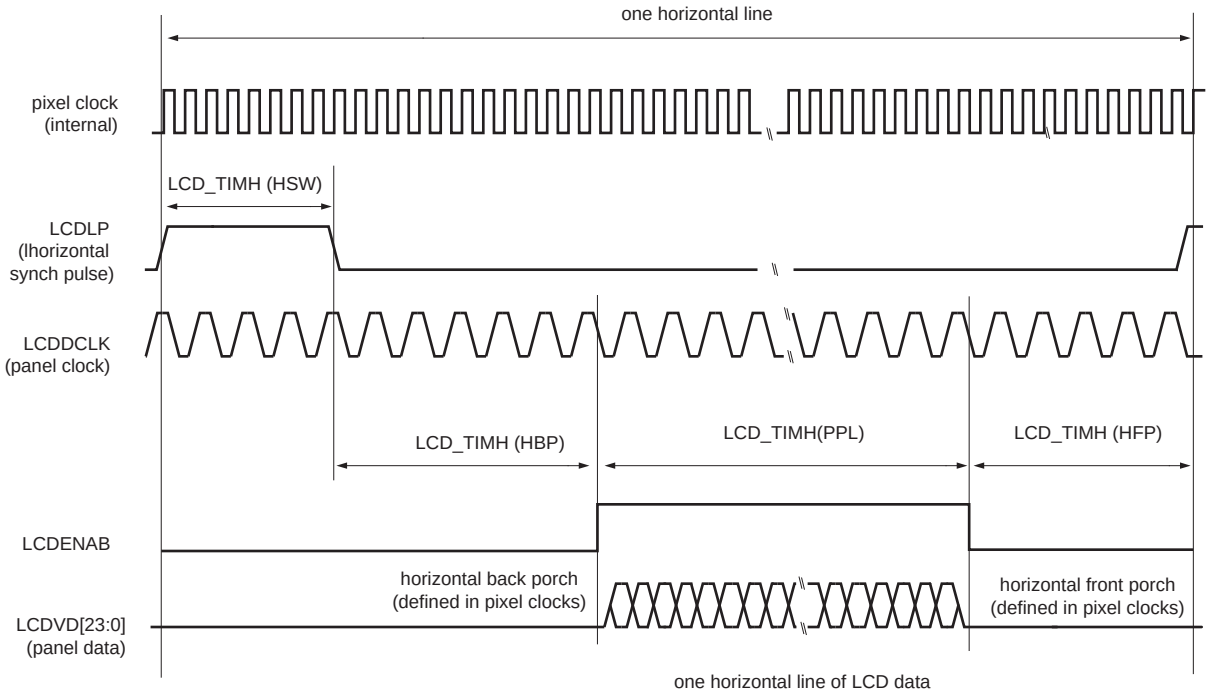
(4) The Polarity of the LCDLP signal is determined by the IHS bit in the LCD_POL register.

Fig 33. Horizontal timing for STN displays



(1) Signal polarities may vary for some displays.

Fig 34. Vertical timing for STN displays



(1) The active data lines will vary with the type of TFT panel.

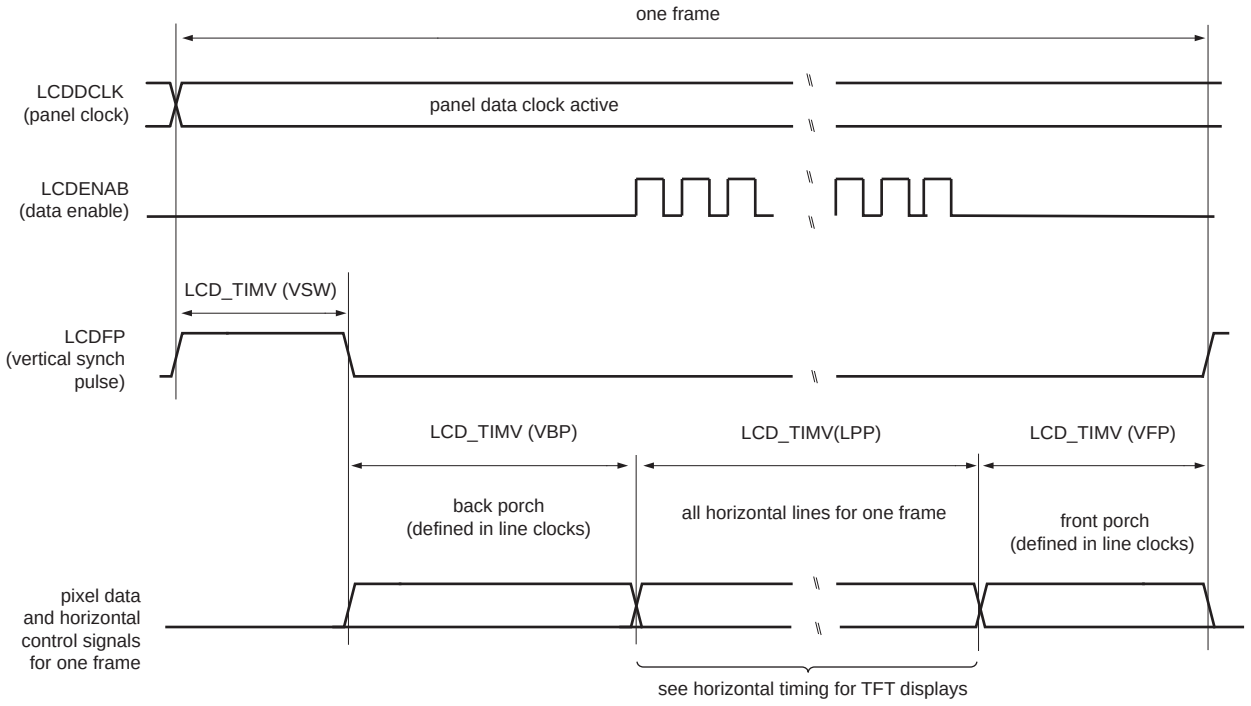
(2) The LCD panel clock is selected and scaled by the LCD controller and used to produce LCDCLK.

(3) The duration of the LCDLP is controlled by the HSW field in the LCD_TIMH register.

(4) The polarity of the LCDLP signal is determined by the IHS bit in the LCD_POL register.

(5) The polarity of the LPCENAB signal is determined by the IOE bit in the LCD_POL register.

Fig 35. Horizontal timing for TFT displays



(1) Polarities may vary for some displays.

(2) LCDENAB is active only when active data is transmitted during each horizontal line.

Fig 36. Vertical timing for TFT displays

10.13 LCD panel signal usage

Table 226. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[23]	-	-	-	-	-	-
LCDVD[22]	-	-	-	-	-	-
LCDVD[21]	-	-	-	-	-	-
LCDVD[20]	-	-	-	-	-	-
LCDVD[19]	-	-	-	-	-	-
LCDVD[18]	-	-	-	-	-	-
LCDVD[17]	-	-	-	-	-	-
LCDVD[16]	-	-	-	-	-	-
LCDVD[15]	-	-	-	-	-	-
LCDVD[14]	-	-	-	-	-	-
LCDVD[13]	-	-	-	-	-	-
LCDVD[12]	-	-	-	-	-	-
LCDVD[11]	-	-	-	-	-	-
LCDVD[10]	-	-	-	-	-	-

Table 226. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[9]	-	-	-	-	-	-
LCDVD[8]	-	-	-	-	-	-
LCDVD[7]	-	-	P0.5 / I2STX_SDA0 / LCDVD[7]	UD[7]	P0.5 / I2STX_SDA0 / LCDVD[7]	UD[7]
LCDVD[6]	-	-	P0.4 / I2SRX_WS0 / LCDVD[6]	UD[6]	P0.4 / I2SRX_WS0 / LCDVD[6]	UD[6]
LCDVD[5]	-	-	P0.3 / I2SRX_CLK0 / LCDVD[5]	UD[5]	P0.3 / I2SRX_CLK0 / LCDVD[5]	UD[5]
LCDVD[4]	-	-	P0.2 / I2SRX_SDA0 / LCDVD[4]	UD[4]	P0.2 / I2SRX_SDA0 / LCDVD[4]	UD[4]
LCDVD[3]	GPO_21 / U4_TX / LCDVD[3]	UD[3]	GPO_21 / U4_TX / LCDVD[3]	UD[3]	GPO_21 / U4_TX / LCDVD[3]	UD[3]
LCDVD[2]	P2[8]	UD[2]	P2[8]	UD[2]	P2[8]	UD[2]
LCDVD[1]	GPO_3 / LCDVD[1]	UD[1]	GPO_3 / LCDVD[1]	UD[1]	GPO_3 / LCDVD[1]	UD[1]
LCDVD[0]	GPO_2 / MAT1.0 / LCDVD[0]	UD[0]	GPO_2 / MAT1.0 / LCDVD[0]	UD[0]	GPO_2 / MAT1.0 / LCDVD[0]	UD[0]
LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP
LCDM	GPO_16 / MC0B / LCDENAB / LCDM	LCDM	GPO_16 / MC0B / LCDENAB / LCDM	LCDM	GPO_16 / MC0B / LCDENAB / LCDM	LCDM
LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP
LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK
LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE
LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR
LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN

Table 227. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[23]	-	-	-	-	-	-
LCDVD[22]	-	-	-	-	-	-
LCDVD[21]	-	-	-	-	-	-
LCDVD[20]	-	-	-	-	-	-
LCDVD[19]	-	-	-	-	-	-
LCDVD[18]	-	-	-	-	-	-
LCDVD[17]	-	-	-	-	-	-
LCDVD[16]	-	-	-	-	-	-
LCDVD[15]	-	-	SYSCLKEN / LCDVD[15]	LD[7]	SYSCLKEN / LCDVD[15]	LD[7]
LCDVD[14]	-	-	GPO_22 / U7_HRTS / LCDVD[14]	LD[6]	GPO_22 / U7_HRTS / LCDVD[14]	LD[6]
LCDVD[13]	-	-	P0.7 / I2STX_WS0 / LCDVD[13]	LD[5]	P0.7 / I2STX_WS0 / LCDVD[13]	LD[5]
LCDVD[12]		-	P0.6 / I2STX_CLK0 / LCDVD[12]	LD[4]	P0.6 / I2STX_CLK0 / LCDVD[12]	LD[4]
LCDVD[11]	U7_TX / MAT1.1 / LCDVD[11]	LD[3]	U7_TX / MAT1.1 / LCDVD[11]	LD[3]	U7_TX / MAT1.1 / LCDVD[11]	LD[3]
LCDVD[10]	U7_RX / CAP0.0 / LCDVD[10]	LD[2]	U7_RX / CAP0.0 / LCDVD[10]	LD[2]	U7_RX / CAP0.0 / LCDVD[10]	LD[2]
LCDVD[9]	GPO_9 / LCDVD[9]	LD[1]	GPO_9 / LCDVD[9]	LD[1]	GPO_9 / LCDVD[9]	LD[1]
LCDVD[8]	GPO_8 / LCDVD[8]	LD[0]	GPO_8 / LCDVD[8]	LD[0]	GPO_8 / LCDVD[8]	LD[0]
LCDVD[7]	-	-	P0.5 / I2STX_SDA0 / LCDVD[7]	UD[7]	P0.5 / I2STX_SDA0 / LCDVD[7]	UD[7]
LCDVD[6]	-	-	P0.4 / I2SRX_WS0 / LCDVD[6]	UD[6]	P0.4 / I2SRX_WS0 / LCDVD[6]	UD[6]
LCDVD[5]	-	-	P0.3 / I2SRX_CLK0 / LCDVD[5]	UD[5]	P0.3 / I2SRX_CLK0 / LCDVD[5]	UD[5]
LCDVD[4]	-	-	P0.2 / I2SRX_SDA0 / LCDVD[4]	UD[4]	P0.2 / I2SRX_SDA0 / LCDVD[4]	UD[4]
LCDVD[3]	GPO_21 / U4_TX / LCDVD[3]	UD[3]	GPO_21 / U4_TX / LCDVD[3]	UD[3]	GPO_21 / U4_TX / LCDVD[3]	UD[3]

Table 227. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[2]	GPO_7 / LCDVD[2]	UD[2]	GPO_7 / LCDVD[2]	UD[2]	GPO_7 / LCDVD[2]	UD[2]
LCDVD[1]	P2[7]	UD[1]	P2[7]	UD[1]	P2[7]	UD[1]
LCDVD[0]	GPO_2 / MAT1.0 / LCDVD[0]	UD[0]	GPO_2 / MAT1.0 / LCDVD[0]	UD[0]	GPO_2 / MAT1.0 / LCDVD[0]	UD[0]
LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP
LCDM	GPO_16 / MC0B / LCDENAB / LCDM	LCDM	GPO_16 / MC0B / LCDENAB / LCDM	LCDM	GPO_16 / MC0B / LCDENAB / LCDM	LCDM
LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP
LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK
LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE
LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR
LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN

Table 228. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[23]	SPI2_CLK / SCK1 / LCDVD[23]	BLUE3	SPI2_CLK / SCK1 / LCDVD[23]	BLUE4	SPI2_CLK / SCK1 / LCDVD[23]	BLUE4	SPI2_CLK / SCK1 / LCDVD[23]	BLUE7
LCDVD[22]	GPIO_4 / SSEL1 / LCDVD[22]	BLUE2	GPIO_4 / SSEL1 / LCDVD[22]	BLUE3	GPIO_4 / SSEL1 / LCDVD[22]	BLUE3	GPIO_4 / SSEL1 / LCDVD[22]	BLUE6
LCDVD[21]	SPI2_DATIN / MISO1 / LCDVD[21]	BLUE1	SPI2_DATIN / MISO1 / LCDVD[21]	BLUE2	SPI2_DATIN / MISO1 / LCDVD[21]	BLUE2	SPI2_DATIN / MISO1 / LCDVD[21]	BLUE5
LCDVD[20]	SPI2_DATIO / MOSI1 / LCDVD[20]	BLUE0	SPI2_DATIO / MOSI1 / LCDVD[20]	BLUE1	SPI2_DATIO / MOSI1 / LCDVD[20]	BLUE1	SPI2_DATIO / MOSI1 / LCDVD[20]	BLUE4

Table 228. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[19]	-	-	PWM_OUT 2 / LCDVD[19]	BLUE0	PWM_OUT2 / LCDVD[19]	BLUE0	PWM_OUT 2 / LCDVD[19]	BLUE3
LCDVD[18]	-	-	-	-	GPO_6 / LCDVD[18]	intensity	GPO_6 / LCDVD[18]	BLUE2
LCDVD[17]	-	-	-	-	-	-	HIGHCOR E / LCDVD[17]	BLUE1
LCDVD[16]	-	-	-	-	-	-	PWM_OUT 1 / LCDVD[16]	BLUE0
LCDVD[15]	SYSCLKEN / LCDVD[15]	GREEN3	SYSCLKEN / LCDVD[15]	GREEN5	SYSCLKEN / LCDVD[15]	GREEN4	SYSCLKEN / LCDVD[15]	GREEN7
LCDVD[14]	GPO_22 / U7_HRTS / LCDVD[14]	GREEN2	GPO_22 / U7_HRTS / LCDVD[14]	GREEN4	GPO_22 / U7_HRTS / LCDVD[14]	GREEN3	GPO_22 / U7_HRTS / LCDVD[14]	GREEN6
LCDVD[13]	P0.7 / I2STX_WS0 / LCDVD[13]	GREEN1	P0.7 / I2STX_WS0 / LCDVD[13]	GREEN3	P0.7 / I2STX_WS0 / LCDVD[13]	GREEN2	P0.7 / I2STX_WS 0 / LCDVD[13]	GREEN5
LCDVD[12]	P0.6 / I2STX_CLK0 / LCDVD[12]	GREEN0	P0.6 / I2STX_CLK 0 / LCDVD[12]	GREEN2	P0.6 / I2STX_CLK0 / LCDVD[12]	GREEN1	P0.6 / I2STX_CLK 0 / LCDVD[12]	GREEN4
LCDVD[11]	-	-	U7_TX / MAT1.1 / LCDVD[11]	GREEN1	U7_TX / MAT1.1 / LCDVD[11]	GREEN0	U7_TX / MAT1.1 / LCDVD[11]	GREEN3
LCDVD[10]	-	-	U7_RX / CAP0.0 / LCDVD[10]	GREEN0	U7_RX / CAP0.0 / LCDVD[10]	intensity	U7_RX / CAP0.0 / LCDVD[10]	GREEN2
LCDVD[9]	-	-	-	-	-	-	GPO_9 / LCDVD[9]	GREEN1
LCDVD[8]	-	-	-	-	-	-	GPO_8 / LCDVD[8]	GREEN0
LCDVD[7]	P0.5 / I2STX_SDA 0 / LCDVD[7]	RED3	P0.5 / I2STX_SDA 0 / LCDVD[7]	RED4	P0.5 / I2STX_SDA0 / LCDVD[7]	RED4	P0.5 / I2STX_SD A0 / LCDVD[7]	RED7
LCDVD[6]	P0.4 / I2SRX_WS0 / LCDVD[6]	RED2	P0.4 / I2SRX_WS0 / LCDVD[6]	RED3	P0.4 / I2SRX_WS0 / LCDVD[6]	RED3	P0.4 / I2SRX_WS 0 / LCDVD[6]	RED6
LCDVD[5]	P0.3 / I2SRX_CLK 0 / LCDVD[5]	RED1	P0.3 / I2SRX_CLK 0 / LCDVD[5]	RED2	P0.3 / I2SRX_CLK0 / LCDVD[5]	RED2	P0.3 / I2SRX_CL K0 / LCDVD[5]	RED5

Table 228. 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	
	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function	LPC32x0 pin used	LCD function
LCDVD[4]	P0.2 / I2SRX_SDA0 / LCDVD[4]	RED0	P0.2 / I2SRX_SDA0 / LCDVD[4]	RED1	P0.2 / I2SRX_SDA0 / LCDVD[4]	RED1	P0.2 / I2SRX_SDA0 / LCDVD[4]	RED4
LCDVD[3]	-	-	GPO_21 / U4_TX / LCDVD[3]	RED0	GPO_21 / U4_TX / LCDVD[3]	RED0	GPO_21 / U4_TX / LCDVD[3]	RED3
LCDVD[2]	-	-	-	-	GPO_7 / LCDVD[2]	intensity	GPO_7 / LCDVD[2]	RED2
LCDVD[1]	-	-	-	-	-	-	P0[5]	RED1
LCDVD[0]	-	-	-	-	-	-	GPO_2 / MAT1.0 / LCDVD[0]	RED0
LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP	GPO_18 / MC0A / LCDLP	LCDLP
LCDENAB	GPO_16 / MC0B / LCDENAB / LCDM	LCDENAB	GPO_16 / MC0B / LCDENAB / LCDM	LCDENAB	GPO_16 / MC0B / LCDENAB / LCDM	LCDENAB	GPO_16 / MC0B / LCDENAB / LCDM	LCDENAB
LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP	GPO_15 / MC1A / LCDFP	LCDFP
LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK	GPO_13 / MC1B / LCDDCLK	LCDDCLK
LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE	GPO_12 / MC2A / LCDLE	LCDLE
LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR	GPO_10 / MC2B / LCDPWR	LCDPWR
LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN	U7_HCTS / CAP0.1 / LCDCLKIN	LCDCLKIN

11.1 Touch screen interface

The LPC3250 includes a TouchScreen Control controller (TSC). The TSC can operate in two different modes. In AUTO mode, the touchscreen will automatically collect the X and Y coordinates of the pressed position. In MANUAL (INACTIVE) mode the controller can be sequenced entirely by software to collect the coordinates of the pressed position.

The analog block of the Analog-to-Digital Converter (ADC) has its own power supply to improve the low noise characteristics of the converter. This voltage should only be supplied internally when the core has voltage. However, the ADC block is not affected by any difference in ramp-up time for VDD_AD and VDD_CORE voltage supplies.

There is robust ESD protection on the touchscreen interface pins.

In normal operation, the touch screen controller hardware automatically measures and determines the X and Y co-ordinates where the touch screen is pressed. In addition the TSC can measure an additional analog input signal on the AUX_IN pin. This controller also operates as a three channel ADC as described in Section 12.1.

The operation of the touch screen controller is described in the following sections.

11.1.1 Analog Interface Circuit

The analog part of the TSC consists of a 10-bit ADC and circuitry to apply a voltage across the touch screen in different directions. When the touch screen detects a pen down operation, the controller applies a voltage across one of the touch screen resistive films. This voltage difference is applied to the positive and negative sense inputs on the ADC. This voltage difference serves as a reference to determine the voltage on the other film, which is proportional to the pressed position in the axis being measured. This design automatically compensates for changes in the VddTS voltage and voltage drop over the transistors. The touch screen controller then takes another measurement using the other film as a reference to measure the voltage on the other axis.

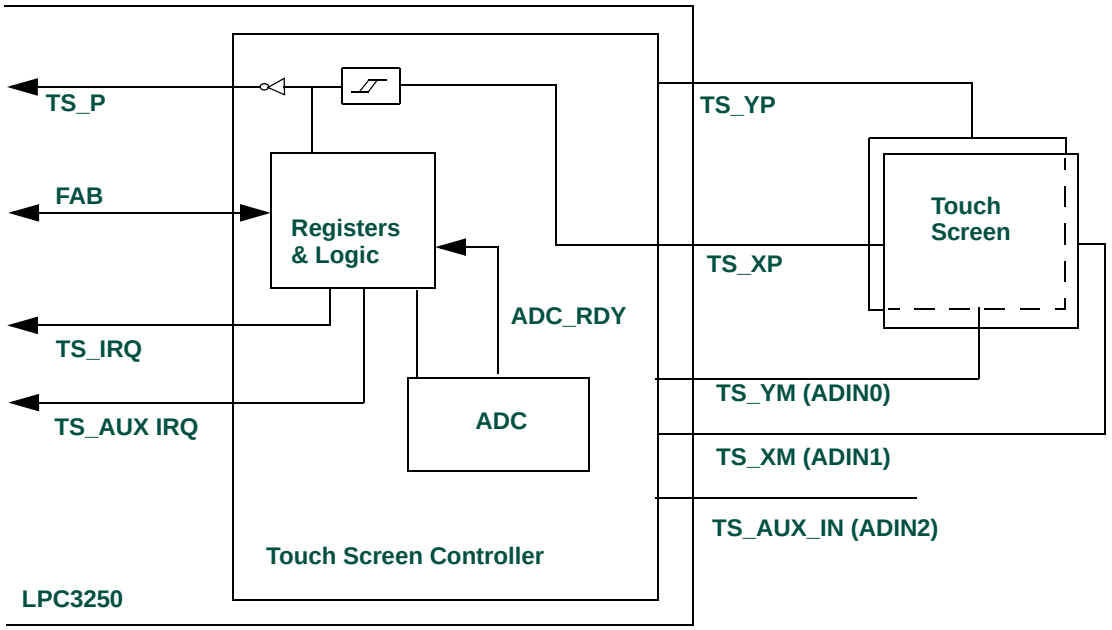


Fig 37. Touch screen signal and pin overview

11.1.1.1 Physical interface to the touch screen

In typical operation, the resistance across the touch screen is in the order of 300 ohms (200 min.) in the X-direction and 650 ohms (400 min.) in the Y-direction.

If two different positions on the touch screen are pressed at the same time the resulting resistance can be near zero. The transistors in the touch screen controller were selected to have a high tolerance to this worst case condition.

Table 229. Touchscreen pin description

Name	Comment
TS_YP	This pin is used to apply VddTS voltage to the Y-plate on the touch screen. Tie this pin to VddTS if the controller is not used
TS_XP	This pin is used to apply VddTS voltage to the X-plate on the touch screen. Tie this pin weakly to ground if the controller is not used
TS_YM (ADIN0)	This pin is used to apply ground to the Y-plate on the touch screen, and it supplies the voltage for the X-value when determining the X-axis co-ordinate. Tie this pin to ground if the controller is not used.
TS_XM (ADIN1)	This pin is used to apply ground to the X-plate on the touch screen, and it supplies the voltage for the Y-value when determining the Y-axis co-ordinate. Tie this pin to ground if the controller is not used
TS_AUX_IN (ADIN2)	This pin contains the voltage for the AUX-value when making an AUX measurement. Tie this pin to ground if the controller is not used

11.1.2 Overview of Operation

When the touch screen controller is in AUTO mode, the X and Y sample measurements are automatically stored in the TSC_SAMPLE_FIFO. If the FIFO interrupt is enabled, the touch screen controller generates a TSC_IRQ interrupt when the FIFO reaches the programmed trigger level or if the FIFO has data and PEN down is no longer detected.

The clock to the TSC is enabled in the ADCLK_CTRL register. The ADC within the TSC runs on this clock.

The touch screen controller logic can select either the 32 KHz (RTC_CLK) or PERIPH_CLK; clock selection is made using the ADCLK_CTRL register.

The ADC within the TSC can operate up to 400 KHz, so a divider is provided to lower the clock rate to a value less than 400KHz when using PERIPH_CLK as the clock source. The value of this divider is set in the ADCLK1_CTRL register.

The Touch Screen Controller register interface runs on PERIPH_CLK.

Even when the touch screen controller is in MANUAL mode (INACTIVE state), software can still control the touch screen by using the ADC_SELECT, ADC_CTRL and ADC_VALUE registers. The intended use of this mode is for testing and making individual touch screen measurements not supported by the controller when in AUTO mode.

The touch controller also has a special “auto position feature”. This feature can restrict wake up to a specified location on the touch screen surface.

11.1.2.1 Mode Selection

The touch screen controller operates in two modes.

- AUTO Mode
- MANUAL Mode

AUTO Mode

When enabled, the touch screen controller hardware automatically handles the measurement sequence to retrieve the co-ordinates as well as determining if the screen is pressed or not. In addition, it will measure the AUX analog input signal if enabled. The main purpose for the automation is to minimize the number of interrupts needed to determine the touch position. This is the normal mode of operation for the TSC.

In AUTO mode AUX_IN measurements will happen independent of the PEN down signal TS_P. The presense of a TS_P signal overrides the collection of AUX data.

The touch screen registers are set to default values after reset and the TSC enters MANUAL mode (INACTIVE state), see Figure 39. If the AUTO mode is then enabled, the touch screen controller will leave the INACTIVE State and enter the AUTO State. While the touch screen controller is running in the AUTO state, it minimizes power consumption by only clocking the ADC when an actual sample is measured, and only clocking the touch screen controller while the touch screen is pressed or the controller is in AUX mode.

TSC_IRQ is the only interrupt needed in AUTO mode, TS_AUX IRQ is optional if sampling the AUX_IN pin. The TS_P IRQ is not used because the sequenced operation of AUTO mode will result in spurious interrupts.

It is possible to switch between manual test mode and auto mode, but may not be desired.

Manual Mode

In this mode all measurement sequences are implemented in software. Here is a typical X and Y measurement sequence followed by an AUX_IN measurement.

- Enable the TSC_P and TSC_IRQ signal interrupts.
- Set the TSC state in ADC_SELECT to TS_DETECT.
- On the Touch Screen pen down (TSC_P IRQ), configure ADC_SELECT register to MEASURE_X and start the ADC by setting the TS_ADC_STROBE bit in ADC_CTRL.
- On the TSC_IRQ interrupt, read the X-axis data from ADC_VALUE register.
- Configure the Y-axis for measurement by configuring ADC_SELECT to MEASURE_Y and start the ADC by setting the TS_ADC_STROBE bit in ADC_CTRL.
- On the TSC_IRQ interrupt, read the Y-axis data from ADC_VALUE register.
- Configure the Touch screen controller to drain the X-plate.
- Configure the AUX input for measurement by configuring ADC_SELECT to MEASURE_AUX and start the ADC by setting the TS_ADC_STROBE bit in ADC_CTRL.
- On the TSC_IRQ interrupt, read the AUX_IN data from ADC_VALUE register.
- Check Pen down status and repeat as necessary.

In MANUAL mode the TSC_ADC_RDY signal is gated to the TSC_IRQ signal and time-out registers are not used.

Note the ADC is reset by hardware when the ADC is powered up using bit 2 in the ADC_CTRL register.

11.1.2.1.1 Touch Screen Controller Special Features

The Touch screen controller also supports several special features.

- Position Detect in Auto Mode
- Auxiliary Analog-to-Digital conversion (AUX_IN)
- Position Detect in Auxiliary Analog-to-Digital conversion (AUX_IN)

Position detect in Auto Mode

The touch controller has a special low power auto position detect feature. This feature limits wake up to a specified location on the touch interface. To use this feature set bits 3 and 0 high in the ADC_CTRL register. When this feature is enabled, each ADC sample is compared against a minimum and maximum value for both the X and Y axis. Only samples found to be between the min./max. values in both directions will go into the TSC_SAMPLE_FIFO. When the TSC_SAMPLE_FIFO reaches the programmed trigger level, the TSC_IRQ signal goes active. This mode can also be used to trigger a start signal to take the system out of STOP mode. These values are stored in registers TSC_MIN_X, TSC_MAX_X, TSC_MIN_Y, and TSC_MAX_Y.

Measuring the AUX analog input

The touch screen controller may also be set to measure the AUX analog input. To enable the AUX measuring mode, set bit 0 and 10 high in the ADC_CTRL register. The sampling frequency of TS_AUX_IN is specified in the TSC_AUX_UTR register. Enable the TS_AUX interrupt. In auto mode the when the ADC completes an AUX_IN measurement it will generate a TS_AUX IRQ, reading the AUX_VALUE register will provide the data and clear the TX_AUX IRQ.

Position Detect in Auxiliary Analog-to-Digital converter

The Auxiliary ADC measurement can also be use in the auto position mode. In this mode the TSC_AUX_MIN and TSC_AUX_MAX can limit the ADC range that will generate a TS_AUX interrupt or start signal.

11.1.2.1.2 Manual Mode Operation Detail

The signals used to control the TSC in manual mode are listed here.

- TS_YPC and TS_YMC signals control the film used for the Y-axis.
- TS_XPC and TS_XMC signals control the film used for the X-axis.
- TS_Ref+[1:0] and TS_Ref-[1:] registers select the reference voltage supplied to the ADC.
- TS_IN[1:0] selects is the input to the ADC.

Table 230 describes these signals, and Figure 38 shows these signals in a simplified block diagram of the TSC.

Table 230. Touchscreen Control and Data Signal Description

Name	Comment
TS_Ref+ [1:0]	Selects which reference voltage to use on Ref+
TS_Ref- [1:0]	Selects which reference voltage to use on Ref-
TS_IN[1:0]	Selects which input voltage to send to the ADC
TS_XPC	Selects the voltage on the TS_XP connection. (X Plus)
TS_XMC	Selects the voltage on the TS_XM connection. (X Minus)
TS_YPC	Selects the voltage on the TS_YP connection. (Y Plus)
TS_YMC	Selects the voltage on the TS_YM connection. (Y Minus)
TS_ADC	Supplies the ADC converted output data.
TSC_P	The inverted logical level on the TS_XP connection. This signal generates a touch interrupt as well as determining if the screen is pressed.
TSC_ADC_RDY	In manual mode, the ADC Ready signal pulses high for one ADC_CLK cycle when an AD-conversion is ready. In auto mode the controller switches off the ADC clock off immediately after a conversion is ready causing ADC_RDY to lock high. The ADC_RDY signal is ignored in auto mode.

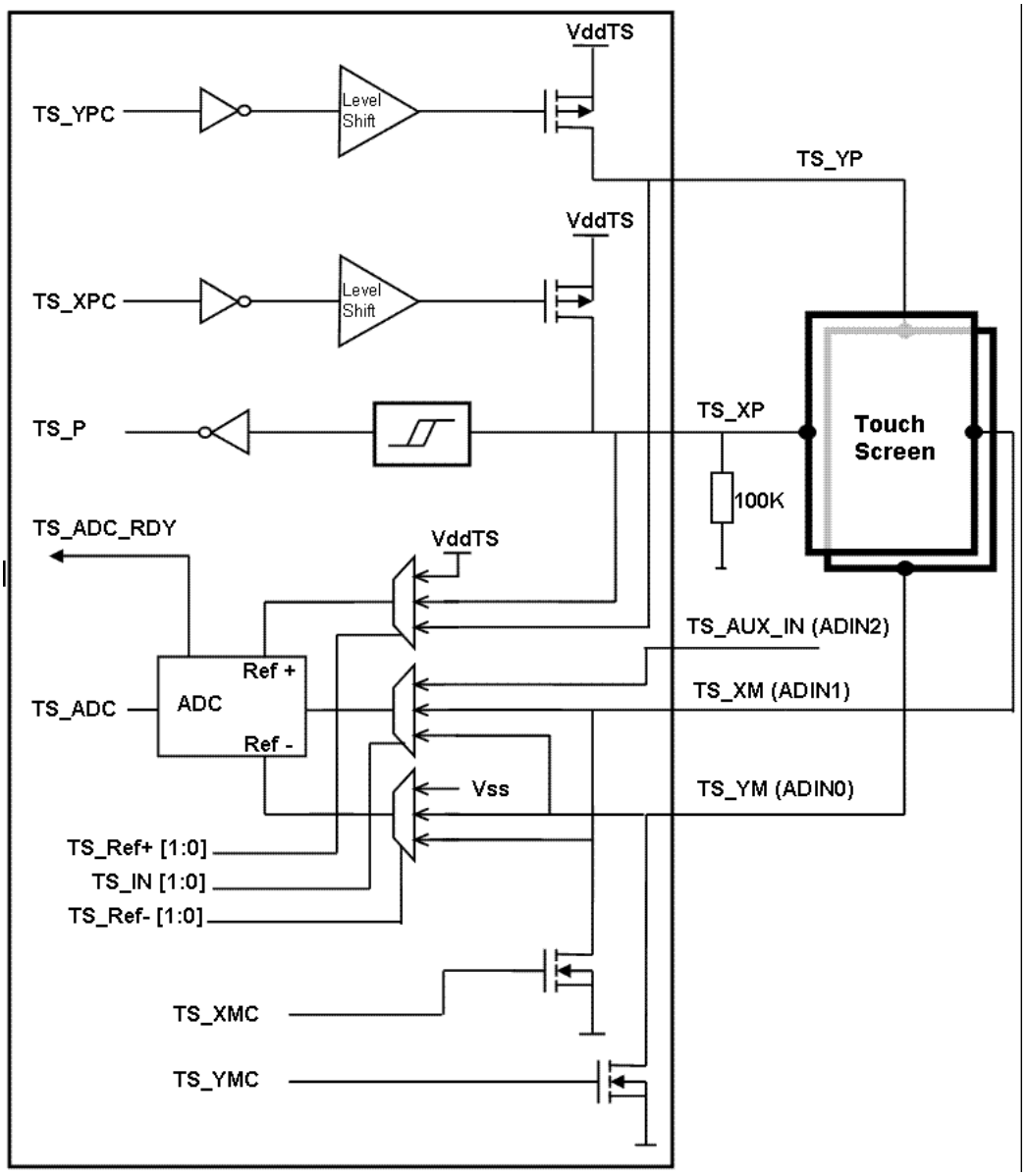


Fig 38. Simplified block diagram of the analog interface to the touch screen

Controlling the Touch ADC reference and Analog in signal multiplexer

There are five configurations needed to perform a manual data collection sequence are defined here.

Configurations needed for manual data collection

TS_DETECT — The TSC is configured to detect PEN DOWN.

MEASURE_X — The TSC is configured to measure the X-axis position.

MEASURE_Y — The TSC is configured to measure the Y-axis position.

MEASURE_AUX_IN — The TSC is configured to measure the AUX_IN ADC value.

DRAIN_X_PLATE — The TSC is configured to drain the X-plate

The ADC_SELECT register is used to configure the signals to take manual measurements with the TSC.

Table 231 lists the value needed for the ADC_SELECT register for each of the five configurations.

Table 231. Touch screen manual configurations

TS Control State	ADC_SELECT Register Value	ADC_SELECT individual bit field values						
		TS_REF– (AD_Ref–)	TS_REF+ (AD_Ref+)	TS_IN (AD_IN)	TS_YMC	TS_YPC	TS_XMC	TS_XPC
TS_DETECT	0x010	00	00	01	0	0	0	0
MEASURE_X	0x007	00	00	00	0	1	1	1
MEASURE_Y	0x158	01	01	01	1	0	0	0
MEASURE_AUX_IN	0x2AE	10	10	10	1	1	1	0
DRAIN_X_PLATE	0x01E	00	00	01	1	1	1	0

Table 232 lists the bias and inputs selected for input to the ADC in each manual configuration.

Table 232. Touch screen reference control state signals

TS Control State	Resulting ADC Configuration		
	REF+ ADC	REF– ADC	IN ADC
TS_DETECT	TS_XP	TS_XM	TS_XM
MEASURE_X	TS_XP	TS_XM	TS_YM
MEASURE_Y	TS_YP	TS_YM	TS_XM
MEASURE_AUX_IN	Vddts	Vss	TS_AUX
DRAIN_X_PLATE	TS_XP	TS_XM	TS_XM

Table 233 shows the resulting external pin state for each configuration.

Table 233. Touch screen external pin configurations

TS control state	Resulting External Pin State			
	TS_XP	TS_XM	TS_YP	TS_YM
TS_DETECT	Z[2]	Z	Vddts	Z
MEASURE_X [1]	Vddts	Vss	Z	Z
MEASURE_Y[1]	Z	Z	Vddts	Vss
MEASURE _AUX_IN	Z	Vss	Z	Vss
DRAIN_X_PLATE[3]	Z	Vss	Z	Vss

[1] The measure X and Y states consume relatively high currents and must only be enabled while the software is performing in a co-ordinate fetch for the touch position. In automatic mode, the controller handles this in hardware.

[2] To be able to detect a touch in TS_DETECT State the TS_XP external pin needs to be weakly pulled down.

[3] The DRAIN_X_PLATE state is used to allow a fast unload of the TOUCH screen plate connected to the TS_XP pin before going to the TS_DETECT state.

11.1.2.2 The Touch Screen Controller state machine

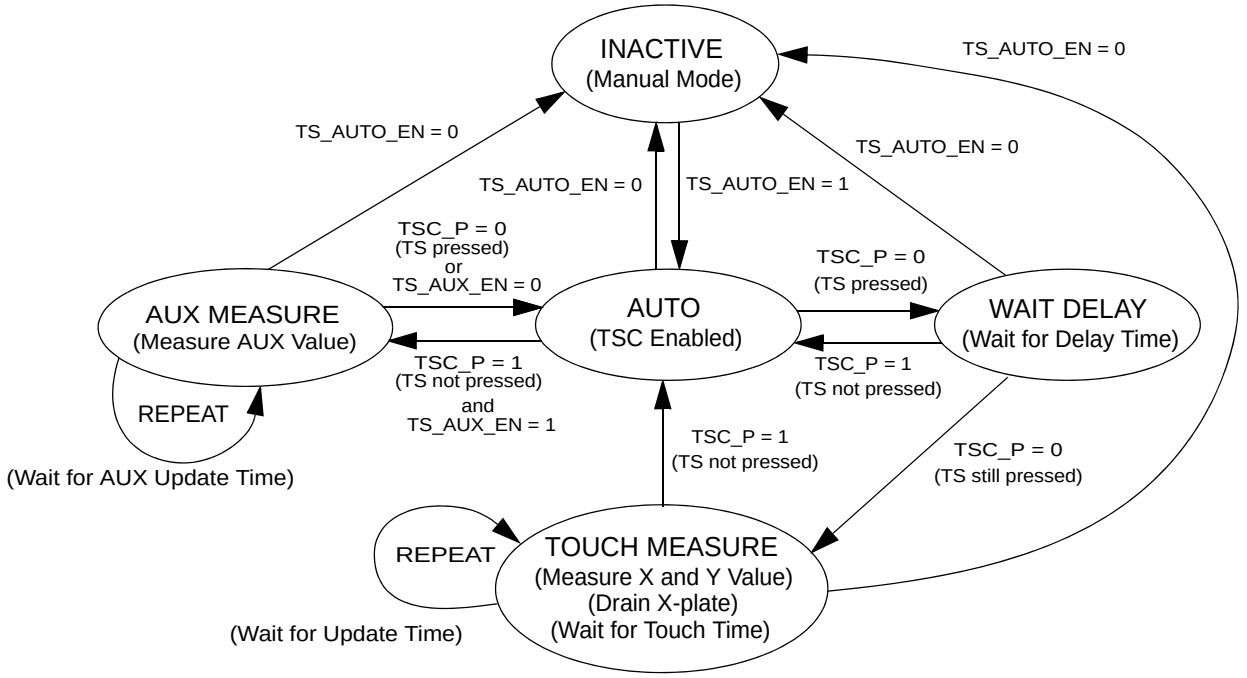


Fig 39. Touch screen controller state diagram

The touch screen controller state diagram can be divided into five main states.

Touch Screen Controller States

INACTIVE — In this state, the TSC and AUX inputs can be measured in using software that controls each step in the Touch screen data collection sequence. This is also referred to as MANUAL mode.

AUTO — In this state, the TSC is enabled and the AUX mode is not enabled, the controller is waiting for the touch screen to be touched or the TS_AUX_EN to be enabled.

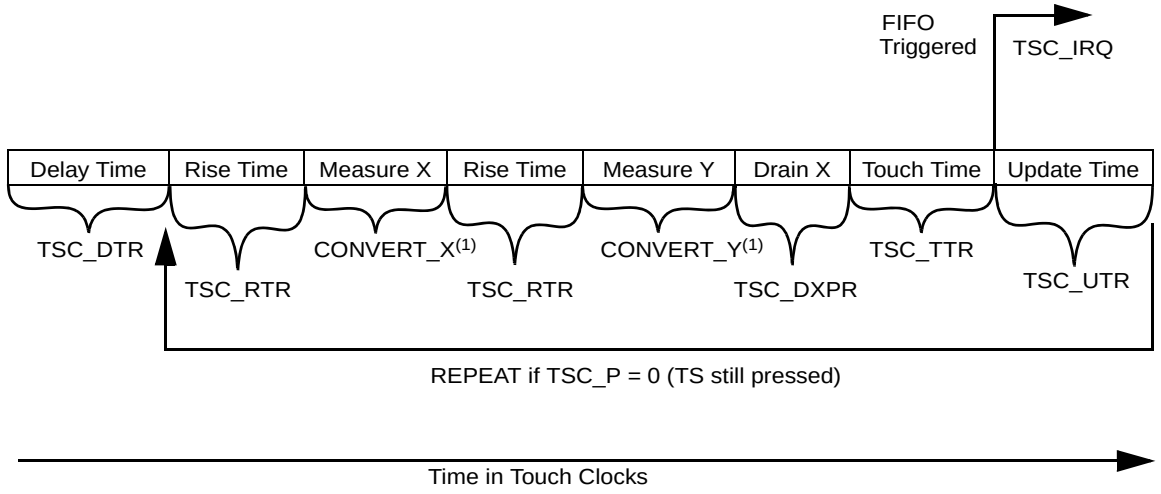
WAIT DELAY — In this state, TSC waits for a specified amount of time to see if it is a real touch.

AUX MEASURE — In this state, both the TSC and the AUX measure mode are enabled, the AUX input pin is measured using the interval defined in the TSC_AUX_UTR register. A touch detected on the touch screen will change the controller to the WAIT DELAY state.

TOUCH MEASURE — In this state, the touch screen controller measure the X and Y value and determine the touched point. The controller will stay in this state as long as the Touch screen remains touched.

11.1.3 Touch Screen Controller Time registers

Figure 40 shows the relationship among the various timing registers used when an automatic sample is taken.



(1) CONVERT_X and CONVERT_Y represent the number of clock cycles needed to perform an Analog to digital conversion for the axis.

Fig 40. Touch screen controller AUTO mode timing diagram

11.1.4 Touch Screen Register Descriptions

Table 234 shows the registers that are used to configure and operate the touch screen controller.

Table 234. Touchscreen registers

Address	Register name	Function	Reset State	Attributes
0x4000 40B4	ADCLK_CTRL	ADC clock control	0	R/W
0x4000 4060	ADCLK_CTRL1	Second ADC clock control register	0	R/W
0x4004 8000	ADC_STAT	A/D Status register	0x80	R
0x4004 8004	ADC_SELECT	A/D Select State register	0x04	R/W
0x4004 8008	ADC_CTRL	A/D Control Register	0x0000	R/W
0x4004 800C	TSC_SAMPLE_FIFO	Touchscreen Sample FIFO	-	R
0x4004 8010	TSC_DTR	Touchscreen Delay Time Register	0x00000	R/W
0x4004 8014	TSC_RTR	Touchscreen Rise Time Register	0x00000	R/W
0x4004 8018	TSC_UTR	Touchscreen Update Time Register	0x00000	R/W
0x4004 801C	TSC_TTR	Touchscreen Touch Time Register	0x00000	R/W
0x4004 8020	TSC_DXP	Touchscreen Drain X Plate Time Register	0x00000	R/W
0x4004 8024	TSC_MIN_X	Touchscreen Minimum X value	0x00000	R/W
0x4004 8028	TSC_MAX_X	Touchscreen Maximum X value	0x00000	R/W
0x4004 802C	TSC_MIN_Y	Touchscreen Minimum X value	0x00000	R/W
0x4004 8030	TSC_MAX_Y	Touchscreen Maximum Y value	0x00000	R/W
0x4004 8034	TSC_AUX_UTR	Touchscreen AUX Update Time Register	0x00000	R/W
0x4004 8038	TSC_AUX_MIN	Touchscreen AUX Minimum value	0x00000	R/W
0x4004 803C	TSC_AUX_MAX	Touchscreen AUX Maximum value	0x00000	R/W
0x4004 8044	TSC_AUX_VALUE	Touchscreen Value Register	0x00000	R/-
0x4004 8048	ADC_VALUE	A/D Data Register	0x00000	R/-

11.1.4.1 ADC Clock Control register (ADCLK_CTRL - 0x4000 40B4)

The ADCLK_CTRL register allows enabling or disabling the clock to the Analog to Digital Converter and Touch Screen.

Table 235. ADC Clock Control register (ADCLK_CTRL - 0x4000 40B4)

Bit	Function	Reset value
0	0 = Disable 32 kHz clock to ADC block. 1 = Enable clock.	0

11.1.4.2 ADC Clock Control1 register (ADCLK_CTRL1 - 0x4000 4060)

The ADCLK_CTRL1 register controls switching the source of the ADC/Touch Screen clock. If the PERIPH_CLK is selected, use the clock divider to reduce the input frequency of the peripheral clock to a value less than 400KHz before using the ADC or touch screen.

Table 236. ADC Clock Control register (ADCLK_CTRL1 - 0x4000 4060)

Bit	Function	Reset value
8	ADCCLK_SEL; ADC clock select 0 = Clock ADC and touch screen from RTC clock. 1 = Clock ADC and touch screen from PERIPH_CLK clock.	0
7:0	ADC_FREQ. Controls the clock divider for ADC when Peripheral clock (bit 8) is enabled. Value in register is one less than divide value. reg value = (divider -1) 00000000 = 1 00000001 = 2 …… 11111110 = 255 11111111 = 256	0

11.1.4.3 A/D Status Register (ADC_STAT - 0x4004 8000)

The ADC_STAT register contains information about the current status of the A/D Converter. The function and description of bits in ADC_STAT are shown in Table 237

Table 237. A/D Status Register (ADC_STAT - 0x4004 8000)

Bits	Function	Description	Reset value
31:9	Reserved	Reserved. The value read from a reserved bit is not defined.	-
8	TS FIFO_OVERRUN	Overrun status on TS FIFO 0 = FIFO not over run 1 = FIFO over run	-
7	TS FIFO_EMPTY	Empty status on TS FIFO 0 = FIFO not empty 1 = FIFO empty	-
6:0	Reserved	Reserved. The value read from a reserved bit is not defined.	0x0

11.1.4.4 A/D Select State Register (ADC_SELECT - 0x4004 8004)

The ADC_SELECT register provides a means of selecting an A/D channel or configure the touchscreen connections to be used for the next conversion. The function of bits in ADC_SELECT are shown in Table 238.

The settings in the ADC_SELECT register do not affect the analog touchscreen interface when the AUTO_EN bit in the ACD_CTRL register is set to logic 1.

Table 238. A/D Select Register (ADC_SELECT - 0x4004 8004)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:8	TS_Ref- (AD_Ref–)	Selects the A/D negative reference voltage, Vref- in TS ADC 00 = TS_XM 01 = TS_YM 10 = Vss 11 = Not used	0
7:6	TS_Ref+ (AD_Ref+)	Selects the A/D positive reference voltage, Vref+ in TS ADC 00 = TS_XP 01 = TS_YP 10 = VDDTS 11 = Not used	0
5:4	TS_IN (AD_IN)	Selects the input TS ADC as follows: 00 = TS_YM 01 = TS_XM 10 = TS_AUX 11 = Not used	0
3	TS_YMC	YM Control 0 = the TS_YM signal is disconnected from GND. 1 = the TS_YM signal is connected to GND.	0
2	TS_YPC	YP Control 0 = the TS_YP signal is connected to VddTS. 1 = the TS_YP signal is disconnected from VddTS.	1
1	TS_XMC	XM Control 0 = the TS_XM signal is disconnected from GND. 1 = the TS_XM signal is connected to GND.	0
0	TS_XPC	XP Control 0 = the TS_XP signal is disconnected from VddTS. (Externally pulled down by 100k resistor.) (Typical) 1 = the TS_XP signal is connected to VddTS (only if TS_XMC bit is also 1).	0

11.1.4.5 A/D Control register (ADC_CTRL - 0x4004 8008)

The ADC_CTRL register contains bits that control the power state of the A/D, start an A/D conversion. The function of bits in ADC_CTRL are shown in Table 239.

Table 239. A/D Control Register (ADC_CTRL - 0x4004 8008)

Bits	Function	Description	Reset value
31:13	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
12:11	TS_FIFO_CTRL	FIFO Control These bits set the level in the FIFO to trig the TSC_IRQ 00 = Send TSC_IRQ on FIFO level 1 01 = Send TSC_IRQ on FIFO level 4 10 = Send TSC_IRQ on FIFO level 8 11 = Send TSC_IRQ on FIFO level 16	0
10	TS_AUX_EN	This bit enables the AUX ANALOG measure in auto mode 0 = The AUX measured controller is disabled. 1 = The AUX measured controller is enabled.	0
9:7	TS_X_ACC	These bits sets the number of bits delivered by the ADC for all modes doing X direction measurement. (AUTO mode only). Fewer ADC bits used means fewer clocks to the ADC and faster acquire time. Note that the MSB bits used will stay in the same bit position in all registers. (They are not shifted down) 000 = ADC delivers 10 bits 001 = ADC delivers 9 bits ... 111 = ADC delivers 3 bits	0
6:4	TS_Y_ACC	These bits set the number of bits delivered by the ADC for all modes doing Y direction measurement. See description for TS_X_ACC.	0
3	TS_POS_DET	This bit has no effect if AUTO_EN = 0 0 = Normal auto mode. 1 = Auto mode including position detect.	0
2	TS_ADC_PDN_CTRL	This bit has no effect if AUTO_EN = 1 0 = the ADC is in power down. (Default) 1 = the ADC is powered up and reset.	0
1	TS_ADC_STROBE	This bit has no effect if AUTO_EN = 1 Setting this bit to logic 1 will start an AD conversion. This bit is write only	0
0	TS_AUTO_EN	0 = The touch screen controller is disabled. The touch screen must be operated in manual mode. 1 = The touch screen controller is enabled. Bits 2:1 in this register are don't care when AUTO_EN is 1.	0

11.1.4.6 Touchscreen controller sample FIFO register (TSC_SAMPLE_FIFO - 0x4004 800C)

This register reads the first value in the TSC sample FIFO, it contains both the X and Y values and whether the screen is pressed or not pressed. If the screen is not pressed, the ADC may not have completed a conversion before the hardware abandons sample collection. In this case, do not to use the last sample.

Note, if the FIFO is full, then new samples will be thrown away! This event will set the overrun bit 8 in the ADC_STAT register.

The register bit fields for TSC_SAMPLE_FIFO are shown in Table 240.

Table 240. Touchscreen sample FIFO Register (TSC_SAMPLE_FIFO - 0x4004 800C)

Bits	Function	Description	Reset value
31	TSC_P_LEVEL	0 = the touch screen is pressed. 1 = the touch screen is not pressed.	0
30	FIFO_EMPTY	0 = FIFO not empty 1 = FIFO empty	0
29	FIFO_OVERRUN	0 = FIFO not over run 1 = FIFO over run	0
29:26	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
25:16	TS_X_VALUE	TS_X_VALUE The ADC value of the X co-ordinate.	0
15:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
9:0	ADC_VALUE	TS_Y_VALUE The ADC value of the Y co-ordinate.	0

11.1.4.7 Touchscreen controller Delay Time register (TSC_DTR - 0x4004 8010)

The delay time register sets the number of clock cycles to wait after the Touchscreen is pressed before starting a sample. Any time shorter than the value stored in this register is ignored by the touchscreen controller. The bit fields in TSC_DTR are shown in Table 241.

The delay time is expressed in number of TOUCH_CLK clock cycles.

Table 241. Touchscreen controller Delay Time Register (TSC_TDTR - 0x4004 8010)

Bits	Function	Description	Reset value
31:20	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
19:0	TSC_DTR	TS Controller Delay Time Register	0

11.1.4.8 Touchscreen controller Rise Time register (TSC_RTR - 0x4004 8014)

The rise time sets the time required for conditions to be stable enough to measure the voltages in the physical TS interface. The bit fields in the TSC_DTR register are shown in Table 242.

The rise time is expressed in number of TOUCH_CLK clock cycles.

Table 242. Touchscreen controller Rise Time Register (TSC_RTR - 0x4004 8014)

Bits	Function	Description	Reset value
31:20	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
19:0	TSC_RTR	Touch Screen Controller rise time	0

11.1.4.9 Touchscreen controller Update Time register (TSC_UTR - 0x4004 8018)

The update time register defines the time interval between touch screen scans made by the TS controller.

The update time is expressed in TOUCH_CLK clock cycles. The fields in TSC_UTR are shown in Table 243.

Table 243. Touchscreen controller Update Time Register (TSC_UTR - 0x4004 801C)

Bits	Function	Description	Reset value
31:20	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
19:0	TSC_UTR	Touch Screen Controller update time.	0

11.1.4.10 Touchscreen controller Touch Time register (TSC_TTR - 0x4004 801C)

The touch time register sets the time that the touchscreen controller waits in the touch detect state before the TSC_P signal is tested in AUTO mode. The time is expressed in TOUCH_CLK clock cycles. The fields in TSC_TTR are shown in Table 244.

Table 244. Touchscreen controller Delay Time Register (TSC_TTR - 0x4004 801C)

Bits	Function	Description	Reset value
31:20	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
19:0	TSC_TTR	TS Controller touch time.	0

11.1.4.11 Touchscreen controller Drain X Plate Time Register (TSC_DXP - 0x4004 8020)

The touch time register sets the time the Drain X plate state is maintained. The time is expressed in TOUCH_CLK clock cycles.The fields in TSC_DXP are shown in Table 245.

Table 245. Touchscreen controller Drain X Plate Time Register (TSC_TTR - 0x4004 8020)

Bits	Function	Description	Reset value
31:20	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
19:0	TSC_DXP	TS Controller Drain X Plate time.	0

11.1.4.12 Touchscreen controller Minimum X value Register (TSC_MIN_X - 0x4004 8024)

This register is only used in Auto-position mode. (ADC_CTRL [3:0] = 1001). The register defines the minimum X value to accept from the ADC. The fields in TSC_MIN_X are shown in Table 246.

Table 246. Touchscreen controller Minimum X value Register (TSC_MIN_X - 0x4004 8024)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_MIN_X	TS Controller Minimum X value.	0

11.1.4.13 Touchscreen controller Maximum X value Register (TSC_MAX_X - 0x4004 8028)

This register is only used in Auto-position mode. (ADC_CTRL [3:0] = 1001). The register defines the maximum X value to accept from the ADC. The fields in TSC_MAX_X are shown in Table 247.

Table 247. Touchscreen controller Maximum X value Register (TSC_MIN_X - 0x4004 8028)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_MAX_X	TS Controller Maximum X value.	0

11.1.4.14 Touchscreen controller Minimum Y value Register (TSC_MIN_Y - 0x4004 802C)

This register is only used in Auto-position mode. (ADC_CTRL [3:0] = 1001). The register defines the minimum Y value to accept from the ADC. The fields in TSC_MIN_Y are shown in Table 248.

Table 248. Touchscreen controller Minimum Y value Register (TSC_MIN_Y - 0x4004 802C)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_MIN_Y	TS Controller Minimum Y value.	0

11.1.4.15 Touchscreen controller Maximum Y value Register (TSC_MAX_X - 0x4004 8030)

This register is only used in Auto-position mode. (ADC_CTRL [3:0] = 1001). The register defines the maximum Y value to accept from the ADC. The fields in TSC_MAX_Y are shown in Table 249.

Table 249. Touchscreen controller Maximum X value Register (TSC_MIN_X - 0x4004 8030)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_MAX_X	TS Controller Maximum X value.	0

11.1.4.16 Touchscreen controller AUX Update Time Register (TSC_AUX_UTR - 0x4004 8034)

This register defines the time interval between scans of the AUX_IN made by the Touch controller when ADC_CTRL[10] = 1.

This register is only used in AUTO mode. (ADC_CTRL [0] = 1). The update time is expressed in TOUCH_CLK clock cycles.

The fields in TSC_AUX_UTR are shown in Table 250.

Table 250. Touchscreen controller AUX Update Time Register (TSC_AUX_UTR - 0x4004 8034)

Bits	Function	Description	Reset value
31:0	TSC_AUX_UTR	TS Controller auxiliary update time.	0

11.1.4.17 Touchscreen controller AUX Minimum value Register (TSC_AUX_MIN - 0x4004 8038)

This register is only used in ANALOG mode. (ADC_CTRL [10] = 1). The register defines the minimum Analog value to accept from the ADC. This register is only used in Auto-position mode. (ADC_CTRL [3:0] = 1001). The fields in TSC_AUX_MIN are shown in Table 251.

Table 251. Touchscreen controller AUX Minimum value Register (TSC_AUX_MIN - 0x4004 8038)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_AUX_MIN	TS Controller Minimum auxiliary ADC value.	0

11.1.4.18 Touchscreen controller AUX Maximum value Register (TSC_AUX_MAX - 0x4004 803C)

This register is only used in ANALOG mode. (ADC_CTRL [10] = 1). The register defines the maximum Analog value to accept from the ADC. This register is only used in Auto-position mode. (ADC_CTRL [3:0] = 1001). The register defines the maximum AUX value to accept from the ADC. The fields in TSC_AUX_MAX are shown in Table 252.

Table 252. Touchscreen controller AUX Maximum value Register (TSC_AUX_MAX - 0x4004 803C)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_AUX_MAX	TS Controller Maximum Auxiliary ADC value.	0

11.1.4.19 Touchscreen controller AUX Value Register (TSC_AUX_VALUE - 0x4004 8044)

This register is only used in ANALOG mode. (ADC_CTRL [10] = 1). The register reads the Analog value from the AUX ADC. The fields in TSC_AUX_VALUE are shown in Table 253.

Table 253. Touchscreen controller AUX Maximum value Register (TSC_AUX_VALUE - 0x4004 8044)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	TSC_AUX_VALUE	TS Controller Auxiliary ADC value.	0

11.1.4.20 Touchscreen controller ADC Value Register (ADC_VALUE - 0x4004 8048)

This register is only used in ANALOG mode. (ADC_CTRL [10] = 1). The register reads the Analog value from the ADC. The fields in ADC_VALUE are shown in Table 254. Reading this bit clears all TS interrupts.

The TSC_P_LEVEL bit in this register is not a duplicate of the TSC_P_LEVEL bit in the TS_SAMPLE_FIFO register. The TSC_P_LEVEL bit in ADC_VALUE is registered, while the TSC_P_LEVEL bit in TSC_SAMPLE_FIFO register is raw data.

Table 254. Touchscreen controller ADC Value Register (TSC_ADC_VALUE - 0x4004 8048)

Bits	Function	Description	Reset value
31:11	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
10	TSC_P_LEVEL	0 = the touch screen is pressed. 1 = the touch screen is not pressed.	-
9:0	ADC_VALUE	The ADC value of the last conversion.	0

12.1 Introduction

The Analog-to-Digital Converter (ADC) is a 3 channel, 10-bit successive approximation converter. The ADC is a sub-component of the Touch Screen controller, and when configured as an Analog-to-Digital converter results have a resolution of 10 bits. Conversion time of the A/D converter is 11 clock times for the full 10-bit conversion. The ADC can convert one of its 3 analog input signals with a maximum conversion rate of 400 kSamples/s for a 10-bit resolution.

The analog portion of the ADC has its own power supply to enhance the low noise characteristics of the converter. This voltage should only be supplied internally when the core has voltage. However, the ADC block is not affected by any difference in ramp-up time for VDD_AD and VDD_CORE voltage supplies.

Figure 41 shows the block diagram of the A/D Converter.

12.1.1 Features

- Low noise A/D converter
- 10-bit resolution
- Three input channels
- 400 kSamples/s maximum conversion rate

12.2 Pin description

Table 255. A/D pin description

Pin name	Type	Description
ADIN0/TS_YM	Analog Input	This pin is A/D input 0. This pin should be tied to ground if it is not used.
ADIN1/TS_XM	Analog Input	This pin is A/D input 1. This pin should be tied to ground if it is not used.
ADIN2/TS_AUX_IN	Analog Input	This pin is A/D input 2. This pin should be tied to ground if it is not used.
VDD_AD	Power	This is the VDD supply for the ADC, also acting as the positive reference voltage.
VSS_AD	Power	This is the VSS supply for the ADC, also acting as the negative reference voltage.

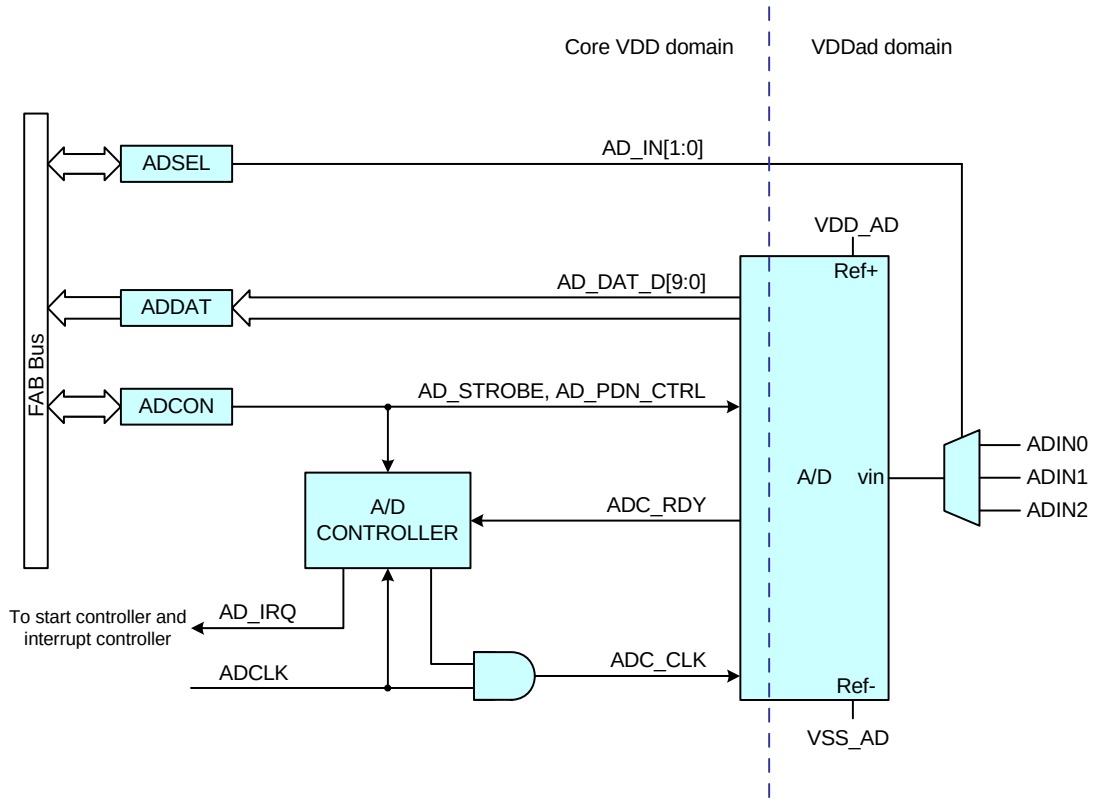


Fig 41. Simplified Block diagram of the Touch screen controller used as an ADC

12.3 Register description

Table 256 shows the registers associated with the A/D Converter and a summary of their functions. Following the table are details for each register.

Table 256. A/D registers

Address offset	Name	Description	Reset value	Type
0x4000 40B4	ADCLK_CTRL	ADC clock control	0	R/W
0x4000 4060	ADCLK_CTRL1	Second ADC clock control register	0	R/W
0x4004 8004	ADC_SELECT	A/D Select Register	0x04	R/W
0x4004 8008	ADC_CTRL	A/D Control Register	0x0000	R/W
0x4004 8048	ADC_VALUE	A/D Data Register	0x00000	R/-

12.3.1 ADC Clock Control register (ADCLK_CTRL - 0x4000 40B4)

The ADCLK_CTRL1 register controls switching the source of the ADC/Touch Screen clock. If the PERIPH_CLK is selected, use the clock divider to reduce the input frequency of the peripheral clock to a value less than or equal to 4.5Mhz before using the ADC or touch screen. One 10-bit conversion needs 11 clock cycles. The maximum conversion

rate of 10-bit samples is 4.5M / 11 or ~400KS/s. During the first clock cycle, the selected input signal is sampled. After the sampling is done the successive approximation algorithm determines the output code during the following 10 clock cycles.

Table 257. ADC Clock Control register (ADCLK_CTRL - 0x4000 40B4)

Bit	Function	Function	Reset value
0	ADC_CLK_EN	ADC clock enable 0 = Disable 32 kHz (RTC) clock to ADC block. 1 = Enable clock.	0

12.3.2 ADC Clock Control1 register (ADCLK_CTRL1 - 0x4000 4060)

The ADCLK_CTRL1 register controls switching the source of the ADC/Touch Screen clock. If the PERIPH_CLK is selected, use the clock divider to reduce the input frequency of the peripheral clock to a value less than 4.5 MHz before using the ADC or touch screen.

Table 258. ADC Clock Control register (ADCLK_CTRL1 - 0x4000 4060)

Bit	Function	Description	Reset value
8	ADCCLK_SEL	ADC clock select 0 = Clock ADC and touch screen from RTC clock. 1 = Clock ADC and touch screen from PERIPH_CLK clock.	0
7:0	ADC_FREQ.	ADC Frequency Controls the clock divider for ADC when Peripheral clock (bit 8) is enabled. Value in register is one less than divide value. reg value = (divider -1) 00000000 = 1 00000001 = 2 …… 11111110 = 255 11111111 = 256	0

12.3.3 A/D Select Register (ADC_SELECT - 0x4004 8004)

The ADC_SELECT register provides a means of selecting an A/D channel to be used for the next conversion. Other bits in ADC_SELECT control internal A/D functions and must be set to the values indicated for proper A/D operation. The function of bits in ADC_SELECT are shown in Table 259.

Table 259. A/D Select Register (ADC_SELECT - 0x4004 8004)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:8	AD_Ref–(TS_Ref-)	Selects the A/D negative reference voltage. Must be set to 10 if ADC is used (VSS_AD). Settings 11, 01, and 00 are undefined. Do not use.	00

Table 259. A/D Select Register (ADC_SELECT - 0x4004 8004)

Bits	Function	Description	Reset value
7:6	AD_Ref+ (TS_Ref+)	Selects the A/D positive reference voltage. Must be set to 10 if ADC is used (VDD_AD). Settings 11, 01, and 00 are undefined. Do not use.	00
5:4	AD_IN (TS_IN)	Selects the A/D input as follows: 00 - ADIN0 01 - ADIN1 10 - ADIN2 11 - Not used	00
3:0	-	A/D internal controls. Must not be changed from the reset value.	0x4

12.3.4 A/D Control register (ADC_CTRL - 0x4004 8008)

The ADC_CTRL register contains bits that control the power state of the A/D, start an A/D conversion. The function of bits in ADC_CTRL are shown in Table 260.

Table 260. A/D Control Register (ADC_CTRL - 0x4004 8008)

Bits	Function	Description	Reset value
31:7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
6:3	-	Internal A/D controls. Must be set to 0x0.	0x0
2	AD_PDN_CTRL (TS_ADC_PDN_CTRL)	0 = the ADC is in power down. 1 = the ADC is powered up and reset.	0
1	AD_STROBE (TS_ADC_STROBE)	Setting this bit to logic 1 will start an A/D conversion. The bit is reset by hardware when the A/D conversion has started.	0
0	-	Internal A/D control. Must be set to 0.	0

12.3.5 ADC Value register (ADC_VALUE - 0x4004 8048)

The ADC_VALUE register contains the result of the last completed A/D conversion. The result field in ADC_VALUE is shown in Table 261.

Table 261. A/D Data Register (ADC_VALUE - 0x4004 8048)

Bits	Function	Description	Reset value
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
9:0	ADC_VALUE	The ADC value of the last conversion.	

12.4 A/D conversion sequence

The following is an example sequence of setting up the ADC, starting a conversion, and acquiring the result value.

- Write a value to the AD_IN field of the ADSEL register to select the desired A/D channel to convert. Make sure to include the required values of other fields in the register.
- Wait for an A/D interrupt signal from AD_IRQ (see Figure 41), or poll the raw interrupt bit 7 in the SIC1_RSR register to determine when the conversion is complete.
- Read the conversion result in the ADC_VALUE register, which will also clear the ADC_INT interrupt.

13.1 Features

- Supports up to 64 keys in 8 × 8 matrix.
- Programmable debounce period.
- A key press can wake up the CPU from stop mode.

13.2 Functional description

13.2.1 Clocking

The Keyboard Scan interface has two clock domains, a 32 kHz (RTC_CLK) domain for scan functionality and a PERIPH_CLK domain for the FAB bus interface including registers. To ensure proper function, the 32 kHz clock should always run.

To wake up the CPU from stop mode on a 'key pressed', a start signal is issued via the NKEY_IRQ signal. This is achieved without the APB bus or PERIPH_CLK active.

13.2.2 Multiplexing of pins

To be able to use a full 8 × 8 matrix, the GPIO_3 and GPIO_2 pins must be connected to Row[7:6]. This is performed by setting the appropriate bits in the P2_MUX_SET register.

13.2.3 Keyboard scan operation

When the internal state machine is in 'Idle state', all KEY_ROW[n] pins are set to 'high' waiting for a key (or multiple keys) to be pressed. 'Key pressed' is detected as a 'high' on the respective KEY_COL[n] input pin. The matrix is scanned by setting one output pin 'high' at a time and then reading all inputs. After a pre-programmed de-bounce period (n identical matrix values are read) the keypad state is stored in the matrix registers (KS_DATAn[7:0]) and an interrupt request is sent to the interrupt controller. The keypad is then continuously scanned waiting for 'extra key pressed' or 'key released'. Any new keypad state is scanned and stored into the matrix registers followed by a new interrupt request to the interrupt controller.

It is possible to detect and separate up to 64 multiple keys pressed. It is possible to read the KEY_COL[n] inputs directly via the FAB bus. The internal de-bounce logic will then be inactive.

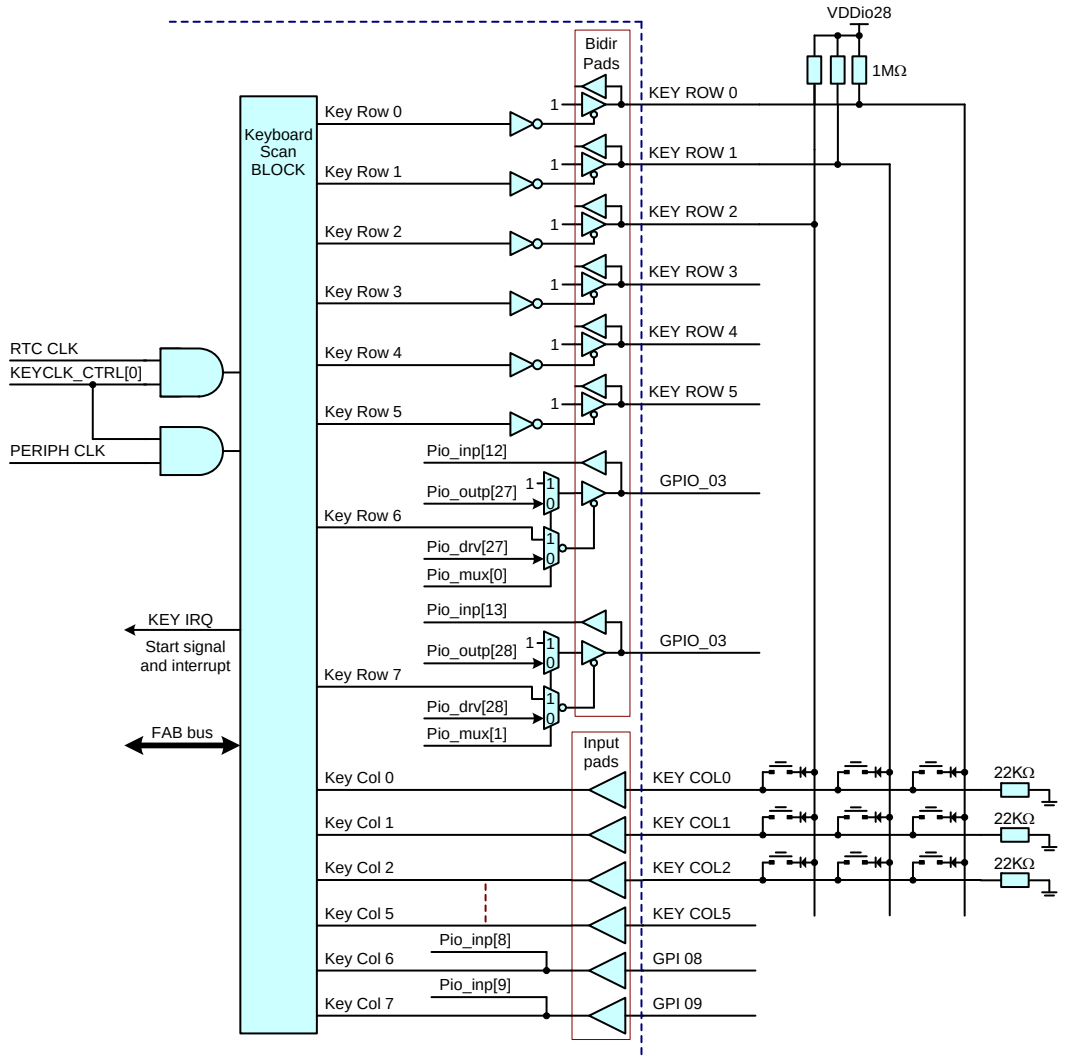


Fig 42. Keyboard scan 8 × 8 block diagram. (Only a 3 × 3 external key matrix is shown).

13.3 Register description

Table 262 shows the registers associated with the Keyboard Scan and a summary of their functions. Following the table are details for each register.

Table 262. Keyboard scan registers

Address	Name	Description	Reset value	Access
0x4005 0000	KS_DEB	Keypad de-bouncing duration register	0x05	R/W
0x4005 0004	KS_STATE_COND	Keypad state machine current state register	0x00	RO
0x4005 0008	KS_IRQ	Keypad interrupt register	0x01	R/W
0x4005 000C	KS_SCAN_CTL	Keypad scan delay control register	0x05	R/W
0x4005 0010	KS_FAST_TST	Keypad scan clock control register	0x02	R/W
0x4005 0014	KS_MATRIX_DIM	Keypad Matrix Dimension select register	0x06	R/W

Table 262. Keyboard scan registers

Address	Name	Description	Reset value	Access
0x4005 0040	KS_DATA0	Keypad data register 0	0x00	RO
0x4005 0044	KS_DATA1	Keypad data register 1	0x00	RO
0x4005 0048	KS_DATA2	Keypad data register 2	0x00	RO
0x4005 004C	KS_DATA3	Keypad data register 3	0x00	RO
0x4005 0050	KS_DATA4	Keypad data register 4	0x00	RO
0x4005 0054	KS_DATA5	Keypad data register 5	0x00	RO
0x4005 0058	KS_DATA6	Keypad data register 6	0x00	RO
0x4005 005C	KS_DATA7	Keypad data register 7	0x00	RO

13.3.1 Keypad De-bouncing Duration register (KS_DEB, RW - 0x4005 0000)

Table 263. Keypad De-bouncing Duration register (KS_DEB, RW - 0x4005 0000)

Bits	Description	Reset value
7:0	Keypad de-bouncing duration. Number of equal matrix values to be read plus 3. This register specifies the number of passes (plus 3) that are used to debounce a matrix of size 1 to 8. Each column read of a matrix takes exactly 1 clock in SCANONCE mode and 2 clocks in SCANMATRIX mode. 0x02 => Debounce completes after 2 + 3 = 5 equal matrix values are read 0xFF => Debounce completes after 256 + 3 = 259 equal matrix values are read	0x5

13.3.2 Keypad State Machine Current State register (KS_STATE_COND, RO - 0x4005 0004)

Table 264. Keypad State Machine Current State register (KS_STATE_COND, RO - 0x4005 0004)

Bits	Name	Description	Reset value
1:0	STATE	00: Idle 01: Scan Once 10: IRQ generation 11: Scan Matrix	0x0

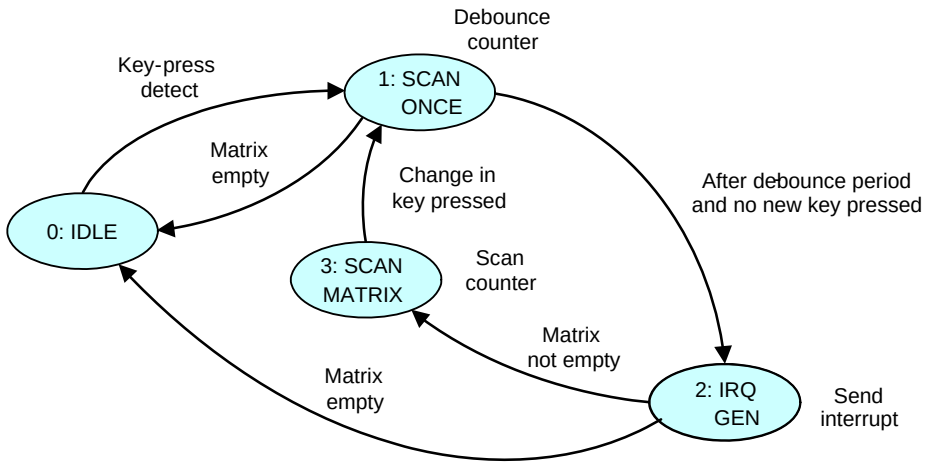


Fig 43. Keyboard scan state diagram

13.3.3 Keypad Interrupt register (KS_IRQ, RW - 0x4005 0008)

Table 265. Keypad Interrupt register (KS_IRQ, RW - 0x4005 0008)

Bits	Name	Description	Reset value
0	KIRQN	0: Active interrupt: Key pressed or released. Any write access to this register will clear the interrupt. In polling mode, this bit needs to be reset after a key has been pressed. 1: No active interrupt.	0x1

13.3.4 Keypad Scan Delay Control register (KS_SCAN_CTL, RW - 0x4005 000C)

Table 266. Keypad Scan Delay Control register (KS_SCAN_CTL, RW - 0x4005 000C)

Bits	Name	Description	Reset value
7:0	SCN_CTL	Time between each keypad scan in STATE: 'Scan Matrix'. Time between each scan = (1 / clock_freq) × 32 × SCN_CTL 32KHz clock source: SCN_CTL = 0x00 => Scan always SCN_CTL = 0x01 => (1 / 32KHz) × 32 × 1 = 1 ms (32 clock cycles × 1) SCN_CTL = 0xFF => (1 / 32KHz) × 32 × 256 = 250 ms (32 clock cycles × 256) (default)	0xFF

13.3.5 Keypad Scan Clock Control register (KS_FAST_TST, RW - 0x4005 0010)

Table 267. Keypad Scan Clock Control register (KS_FAST_TST, RW - 0x4005 0010)

Bits	Description	Reset value
0	0: No forced Jump (default). 1: Jump to STATE: Scan Once.	0
1	0: PCLK (do **not** use in this configuration) 1: Use the 32 KHz RTC clock as the clock source (default).	1

13.3.6 Keypad Matrix Dimension Select register (KS_MATRIX_DIM, RW - 0x4005 0014)

Table 268. Keypad Matrix Dimension Select register (KS_MATRIX_DIM, RW - 0x4005 0014)

Bits	Name	Description	Reset value
3:0	MX_DIM	0x01 => 1 x 1 matrix dimension 0x06 => 6 x 6 matrix dimension (default) 0x08 => 8 x 8 matrix dimension (Max)	0x06

13.3.7 Keypad Data Register 0 (KS_DATA0, RO - 0x4005 0040)

Table 269. Keypad Data Register 0 (KS_DATA0, RO - 0x4005 0040)

Bits	Description	Reset value
7:0	KEY_R0_C<7:0> Image of Column <7:0> on Row 0. Captured on Row 0 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 0) 0: Column <7:0> = ‘low’ (no key pressed on Row 0)	0x0

13.3.8 Keypad Data Register 1 (KS_DATA1, RO - 0x4005 0044)

Table 270. Keypad Data Register 1 (KS_DATA1, RO - 0x4005 0044)

Bits	Description	Reset value
7:0	KEY_R1_C<7:0> Image of Column <7:0> on Row 1. Captured on Row 1 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 1) 0: Column <7:0> = ‘low’ (no key pressed on Row 1)	0x0

13.3.9 Keypad Data Register 2 (KS_DATA2, RO - 0x4005 0048)

Table 271. Keypad Data Register 2 (KS_DATA2, RO - 0x4005 0048)

Bits	Description	Reset value
7:0	KEY_R2_C<7:0> Image of Column <7:0> on Row 2. Captured on Row 2 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 2) 0: Column <7:0> = ‘low’ (no key pressed on Row 2)	0x0

13.3.10 Keypad Data Register 3 (KS_DATA3, RO - 0x4005 004C)

Table 272. Keypad Data Register 3 (KS_DATA3, RO - 0x4005 004C)

Bits	Description	Reset value
7:0	KEY_R3_C<7:0> Image of Column <7:0> on Row 3. Captured on Row 3 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 3) 0: Column <7:0> = ‘low’ (no key pressed on Row 3)	0x0

13.3.11 Keypad Data Register 4 (KS_DATA4, RO - 0x4005 0050)

Table 273. Keypad Data Register 4 (KS_DATA4, RO - 0x4005 0050)

Bits	Description	Reset value
7:0	KEY_R4_C<7:0> Image of Column <7:0> on Row 4. Captured on Row 4 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 4) 0: Column <7:0> = ‘low’ (no key pressed on Row 4)	0x0

13.3.12 Keypad Data Register 5 (KS_DATA5, RO - 0x4005 0054)

Table 274. Keypad Data Register 5 (KS_DATA5, RO - 0x4005 0054)

Bits	Description	Reset value
7:0	KEY_R5_C<7:0> Image of Column <7:0> on Row 5. Captured on Row 5 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 5) 0: Column <7:0> = ‘low’ (no key pressed on Row 5)	0x0

13.3.13 Keypad Data Register 6 (KS_DATA6, RO - 0x4005 0058)

Table 275. Keypad Data Register 6 (KS_DATA6, RO - 0x4005 0058)

Bits	Description	Reset value
7:0	KEY_R6_C<7:0> Image of Column <7:0> on Row 6. Captured on Row 6 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 6) 0: Column <7:0> = ‘low’ (no key pressed on Row 6)	0x0

13.3.14 Keypad Data Register 7 (KS_DATA7, RO - 0x4005 005C)

Table 276. Keypad Data Register 7 (KS_DATA7, RO - 0x4005 005C)

Bits	Description	Reset value
7:0	KEY_R0_C<7:0> Image of Column <7:0> on Row 7. Captured on Row 7 ‘high’ 1: Column <7:0> = ‘high’ (key pressed on Row 7) 0: Column <7:0> = ‘low’ (no key pressed on Row 7)	0x0

13.4 Example timing for Keyscan matrix

The following Equations can be used to calculate keyscan timing.

Equation 10 shows timing calculation for Scan once mode

$$\text{ScanOnce Time} = (1/32\text{K}) \times ((\text{MX\_DIM} \times (\text{KS\_DEB} + 3)) \quad (10)$$

Equation 11 shows timing calculation for Scan Matrix mode

$$\text{ScanMatrix Time} = (1/32\text{K}) \times (2 \times \text{MX\_DIM} \times (\text{KS\_DEB} + 3)) \quad (11)$$

13.4.1 Timing example for 4 x 4 matrix keyscan

For the following conditions:

Clock — 32.768 KHz

Number of Rows — MX_DIM = 4 (matrix size)

Debounce Duration — KS_DEB = 5

Scan Delay — SCN_CTL = 1

The first scan pass is 1.95 ms.

Scan Once + Scan delay = (1/32768) x (4 x (5 + 3)) + ((1/32768) x (1 x 32)) = 1.95 ms

Each subsequent pass through the scan matrix is 2.92 ms.

Scan Matrix + Scan delay = (1/32768) x (2 x 4 x (5 + 3)) + ((1/32768) x (1 x 32)) = 2.92 ms

14.1 Introduction

The Ethernet interface 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 connected to the AHB matrix and has access to internal SRAM (IRAM) and memory connected to the External Memory Controller for Ethernet data, control, and status information. Other AHB traffic in the LPC32x0 can take place using other masters, effectively separating Ethernet activity from the rest of the system.

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.

Remark: The Ethernet controller is available on LPC3240 and LPC3250 parts only.

Table 277. 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.
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

Table 277. Ethernet acronyms, abbreviations, and definitions

Acronym or Abbreviation	Definition
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

14.2 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 IRAM or EMC memory.
  - DMA managers with scatter/gather DMA and arrays of frame descriptors.
  - Memory traffic optimized by buffering and pre-fetching.
- 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 back-off 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.

14.3 Architecture and operation

Figure 44 shows the internal architecture of the Ethernet block.

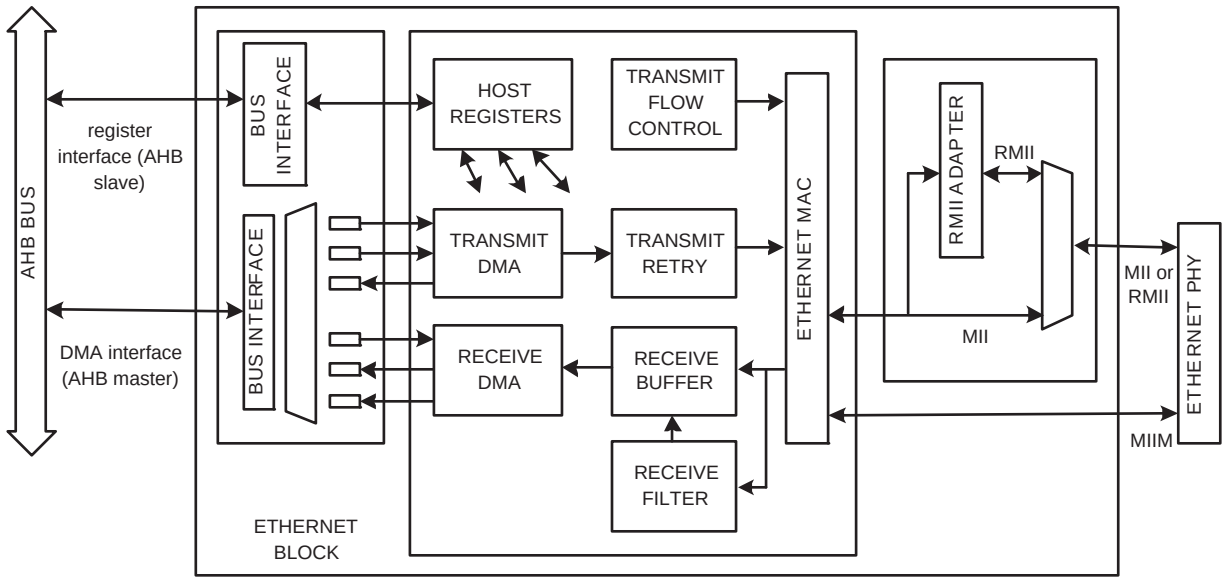


Fig 44. Ethernet block diagram

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 datapath as well as the MAC.
- The DMA to AHB interface. This provides an AHB master connection that allows the Ethernet block to access IRAM or EMC memory for reading of descriptors, writing of status, and reading and writing data buffers.
- The Ethernet MAC and attached RMII adapter. The MAC interfaces to the off-chip PHY.
- The transmit datapath, 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 datapath, 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.

14.4 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 LPC32x0.

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.

14.5 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.

14.6 Ethernet Packet

Figure 45 illustrates the different fields in an Ethernet packet.

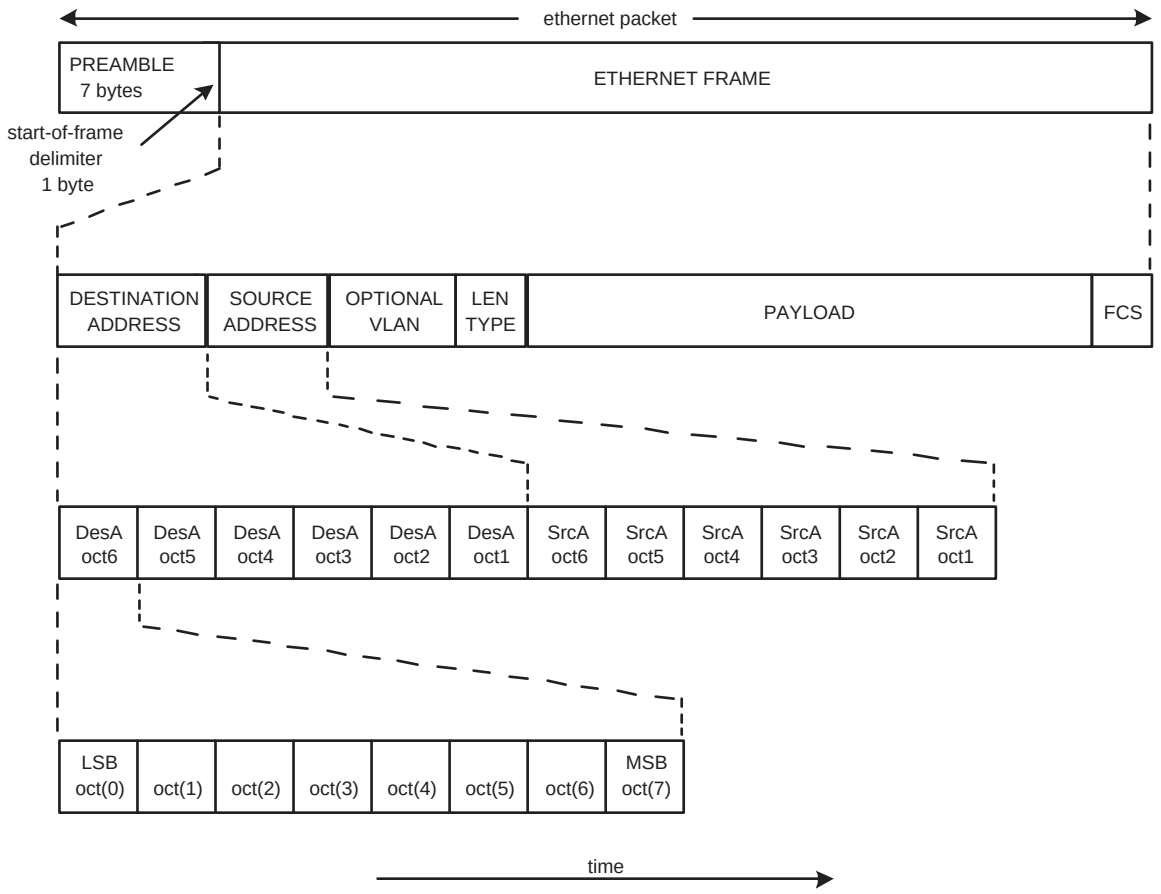


Fig 45. Ethernet packet fields

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.

14.7 Overview

14.7.1 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 pre-fetching 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 IEEE 802.3/clause 22 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 MII bus (MIIM).

14.7.2 Example PHY Devices

Some examples of compatible PHY devices are shown in Table 278.

Table 278. Example PHY Devices

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

14.8 Pin description

Table 279 shows the signals used for connecting the Media Independent Interface (MII), and Table 280 shows the signals used for connecting the Reduced Media Independent Interface (RMII) to the external PHY.

Remark: The Ethernet interface must be configured by enabling the clocks **and** connecting the Ethernet pins to port pins using the MAC_CLK_CTRL register (see Table 642 and Table 643). Enabling clocks without connecting the Ethernet signals to external pins will lock the Ethernet interface and, in Debug mode, cause JTAG to lose communication with the target.

Table 279. Ethernet MII pin descriptions

Pin Name	Type	Pin Description
ENET_TX_EN	Output	Transmit data enable.
ENET_TXD[3:0]	Output	Transmit data, 4 bits.
ENET_TX_ER	Output	Transmit error.
ENET_TX_CLK	Input	Transmit clock.
ENET_RX_DV	Input	Receive data valid.
ENET_RXD[3:0]	Input	Receive data.
ENET_RX_ER	Input	Receive error.
ENET_RX_CLK	Input	Receive clock
ENET_COL	Input	Collision detect.
ENET_CRS	Input	Carrier sense.

Table 280. Ethernet RMII pin descriptions

Pin Name	Type	Pin Description
ENET_TX_EN	Output	Transmit data enable
ENET_TXD[1:0]	Output	Transmit data, 2 bits
ENET_RXD[1:0]	Input	Receive data, 2 bits.
ENET_RX_ER	Input	Receive error.
ENET_CRS	Input	Carrier sense/data valid.
ENET_REF_CLK/ ENET_RX_CLK	Input	Reference clock

Table 281 shows the signals used for Media Independent Interface Management (MIIM) of the external PHY.

Table 281. Ethernet MIIM pin descriptions

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

14.9 Registers and software interface

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.

14.9.1 Register map

Table 282 lists the registers, register addresses and other basic information. The total AHB address space consumes 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 282. Register definitions

Address	Register Type	Name	R/W	Description
	MAC Configuration			
0x4000 4090		MAC_CLK_CTRL	R/W	Controls Ethernet CLKs and Pin Multiplexing
	MAC registers			
0x3106 0000		MAC1	R/W	MAC configuration register 1.
0x3106 0004		MAC2	R/W	MAC configuration register 2.
0x3106 0008		IPGT	R/W	Back-to-Back Inter-Packet-Gap register.
0x3106 000C		IPGR	R/W	Non Back-to-Back Inter-Packet-Gap register.
0x3106 0010		CLRT	R/W	Collision window / Retry register.
0x3106 0014		MAXF	R/W	Maximum Frame register.
0x3106 0018		SUPP	R/W	PHY Support register.
0x3106 001C		TEST	R/W	Test register.
0x3106 0020		MCFG	R/W	MII Mgmt Configuration register.
0x3106 0024		MCMD	R/W	MII Mgmt Command register.

Table 282. Register definitions

Address	Register Type	Name	R/W	Description
0x3106 0028		MADR	R/W	MII Mgmt Address register.
0x3106 002C		MWTD	WO	MII Mgmt Write Data register.
0x3106 0030		MRDD	RO	MII Mgmt Read Data register.
0x3106 0034		MIND	RO	MII Mgmt Indicators register.
0x3106 0038 to 0x3106 003C		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
0x3106 0040		SA0	R/W	Station Address 0 register.
0x3106 0044		SA1	R/W	Station Address 1 register.
0x3106 0048		SA2	R/W	Station Address 2 register.
0x3106 004C to 0x3106 00FC		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
	Control Registers			
0x3106 0100		Command	R/W	Command register.
0x3106 0104		Status	RO	Status register.
0x3106 0108		RxDescriptor	R/W	Receive descriptor base address register.
0x3106 010C		RxStatus	R/W	Receive status base address register.
0x3106 0110		RxDescriptorNumber	R/W	Receive number of descriptors register.
0x3106 0114		RxProduceIndex	RO	Receive produce index register.
0x3106 0118		RxConsumeIndex	R/W	Receive consume index register.
0x3106 011C		TxDescriptor	R/W	Transmit descriptor base address register.
0x3106 0120		TxStatus	R/W	Transmit status base address register.
0x3106 0124		TxDescriptorNumber	R/W	Transmit number of descriptors register.
0x3106 0128		TxProduceIndex	R/W	Transmit produce index register.
0x3106 012C		TxConsumeIndex	RO	Transmit consume index register.
0x3106 0130 to 0x3106 0154		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
0x3106 0158		TSV0	RO	Transmit status vector 0 register.
0x3106 015C		TSV1	RO	Transmit status vector 1 register.
0x3106 0160		RSV	RO	Receive status vector register.
0x3106 0164 to 0x3106 016C		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
0x3106 0170		FlowControlCounter	R/W	Flow control counter register.
0x3106 0174		FlowControlStatus	RO	Flow control status register.
0x3106 0178 to 0x3106 01FC		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
	Rx filter Registers			
0x3106 0200		RxFliterCtrl		Receive filter control register.

Table 282. Register definitions

Address	Register Type	Name	R/W	Description
0x3106 0204		RxFilterWoLStatus		Receive filter WoL status register.
0x3106 0208		RxFilterWoLClear		Receive filter WoL clear register.
0x3106 020C		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
0x3106 0210		HashFilterL		Hash filter table LSBs register.
0x3106 0214		HashFilterH		Hash filter table MSBs register.
0x3106 0218 to 0x3106 0FDC		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
	Module control Registers			
0x3106 0FE0		IntStatus	RO	Interrupt status register.
0x3106 0FE4		IntEnable	R/W	Interrupt enable register.
0x3106 0FE8		IntClear	WO	Interrupt clear register.
0x3106 0FEC		IntSet	WO	Interrupt set register.
0x3106 0FF0		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
0x3106 0FF4		PowerDown	R/W	Power-down register.
0x3106 0FF8		-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

The forth 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.

14.10 Ethernet MAC register definitions

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

14.10.1 MAC Configuration Register 1 (MAC1 - 0x3106 0000)

The MAC configuration register 1 (MAC1) has an address of 0x3106 0000. Its bit definition is shown in Table 283.

Table 283. MAC Configuration register 1 (MAC1 - address 0x3106 0000) bit description

Bit	Symbol	Function	Reset value
0	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	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	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	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	0x0
8	RESET TX	Setting this bit will put the Transmit Function logic in reset.	0
9	RESET MCS / TX	Setting this bit resets the MAC Control Sublayer / Transmit logic. The MCS logic implements flow control.	0
10	RESET RX	Setting this bit will put the Ethernet receive logic in reset.	0
11	RESET MCS / RX	Setting this bit resets the MAC Control Sublayer / Receive logic. The MCS logic implements flow control.	0x0
13:12	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
14	SIMULATION RESET	Setting this bit will cause a reset to the random number generator within the Transmit Function.	0
15	SOFT RESET	Setting this bit will put all modules within the MAC in reset except the Host Interface.	1
31:16	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

14.10.2 MAC Configuration Register 2 (MAC2 - 0x3106 0004)

The MAC configuration register 2 (MAC2) has an address of 0x3106 0004. Its bit definition is shown in Table 284.

Table 284. MAC Configuration register 2 (MAC2 - address 0x3106 0004) bit description

Bit	Symbol	Function	Reset value
0	FULL-DUPLEX	When enabled (set to '1'), the MAC operates in Full-Duplex mode. When disabled, the MAC operates in Half-Duplex mode.	0
1	FRAME LENGTH 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	HUGE FRAME ENABLE	When enabled (set to '1'), frames of any length are transmitted and received.	0
3	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	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

Table 284. MAC Configuration register 2 (MAC2 - address 0x3106 0004) bit description

Bit	Symbol	Function	Reset value
5	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 286 - Pad Operation for details on the pad function.	0
6	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 286 - Pad Operation for more information on the various padding features. **Note:** This bit is ignored if PAD / CRC ENABLE is cleared.	0
7	AUTO DETECT PAD 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 286 - 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	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	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. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
12	NO BACKOFF	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	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	EXCESS DEFER	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. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

Table 285. 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.

14.10.3 Back-to-Back Inter-Packet-Gap Register (IPGT - 0x3106 0008)

The Back-to-Back Inter-Packet-Gap register (IPGT) has an address of 0x3106 0008. Its bit definition is shown in Table 286.

Table 286. Back-to-back Inter-packet-gap register (IPGT - address 0x3106 0008) bit description

Bit	Symbol	Function	Reset value
6:0	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).	0x0
31:7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

14.10.4 Non Back-to-Back Inter-Packet-Gap Register (IPGR - 0x3106 000C)

The Non Back-to-Back Inter-Packet-Gap register (IPGR) has an address of 0x3106 000C. Its bit definition is shown in Table 287.

Table 287. Non Back-to-back Inter-packet-gap register (IPGR - address 0x3106 000C) bit description

Bit	Symbol	Function	Reset value
6:0	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).	0x0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
14:8	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)	0x0
31:15	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

14.10.5 Collision Window / Retry Register (CLRT - 0x3106 0010)

The Collision window / Retry register (CLRT) has an address of 0x3106 0010. Its bit definition is shown in Table 288.

Table 288. Collision Window / Retry register (CLRT - address 0x3106 0010) bit description

Bit	Symbol	Function	Reset value
3:0	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. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
13:8	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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.10.6 Maximum Frame Register (MAXF - 0x3106 0014)

The Maximum Frame register (MAXF) has an address of 0x3106 0014. Its bit definition is shown in Table 289.

Table 289. Maximum Frame register (MAXF - address 0x3106 0014) bit description

Bit	Symbol	Function	Reset value
15:0	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	0x0

14.10.7 PHY Support Register (SUPP - 0x3106 0018)

The PHY Support register (SUPP) has an address of 0x3106 0018. The SUPP register provides additional control over the RMII interface. The bit definition of this register is shown in Table 290.

Table 290. PHY Support register (SUPP - address 0x3106 0018) bit description

Bit	Symbol	Function	Reset value
7:0	-	Unused	0x0
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	0x0

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

14.10.8 Test Register (TEST - 0x3106 001C)

The Test register (TEST) has an address of 0x3106 001C. The bit definition of this register is shown in Table 291. These bits are used for testing purposes only.

Table 291. Test register (TEST - address 0x3106) bit description

Bit	Symbol	Function	Reset value
0	SHORTCUT PAUSE QUANTA	This bit reduces the effective PAUSE quanta from 64 byte-times to 1 byte-time.	0
1	TEST PAUSE	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	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	0x0

14.10.9 MII Mgmt Configuration Register (MCFG - 0x3106 0020)

The MII Mgmt Configuration register (MCFG) has an address of 0x3106 0020. The bit definition of this register is shown in Table 292.

Table 292. MII Mgmt Configuration register (MCFG - address 0x3106 0020) bit description

Bit	Symbol	Function	Reset value
0	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	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
4:2	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 25 MHz, however. Refer to Table 293 below for the definition of values for this field. Note: Using a Host Clock (HCLK) of 104 MHz results in a minimum MDC clock of 3.72 Mhz.	0
14:5	-	Unused	0x0
15	RESET MII MGMT	This bit resets the MII Management hardware.	0
31:16	-	Unused	0x0

Table 293. Clock select encoding

Clock Select	Bit 4	Bit 3	Bit 2
Host Clock divided by 4	0	0	x
Host Clock divided by 6	0	1	0
Host Clock divided by 8	0	1	1
Host Clock divided by 10	1	0	0
Host Clock divided by 14	1	0	1
Host Clock divided by 20	1	1	0
Host Clock divided by 28	1	1	1

14.10.10 MII Mgmt Command Register (MCMD - 0x3106 0024)

The MII Mgmt Command register (MCMD) has an address of 0x3106 0024. The bit definition of this register is shown in Table 294.

Table 294. MII Mgmt Command register (MCMD - address 0x3106 0024) 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	0x0

14.10.11 MII Mgmt Address Register (MADR - 0x3106 0028)

The MII Mgmt Address register (MADR) has an address of 0x3106 0028. The bit definition of this register is shown in Table 295.

Table 295. MII Mgmt Address register (MADR - address 0x3106 0028) bit description

Bit	Symbol	Function	Reset value
4:0	REGISTER ADDRESS	This field represents the 5 bit Register Address field of Mgmt cycles. Up to 32 registers can be accessed.	0x0
7:5	-	Unused	0x0
12:8	PHY ADDRESS	This field represents the 5 bit PHY Address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved).	0x0
31:13	-	Unused	0x0

14.10.12 MII Mgmt Write Data Register (MWTD - 0x3106 002C)

The MII Mgmt Write Data register (MWTD) is a Write Only register with an address of 0x3106 002C. The bit definition of this register is shown in Table 296.

Table 296. MII Mgmt Write Data register (MWTD - address 0x3106 002C) bit description

Bit	Symbol	Function	Reset value
15:0	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).	0x0
31:16	-	Unused	0x0

14.10.13 MII Mgmt Read Data Register (MRDD - 0x3106 0030)

The MII Mgmt Read Data register (MRDD) is a Read Only register with an address of 0x3106 0030. The bit definition of this register is shown in Table 297.

Table 297. MII Mgmt Read Data register (MRDD - address 0x3106 0030) bit description

Bit	Symbol	Function	Reset value
15:0	READ DATA	Following an MII Mgmt Read Cycle, the 16 bit data can be read from this location.	0x0
31:16	-	Unused	0x0

14.10.14 MII Mgmt Indicators Register (MIND - 0x3106 0034)

The MII Mgmt Indicators register (MIND) is a Read Only register with an address of 0x3106 0034. The bit definition of this register is shown in Table 298.

Table 298. MII Mgmt Indicators register (MIND - address 0x3106 0034) 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	NOT VALID	When '1' is returned - indicates MII Mgmt Read cycle has not completed and the Read Data is not yet valid.	0
3	MII Link Fail	When '1' is returned - indicates that an MII Mgmt link fail has occurred.	0
31:4	-	Unused	0x0

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

14.10.15 Station Address 0 Register (SA0 - 0x3106 0040)

The Station Address 0 register (SA0) has an address of 0x3106 0040. The bit definition of this register is shown in Table 299.

Table 299. Station Address register (SA0 - address 0x3106 0040) bit description

Bit	Symbol	Function	Reset value
7:0	STATION ADDRESS, 2nd octet	This field holds the second octet of the station address.	0x0
15:8	STATION ADDRESS, 1st octet	This field holds the first octet of the station address.	0x0
31:16	-	Unused	0x0

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 45.

14.10.16 Station Address 1 Register (SA1 - 0x3106 0044)

The Station Address 1 register (SA1) has an address of 0x3106 0044. The bit definition of this register is shown in Table 300.

Table 300. Station Address register (SA1 - address 0x3106 0044) bit description

Bit	Symbol	Function	Reset value
7:0	STATION ADDRESS, 4th octet	This field holds the fourth octet of the station address.	0x0
15:8	STATION ADDRESS, 3rd octet	This field holds the third octet of the station address.	0x0
31:16	-	Unused	0x0

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 45.

14.10.17 Station Address 2 Register (SA2 - 0x3106 0048)

The Station Address 2 register (SA2) has an address of 0x3106 0048. The bit definition of this register is shown in Table 301.

Table 301. Station Address register (SA2 - address 0x3106 0048) bit description

Bit	Symbol	Function	Reset value
7:0	STATION ADDRESS, 6th octet	This field holds the sixth octet of the station address.	0x0
15:8	STATION ADDRESS, 5th octet	This field holds the fifth octet of the station address.	0x0
31:16	-	Unused	0x0

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 45.

14.11 Control register definitions

14.11.1 Command Register (Command - 0x3106 0100)

The Command register (Command) register has an address of 0x3106 0100. Its bit definition is shown in Table 302.

Table 302. Command register (Command - address 0x3106 0100) bit description

Bit	Symbol	Function	Reset value
0	RxEnable	Enable receive.	0
1	TxEnable	Enable transmit.	0
2	-	Unused	0x0
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.	0
5	RxReset	When a '1' is written, the receive datapath is reset.	0
6	PassRuntFrame	When set to '1', passes runt frames smaller 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	0x0

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

14.11.2 Status Register (Status - 0x3106 0104)

The Status register (Status) is a Read Only register with an address of 0x3106 0104. Its bit definition is shown in Table 303.

Table 303. Status register (Status - address 0x3106 0104) 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	0x0

The values represent the status of the two channels/datapaths. 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.

14.11.3 Receive Descriptor Base Address Register (RxDescriptor - 0x3106 0108)

The Receive Descriptor base address register (RxDescriptor) has an address of 0x3106 0108. Its bit definition is shown in Table 304.

Table 304. Receive Descriptor Base Address register (RxDescriptor - address 0x3106 0108) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to ’00’	-
31:2	RxDescriptor	MSBs of receive descriptor base address.	0x0

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.

14.11.4 Receive Status Base Address Register (RxStatus - 0x3106 010C)

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 305. receive Status Base Address register (RxStatus - address 0x3106 010C) bit description

Bit	Symbol	Function	Reset value
2:0	-	Fixed to ’000’	-
31:3	RxStatus	MSBs of receive status base address.	0x0

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

14.11.5 Receive Number of Descriptors Register (RxDescriptor - 0x3106 0110)

The Receive Number of Descriptors register (RxDescriptorNumber) has an address of 0x3106 0110. Its bit definition is shown in Table 306.

Table 306. Receive Number of Descriptors register (RxDescriptor - address 0x3106 0110) bit description

Bit	Symbol	Function	Reset value
15:0	RxDescriptorNumber	Number of descriptors in the descriptor array for which RxDescriptor is the base address. The number of descriptors is minus one encoded.	0x0
31:16	-	Unused	0x0

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.

14.11.6 Receive Produce Index Register (RxProduceIndex - 0x3106 0114)

The Receive Produce Index register (RxProduceIndex) is a Read Only register with an address of 0x3106 0114. Its bit definition is shown in Table 307.

Table 307. Receive Produce Index register (RxProduceIndex - address 0x3106 0114) bit description

Bit	Symbol	Function	Reset value
15:0	RxProduceIndex	Index of the descriptor that is going to be filled next by the receive datapath.	0x0
31:16	-	Unused	0x0

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.

14.11.7 Receive Consume Index Register (RxConsumeIndex - 0x3106 0118)

The Receive consume index register (RxConsumeIndex) has an address of 0x3106 0118. Its bit definition is shown in Table 308.

Table 308. Receive Consume Index register (RXConsumeIndex - address 0x3106 0118) bit description

Bit	Symbol	Function	Reset value
15:0	RxConsumeIndex	Index of the descriptor that is going to be processed next by the receive	
31:16	-	Unused	0x0

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.

14.11.8 Transmit Descriptor Base Address Register (TxDescriptor - 0x3106 011C)

The Transmit Descriptor base address register (TxDescriptor) has an address of 0x3106 011C. Its bit definition is shown in Table 309.

Table 309. Transmit Descriptor Base Address register (TxDescriptor - address 0x3106 011C) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to '00'	-
31:2	TxDescriptor	MSBs of transmit descriptor base address.	0x0

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.

14.11.9 Transmit Status Base Address Register (TxStatus - 0x3106 0120)

The Transmit Status base address register (TxStatus) has an address of 0x3106 0120. Its bit definition is shown in Table 310.

Table 310. Transmit Status Base Address register (TxStatus - address 0x3106 0120) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to '00'	-
31:2	TxStatus	MSBs of transmit status base address.	0x0

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.

14.11.10 Transmit Number of Descriptors Register (TxDescriptorNumber - 0x3106 0124)

The Transmit Number of Descriptors register (TxDescriptorNumber) has an address of 0x3106 0124. Its bit definition is shown in Table 311.

Table 311. Transmit Number of Descriptors register (TxDescriptorNumber - address 0x3106 0124) bit description

Bit	Symbol	Function	Reset value
15:0	TxDescriptorNumber	Number of descriptors in the descriptor array for which TxDescriptor is the base address. The register is minus one encoded.	
31:16	-	Unused	0x0

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.

14.11.11 Transmit Produce Index Register (TxProduceIndex - 0x3106 0128)

The Transmit Produce Index register (TxProduceIndex) has an address of 0x3106 0128. Its bit definition is shown in Table 312.

Table 312. Transmit Produce Index register (TxProduceIndex - address 0x3106 0128) bit description

Bit	Symbol	Function	Reset value
15:0	TxProduceIndex	Index of the descriptor that is going to be filled next by the transmit software driver.	0x0
31:16	-	Unused	0x0

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.

14.11.12 Transmit Consume Index Register (TxConsumeIndex - 0x3106 012C)

The Transmit Consume Index register (TxConsumeIndex) is a Read Only register with an address of 0x3106 012C. Its bit definition is shown in Table 313.

Table 313. Transmit Consume Index register (TxConsumeIndex - address 0x3106 012C) bit description

Bit	Symbol	Function	Reset value
15:0	TxConsumeIndex	Index of the descriptor that is going to be transmitted next by the transmit datapath.	0x0
31:16	-	Unused	0x0

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.

14.11.13 Transmit Status Vector 0 Register (TSV0 - 0x3106 0158)

The Transmit Status Vector 0 register (TSV0) is a Read Only register with an address of 0x3106 0158. 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 314 lists the bit definitions of the TSV0 register.

Table 314. Transmit Status Vector 0 register (TSV0 - address 0x3106 0158) bit description

Bit	Symbol	Function	Reset value
0	CRC error	The attached CRC in the packet did not match the internally generated CRC.	0
1	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	Length out of range[1]	Indicates that frame type/length field was larger than 1500 bytes.	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	Packet Defer	Packet was deferred for at least one attempt, but less than an excessive defer.	0
7	Excessive Defer	Packet was deferred in excess of 6071 nibble times in 100 Mbps or 24287 bit times in 10 Mbps mode.	0
8	Excessive Collision	Packet was aborted due to exceeding of maximum allowed number of collisions.	0
9	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	Total bytes	The total number of bytes transferred including collided attempts.	0x0
28	Control frame	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.

14.11.14 Transmit Status Vector 1 Register (TSV1 - 0x3106 015C)

The Transmit Status Vector 1 register (TSV1) is a Read Only register with an address of 0x3106 015C. 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 315 lists the bit definitions of the TSV1 register.

Table 315. Transmit Status Vector 1 register (TSV1 - address 0x3106 015C) bit description

Bit	Symbol	Function	Reset value
15:0	Transmit byte count	The total number of bytes in the frame, not counting the collided bytes.	0x0
19:16	Transmit collision count	Number of collisions the current packet incurred during transmission attempts. The maximum number of collisions (16) cannot be represented.	0x0
31:20	-	Unused	0x0

14.11.15 Receive Status Vector Register (RSV - 0x3106 0160)

The Receive status vector register (RSV) is a Read Only register with an address of 0x3106 0160. 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 316 lists the bit definitions of the RSV register.

Table 316. Receive Status Vector register (RSV - address 0x3106 0160) bit description

Bit	Symbol	Function	Reset value
15:0	Received byte count	Indicates length of received frame.	0x0
16	Packet previously ignored	Indicates that a packet was dropped.	0
17	RXDV event previously seen	Indicates that the last receive event seen was not long enough to be a valid packet.	0
18	Carrier event previously seen	Indicates that at some time since the last receive statistics, a carrier event was detected.	0
19	Receive code violation	Indicates that MII data does not represent a valid receive code.	0
20	CRC error	The attached CRC in the packet did not match the internally generated CRC.	0
21	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	Length out of range[1]	Indicates that frame type/length field was larger than 1518 bytes.	0
23	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	Dribble Nibble	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	Control frame	The frame was a control frame.	0
28	PAUSE	The frame was a control frame with a valid PAUSE opcode.	0

Table 316. Receive Status Vector register (RSV - address 0x3106 0160) bit description

Bit	Symbol	Function	Reset value
29	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	0x0

[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.

14.11.16 Flow Control Counter Register (FlowControlCounter - 0x3106 0170)

The Flow Control Counter register (FlowControlCounter) has an address of 0x3106 0170. Table 317 lists the bit definitions of the register.

Table 317. Flow Control Counter register (FlowControlCounter - address 0x3106 0170) bit description

Bit	Symbol	Function	Reset value
15:0	MirrorCounter	In full duplex mode the MirrorCounter specifies the number of cycles before re-issuing the Pause control frame.	0x0
31:16	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.	0x0

14.11.17 Flow Control Status Register (FlowControlStatus - 0x3106 0174)

The Flow Control Status register (FlowControlStatus) is a Read Only register with an address of 0x3106 8174. Table 318 lists the bit definitions of the register.

Table 318. Flow Control Status register (FlowControlStatus - address 0x3106 8174) bit description

Bit	Symbol	Function	Reset value
15:0	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.	0x0
31:16	-	Unused	0x0

14.12 Receive filter register definitions

14.12.1 Receive Filter Control Register (RxFilterCtrl - 0x3106 0200)

The Receive Filter Control register (RxFilterCtrl) has an address of 0x3106 0200. Table 319 lists the definition of the individual bits in the register.

Table 319. Receive Filter Control register (RxFilterCtrl - address 0x3106 0200) bit description

Bit	Symbol	Function	Reset value
0	AcceptUnicastEn	When set to '1', all unicast frames are accepted.	0
1	AcceptBroadcastEn	When set to '1', all broadcast frames are accepted.	0
2	AcceptMulticastEn	When set to '1', all multicast frames are accepted.	0
3	AcceptUnicastHashEn	When set to '1', unicast frames that pass the imperfect hash filter are accepted.	0
4	AcceptMulticastHashEn	When set to '1', multicast frames that pass the imperfect hash filter are accepted.	0
5	AcceptPerfectEn	When set to '1', the frames with a destination address identical to the station address are accepted.	0
11:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
12	MagicPacketEnWoL	When set to '1', the result of the magic packet filter will generate a WoL interrupt when there is a match.	0
13	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	0x0

14.12.2 Receive Filter WoL Status Register (RxFilterWoLStatus - 0x3106 0204)

The Receive Filter Wake-up on LAN Status register (RxFilterWoLStatus) is a Read Only register with an address of 0x3106 0204.

Table 320 lists the definition of the individual bits in the register.

Table 320. Receive Filter WoL Status register (RxFilterWoLStatus - address 0x3106 0204) bit description

Bit	Symbol	Function	Reset value
0	AcceptUnicastWoL	When the value is '1', a unicast frames caused WoL.	0
1	AcceptBroadcastWoL	When the value is '1', a broadcast frame caused WoL.	0
2	AcceptMulticastWoL	When the value is '1', a multicast frame caused WoL.	0
3	AcceptUnicastHashWoL	When the value is '1', a unicast frame that passes the imperfect hash filter caused WoL.	0
4	AcceptMulticastHashWoL	When the value is '1', a multicast frame that passes the imperfect hash filter caused WoL.	0
5	AcceptPerfectWoL	When the value is '1', the perfect address matching filter caused WoL.	0
6	-	Unused	0x0
7	RxFilterWoL	When the value is '1', the receive filter caused WoL.	0
8	MagicPacketWoL	When the value is '1', the magic packet filter caused WoL.	0
31:9	-	Unused	0x0

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

14.12.3 Receive Filter WoL Clear Register (RxFilterWoLClear - 0x3106 0208)

The Receive Filter Wake-up on LAN Clear register (RxFilterWoLClear) is a Write Only register with an address of 0x3106 0208.

Table 321 lists the definition of the individual bits in the register.

Table 321. Receive Filter WoL Clear register (RxFilterWoLClear - address 0x3106 0208) bit description

Bit	Symbol	Function	Reset value
0	AcceptUnicastWoLClr	When a '1' is written to one of these bits (0 to 5), the corresponding status bit in the RxFilterWoLStatus register is cleared.	0
1	AcceptBroadcastWoLClr		0
2	AcceptMulticastWoLClr		0
3	AcceptUnicastHashWoLClr		0
4	AcceptMulticastHashWoLClr		0
5	AcceptPerfectWoLClr		0
6	-	Unused	0x0
7	RxFilterWoLClr	When a '1' is written to one of these bits (7 and/or 8), the corresponding status bit in the RxFilterWoLStatus register is cleared.	0
8	MagicPacketWoLClr		0
31:9	-	Unused	0x0

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

14.12.4 Hash Filter Table LSBs Register (HashFilterL - 0x3106 0210)

The Hash Filter table LSBs register (HashFilterL) has an address of 0x3106 0210. Table 322 lists the bit definitions of the register. Details of Hash filter table use can be found in Section 14.16.10 "Receive filtering" on page 345.

Table 322. Hash Filter Table LSBs register (HashFilterL - address 0x3106 0210) bit description

Bit	Symbol	Function	Reset value
31:0	HashFilterL	Bit 31:0 of the imperfect filter hash table for receive filtering.	0x0

14.12.5 Hash Filter Table MSBs Register (HashFilterH - 0x3106 0214)

The Hash Filter table MSBs register (HashFilterH) has an address of 0x3106 0214. Table 323 lists the bit definitions of the register. Details of Hash filter table use can be found in Section 14.16.10 "Receive filtering" on page 345.

Table 323. Hash Filter MSBs register (HashFilterH - address 0x3106 0214) bit description

Bit	Symbol	Function	Reset value
31:0	HashFilterH	Bit 63:32 of the imperfect filter hash table for receive filtering.	0x0

14.13 Module control register definitions

14.13.1 Interrupt Status Register (IntStatus - 0x3106 0FE0)

The Interrupt Status register (IntStatus) is a Read Only register with an address of 0x3106 0FE0. The interrupt status register bit definition is shown in Table 324. 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 324. Interrupt Status register (IntStatus - address 0x3106 0FE0) 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	0x0
12	SoftInt	Interrupt triggered by software writing a 1 to the SoftintSet bit in the IntSet register.	0
13	WakeupInt	Interrupt triggered by a Wakeup event detected by the receive filter.	0
31:14	-	Unused	0x0

The interrupt status register is read-only. Setting can be done via the IntSet register. Reset can be accomplished via the IntClear register.

14.13.2 Interrupt Enable Register (IntEnable - 0x3106 0FE4)

The Interrupt Enable register (IntEnable) has an address of 0x3106 0FE4. The interrupt enable register bit definition is shown in Table 325.

Table 325. Interrupt Enable register (intEnable - address 0x3106 0FE4) 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	0x0
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 Wakeup event detected by the receive filter.	0
31:14	-	Unused	0x0

14.13.3 Interrupt Clear Register (IntClear - 0x3106 0FE8)

The Interrupt Clear register (IntClear) is a Write Only register with an address of 0x3106 0FE8. The interrupt clear register bit definition is shown in Table 326.

Table 326. Interrupt Clear register (IntClear - address 0x3106 0FE8) bit description

Bit	Symbol	Function	Reset value
0	RxOverrunIntClr	Writing a '1' to one of these bits clears (0 to 7) the corresponding status bit in interrupt status register IntStatus.	0
1	RxErrorIntClr		0
2	RxFinishedIntClr		0
3	RxDoneIntClr		0
4	TxUnderrunIntClr		0
5	TxErrorIntClr		0
6	TxFinishedIntClr		0
7	TxDoneIntClr		0

Table 326. Interrupt Clear register (IntClear - address 0x3106 0FE8) bit description

Bit	Symbol	Function	Reset value
11:8	-	Unused	0x0
12	SoftIntClr	Writing a '1' to one of these bits (12 and/or 13) clears the corresponding status bit in interrupt status register IntStatus.	0
13	WakeupIntClr		0
31:14	-	Unused	0x0

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.

14.13.4 Interrupt Set Register (IntSet - 0x3106 0FEC)

The Interrupt Set register (IntSet) is a Write Only register with an address of 0x3106 0FEC. The interrupt set register bit definition is shown in Table 327.

Table 327. Interrupt Set register (IntSet - address 0x3106 0FEC) bit description

Bit	Symbol	Function	Reset value
0	RxOverrunIntSet	Writing a '1' to one of these bits (0 to 7) sets the corresponding status bit in interrupt status register IntStatus.	0
1	RxErrorIntSet		0
2	RxFinishedIntSet		0
3	RxDoneIntSet		0
4	TxUnderrunIntSet		0
5	TxErrorIntSet		0
6	TxFinishedIntSet		0
7	TxDoneIntSet		0
11:8	-	Unused	0x0
12	SoftIntSet	Writing a '1' to one of these bits (12 and/or 13) sets the corresponding status bit in interrupt status register IntStatus.	0
13	WakeupIntSet		0
31:14	-	Unused	0x0

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.

14.13.5 Power Down Register (PowerDown - 0x3106 0FF4)

The Power-Down register (PowerDown) is used to block all AHB accesses except accesses to the PowerDown register. The register has an address of 0x3106 0FF4. The bit definition of the register is listed in Table 328.

Table 328. Power Down register (PowerDown - address 0x3106 0FF4) bit description

Bit	Symbol	Function	Reset value
30:0	-	Unused	0x0
31	PowerDownMACAHB	If true, all AHB accesses will return a read/write error, except accesses to the PowerDown register.	0

Setting the bit will return an error on all read and write accesses on the MACAHB interface except for accesses to the PowerDown register.

14.14 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.

14.14.1 Receive descriptors and statuses

Figure 46 depicts the layout of the receive descriptors in memory.

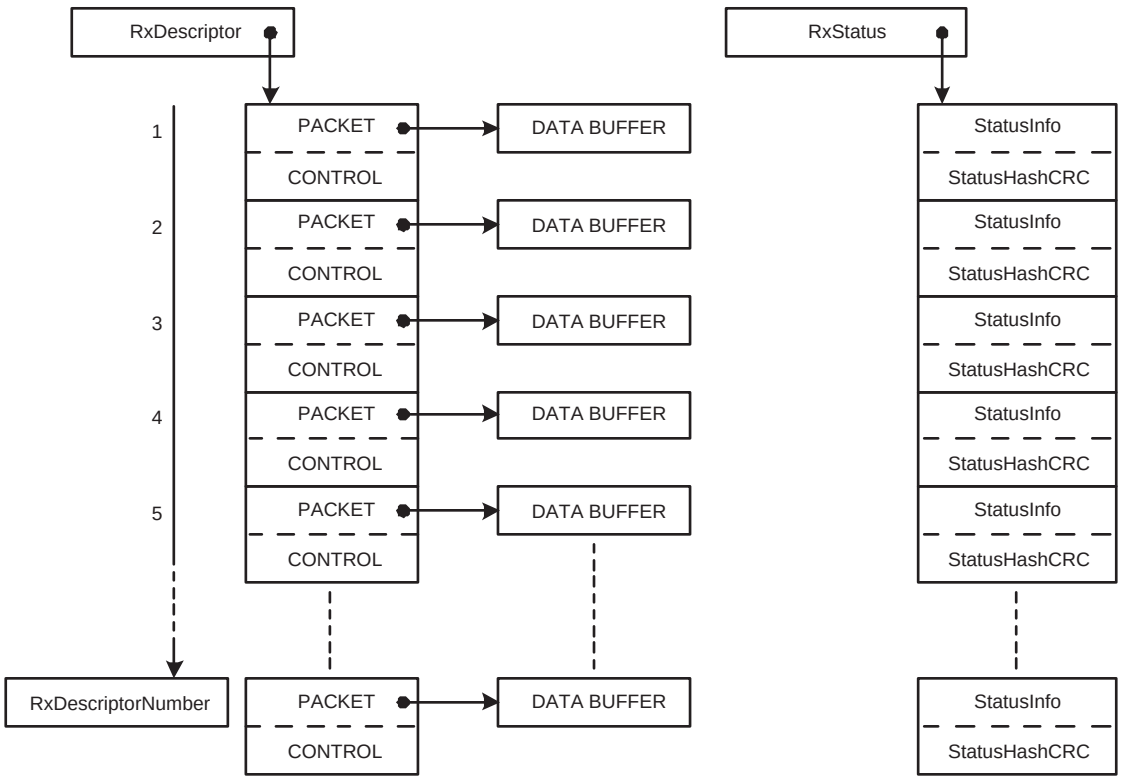


Fig 46. Receive descriptor memory layout

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 datapath 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 329.

Table 329. 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 330.

The data buffer pointer (Packet) is a 32 bits byte aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in Table 330.

Table 330. 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 331 lists the fields in the receive status elements from the status array.

Table 331. Receive Status Fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Receive status return flags, see Table 333.
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 332:

Table 332. 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 datapath reflecting the status of the reception. Table 333 lists the bit definitions in the StatusInfo word.

Table 333. 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 MII 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.

14.14.2 Transmit descriptors and statuses

Figure 47 depicts the layout of the transmit descriptors in memory.

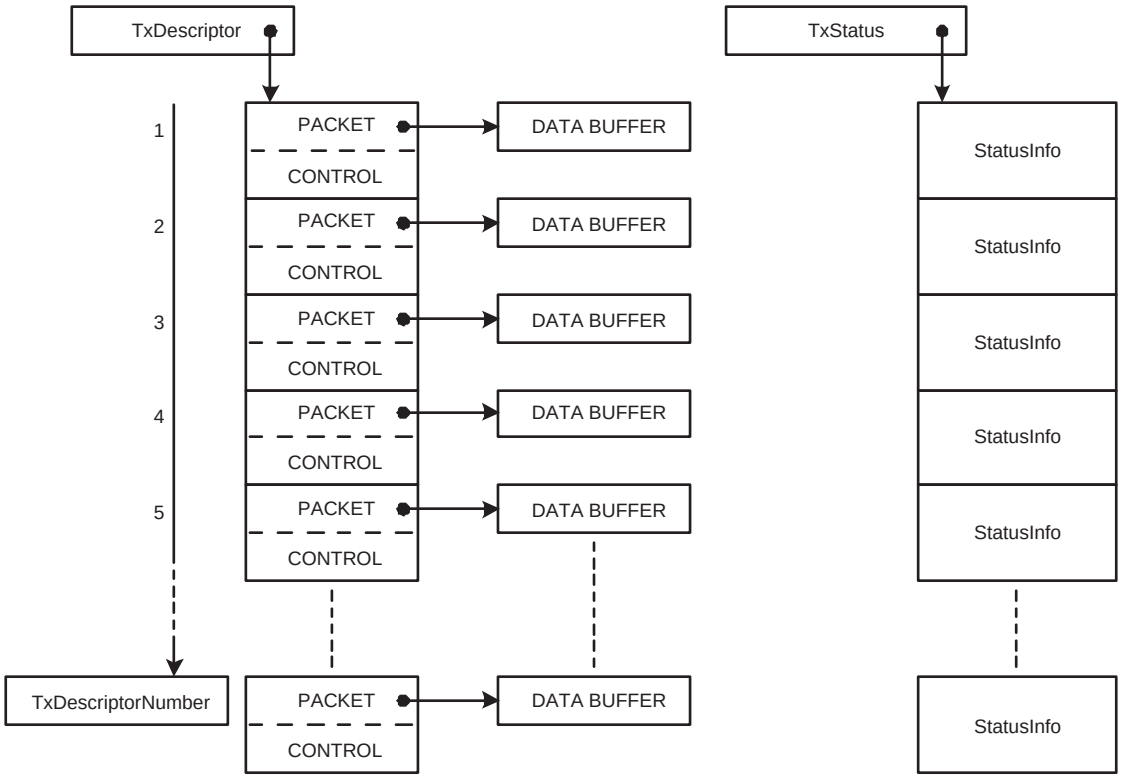


Fig 47. Transmit descriptor memory layout

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 datapath 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 334.

Table 334. 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 335.

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 335.

Table 335. 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 336 shows the one field transmit status.

Table 336. Transmit status fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Transmit status return flags, see Table 337.

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 datapath reflecting the status of the transmission. Table 337 lists the bit definitions in the StatusInfo word.

Table 337. 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.

14.15 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.

14.15.1 Overview

The Ethernet block can transmit and receive Ethernet packets from an off-chip Ethernet PHY connected through the MII or RMII interface. MII or RMII mode can be selected from 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 Ethernet MAC module 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 pre-fetch and buffer up to three descriptors.

14.15.2 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 282 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 282.
- If the PowerDown bit is set all accesses to AHB registers will result in an error response except for accesses to the PowerDown register.

14.16 Interrupts

The Ethernet block has a single interrupt request output to the CPU via the Interrupt Controller.

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 datapaths 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.

14.16.1 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. Arbitration on the AHB bus gives priority to the DMA hardware in the case of simultaneous requests. 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 datapaths 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.

14.16.2 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: 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 Ethernet MAC module 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 RMII or MII 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 datapaths

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 RMII and MII PHYs. During initialization software must select MII or RMII mode by programming the Command register. After initialization, the RMII or MII mode should not be modified.

Before switching to RMII mode the default soft reset (MAC1 register bit 15) has to be deasserted when the Ethernet block is in MII mode. The phy_tx_clk and phy_rx_clk are necessary during this operation. In case an RMII PHY is used (which does not provide these clock signals), phy_tx_clk and phy_rx_clk can be connected to the phy_ref_clk.

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 datapath. 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 datapath 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 datapath 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 datapath 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 datapaths, 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 datapaths.

14.16.3 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 datapath is disabled, the device driver should not forget to enable the transmit datapath 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 datapath. The status of the transmit datapath can be monitored by the device driver reading the TxStatus bit in the Status register.

As soon as the transmit datapath 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 datapath 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 (R)MII 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 resend the frame or have upper software layers resend 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 datapath 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 Vectored Interrupt Controller).

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 48 illustrates the transmit process in an example transmitting uses a frame header of 8 bytes and a frame payload of 12 bytes.

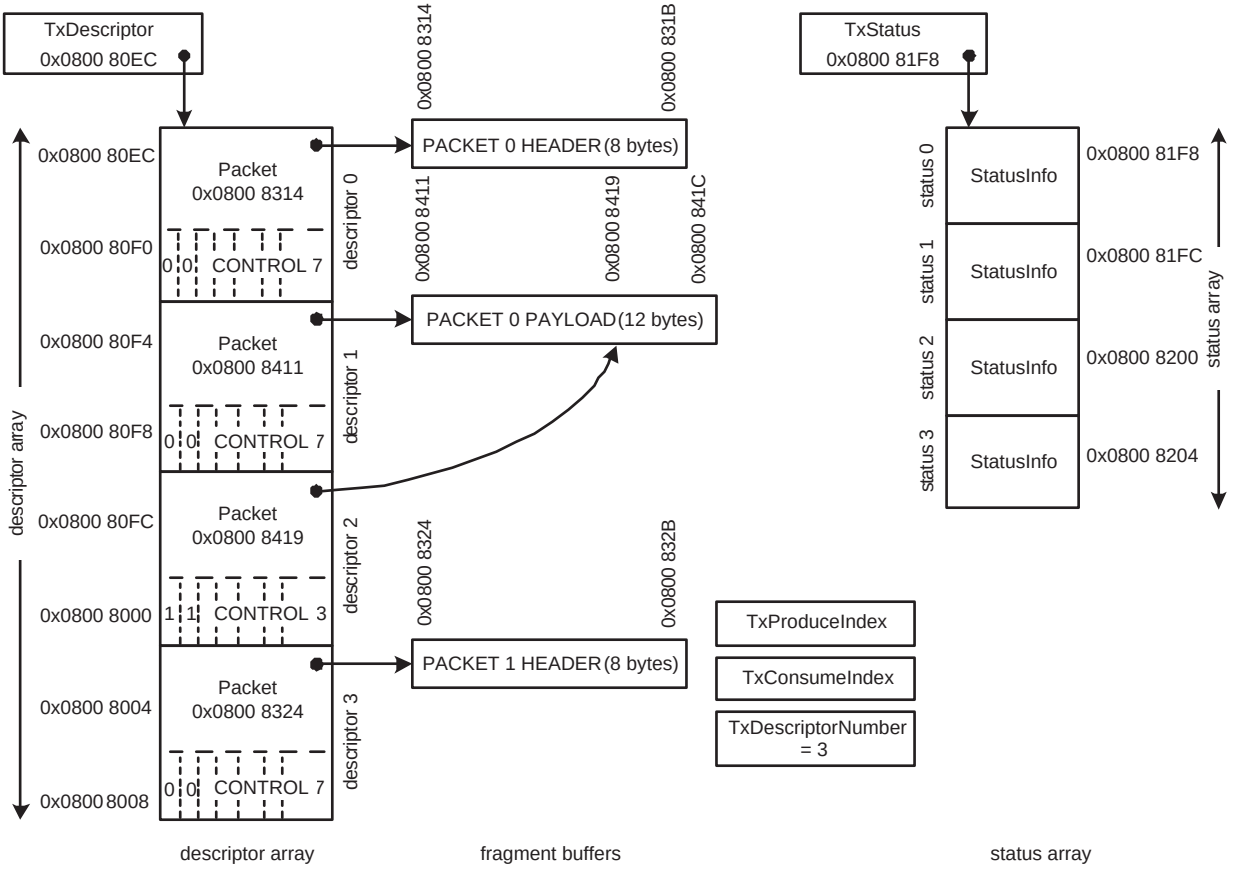


Fig 48. Transmit example memory and registers

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 (0x0800 80EC) to the TxDescriptor register and the base address of the status array (0x0800 81F8) 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 datapath may be enabled by setting the TxEnable bit in the Command register. If the transmit datapath 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 datapath was not enabled during initialization the device driver needs to enable the datapath now.

If the transmit datapath 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 (R)MII 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 nibbles. On the MII interface the Ethernet block adds the preamble, frame delimiter leader, and the CRC trailer if hardware CRC is enabled. Once transmission on the MII 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.

For an RMII PHY, the data communication between the Ethernet block and the PHY is communicated at half the data-width (2 bits) and twice the clock frequency (50 MHz). In 10 Mbps mode data will only be transmitted once every 10 clock cycles.

14.16.4 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 datapath 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 datapath, 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 datapath.

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 (R)MII 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 (R)MII 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 datapath 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 (R)MII 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 datapath 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 datapath 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 Vectored Interrupt Controller).

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 49 illustrates the receive process in an example receiving a frame of 19 bytes.

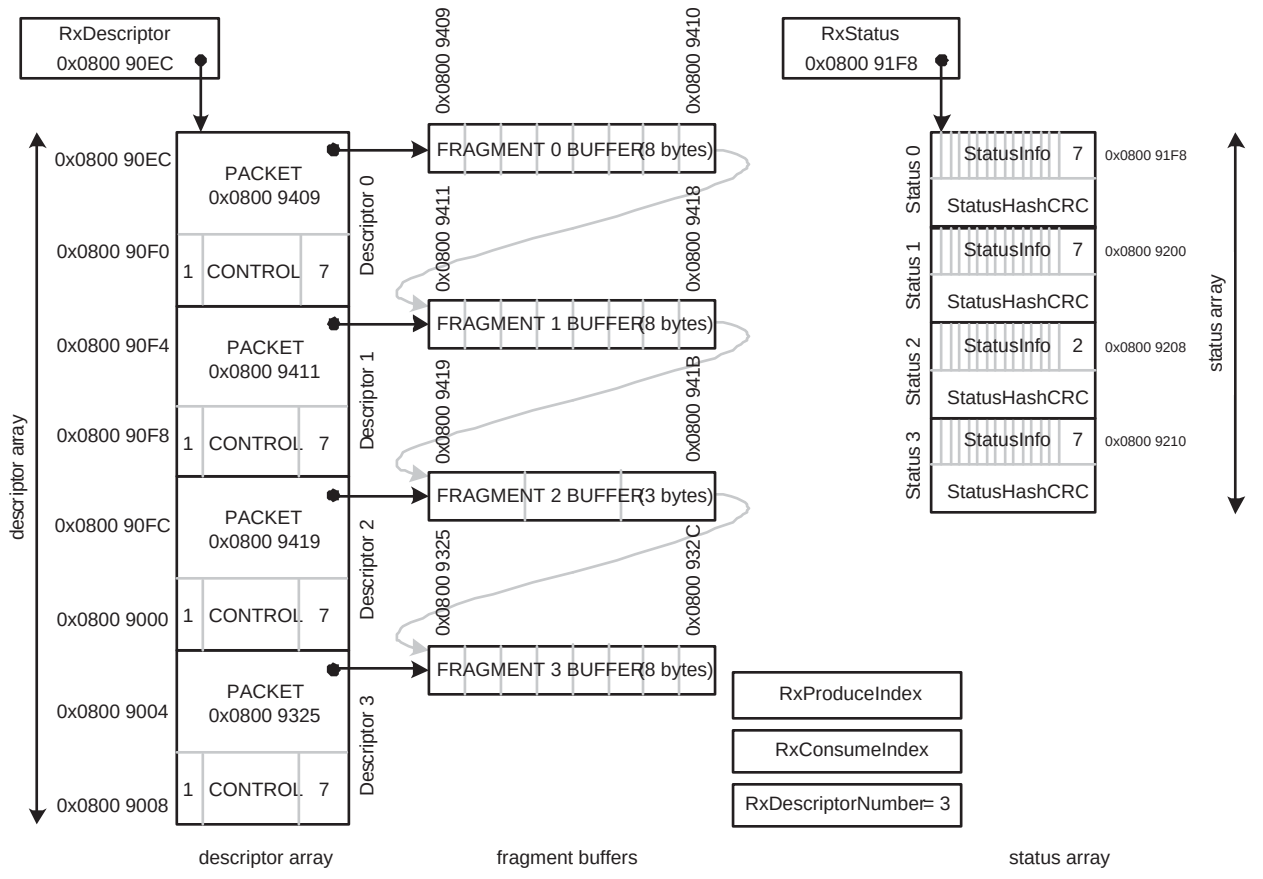


Fig 49. Receive Example Memory and Registers

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 (0x0800 90EC) in the RxDescriptor register, and the base address of the status array (0x0800 91F8) 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 (R)MII 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 (R)MII 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) is 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).

14.16.5 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 Vectored Interrupt Controller). 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.

14.16.6 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.

14.16.7 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.

14.16.8 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 datapath 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 deasserted, 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 deasserted 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 50 illustrates the transmit flow control.

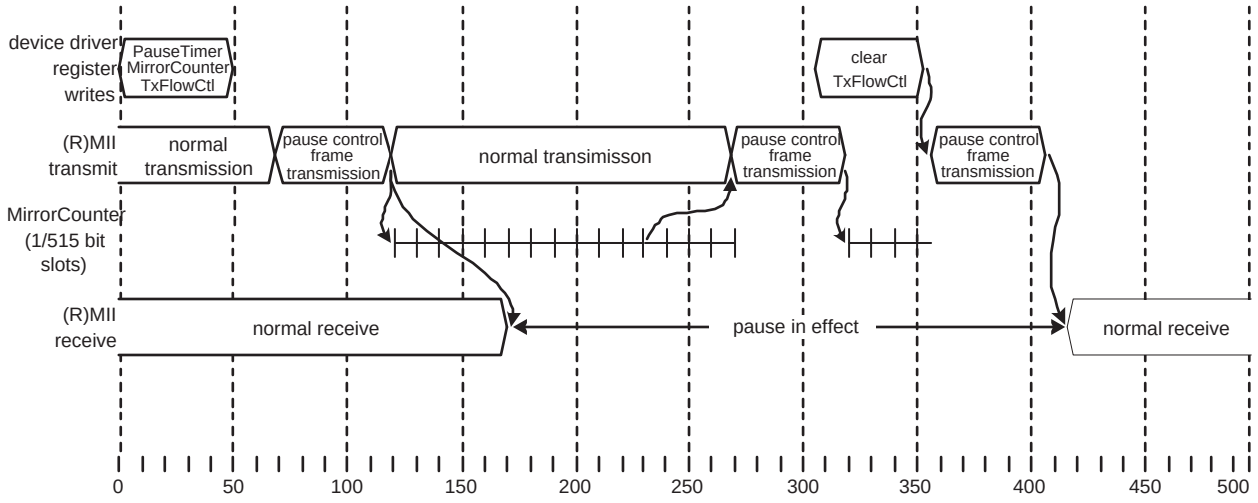


Fig 50. Transmit Flow Control

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 datapath will continue normal transmissions.

As soon as software disables transmit flow control a zero pause control frame is transmitted to resume the receive process.

14.16.9 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 deasserted. 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.

14.16.10 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 bits hash code which can be used as an index into a 64 entry programmable hash table. Figure 51 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 datapath 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.

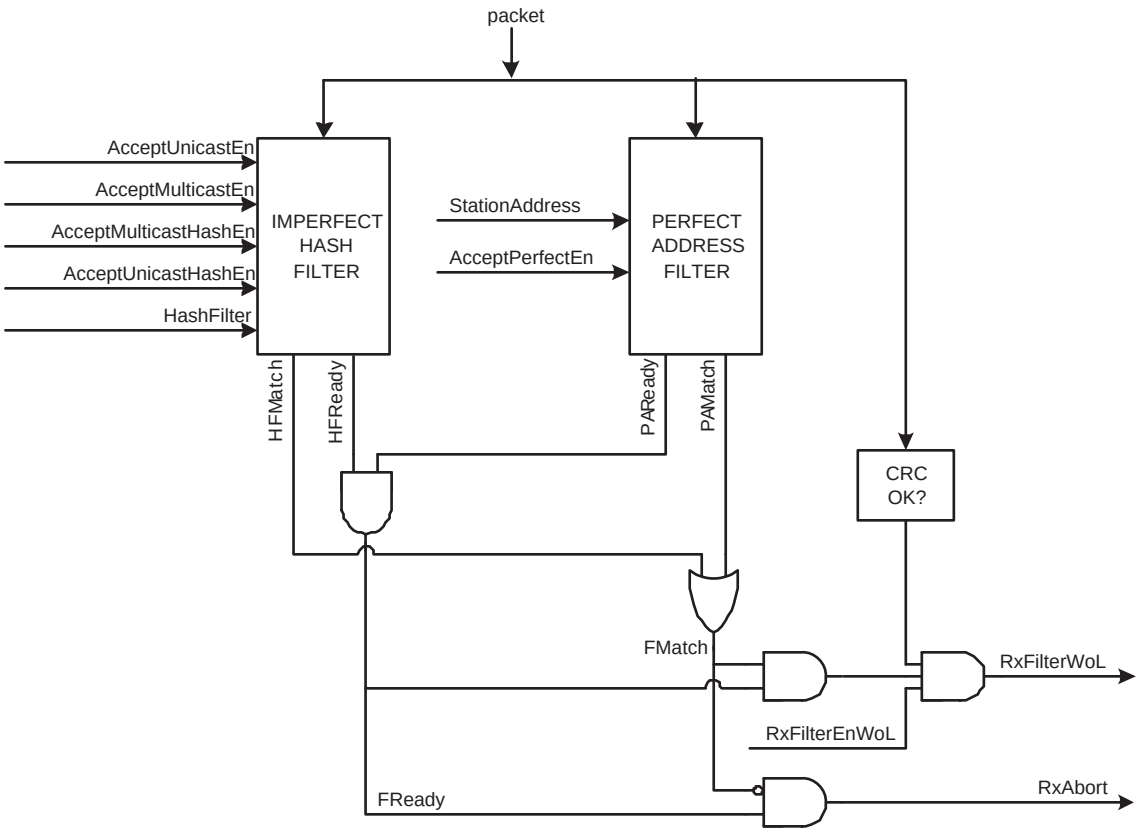


Fig 51. Receive filter block diagram

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 bits 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 datapath 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.

14.16.11 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.

14.16.12 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 51 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 datapath 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 51: 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>
```

14.16.13 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 datapath can be monitored by the device driver by reading the RxStatus bit of the Status register. Figure 52 illustrates the state machine for the generation of the RxStatus bit.

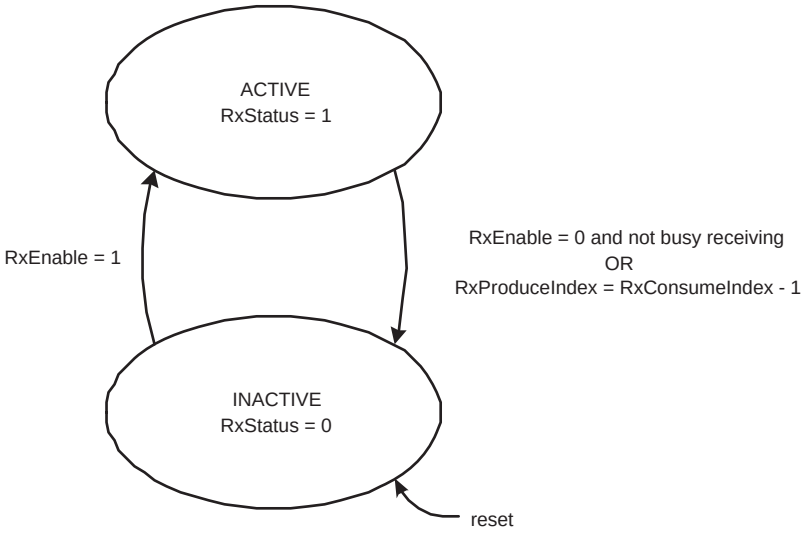


Fig 52. Receive Active/Inactive state machine

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 datapath is busy receiving a packet while the receive datapath 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 52, a soft reset is like a hardware reset assertion, i.e. after a soft reset the receive datapath is inactive until the datapath is re-enabled.

Enabling and disabling transmission

After reset, the transmit function of the Ethernet block is disabled. The Tx transmit datapath can be enabled by the device driver setting the TxEnable bit in the Command register to 1.

The status of the transmit datapaths can be monitored by the device driver reading the TxStatus bit of the Status register. Figure 53 illustrates the state machine for the generation of the TxStatus bit.

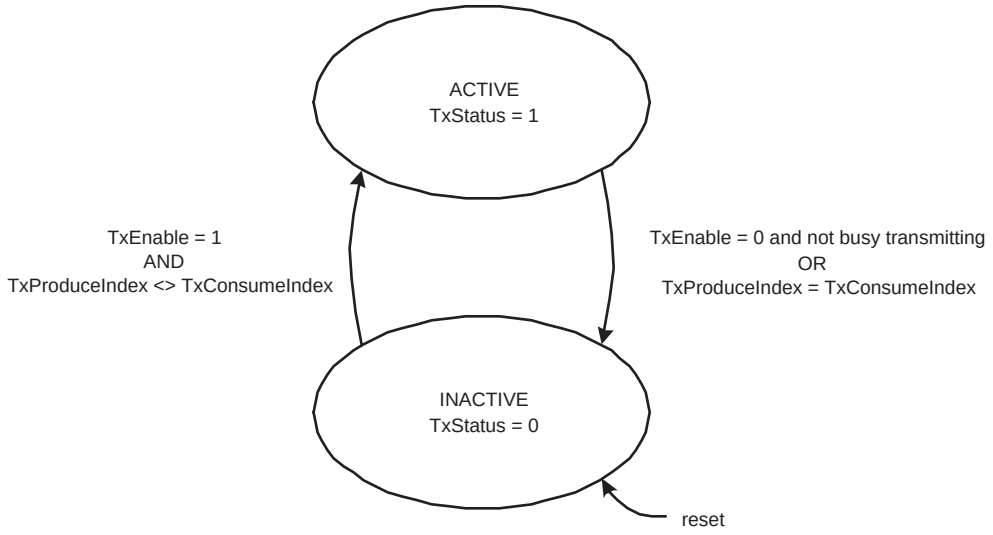


Fig 53. Transmit Active/Inactive state machine

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 datapath 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 53, a soft reset is like a hardware reset assertion, i.e. after a soft reset the transmit datapath is inactive until the datapath is re-enabled.

14.16.14 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.

14.16.15 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.

14.16.16 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.

14.16.17 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.

14.16.18 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 bit(s) 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 datapath, excluding the MAC portions, including all (read-only) registers in the transmit datapath, as well as the TxProduceIndex register in the host registers module. A soft reset of the transmit datapath will abort all AHB transactions of the transmit datapath. The reset bit will be cleared autonomously by the Ethernet block. A soft reset of the Tx datapath will clear the TxStatus bit in the Status register.
- RxReset: Writing a ‘1’ to the RxReset bit will reset the receive datapath, excluding the MAC portions, including all (read-only) registers in the receive datapath, as well as the RxConsumeIndex register in the host registers module. A soft reset of the receive datapath will abort all AHB transactions of the receive datapath. The reset bit will be cleared autonomously by the Ethernet block. A soft reset of the Rx datapath will clear the RxStatus bit in the Status register.

- RegReset: Resets all of the datapaths 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 datapath. 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.
- Reinitialize the MAC registers (0x000 to 0x0FC).
- Reset the 'SOFT RESET' bit in the MAC1 register to 0.

To reset just the transmit datapath, 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 datapath, 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.

14.16.19 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).

14.17 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.

14.17.1 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 either MII or RMII. An MII operates at 25 MHz, transferring a byte in 2 clock cycles. An RMII operates at 50 MHz, transferring a byte in 4 clock cycles. The data transfer rate is the same in both cases: 12.5 MB/s.

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 MB/s.
- 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 MB/s.
- 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 MB/s.
- 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 MB/s.
- Tx data read:
  - Data transmitted in an Ethernet frame, the size is variable.

- Basic Ethernet rate = 12.5 Mb/s.

- Rx data write:
  - Data to be received in an Ethernet frame, the size is variable.
  - Basic Ethernet rate = 12.5 MB/s.

This gives a total rate of 30.5 MB/s for the traffic generated by the Ethernet DMA function.

14.17.2 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 MB/s.
- 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 MB/s.

This gives a total rate of 36 MB/s for the traffic generated by the Ethernet DMA function.

14.17.3 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 running at 208MHz, which means HCLK is set at 104 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 HCLKs, which equals 2 times the HCLK rate. With a 104 MHz HCLK, the bandwidth is 208 MB/s, giving about 32% utilization for Ethernet traffic during simultaneous transmit and receive operations.

14.18 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.

15.1 Introduction

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 schedules transactions in 1 ms frames. Each frame contains a SoF marker and transactions that transfer data to/from device endpoints. Each device can have a maximum of 16 logical or 32 physical endpoints. There are 4 types of transfers defined for the endpoints. The control transfers are used to configure the device. The interrupt transfers are used for periodic data transfer. The bulk transfers are used when rate of transfer is not critical. The isochronous transfers have guaranteed delivery time but no error correction.

The device controller enables 12 Mb/s data exchange with a USB host controller. It consists of register interface, serial interface engine, endpoint buffer memory and DMA controller. The serial interface engine decodes the USB data stream and writes data to the appropriate end point buffer memory. The status of a completed USB transfer or error condition is indicated via status registers. An interrupt is also generated if enabled. The DMA controller when enabled transfers data between the endpoint buffer and the USB RAM.

Table 338. 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
DC	Device Core
DDP	DD Pointer
DMA	Direct Memory Access
EoP	End of Package
EP	End Point
FS	Full Speed
HREADY	When HIGH the HREADY signal indicates that a transfer has finished on the AHB bus. This signal may be driven LOW to extend a transfer.
LED	Light Emitting Diode
LS	Low Speed
MPS	Maximum Packet Size
PLL	Phase Locked Loop
RAM	Random Access Memory
SoF	Start of Frame
SIE	Serial Interface Engine

Table 338. USB related acronyms, abbreviations, and definitions used in this chapter

Acronym/abbreviation	Description
SRAM	Synchronous RAM
UDCA	USB Device Communication Area
USB	Universal Serial Bus

15.1.1 Features

- Fully compliant with USB 2.0 Full Speed specification.
- 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.
- RAM message buffer size based on endpoint realization and maximum packet size.
- Supports bus-powered capability with low suspend current.
- Support DMA transfer on all non-control endpoints.
- One Duplex DMA channel serves all endpoints.
- Allows dynamic switching between CPU controlled and DMA modes.
- Double buffer implementation for Bulk and Isochronous endpoints.

15.1.2 Fixed endpoint configuration

Table 339. Pre-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

Table 339. Pre-fixed endpoint configuration *…continued*

Logical endpoint	Physical endpoint	Endpoint type	Direction	Packet size (bytes)	Double buffer
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

15.1.3 Architecture

The architecture of the USB device controller is shown below in the block diagram.

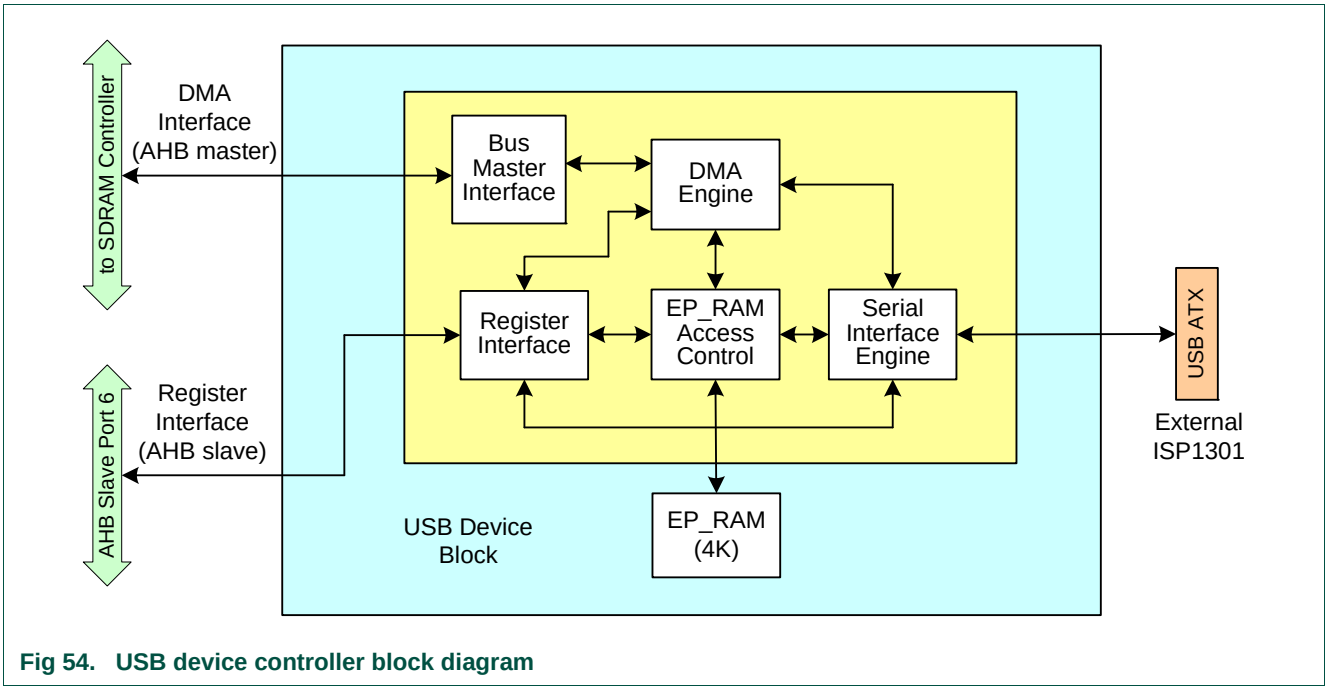


Fig 54. USB device controller block diagram

15.2 Data flow

USB is a host controlled protocol, i.e., irrespective of whether the data transfer is from the host to the device or device to the host, transfer sequence is always initiated by the host. During data transfer from device to the host, the host sends an IN token to the device, following which the device responds with the data.

15.2.1 Data flow from USB host to the device

The USB ATX receives the bi-directional D+ and D- signal of the USB bus. The USB device Serial Interface Engine (SIE) receives the serial data from the ATX and converts it into a parallel data stream. The parallel data is sent to the RAM interface which in turn transfers the data to the endpoint buffer. The endpoint buffer is implemented as an SRAM based FIFO. Each realized endpoint will have a reserved space in the RAM. So the total RAM space required depends on the number of realized endpoints, maximum packet size of the endpoint and whether the endpoint supports double buffering. Data is written to the buffers with the header showing how many bytes are valid in the buffer.

For non-isochronous endpoints, when a full data packet is received without any errors, the endpoint generates a request for data transfer from its FIFO by generating an interrupt to the system.

Isochronous endpoint will have one packet of data to be transferred in every frame. So the data transfer has to be synchronized to the USB frame rather than packet arrival. Therefore for every 1 ms there will be an interrupt to the system.

The data transfer follows the little endian format. The first byte received from the USB bus will be available in the LS byte of the receive data register.

15.2.2 Data flow from device to the host

For data transfer from an endpoint to the host, the host will send an IN token to that endpoint. If the FIFO corresponding to the endpoint is empty, the device will return a NAK and will raise an interrupt to the system. On this interrupt the CPU fills a packet of data in the endpoint FIFO. The next IN token that comes after filling this packet will transfer this packet to the host.

The data transfer follows the little endian format. The first byte sent on the USB bus will be the LS byte of the transmit data register.

15.2.3 Slave mode transfer

Slave data transfer is done through the interrupt issued from the USB device to the CPU.

Reception of valid (error-free) data packet in any of the OUT non-isochronous endpoint buffer generates an interrupt. Upon receiving the interrupt, the software can read the data using receive length and data registers. When there is no empty buffer (for a given OUT non-isochronous endpoint), any data arrival generates an interrupt only if the Interrupt on NAK feature for that endpoint type is enabled and the existing interrupt is cleared. For OUT isochronous endpoints, the data will always be written irrespective of the buffer status. There will be no interrupt generated specific to OUT isochronous endpoints other than the frame interrupt.

Similarly, when a packet is successfully transferred to the host from any of the IN non-isochronous endpoint buffer, an interrupt is generated. When there is no data available in any of the buffers (for a given IN non-isochronous endpoint), a data request generates an interrupt only if Interrupt on NAK feature for that endpoint type is enabled and existing interrupt is cleared. Upon receiving the interrupt, the software can load any data to be sent using transmit length and data registers. For IN isochronous endpoints, the data available in the buffer will be sent only if the buffer is validated; otherwise, an empty packet will be sent. Like OUT isochronous endpoints, there will be no interrupt generated specific to IN isochronous endpoints other than the frame interrupt.

15.2.4 DMA mode transfer

Under DMA mode operation the USB device will act as a master on the AHB bus and transfers the data directly from the memory to the endpoint buffer and vice versa. A duplex channel DMA acts as a AHB master on the bus.

The endpoint 0 of USB (default control endpoint) will receive the setup packet. It will not be efficient to transfer this data to the USB RAM since the CPU has to decode this command and respond back to the host. So, this transfer will happen in the slave mode only.

For each isochronous endpoint, one packet transfer happens every frame. Hence, the DMA transfer has to be synchronized to the frame interrupt.

The DMA engine also supports Auto Transfer Length Extraction (ATLE) mode for bulk transfers. In this mode the DMA engine recovers the transfer size from the incoming packet stream.

15.2.5 Interrupts

The USB device has three interrupt output lines. The interrupts usb_dev_lp_int and usb_dev_hp_int facilitates transfer of data in slave mode. These two interrupt lines are provided to allow two different priority (high/low) levels in slave mode transfer. Each of the individual endpoint interrupts can be routed to either high priority or low priority levels using corresponding bits in the endpoint interrupt priority register. The interrupt level is triggered with active HIGH polarity. The external interrupt generation takes place only if the necessary ‘enable’ bits are set in the Device Interrupt Enable register. Otherwise, they will be registered only in the status registers. The usb_dev_dma_int is raised when an end_of_transfer or a system error has occurred. DMA data transfer is not dependent on this interrupt. These interrupts also contribute to the USB_INT which can act as a start source in STOP mode.

15.3 Interfaces

15.3.1 Pin description

Table 340. USB external interface

Name	Direction	Description
USB_I2C_SDA	I/OT	I^2C serial bus data[1]
USB_I2C_SCL	I/OT	I^2C serial bus clock[1]
USB_ATX_INT_N	I	Interrupt from transceiver

Table 340. USB external interface

Name	Direction	Description
USB_OE_TP_N	I/O	Transmit enable for DAT/SE0
USB_DAT_VP	I/O	TX data / D+ receive
USB_SE0_VM	I/O	S. E. Zero transmit / D− receive

[1] Open drain pin requiring an external pull-up resistor

15.3.2 AHB interface

Accessing all of the registers in USB device controller is done through the AHB interface. AHB is also used for data transfer to all endpoints in the slave mode. All AHB signals are timed by the AHB clock “HCLK”.

The minimum AHB clock frequency should be 18 MHz if the USB block is enabled.

15.3.3 Clock

The USB device controller clock is a 48MHz input clock derived from the Main oscillator clock OSC_CLK. This clock will be used to recover the 12MHz clock from the USB bus.

The AHB clock is also needed to access all the USB device registers.

15.3.4 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 the bus-powered device.

1. A device in the non-configured state should draw a maximum of 100mA from the bus.
2. The configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500mA.
3. A suspended device should draw only a maximum of 500μA.

15.3.4.1 Suspend and resume (Wake-up)

A device can go into suspend state if there is no activity for more than 3ms. In a full speed device, a frame token (SoF packet) starts at every millisecond. So, they are less likely to go into suspend state. But there are two situations during which they do go into the suspend state.

In the global suspend mode, the USB host suspends the full USB system by stopping the transmission of SoF packets. In the selective suspend mode, the host disables the hub port in which the device is connected, thus blocking the transmission of SoF packets and data to the device.

A suspended device can be resumed or woken up if the host starts sending USB packets again (host initiated wake-up).

15.3.4.2 Power management support

When the device is going to the suspend state, there will be an interrupt to the USB device controller when there is no activity on the bus for more than 3ms.

If there is no bus activity again for the next 2ms, the usb_dev_needclk signal will go low. This indicates that the USB main clock can be switched off. Once the USB main clock is switched off, internal registers in the USB clock domain will not be visible anymore to the software.

15.3.4.3 Remote wake-up

The USB device controller supports software initiated remote wake-up. Remote wake-up involves a resume signal initiated from the device. This is done by resetting the suspend bit in the device status register. Before writing into the register, all the clocks to the USB device have to be enabled. In order to keep the usb_dev_needclk high, the AP_CLK bit in the set mode register needs to be set to high so that the 48 Mhz PLL clock to the USB device controller is always enabled.

15.3.5 Software interface

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

15.3.5.1 Register map

The following registers are located in the AHB clock domain. The minimum AHB clock frequency should be 18 MHz. They can be accessed directly by the CPU. All registers are 32 bit wide and aligned in the word address boundaries.

USB slave mode registers are located in the address region 0x3102 0200 to 0x3102 024C. All unused address in this region reads “DEADABBA”.

DMA related registers are located in the address region 0x3102 0250 to 0x3102 02FC. All unused address in this region reads invalid data.

Table 341. USB device register address definitions

Name	Description	Address	R/W[1]	Function
Device interrupt registers				
USBDevIntSt	Device Interrupt Status	0x3102 0200	R	Interrupt status register for the device
USBDevInt En	Device Interrupt Enable	0x3102 0204	R/W	Enable external interrupt generation
USBDevIntClr	Device Interrupt Clear	0x3102 0208	C	Clears device interrupt status
USBDevIntSet	Device Interrupt Set	0x3102 020C	S	Sets device interrupt status
USBDevIntPri	Device Interrupt Priority	0x3102 022C	W	Interrupt priority register
Endpoint interrupt registers				
USBEpIntSt	Endpoint Interrupt Status	0x3102 0230	R	Interrupt status register for endpoints
USBEpIntEn	Endpoint Interrupt Enable	0x3102 0234	R/W	Enable endpoint interrupt generation
USBEpIntClr	Endpoint Interrupt Clear	0x3102 0238	C	Clears endpoint interrupt status
USBEpIntSet	Endpoint Interrupt Set	0x3102 023C	S	Sets endpoint interrupt status
USBEpIntPri	Endpoint Interrupt Priority	0x3102 0240	W	Defines in which interrupt line the endpoint interrupt will be routed
Endpoint realization registers				
USBReEp	Realize Endpoint	0x3102 0244	R/W	Defines which endpoints are to be realized

Table 341. USB device register address definitions

Name	Description	Address	R/W[1]	Function
USBEpInd	Endpoint Index	0x3102 0248	W	Pointer to the maxpacketsize register array
USBEpMaxPSize	MaxPacket Size	0x3102 024C	R/W	Max packet size register array
Data transfer registers				
USBRxData	Receive Data	0x3102 0218	R	Register from which data corresponding to the OUT endpoint packet is to be read
USBRxPLen	Receive PacketLength	0x3102 0220	R	Register from which packet length corresponding to the OUT endpoint packet is to be read
USBTxData	Transmit Data	0x3102 021C	W	Register to which data to the IN endpoint is to be written
USBTxPLen	Transmit PacketLength	0x3102 0224	W	Register to which packet length for IN endpoint is to be written
USBCtrl	USB Control	0x3102 0228	R/W	Controls read-write operation
Command registers				
USBCmdCode	Command Code	0x3102 0210	W	Register to which command has to be written
USBCmdData	Command Data	0x3102 0214	R	Register from which data resulting from the execution of command to be read
DMA registers				
USBDMARSt	DMA Request Status	0x3102 0250	R	The DMA request status register
USBDMARClr	DMA Request Clear	0x3102 0254	C	DMA request clear register
USBDMARSet	DMA Request Set	0x3102 0258	S	DMA Request set register
USBUDCAH	UDCA_Head	0x3102 0280	R/W	DD pointer address location
USBEpDMASt	EP DMA Status	0x3102 0284	R	DMA enable status for each endpoint
USBEpDMAEn	EP DMA Enable	0x3102 0288	S	Endpoint DMA enable register
USBEpDMADis	EP DMA Disable	0x3102 028C	C	Endpoint DMA disable register
USBDMAIntSt	DMA Interrupt Status	0x3102 0290	R	DMA Interrupt status register
USBDMAIntEn	DMA Interrupt Enable	0x3102 0294	R/W	DMA Interrupt enable register
USBEoTIntSt	End Of Transfer Interrupt Status	0x3102 02A0	R	DMA transfer complete interrupt status register
USBEoTIntClr	End Of Transfer Interrupt Clear	0x3102 02A4	C	DMA transfer complete interrupt clear register
USBEoTIntSet	End Of Transfer Interrupt Set	0x3102 02A8	S	DMA transfer complete interrupt set register
USBNDDRIntSt	New DD Request Interrupt Status	0x3102 02AC	R	New DD request interrupt status register
USBNDDRIntClr	New DD Request Interrupt Clear	0x3102 02B0	C	New DD request interrupt clear register
USBNDDRIntSet	New DD Request Interrupt Set	0x3102 02B4	S	New DD request interrupt set register
USBSysErrIntSt	System Error Interrupt Status	0x3102 02B8	R	System error interrupt status register
USBSysErrIntClr	System Error Interrupt Clear	0x3102 02BC	C	System error interrupt clear register
USBSysErrIntSet	System Error Interrupt Set	0x3102 02C0	S	System error interrupt set register

[1] The R/W column in Table 339 lists the accessibility of the register:

a) Registers marked ‘R’ for access will return their current value when read.

b) Registers marked ‘S’ for access allows individual bits to be set to ‘1’ for each corresponding register bit. Bits set to ‘0’ will not affect the value of the corresponding register bit. Reading an ‘S’ marked register will return an invalid value.

c) Registers marked ‘C’ for access allows individual bits to be cleared by writing a value that has bits set to ‘1’ for each corresponding register bit that needs to be set to ‘0’. Bits set to ‘0’ will not affect the value of the corresponding register bit. Reading a ‘C’ marked register will return invalid value.

d) Registers marked ‘R/W’ allow both read and write.

15.3.6 USB device register definitions

15.3.6.1 USB Device Interrupt Status Register - (USBDevIntSt - 0x3102 0200, R)

Interrupt status register holds the value of the interrupt. ‘0’ indicates no interrupt and ‘1’ indicates the presence of the interrupt.

Table 342. USB Device Interrupt Status Register - (USBDevIntSt - 0x3102 0200, R)

Bits	Name	Function	Reset value
31:10	-	Reserved	0x0
9	ERR_INT	Error Interrupt. Any bus error interrupt from the USB device. Refer to section Section 15.3.8.1.9 “ReadErrorStatus”.	0
8	EP_RLZED	Endpoints realized. Set when Realize endpoint register or Maxpacket size register is updated.	0
7	TxENDPKT	The number of data bytes transferred to the FIFO equals the number of bytes programmed in the TxPacket length register.	0
6	RxENDPKT	The current packet in the FIFO is transferred to the CPU.	0
5	CDFULL	Command data register is full (Data can be read now).	0
4	CCEMPTY	The command code register is empty (New command can be written).	1
3	DEV_STAT	Set when USB Bus reset, USB suspend change or Connect change event occurs. Refer to section Section 15.3.8.1.6 “Set Device Status”.	0
2	EP_SLOW	This is the Slow interrupt transfer for the endpoint. If an Endpoint Interrupt Priority Register bit is not set, the endpoint interrupt will be routed to this bit.	0
1	EP_FAST	This is the fast interrupt transfer for the endpoint. If an Endpoint Interrupt Priority register bit is set, the endpoint interrupt will be routed to this bit.	0
0	FRAME	The frame interrupt occurs every 1 ms. This is to be used in isochronous packet transfer.	0

15.3.6.2 USB Device Interrupt Enable Register - (USBDevIntEn - 0x3102 0204, R/W)

If the Interrupt Enable bit value is set, an external interrupt is generated (on Fast or Slow Interrupt line) when the corresponding bit in the interrupt status register is set. If it is not set, no external interrupt is generated but interrupt will still be held in the interrupt status register. The bit field definition is same as the device interrupt status register as shown in Table 342. All bits of this register are cleared after reset.

Table 343. USB Device Interrupt Enable Register - (USBDevIntEn - 0x3102 0204, R/W)

Bits	Name	Function	Reset value
31:0	See USBDevIntSt register bit allocation.	0 - No external interrupt is generated. 1 - Enables an external interrupt to be generated (Fast or Slow) when the corresponding bit in the USBDevIntSt register is set. If this bit is not set, no external interrupt is generated, but the interrupt status will be held in the interrupt status register.	0x0

15.3.6.3 USB Device Interrupt Clear Register - (USBDevIntClr - 0x3102 0208, C)

Setting a particular bit to '1' in this register causes the clearing of the interrupt by resetting the corresponding bit in the interrupt status register. Writing a '0' will not have any influence. The bit field definition is same as the device interrupt status register as shown in Table 342.

Table 344. USB Device Interrupt Clear Register - (USBDevIntClr - 0x3102 0208, C)

Bits	Name	Function	Reset value
31:0	See USBDevIntSt register bit allocation.	0 - No effect. 1 - The corresponding bit in the USBDevIntSt register is cleared.	0x0

15.3.6.4 USB Device Interrupt Set Register - (USBDevIntSet - 0x3102 020C, S)

Setting a particular bit to '1' in this register will set the corresponding bit in the interrupt status register. Writing a '0' will not have any influence. The bit field definition is same as the device interrupt status register as shown in Table 342.

Table 345. USB Device Interrupt Set Register - (USBDevIntSet - 0x3102 020C, S)

Bits	Name	Function	Reset value
31:0	See USBDevIntSt register bit allocation.	0 - No effect. 1 - The corresponding bit in the USBDevIntSt register is set.	0x0

15.3.6.5 USB Device Interrupt Priority Register - (USBDevIntPri - 0x3102 022C, W)

If the corresponding bit is set to '1', the corresponding interrupt will be routed to the high priority interrupt line. If the bit is '0' the interrupt will be routed to the low priority interrupt line. Only one of the EP_FAST or FRAME can be routed to the high priority interrupt line. Setting both bits at the same time is not allowed. If the software attempts to set both the bits to '1', none of them will be routed to the high priority interrupt line. All enabled endpoint interrupts will be routed to the low priority interrupt line if the EP_FAST bit is set to 0, irrespective of the Endpoint Interrupt Priority register setting.

Table 346. USB Device Interrupt Priority Register - (USBDevIntPri - 0x3102 022C, W)

Bits	Name	Function	Reset value
7:2	-	Reserved	0x0
1	EP_FAST	0 - EP_FAST interrupt is routed to the low priority interrupt line. 1 - EPFAST interrupt is routed to the high priority interrupt line. This is the fast interrupt transfer for the endpoint. If an Endpoint Interrupt Priority register bit is set, the endpoint interrupt will be routed to the high priority interrupt line.	0
0	FRAME	0 - FRAME interrupt is routed to the low priority interrupt line. 1 - FRAME interrupt is routed to the high priority interrupt line. The frame interrupt occurs every 1 ms. This is to be used in an isochronous packet transfer.	0

15.3.6.6 USB Endpoint Interrupt Status Register - (USBEpIntSt - 0x3102 0230, R)

Each physical non-isochronous endpoint is represented by one bit in this register to indicate that it has generated the interrupt. All non-isochronous OUT endpoints give an interrupt when they receive a packet without any error. All non-isochronous IN endpoints will give an interrupt when a packet is successfully transmitted or a NAK handshake is sent on the bus provided that the interrupt on NAK feature is enabled. Isochronous endpoint transfer takes place with respect to frame interrupt.

Table 347. USB Endpoint Interrupt Status Register - (USBEpIntSt - 0x3102 0230, R)

Bits	Name	Function	Reset value
31	EP 15TX	Endpoint 15, Data Transmitted Interrupt bit or sent a NAK.	0
30	EP 15RX	Endpoint 15, Data Received Interrupt bit.	0
29	EP 14TX	Endpoint 14, Data Transmitted Interrupt bit or sent a NAK.	0
28	EP 14RX	Endpoint 14, Data Received Interrupt bit.	0
27	EP 13TX	Endpoint 13, Data Transmitted Interrupt bit or sent a NAK.	0
26	EP 13RX	Endpoint 13, Data Received Interrupt bit.	0
25	EP 12TX	Endpoint 12, Isochronous endpoint.	NA
24	EP 12RX	Endpoint 12, Isochronous endpoint.	NA
23	EP 11TX	Endpoint 11, Data Transmitted Interrupt bit or sent a NAK.	0
22	EP 11RX	Endpoint 11, Data Received Interrupt bit.	0
21	EP 10TX	Endpoint 10, Data Transmitted Interrupt bit or sent a NAK.	0
20	EP 10RX	Endpoint 10, Data Received Interrupt bit.	0
19	EP 9TX	Endpoint 9, Isochronous endpoint.	NA
18	EP 9RX	Endpoint 9, Isochronous endpoint.	NA
17	EP 8TX	Endpoint 8, Data Transmitted Interrupt bit or sent a NAK.	0
16	EP 8RX	Endpoint 8, Data Received Interrupt bit.	0
15	EP 7TX	Endpoint 7, Data Transmitted Interrupt bit or sent a NAK.	0
14	EP 7RX	Endpoint 7, Data Received Interrupt bit.	0
13	EP 6TX	Endpoint 6, Isochronous endpoint.	NA
12	EP6 RX	Endpoint 6, Isochronous endpoint.	NA
11	EP 5TX	Endpoint 5, Data Transmitted Interrupt bit or sent a NAK.	0
10	EP 5RX	Endpoint 5, Data Received Interrupt bit.	0

Table 347. USB Endpoint Interrupt Status Register - (USBEpIntSt - 0x3102 0230, R) *…continued*

Bits	Name	Function	Reset value
9	EP 4TX	Endpoint 4, Data Transmitted Interrupt bit or sent a NAK.	0
8	EP 4RX	Endpoint 4, Data Received Interrupt bit.	0
7	EP 3TX	Endpoint 3, Isochronous endpoint.	NA
6	EP 3RX	Endpoint 3, Isochronous endpoint.	NA
5	EP 2TX	Endpoint 2, Data Transmitted Interrupt bit or sent a NAK.	0
4	EP 2RX	Endpoint 2, Data Received Interrupt bit.	0
3	EP 1TX	Endpoint 1, Data Transmitted Interrupt bit or sent a NAK.	0
2	EP 1RX	Endpoint 1, Data Received Interrupt bit.	0
1	EP 0TX	Endpoint 0, Data Transmitted Interrupt bit or sent a NAK.	0
0	EP 0RX	Endpoint 0, Data Received Interrupt bit.	0

15.3.6.7 USB Endpoint Interrupt Enable Register - (USBEpIntEn - 0x3102 0234, R/W)

Setting bits in this register will cause the corresponding bit in the interrupt status register to transfer its status to the device interrupt status register. Either the EP_FAST or EP_SLOW bit will be set depending on the value in the endpoint interrupt priority register. Setting this bit to ‘1’ implies operating in the slave mode. The bit field definition is the same as the Endpoint Interrupt Status Register as shown in Table 347.

Table 348. USB Endpoint Interrupt Enable Register - (USBEpIntEn - 0x3102 0234, R/W)

Bits	Name	Function	Reset value
31:0	See USBEpIntSt register bit allocation.	0 - No effect. 1 - The corresponding bit in the USBEpIntSt register transfers its status to the USBDevIntSt register. Setting any bit to 1 in the USBEpIntEn register implies operating in Slave mode.	0x0

15.3.6.8 USB Endpoint Interrupt Clear Register - (USBEpIntClr - 0x3102 0238, C)

Writing a ‘1’ to this bit clears the bit in the endpoint interrupt status register. Writing 0 will not have any impact. When the endpoint interrupt is cleared from this register, the hardware will clear the CDFULL bit in the device interrupt status register. On completion of this action, the CDFULL bit will be set and the command data register will have the status of the endpoint.

Endpoint interrupt register and CDFULL bit of Device Interrupt status register are related through clearing of interrupts in USB clock domain. Whenever software attempts to clear a bit of Endpoint interrupt register, hardware will clear CDFULL bit before it starts issuing "Select Endpoint/Clear Interrupt" command (refer to Section 15.3.8.1.11) and sets the same bit when command data is available for reading. Software will have to wait for CDFULL bit to be set to '1' (whenever it expects data from hardware) before it can read Command Data register.

Remark: Even though endpoint interrupts are "accessible" via either registers or protocol engine commands, keep in mind that the register is an "image" of what is happening at the protocol engine side. Therefore read the endpoint interrupt status register to know which endpoint has to be served, and then select one of two ways to clear the endpoint interrupt:

- Send the “SelectEndpoint/ClearInterrupt” command to the protocol engine in order to properly clear the interrupt, then read the CMD_DATA to get the status of the interrupt when CDFULL bit is set.

- Write a 1 to the corresponding bit in the endpoint interrupt clear register, wait until CDFULL bit is set, then read the CMD_DATA to get the status of the interrupt.

For bit definition of endpoint status read from command data register, refer to Table 387. Each physical endpoint has its own reserved bit in this register. The bit field definition is the same as the Endpoint Interrupt Status Register as shown in Table 347.

Table 349. USB Endpoint Interrupt Clear Register - (USBEpIntClr - 0x3102 0238, C)

Bits	Name	Function	Reset value
31:0	See USBEpIntSt register bit allocation.	0 - No effect. 1 - Clears the corresponding bit in the USBEpIntSt register.	0x0

Software is allowed to issue clear operation on multiple endpoints as well. However, only the status of the endpoint with the lowest number can be read at the end of this operation. Therefore, if the status of all the endpoints is needed, clearing the interrupts on multiple endpoints at once is not recommended. This is explained further in the following example:

Assume bits 5 and 10 of Endpoint Interrupt Status register are to be cleared. The software can issue Clear operation by writing in Endpoint Interrupt Clear register (with corresponding bit positions set to '1'). Then hardware will do the following:

1. Clears CDFULL bit of Device Interrupt Status register.
2. Issues 'Select Endpoint/Interrupt Clear' command for endpoint 10.
3. Waits for command to get processed and CDFULL bit to get set.
4. Now, endpoint status (for endpoint 10) is available in Command Data register (note that hardware does not wait for the software to finish reading endpoint status in Command Data register for endpoint 10).
5. Clears CDFULL bit again.
6. Issues 'Select Endpoint/Interrupt Clear' command for endpoint 5.
7. Waits for command to get processed and CDFULL bit to get set.
8. Now, endpoint status (for endpoint 5) is available in Command Data register for the software to read.

15.3.6.9 USB Endpoint Interrupt Set Register - (USBEpIntSet - 0x3102 023C, S)

Writing a ‘1’ to a bit in this register sets the corresponding bit in the endpoint interrupt status register. Writing 0 will not have any impact. Each endpoint has its own bit in this register. The bit field definition is the same as the Endpoint Interrupt Status Register as shown in Table 347.

Table 350. USB Endpoint Interrupt Set Register - (USBEpIntSet - 0x3102 023C, S)

Bits	Name	Function	Reset value
31:0	See USBEpIntSt register bit allocation.	0 - No effect. 1 - Sets the corresponding bit in the USBEpIntSt register.	0x0

15.3.6.10 USB Endpoint Interrupt Priority Register - (USBEpIntPri - 0x3102 0240, W)

This register determines whether the interrupt has to be routed to the fast interrupt line (EP_FAST) or to the slow interrupt line (EP_SLOW). If set 1 the interrupt will be routed to the fast interrupt bit of the device status register. Otherwise it will be routed to the slow endpoint interrupt bit. Note that routing of multiple endpoints to EP_FAST or EP_SLOW is

possible. The bit field definition is the same as the Endpoint Interrupt Status Register as shown in Table 347. The Device Interrupt Priority register may override this register setting. Refer to Section 15.3.6.5 for more details.

Table 351. USB Endpoint Interrupt Priority Register - (USBEpIntPri - 0x3102 0240, W)

Bits	Name	Function	Reset value
31:0	See USBEpIntSt register bit allocation.	0 - The corresponding interrupt will be routed to the slow endpoint interrupt bit in the USBEpIntSet register. 1 - The corresponding interrupt will be routed to the fast endpoint interrupt bit in the USBEpIntSet register.	0x0

15.3.6.11 USB Realize Endpoint Register - (USBReEp - 0x3102 0244, R/W)

Though fixed-endpoint configuration implements 32 endpoints, it is not a must that all have to be used. If the endpoint has to be used, it should have buffer space in the EP_RAM. The EP_RAM space can be optimized by realizing a subset of endpoints. This is done through programming the Realize Endpoint register. Each physical endpoint has one bit as shown in Table 352. The USBReEp is a R/W register.

Table 352. USB Realize Endpoint Register - (USBReEp - 0x3102 0244, R/W)

Bits	Name	Function	Reset value
0	EP0	Control endpoint is realized by default after power on.	1
1	EP1	Control endpoint is realized by default after power on.	1
31:2	EPxx	Where xx can take a value between 2 and 31. 0 => endpoint unrealized. 1 => endpoint realized.	0

At power on only the default control endpoint is realized. Other endpoints if required have to be realized by programming the corresponding bit in the Realize Endpoint register. Realization of endpoints is a multi-cycle operation. The pseudo code of endpoint realization is shown below.

```
for every endpoint to be realized,
{
    /* OR with the existing value of the register */
    RealizeEndpointRegister |= (UInt32) ((0x1 << endpt));
    /* Load endpoint index Reg with physical endpoint no.*/
    EndpointIndexRegister = (UInt32) endpointnumber;

    /* load the max packet size Register */
    Endpoint MaxPacketSizeReg = PacketSize;

    /* check whether the EP_RLSED bit is set */
    while (!(DeviceInterruptStatusReg & PFL_HW_EP_RLSED_BIT))
    {
        /* wait till endpoint realization is complete */
    }
    /* Clear the EP_RLSED bit */
    Clear EP_RLSED bit in DeviceInterrupt Status Reg;
}
```

Device will not respond to any tokens to the un-realized endpoint. ‘Configure Device’ command can only enable all realized and enabled endpoints. See Section 15.3.8.1.2 for more details.

15.3.7 EP_RAM requirements

The USB device controller uses dedicated RAM based FIFO (EP_RAM) as an endpoint buffer. Each endpoint has a reserved space in the EP_RAM. The EP_RAM size requirement for an endpoint depends on its Maxpacketsize and whether it is double buffered or not. 32 words of EP_RAM are used by the device for storing the 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.

EP_ RAM size (in words) required for the physical endpoint can be expressed as

epramsize = ((Maxpacketsize + 3)/4 + 1) × db_status

where db_status = 1 for single buffered endpoint and 2 for double buffered endpoint.

Since all the realized endpoints occupy EP_RAM space, the total EP_RAM requirement is

$$\text{Total EP\_RAM size} = 32 + \sum_{(n+0)}^{N} epramsize(n) \quad (12)$$

where N is the number of realized endpoints. Total EP_RAM size should not exceed 4K bytes (1K words).

EP_RAM can be accessed by 3 sources, which are SIE, DMA engine and CPU. Among them, CPU has the highest priority followed by the SIE and DMA engine. The DMA engine has got the lowest priority. Then again, under the above mentioned 3 request sources, write request has got higher priority than read request. Typically, CPU does single word read or write accesses, the DMA logic can do 32-byte burst access. The CPU and DMA engine operates at a higher clock frequency as compared to the SIE engine. The CPU cycles are valuable and so the CPU is given the highest priority. The CPU clock frequency is higher than the SIE operating frequency (12 MHz). The SIE will take 32 clock cycles for a word transfer. In general, this time translates to more than 32 clock cycles of the CPU in which it can easily do several accesses to the memory.

15.3.7.1 USB Endpoint Index Register - (USBEpInd - 0x3102 0248, W)

Each endpoint has a register carrying the Maxpacket size value for that endpoint. This is in fact a register array. Hence before writing, this register has to be ‘addressed’ through the Endpoint Index register.

The endpoint index register will hold the physical endpoint number. Writing into the Maxpacket size register will set the array element pointed by the Endpoint Index register.

Table 353. USB Endpoint Index Register - (USBEpInd - 0x3102 0248, W)

Bits	Name	Function	Reset value
31:5	-	Reserved.	NA
4:0	Phy endpoint	The physical endpoint number (0-31).	0x0

15.3.7.2 USB MaxPacketSize Register - (USBMaxPSize - 0x3102 024C, R/W)

At power on control endpoint is assigned the Maxpacketsize of 8 bytes. Other endpoints are assigned 0. Modifying MaxPacketSize register content will cause the buffer address of the internal RAM to be recalculated. This is essentially a multi-cycle process. At the end of it, the EP_RLZED bit will be set in the Device Interrupt Status register. MaxPacket Register Array Indexing is shown in Figure 55.

Table 354. USB MaxPacketSize Register - (USBMaxPSize - 0x3102 024C, R/W)

Bits	Name	Function	Reset value
31:10	-	Reserved.	NA
9:0	MaxPacketSize	The maximum packet size value.	0x8

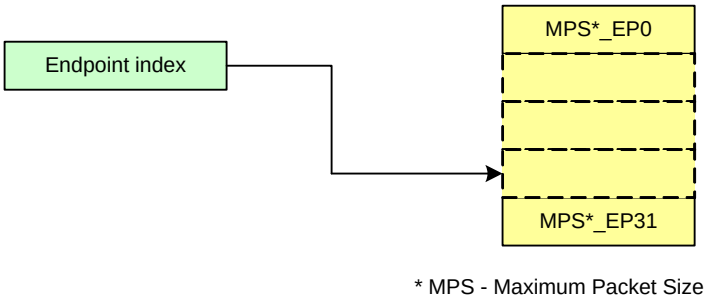


Fig 55. Maxpacket register array indexing

15.3.7.3 USB Receive Data Register - (USBRxData - 0x3102 0218, R)

For an OUT transaction, CPU reads the endpoint data from this register. Data from the endpoint RAM is fetched and filled in this register. There is no interrupt when the register is full.

Table 355. USB Receive Data Register - (USBRxData - 0x3102 0218, R)

Bits	Name	Function	Reset value
31:0	Receive data	Receive Data.	0x0

15.3.7.4 USB Receive Packet Length Register - (USBRxPLen - 0x3102 0220, R)

This register gives the number of bytes remaining in the EP_RAM for the current packet being transferred and whether the packet is valid or not. This register will get updated at every word that gets transferred to the system. Software can use this register to get the number of bytes to be transferred. When the number of bytes reaches zero, an end of packet interrupt is generated.

Table 356. USB Receive Packet Length Register - (USBRxPLen - 0x3102 0220, R)

Bits	Name	Function	Reset value
31:12	-	Reserved.	NA
11	PKT_RDY	Packet length field in the register is valid and packet is ready for reading.	0
10	DV	'1' - Data is valid; '0' - Data is invalid. Non-isochronous end point will not raise an interrupt when an erroneous data packet is received. But invalid data packet can be produced with bus reset. For isochronous endpoint, data transfer will happen even if an erroneous packet is received. In this case DV bit will not be set for the packet.	0
9:0	PKT_LNGTH	The remaining amount of data in bytes still to be read from the RAM.	0x0

15.3.7.5 USB Transmit Data Register - (USBTxData - 0x3102 021C, W)

For an IN transaction the CPU writes the data into this register. This data will be transferred into the EP_RAM before the next writing occurs. There is no interrupt when the register is empty.

Table 357. USB Transmit Data Register - (USBTxData - 0x3102 021C, W)

Bits	Name	Function	Reset value
31:0	Transmit Data	Transmit Data.	0x0

15.3.7.6 USB Transmit Packet Length Register - (USBTxPLen - 0x3102 0224, W)

The software should first write the packet length (<= Maximum Packet Size) in the Transmit Packet Length register followed by the data write(s) to the Transmit Data register. This register counts the number of bytes transferred from the CPU to the EP_RAM. The software can read this register to determine the number of bytes it has transferred to the EP_RAM. After each write to the Transmit Data register the hardware will decrement the contents of the Transmit Packet Length register. For lengths larger than the Maximum Packet Size, the software should submit data in steps of Maximum Packet Size and the remaining extra bytes in the last packet. For example, if the Maximum Packet Size is 64 bytes and the data buffer to be transferred is of length 130 bytes, then the software submits 64 bytes packet twice followed by 2 bytes in the last packet. So, a total of 3 packets are sent on USB.

Table 358. USB Transmit Packet Length Register - (USBTxPLen - 0x3102 0224, W)

Bits	Name	Function	Reset value
31:10	-	Reserved.	NA
9:0	PKT_LNGTH	The remaining amount of data in bytes to be written to the EP_RAM.	0x0

15.3.7.7 USB Control Register - (USBCtrl - 0x3102 0228, R/W)

This register controls the data transfer operation of the USB device.

Table 359. USB Control Register - (USBCtrl - 0x3102 0228, R/W)

Bits	Name	Function	Reset value
31:6	-	Reserved.	NA
5:2	LOG_ENDPOINT	Logical Endpoint Number.	0x0
1	WR_EN	Write Enable; 1 - Write mode is enabled; 0 - disabled	0
0	RD_EN	Read Enable; 1 - Read mode is enabled; 0 - disabled	0

15.3.7.8 Slave mode data transfer

When the software wants to read the data from an endpoint buffer it should make the Read Enable bit high and should program the LOG_ENDPOINT in the USB control register. The control logic will first fetch the packet length to the receive packet length register. The PKT_RDY bit (Table 356) in the packet length register is set along with this. Also the hardware fills the receive data register with the first word of the packet.

The software can now start reading the Receive Data register. When the end of packet is reached the Read Enable bit will be disabled by the control logic and RxENDPKT bit is set in the Device interrupt status register. The software should issue a Clear Buffer (Section 15.3.8.1.13) command. The endpoint is now ready to accept the next packet.

If the software makes the Read Enable bit low midway, the reading will be terminated. In this case the data will remain in the EP_RAM. When the Read Enable signal is made high again for this endpoint, data will be read from the beginning.

For writing data to an endpoint buffer, Write Enable bit should be made high and software should write to the Tx Packet Length register the number of bytes it is going to send in the packet. It can then write data continuously in the Transmit Data register.

When the control logic receives the number of bytes programmed in the Tx Packet length register, it will reset the Write Enable bit. The TxENDPKT bit is set in the Device interrupt status register. The software should issue a Validate Buffer (Section 15.3.8.1.14) command. The endpoint is now ready to send the packet. If the software resets this bit midway, writing will start again from the beginning.

A synchronization mechanism is used to transfer data between the two clock domains i.e. AHB slave clock and the USB bit clock at 12 MHz. This synchronization process takes up to 5 clock cycles of the slow clock (i.e. 12 MHz) for reading/writing from/to a register before the next read/write can happen. The AHB HREADY output from the USB device is driven appropriately to take care of the timing.

Both Read Enable and Write Enable bits can be high at the same time for the same logical endpoint. The interleaved read and write operation is possible.

15.3.7.9 USB Command Code Register - (USBCmdCode - 0x3102 0210, W)

This register is used for writing the commands. The commands written here will get propagated to the Protocol Engine and will be executed there. After executing the command, the register will be empty, and the “CCEMTY” bit of the Interrupt status register is set high. See Section 15.3.8 “Protocol engine command description” on page 384

Table 360. USB Command Code Register - (USBCmdCode - 0x3102 0210, W)

Bits	Name	Function	Reset value
31:24	-	Reserved.	0x0
23:16	CMD_CODE	The code for the command.	0x0
15:8	CMD_PHASE	The command phase.	0x0
7:0	-	Reserved.	0x0

15.3.7.10 USB Command Data Register - (USBCmdData - 0x3102 0214, R)

This is a read-only register which will carry the data retrieved after executing a command. When the data are ready to read, the “CD_FULL” bit of the device interrupt status register is set. The CPU can poll this bit or enable an interrupt corresponding to this to sense the arrival of the data. The data is always one-byte wide. See Section 15.3.8 “Protocol engine command description” on page 384.

Table 361. USB Command Data Register - (USBCmdData - 0x3102 0214, R)

Bits	Name	Function	Reset value
31:8	-	Reserved.	0x0
7:0	Command Data	Command Data.	0x0

15.3.7.11 USB DMA Request Status Register - (USBDMARSt - 0x3102 0250, R)

This register is set by the hardware whenever a packet (OUT) or token (IN) is received on a realized endpoint. It serves as a flag for DMA engine to start the data transfer if the DMA is enabled for this particular endpoint. Each endpoint has one reserved bit in this register. Hardware sets this bit when a realized endpoint needs to be serviced through DMA. Software can read the register content. DMA cannot be enabled for control endpoints (EP0 and EP1). For easy readability the control endpoint is shown in the register contents.

Table 362. USB DMA Request Status Register - (USBDMARSt - 0x3102 0250, R)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31	0
30:2	EPxx	Where xx can take a value between 2 and 30. 0 => No request 1 => DMA requested	0x0
1	EP1	Control endpoint IN (DMA cannot be enabled for this endpoint).	0
0	EP0	Control endpoint OUT (DMA cannot be enabled for this endpoint).	0

15.3.7.12 USB DMA Request Clear Register - (USBDMARClr - 0x3102 0254, C)

Writing ‘1’ into the register will clear the corresponding interrupt from the DMA request register. Writing ‘0’ will not have any effect. After a packet transfer, the hardware clears the particular bit in DMA Request Status register. Software does not need to clear this bit. The bit field definition is same as the DMA Request Status Register as shown in Table 362.

Table 363. USB DMA Request Clear Register - (USBDMARClr - 0x3102 0254, C)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31	0
30:2	EPxx	Where xx can take a value between 2 and 30. 0 => No effect 1 => Clear the corresponding interrupt from the DMA register.	0x0
1	EP1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0
0	EP0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).	0

The software should not clear the DMA request clear bit while the DMA operation is in progress. But if this bit is cleared, the behavior of the DMA engine will depend on at what time the clearing is done. There can be more than one DMA requests pending at any given time. The DMA engine processes these requests serially (i.e starting from EP2 to EP31). If the DMA request for a particular endpoint is cleared before DMA operation has started for that request, then the DMA engine will never know about the request and no DMA operation on that endpoint will be done (till the next request appears). On the other hand, if the DMA request for a particular endpoint is cleared after the DMA operation corresponding to that request has begun, it does not matter even if the request is cleared, since the DMA engine has registered the endpoint number internally and will not sample the same request before finishing the current DMA operation.

15.3.7.13 USB DMA Request Set Register - (USBDMARSet - 0x3102 0258, S)

Writing ‘1’ into the register will set the corresponding interrupt from the DMA request register. Writing ‘0’ will not have any effect. The bit field definition is same as the DMA Request Status Register as shown in Table 362.

Table 364. USB DMA Request Clear Register - (USBDMARClr - 0x3102 0254, C)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31	0
30:2	EPxx	Where xx can take a value between 2 and 30. 0 => No effect 1 => Set the corresponding interrupt from the DMA register.	0x0
1	EP1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0
0	EP0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).	0

The "DMA Request Set Register" is normally used for the test purpose. It is also useful in the normal operation mode to avoid a "lock" situation if the DMA is programmed after that the USB packets are already received. Normally the arrival of a packet generates an interrupt when it is completely received. This interrupt is used by the DMA to start working. This works fine as long as the DMA is programmed before the arrival of the packet (2 packets - if double buffered). If the DMA is programmed "too late", the interrupts were already generated in slave mode (but not handled because the intention was to use the

DMA) and when the DMA is programmed no interrupts are generated to "activate" it. In this case the usage of the DMA Request Set Register is useful to manually start the DMA transfer.

15.3.7.14 USB UDCA Head Register - (USBUDCAH - 0x3102 0280, R/W)

The UDCA (USB Device Communication Area) Head register maintains the address where UDCA is allocated in the USB RAM (Figure 56). The USB RAM is part of the system memory which is used for the USB purposes. It can be located in IRAM or memory attached to the EMC controller. Note, however, DMA on endpoint 0 is not feasible. The UDCA has to be aligned to 128-byte boundary and should be of size 128 bytes (32 words that correspond to 32 physical endpoints). Each word can point to a DMA descriptor of a physical endpoint or can point to NULL (i.e. zero value) when the endpoint is not enabled for DMA operation. This implies that the DMA descriptors need to be created only for the DMA enabled endpoints. Gaps can be there while realizing the endpoints and there is no need to keep dummy DMA descriptors. The DMA engine will not process the descriptors of the DMA disabled endpoints. The reset value for this register is 0. Refer to Section 15.3.9 and Section 15.4 for more details on DMA descriptors.

Table 365. USB UDCA Head Register - (USBUDCAH - 0x3102 0280, R/W)

Bits	Name	Function	Reset value
31:7	UDCA Header	Start address of the UDCA Header.	0x0
6:0	-	UDCA header is aligned in 128-byte boundaries.	0x0

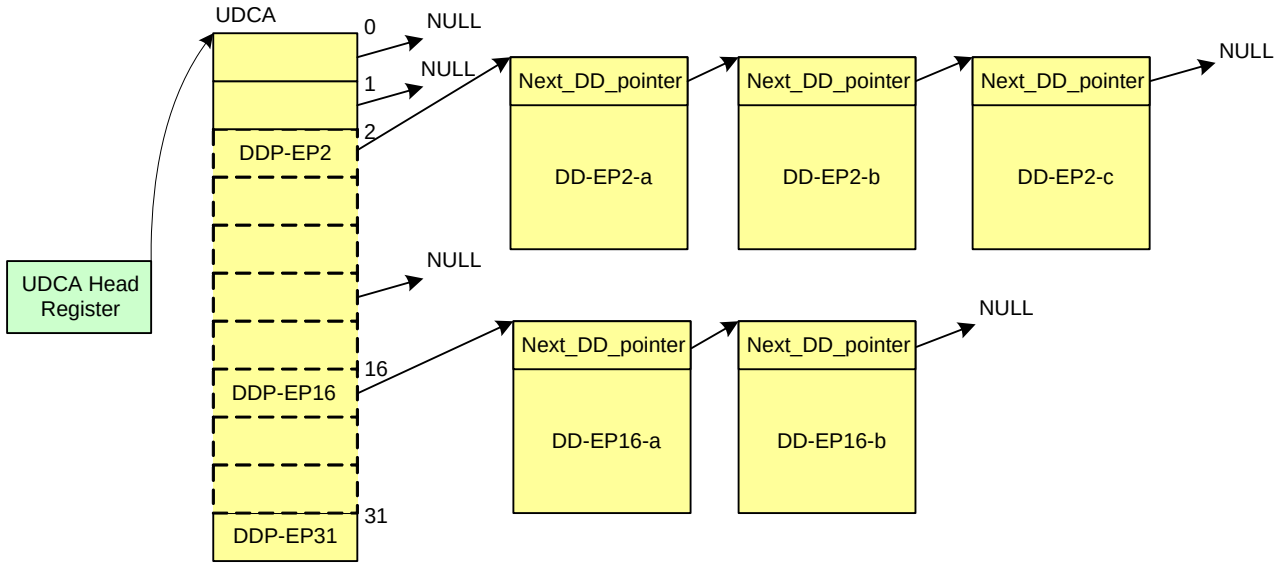


Fig 56. UDCA Head register and DMA descriptors

15.3.7.15 USB EP DMA Status register - (USBEpDMASt - 0x3102 0284, R)

This register indicates whether the DMA for a particular endpoint is enabled or disabled. Each endpoint has one bit assigned in the EP DMA Status register. DMA transfer can start only if this bit is set. Hence, it is referred as DMA_ENABLE bit. If the bit in the EP DMA Status register is made '0' (by writing into EP DMA Disable register) in between a packet

transfer, the current packet transfer will still be completed. After the current packet, DMA gets disabled. In other words, the packet transfer when started will end unless an error condition occurs. When error condition is detected the bit will be reset by the hardware.

Software does not have direct write permission to this register. It has to set the bit through EP DMA Enable register. Resetting of the bit is done through 'EP DMA Disable' register.

Table 366. USB EP DMA Status register - (USBEpDMASt - 0x3102 0284, R)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 2 and 31. 0 => The DMA for Endpoint EPxx is disabled 1 => The DMA for Endpoint EPxx is enabled	0x0
1	EP1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP0 must be 0).	0
0	EP0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0

15.3.7.16 USB EP DMA Enable Register - (USBEpDMAEn - 0x3102 0288, S)

Writing '1' to this register will enable the DMA operation for the corresponding endpoint. Writing '0' will not have any effect. The bit field definition is same as the EP_DMA Status Register as shown in Table 366.

Table 367. USB EP DMA Enable Register - (USBEpDMAEn - 0x3102 0288, S)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 2 and 31. 0 => No effect. 1 => Enable DMA operation for endpoint EPxx.	0x0
1	EP1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP0 must be 0).	0
0	EP0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0

15.3.7.17 USB EP DMA Disable Register - (USBDEpDMADis - 0x3102 028C, C)

Writing '1' to this register will disable the DMA operation for the corresponding endpoint. Writing '0' will have the effect of resetting the DMA_PROCEED flag. The bit field definition is same as the EP_DMA Status Register as shown in Table 366.

Table 368. USB EP DMA Disable Register - (USBDEpDMADis - 0x3102 028C, C)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 2 and 31. 0 => No effect. 1 => Disable DMA operation for endpoint EPxx.	0x0
1	EP1	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP0 must be 0).	0
0	EP0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0

15.3.7.18 USB DMA Interrupt Status Register - (USBDMAIntSt - 0x3102 0290, R)

Bit 0 “End of Transfer Interrupt” will be set by hardware if any of the 32 bits in the End Of Transfer Interrupt Status register is ‘1’. The same logic applies for Bit 1 and 2 of the DMA Interrupt Status register. The hardware checks the 32 bits of New DD Request Interrupt Status register to set/clear the bit 1 of DMA Interrupt Status register and similarly the 32 bits of System Error Interrupt Status register to set/clear the bit 2 of DMA Interrupt Status register.

Table 369. USB DMA Interrupt Status Register - (USBDMAIntSt - 0x3102 0290, R)

Bits	Name	Function	Reset value
31:3	-	Reserved.	0x0
2	System_Error_Interrupt	System error interrupt. 0 - All bits in the USBSysErrIntSt register are 0. 1 - At least one bit in the USBSysErrIntSt register is set.	0
1	New DD Request Interrupt	New DD Request Interrupt. 0 - All bits in the USBNDDRIntSt are 0. 1 - At least one bit in the USBNDDRIntSt is set.	0
0	End of Transfer Interrupt	End of Transfer Interrupt. 0 - All bits in the USBSysErrIntSt are 0. 1 - At least one bit in the USBSysErrIntSt is set.	0

15.3.7.19 USB DMA Interrupt Enable Register - (USBDMAIntEn - 0x3102 0294, R/W)

Setting the bit in this register will cause external interrupt to happen for the bits set in the USB DMA Interrupt Status register. The bit field definition is same as the DMA Interrupt Status Register as shown in Table 369.

Table 370. USB DMA Interrupt Enable Register - (USBDMAIntEn - 0x3102 0294, R/W)

Bits	Name	Function	Reset value
31:3	-	Reserved.	0x0
2	System_Error_Interrupt	System error interrupt. 0 - The System Error Interrupt is disabled. 1 - The System Error Interrupt is enabled.	0
1	New DD Request Interrupt	New DD Request Interrupt. 0 - The New DD Request interrupt is disabled. 1 - The New DD Request Interrupt is enabled.	0
0	End of Transfer Interrupt	End of Transfer Interrupt. 0 - The End of Transfer Interrupt is disabled. 1 - The End of transfer Interrupt is enabled.	0

15.3.7.20 USB New DD Request Interrupt Status Register - (USBNDDRIntSt - 0x3102 02AC, R)

This interrupt bit is set when a transfer is requested from the USB device and no valid DD is detected for this endpoint.

Table 371. USB New DD Request Interrupt Status Register - (USBNDDRIntSt - 0x3102 02AC, R)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No new DD request for Endpoint xx. 1 => New DD Request for Endpoint xx.	0x0
0	EP0	Endpoint 0.	0

15.3.7.21 USB New DD Request Interrupt Clear Register - (USBNDDRIntClr - 0x3102 02B0, C)

Writing ‘1’ into the register will clear the corresponding interrupt from the status register. Writing ‘0’ will not have any effect. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 372. USB New DD Request Interrupt Clear Register - (USBNDDRIntClr - 0x3102 02B0, C)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No effect. 1 => Clear the EPxx new DD Interrupt request in the USBNDDRIntSt register.	0x0
0	EP0	Endpoint 0.	0

15.3.7.22 USB New DD Request Interrupt Set Register - (USBNDDRIntSet - 0x3102 02B4, S)

Writing ‘1’ into the register will set the corresponding interrupt from the status register. Writing ‘0’ will not have any effect. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 373. USB New DD Request Interrupt Set Register - (USBNDDRIntSet - 0x3102 02B4, S)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No effect. 1 => Set the EPxx new DD Interrupt request in the USBNDDRIntSt register.	0x0
0	EP0	Endpoint 0.	0

15.3.7.23 USB End Of Transfer Interrupt Status Register - (USBEoTIntSt - 0x3102 02A0, R)

When the DMA transfer completes for the descriptor either normally (descriptor is retired) or because of an error this interrupt occurs. The cause of the interrupt generation will be recorded in the DD_Status field of the descriptor. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 374. USB End Of Transfer Interrupt Status Register - (USBEoTIntSt - 0x3102 02A0, R)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => There is no End of Transfer Interrupt request for endpoint xx. 1 => There is an End of Transfer Interrupt request for endpoint xx.	0x0
0	EP0	Endpoint 0.	0

15.3.7.24 USB End Of Transfer Interrupt Clear Register - (USBEoTIntClr - 0x3102 02A4, C)

Writing ‘1’ into the register will clear the corresponding interrupt from the status register. Writing ‘0’ will not have any effect. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 375. USB End Of Transfer Interrupt Clear Register - (USBEoTIntClr - 0x3102 02A4, C)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No effect. 1 => Clear the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	0x0
0	EP0	Endpoint 0.	0

15.3.7.25 USB End Of Transfer Interrupt Set Register - (USBEoTIntSet - 0x3102 02A8, S)

Writing ‘1’ into the register will set the corresponding interrupt from the status register. Writing ‘0’ will not have any effect. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 376. USB End Of Transfer Interrupt Set Register - (USBEoTIntSet - 0x3102 02A8, S)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No effect. 1 => Set the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	0x0
0	EP0	Endpoint 0.	0

15.3.7.26 USB System Error Interrupt Status Register - (USBSysErrIntClr - 0x3102 02B8, R)

If a system error (AHB bus error) occurs when transferring the data or when fetching or updating the DD this interrupt bit is set. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 377. USB System Error Interrupt Status Register - (USBSysErrIntClr - 0x3102 02B8, R)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => There is no System Error Interrupt request for endpoint xx. 1 => There is a System Error Interrupt request for endpoint xx.	0x0
0	EP0	Endpoint 0.	0

15.3.7.27 USB System Error Interrupt Clear Register - (USBSysErrIntClr - 0x3102 02BC, C)

Writing ‘1’ into the register will clear the corresponding interrupt from the status register. Writing ‘0’ will not have any effect. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 378. USB System Error Interrupt Clear Register - (USBSysErrIntClr - 0x3102 02BC, C)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No effect. 1 => Clear the EPxx System Error Interrupt request in the USBSysErrIntSt register.	0x0
0	EP0	Endpoint 0.	0

15.3.7.28 USB System Error Interrupt Set Register - (USBSysErrIntSet - 0x3102 02C0, S)

Writing ‘1’ into the register will set the corresponding interrupt from the status register. Writing ‘0’ will not have any effect. The bit field definition is same as the New DD Request Interrupt Status Register as shown in Table 371.

Table 379. USB System Error Interrupt Set Register - (USBSysErrIntSet - 0x3102 02C0, S)

Bits	Name	Function	Reset value
31	EP31	Endpoint 31.	0
xx	EPxx	Where xx can take a value between 1 and 30. 0 => No effect. 1 => Set the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	0x0
0	EP0	Endpoint 0.	0

15.3.8 Protocol engine command description

The protocol engine operates based on the commands issued from the CPU.

These commands have to be written into the Command Code Register. The read data when present will be available in the Command Data register after the successful execution of the command. Table 380 lists all protocol engine commands.

15.3.8.1 Read Current Frame Number command example

Here is an example of the Read Current Frame Number command (reading 2 bytes):

```
USBDevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL int.
USBCmdCode = 0x00F50500;
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY interrupt bit.
USBCmdCode = 0x00F50200;
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
CurFrameNum = USBCmdData;      // Read Frame number LSB byte.
USBDevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL int.
USBCmdCode = 0x00F50200;
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
Temp = USBCmdData;             // Read Frame number MSB byte
USBDevIntClr = 0x20;           // Clear CDFULL interrupt bits.
CurFrameNum = CurFrameNum | (Temp << 8);
```

Table 380. Protocol engine command description

Command Name	Recipient	Command	Data phase (coding)
Device commands			
Set Address	Device	00 D0 05 00	Write 1 byte - 00 <Byte> 01 00
Configure Device	Device	00 D8 05 00	Write 1 byte - 00 <Byte> 01 00
Set Mode	Device	00 F3 05 00	Write 1 byte - 00 <Byte> 01 00
Read Current Frame Number	Device	00 F5 05 00	Read 1 or 2 bytes - 00 F5 02 00
Read Test Register	Device	00 FD 05 00	Read 2 bytes - 00 FD 02 00
Set Device Status	Device	00 FE 05 00	Write 1 byte - 00 <Byte> 01 00
Get Device Status	Device	00 FE 05 00	Read 1 byte - 00 FE 02 00
Get Error Code	Device	00 FF 05 00	Read 1 byte - 00 FF 02 00
ReadErrorStatus	Device	00 FB 05 00	Read 1 byte - 00 FB 02 00
Endpoint commands			
Select Endpoint	Endpoint 0	00 00 05 00	Read 1 byte (optional) - 00 00 02 00
	Endpoint 1	00 01 05 00	Read 1 byte (optional) - 00 01 02 00
	Endpoint 2	00 02 05 00	Read 1 byte (optional) - 00 02 02 00
	Endpoint xx	00 xx 05 00	Read 1 byte (optional) - 00 xx 02 00 xx - Physical endpoint number
	Endpoint 32	00 1F 05 00	Read 1 byte (optional) - 00 1F 02 00
Select Endpoint/Clear Interrupt	Endpoint 0	00 40 05 00	Read 1 byte - 00 40 02 00
	Endpoint 1	00 41 05 00	Read 1 byte - 00 41 02 00
	Endpoint 2	00 42 05 00	Read 1 byte - 00 42 02 00
	Endpoint xx	00 xx 05 00	Read 1 byte - 00 xx 02 00 xx - (Physical endpoint number + 40h)
	Endpoint 31	00 5F 05 00	Read 1 byte - 00 5F 02 00

Table 380. Protocol engine command description *...continued*

Command Name	Recipient	Command	Data phase (coding)
Set Endpoint Status	Endpoint 0	00 40 05 00	Write 1 byte - 00 <Byte> 01 00
	Endpoint 1	00 41 05 00	Write 1 byte - 00 <Byte> 01 00
	Endpoint 2	00 42 05 00	Write 1 byte - 00 <Byte> 01 00
	Endpoint xx	00 xx 05 00	Write 1 byte - 00 <Byte> 01 00 xx - (Physical endpoint number + 40h)
	Endpoint 31	00 5F 05 00	Write 1 byte - 00 <Byte> 01 00
Clear Buffer	Selected Endpoint	00 F2 05 00	Read 1 byte (optional) - 00 F2 02 00
Validate Buffer	Selected Endpoint	00 FA 05 00	None

15.3.8.1.1 Set Address

Command: D0h

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 phase of the setup token. (Alternately, issuing the Set Address command twice will set the address in the device). At power_on reset, the DEV_EN is set to 0. After bus reset, the address is reset to “000_0000”. The enable bit is set. The device will respond on packets for function address “000_0000”, endpoint 0 (default endpoint).

Table 381. Device Set Address Register

Bits	Name	Function	Reset value
7	DEV_EN	Device Enable.	0
6:0	DEV_ADDR	Device address set by the software.	0x0

15.3.8.1.2 Configure Device

Command: D8h

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 382. Configure Device Register

Bits	Name	Function	Reset value
7:1	-	Reserved.	0x0
0	CONF_DEVICE	Device is configured. This bit is set after the set configuration command is executed.	0

15.3.8.1.3 Set Mode

Command: F3h

Data: Write 1 byte

Table 383. Set Mode Register

Bits	Name	Function	Reset value
7:1	-	Reserved.	0
6	INAK_BO[1]	Interrupt on NAK for Bulk OUT endpoints. '0' Only successful transactions generate an interrupt. '1' Both successful and NAKed OUT transactions generate interrupts.	0
5	INAK_BI	Interrupt on NAK for Bulk IN endpoints. '0' Only successful transactions generate an interrupt. '1' Both successful and NAKed IN transactions generate interrupts.	0
4	INAK_IO[2]	Interrupt on NAK for Interrupt OUT endpoints. '0' Only successful transactions generate an interrupt. '1' Both successful and NAKed OUT transactions generate interrupts.	0
3	INAK_II	Interrupt on NAK for Interrupt IN endpoint. '0' Only successful transactions generate an interrupt. '1' Both successful and NAKed IN transactions generate interrupts.	0
2	INAK_CO	Interrupt on NAK for Control OUT endpoint '0' Only successful transactions generate an interrupt '1' Both successful and NAKed OUT transactions generate interrupts.	0
1	INAK_CI	Interrupt on NAK for Control IN endpoint. '0' Only successful transactions generate an interrupt. '1' Both successful and NAKed IN transactions generate interrupts.	0
0	AP_CLK	Always PLL Clock. '0' usb_needclk is functional; 48 Mhz Clock can be stopped when the device enters suspend state. '1' usb_needclk always have the value '1'. 48 Mhz Clock cannot be stopped in case when the device enters suspend state.	0

[1] This bit should be reset to 0 if the DMA is enabled for any of the Bulk OUT endpoints.

[2] This bit should be reset to 0 if the DMA is enabled for any of the Interrupt OUT endpoints.

15.3.8.1.4 Read Current Frame Number

Command: F5h

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.

15.3.8.1.5 Read Test Register

Command: FDh

Data: Read 2 bytes

The test register is 16 bits wide. It returns the value of 0xA50F, if the USB clocks (48 Mhz and hclk) are fine.

15.3.8.1.6 Set Device Status

Command: FEh

Data: Write 1 byte

The Set Device Status command sets bits in the Device Status Register.

Table 384. Set Device Status Register

Bits	Name	Function	Reset value
7:5	-	Reserved.	0x0
4	RST	Bus Reset: The reset bit is set when the device receives a bus reset. It is cleared when read. 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 enabled. Data toggling is reset for all endpoints. All buffers are cleared. There is no change to the endpoint interrupt status. Generate Interrupt (DEV_STAT).	0
3	SUS_CH	Suspend Change: The suspend change bit is set to '1' when the suspend bit toggles. The suspend bit can toggle because: The device goes into the suspended state. The device is disconnected. The device receives resume signalling on its upstream port. The Suspend Change bit is reset after the register has been read. Generate Interrupt (DEV_STAT).	0

Table 384. Set Device Status Register

Bits	Name	Function	Reset value
2	SUS	Suspend: The Suspend bit represents the current suspend state. It is set to '1' when the device hasn't seen any activity on its upstream port for more than 3 ms. It is reset to '0' on any activity. When the device is suspended (suspend bit = '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 (connect bit = '1'). When the device is not connected or not suspended, writing a '0' has no effect. Writing a '1' into this register has never an effect.	0
1	CON_CH	Connect Change: This bit is set when the device's pull-up resistor is disconnected because VBus disappeared. It is reset when read. Generate Interrupt (DEV_STAT).	0
0	CON	Connect: The Connect bit indicates the current connect status of the device. It controls the SoftConnect_N output pin, used for SoftConnect. Writing a '1' will make SoftConnect_N active. Writing a '0' will make SoftConnect_N inactive. Reading the connect bit returns the current connect status.	0

15.3.8.1.7 Get Device Status

Command: FEh

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 384.

It is important to note that when the DEV_STAT status interrupt has been detected in the USB Device Interrupt Status register, the DEV_STAT bit will be set. This interrupt needs to be cleared first by setting the DEV_STAT bit in the "USB Device Interrupt Clear" register before sending the "Get Device Status" command to the protocol engine.

15.3.8.1.8 Get Error Code

Command: FFh

Data: Read 1 bytes

Different error conditions can arise inside the protocol engine. The 'Get Error Code' command returns the error code which last occurred. The 4 least significant bits form the error code.

Table 385. Get Error Code Register

Bits	Name	Function	Reset value
7:5	-	Reserved.	0x0
4	EA	The Error Active bit will be reset once this register is read.	0

Table 385. Get Error Code Register

Bits	Name	Function		Reset value
3:0	EC	Error Code	Description	0x0
		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.	

15.3.8.1.9 ReadErrorStatus

Command: FBh

Data: Read 1 byte

This command reads the 8 bit Error register from the USB device. If any of these bits is set, there will be an interrupt to the CPU. The error bits are reset after reading the register.

Table 386. ReadErrorStatus Register

Bits	Name	Function	Reset value
7	TGL_ERR	Wrong toggle bit in data PID, ignored data.	0
6	BTSTF	Bit stuff error.	0
5	B_OVRN	Buffer Overrun.	0
4	EOP	End of packet error.	0
3	TIMOUT	Time out error.	0
2	DCRC	Data CRC error.	0
1	UEPKT	Unexpected Packet - any packet sequence violation from the specification	0
0	PID_ERR	PID encoding error or Unknown PID or Token CRC.	0

15.3.8.1.10 Select Endpoint

Command: 00-1Fh

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 in the buffer. The command code of 'select endpoint' is equal to the physical endpoint number. In the case of single buffer, B_2_FULL bit is not valid.

Table 387. Select Endpoint Register

Bits	Name	Function	Reset value
7	-	Reserved.	0
6	B_2_FULL	The buffer 2 status '1' = Full; '0' = Empty.	0
5	B_1_FULL	The buffer 1 status '1' = Full; '0' = Empty.	0
4	EPN	EP NAKed '1' - The device has sent 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 to an empty IN buffer, the device returns NAK. This bit is set when a NAK is sent and the interrupt on NAK feature is enabled. This 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.	0
3	PO	Packet over-written: '1': The previously received packet was over-written by a setup packet. The value of this bit is cleared by the 'Select Endpoint/Clear Interrupt' command.	0
2	STP	Setup: '1': The last received packet for the selected endpoint was a setup packet. The value of this bit is updated after each successfully received packet (i.e. an ACKed package on that particular physical endpoint). It is cleared by doing a Select Endpoint/Clear Interrupt on this endpoint.	0
1	ST	'1': The selected endpoint is stalled.	0
0	F/E	Full/Empty: For OUT endpoint if the next read buffers is full this bit is set to 1. For IN endpoint if the next write buffer is empty this bit is set to 0. The F/E bit gives the ORed result of B_1_FULL and B_2_FULL bits.	0

15.3.8.1.11 Select Endpoint/Clear Interrupt

Command: 40-5Fh

Data: Read 1 byte

Commands 40h to 5Fh are identical to their Select Endpoint equivalents, with the following differences:

- They clear the associated interrupt in the USB clock domain only.
- In case of a control out endpoint, they clear the setup and over-written bits.
- Reading one byte is obligatory.

15.3.8.1.12 Set Endpoint Status

Command: 40-5Fhh

Data: Write 1 byte

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 40h and the physical endpoint number in hex value. Not all bits can be set for all types of endpoints.

Table 388. Set Endpoint Status Register

Bits	Name	Function	Reset value
7	CND_ST	Conditional Stall: ‘1’ - Stall both control endpoints, unless the ‘Setup Packet’ bit is set. It is defined only for control OUT endpoints.	0
6	RF_MO	Rate Feedback Mode: ‘0’ - Interrupt endpoint in ‘toggle mode’ ‘1’ - Interrupt endpoint in ‘rate feedback mode’, meaning, transfer takes place without data toggle bit.	0
5	DA	Disabled: ‘1’: The endpoint is disabled.	0
4:1	-	Reserved.	0x0
0	ST	Stalled: ‘1’: The endpoint is stalled. 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 un-stall it. When a stalled endpoint is unstalled - either by the Set Endpoint Statuscommand 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 on the interrupt status of the endpoint. Even when unstalled, setting the stalled bit to ‘0’ initializes the endpoint. When an endpoint is stalled by the Set Endpoint Status command it is also re initialized. The command to set the conditional stall bit will be ignored if the ‘Setup Packet’ bit is set (the EP will not be reset and no status bits will change).	0

15.3.8.1.13 Clear Buffer

Command: F2h

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 CPU has read the data, it should free the buffer and clear the “Buffer Full” bit by using the Clear Buffer command. 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 overwritten bit is used only in control transfers. According to the USB specification, 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 (see Section 15.3.8.1.11), read the new SETUP data and again check the status of the PO bit.

Table 389. Clear Buffer Register

Bits	Name	Function	Reset value
7:1	-	Reserved.	0x0
0	PO	Packet over-written. This bit is only applicable to the control endpoint EP0. 0 - The previously received packet is intact. 1 - The previously received packet was over-written by a later SETUP packet.	0

Here is an example in slave mode when an OUT packet is received on the USB device:

- Set the RD_EN bit and corresponding bits LOG_ENDPOINT number in the “USB Control” register.
- Check the PKT_RDY bit in the “Receive Packet Length” register
- Get the length of the receive packet from the “Receive Packet Length” register when the PKT_RDY bit is set.
- Read data from the “Receive Data” register based on the length.
- Send the “Select Endpoint” command to the protocol engine based on the LOG_ENDPOINT.
- Send the “Clear Buffer” command to the protocol engine for the new incoming packets.

15.3.8.1.14 Validate Buffer

Command: FAh

Data: None

When the CPU has written data into an IN buffer, it should validate the buffer through Command “Validate Buffer”. This will tell the hardware that the buffer is ready for dispatching. The hardware will send the content of the buffer when the next IN token is received. Internally, there is a hardware FIFO status, it has a "Buffer Full" bit. This bit is set by the "Validate Buffer" command, and cleared when the data have been dispatched.

When the CPU has written data into an IN buffer, it should set the buffer full flag by the Validate Buffer command. This indicates that the data in the buffer is valid and can be sent to the host when the next IN token is received.

A control IN buffer cannot be validated when the Packet Over-written bit of its corresponding OUT buffer is set or when the Set up packet is pending in the buffer. For the control endpoint the validated buffer will be invalidated when a Setup packet is received.

Here is an example describing when an IN packet is ready to transmit to the USB host in slave mode:

- Set the WR_EN bit and corresponding bits LOG_ENDPOINT number in the “USB Control” register.
- Set the length of the transmit packet in the “Transmit Packet Length” register.
- Write data to the “Transmit Data” register based on the length.

- Send the “Select Endpoint” command to the protocol engine based on the LOG_ENDPOINT.
- Send the “Validate Buffer” command to the protocol engine and tell the hardware that the buffer is ready to dispatch.

15.3.9 DMA descriptor

A DMA transfer can be characterized by a structure describing these parameters. This structure is called the DMA Descriptor (DD).

The DMA descriptors are placed in the USB RAM. These descriptors can be located anywhere in the USB RAM in the word-aligned boundaries. The USB RAM is part of the system memory which is used for the USB purposes. It can be located in IRAM or memory attached to the EMC controller.

DD for non-isochronous endpoints are four-word long and isochronous endpoints are five-word long.

Total USB RAM required for DD = (No. of non-iso endpoints x 4 + No. of iso endpoints x5)

There are certain parameters associated with a DMA transfer. These are:

- The start address of the DMA buffer in the USB RAM.
- The length of the DMA Buffer in the USB RAM.
- The start address of the next DMA buffer.
- Control information.
- DMA count information (Number of bytes transferred).
- DMA status information.

Table 390 lists the DMA descriptor fields.

Table 390. DMA descriptor

Word position	Access (H/W)[1]	Access (S/W)	Bit position	Description
0	R	R/W	31:0	Next_DD_pointer (USB RAM address).
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.
	R	R/W	4	Isochronous_endpoint (1 - isochronous; 0 - non-isochronous).
	R	R/W	15:5	Max_packet_size
	R/W*	R/W	31:16	DMA_buffer_length in bytes.
2	R/W	R/W	31:0	DMA_buffer_start_addr.

Table 390. DMA descriptor *…continued*

Word position	Access (H/W)[1]	Access (S/W)	Bit position	Description
3	R/W	R/I	0	DD_retired (To be initialized to 0).
	W	R/I	4:1	DD_status (To be initialized to 0) 0000 - Not serviced. 0001 - Being serviced. 0010 - Normal completion. 0011 - Data under run (short packet). 1000 - Data over run. 1001 - System error.
	R/W	R/I	5	Packet_valid (To be initialized to 0).
	R/W	R/I	6	LS_byte_extracted (ATLE mode) (To be initialized to 0).
	R/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.
	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] R - Read; W - Write; W* - Write only in ATLE mode; I - Initialize

15.3.9.1 Next_DD_pointer

Pointer to the memory location from where the next DMA descriptor has to be fetched.

15.3.9.2 DMA_mode

Defines in which mode the DMA has to operate. Two modes have been defined, Normal and ATLE. In the normal mode the DMA engine will not split a packet into two different DMA buffers. In the ATLE mode splitting of the packet into two buffers can happen. This is because two transfers can be concatenated in the packet to improve the bandwidth. See Section 15.4.3.2 “Concatenated transfer (ATLE) mode operation” on DMA operation.

15.3.9.3 Next_DD_valid

This bit indicates whether the software has prepared the next DMA descriptor. If it is valid, the DMA engine once finished with the current descriptor will load the new descriptor.

15.3.9.4 Isochronous_endpoint

The descriptor belongs to an isochronous endpoint. Hence, 5 words have to be read.

15.3.9.5 Max_packet_size

This field is the maximum packet size of the endpoint. This parameter has to be 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. The max_packet_size field should be the same as the value set in the MaxPacketsize register for the endpoint.

15.3.9.6 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. This will be set by the software in the normal mode operation for both IN and OUT endpoints. In the ATLE mode operation the buffer_length is set by software for IN endpoints. For OUT endpoints this is set by the hardware from the extracted length of the data stream. In case of the Isochronous endpoints the DMA_buffer_length is specified in terms of number of packets.

15.3.9.7 DMA_buffer_start_addr

The address from where the data has to be picked up or to be stored. This field is updated packet-wise by DMA engine.

15.3.9.8 DD_retired

This bit is set when the DMA engine finishes the current descriptor. This will happen when the end of the buffer is reached or a short packet is transferred (no isochronous endpoints) or an error condition is detected.

15.3.9.9 DD_status

The status of the DMA transfer is encoded in this field. The following status are defined.

- **Not serviced** - No packet has been transferred yet. DD is in the initial position itself.
- **Being serviced** - This status indicates that at least one packet is transferred.
- **Normal completion** - The DD is retired because the end of the buffer is reached and there were no errors. DD_retired bit also is set.
- **Data under run** - Before reaching the end of the buffer, transfer is terminated because a short packet is received. DD_retired bit also is set.
- **Data over run** - End of the DMA buffer is reached in the middle of a packet transfer. This is an error situation. DD_retired bit will be set. The DMA count will show the value of DMA buffer length. The packet has to be re-transmitted from the FIFO. DMA_ENABLE bit is reset.
- **System error** - Transfer is terminated because of an error in the system bus. DD_retired bit is not set in this case. DMA_ENABLE bit is reset. Since system error can happen while updating the DD, the DD fields in the USB RAM may not be very reliable.

15.3.9.10 Packet_valid

This bit indicates whether the last packet transferred to the memory is received with errors or not. This bit will be set if the packet is valid, i.e., it was received without errors. Since non-isochronous endpoint will not generate DMA request for packet with errors, this field will not make much sense as it will be set for all packets transferred. But for isochronous endpoints this information is useful. See Section 15.4.4 for isochronous endpoint operation.

15.3.9.11 LS_byte_extracted

Applicable only in the ATLE mode. This bit set indicates that the Least Significant Byte (LSB) of the transfer length has been already extracted. The extracted size will be reflected in the ‘dma_buffer_length’ field in the bits 23:16.

15.3.9.12 MS_byte_extracted

Applicable only in the ATLE mode. This bit set indicates that the Most Significant Byte (MSB) of the transfer size has been already extracted. The size extracted will be reflected in the ‘dma_buffer_length’ field at 31:24. Extraction stops when both ‘LS_Byte_extracted’ and ‘MS_byte_extracted’ fields are set.

15.3.9.13 Present_DMA_count

The number of bytes transferred by the DMA engine at any point of time. This is updated packet-wise by the DMA engine when it updates the descriptor. In case of the Isochronous endpoints the Present_DMA_count is specified in terms of number of packets transferred.

15.3.9.14 Message_length_position

This applies only in the ATLE mode. This field gives the offset of the message length position embedded in the packet. This is applicable only for OUT endpoints. Offset 0 indicates that the message length starts from the first byte of the packet onwards.

15.3.9.15 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 59. This is applicable to isochronous endpoints only.

15.4 DMA operation

15.4.1 Triggering the DMA engine

An endpoint will raise a DMA request when the slave mode transfer is disabled by setting the corresponding bit in "Endpoint Interrupt Enable" register to 0.

The DMA transfer for an OUT endpoint is triggered when it receives a packet without any errors (i.e., the buffer is full) and the ‘DMA_ENABLE’ (EP DMA Status register) bit is set for this endpoint.

Transfer for an IN endpoint is triggered when the host requests for a packet of data and the ‘DMA_ENABLE’ bit is set for this endpoint.

In DMA mode, the bits corresponding to Interrupt on NAK for Bulk OUT and Interrupt OUT endpoints (bit INAK_BO and INAK_IO) in Set Mode register (Section 15.3.8.1.3) should be reset to 0.

15.4.2 Arbitration between endpoints

If more than one endpoint is requested for data transfer at the same time, the endpoint with lower physical endpoint number value gets the priority.

15.4.3 Non isochronous endpoint operation

15.4.3.1 Normal mode operation

15.4.3.1.1 Setting up DMA transfer

The software prepares the DDs for the physical endpoints that need DMA transfer. These DDs are present in the USB RAM. Also, the start address of the first DD is programmed into the DDP location for the corresponding endpoint. The software will then set the DMA_ENABLE bit for this endpoint in the EP DMA Status register. The ‘DMA_mode’ bit in the descriptor has to be set to ‘00’ for normal mode operation. It should also initialize all the bits in the DD as given in the table.

15.4.3.1.2 Finding DMA descriptor

When there is a trigger for a DMA transfer for an endpoint, DMA engine will first determine whether a new descriptor has to the fetched or not. A new descriptor need not have to be fetched if the last transfer was also made for the same endpoint and the DD is not yet in the ‘retired’ state. A flag called ‘DMA_PROCEED’ is used to identify this (see Section 15.4.3.1.4).

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 DD from this location. A DD start address at location zero is considered invalid. In this case a ‘new_dd_request’ interrupt is raised. All other word boundaries are valid.

If at any point of time the DD is to be fetched, the status of DD (word 3) is read first and the status of the ‘DD_retired’ bit is checked. If this is not set, DDP points to a valid DD. If the ‘DD_retired’ bit is set, the DMA engine will read the ‘control’ field (word 1) of the DD.

If the bit ‘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 DMA descriptor (4 words) will in turn be fetched from this address pointed by DDP. The DD will give the details of the 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 the ‘next_dd_valid’ is not set and the DD_retired bit is set, the DMA engine will raise the ‘NEW_DD_REQUEST’ interrupt for this endpoint. It also disables the DMA_ENABLE bit.

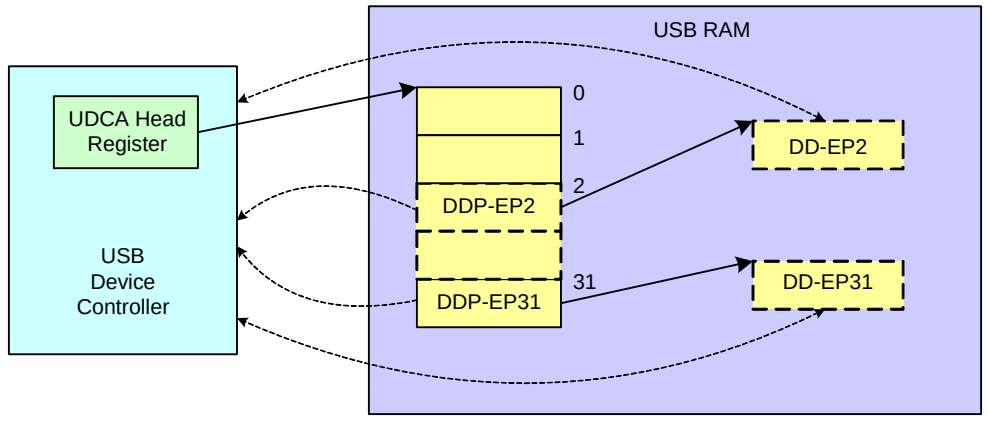


Fig 57. Finding the DMA descriptor

15.4.3.1.3 Transferring the data

In case of OUT endpoints, the current packet will be read from the EP_RAM by the DMA Engine and will get transferred to the USB RAM memory locations starting from the address pointed by ‘dma_buffer_start_addr’. In case of IN endpoints, the data will be fetched from the USB RAM and will be written to the EP_RAM. The ‘dma_buffer_start_addr’ and ‘present_dma_count’ will get updated while the transfer progresses.

15.4.3.1.4 Optimizing descriptor fetch

A DMA transfer normally involves multiple packet transfers. If a DD once fetched is equipped to do multiple transfers, the hardware will not fetch DD for all the succeeding packets. It will do the fetching only if the previous packet transferred on this channel does not belong to this endpoint. This is on the assumption that the current contents of the hardware resource and that of the descriptor to be fetched will be the same. In such a case DMA engine can proceed without fetching the new descriptor if it has not transferred enough data specified in the ‘dma_buffer_length’ field of the descriptor. To keep this information the hardware will have a flag set called ‘DMA_PROCEED’.

This flag will be reset after the required number of bytes specified in the ‘dma_buffer_length’ field is transferred. It is also reset when the software writes into the EP DMA Disable register. This will give the software control over the reading of DD by the hardware. Hardware will be forced to read the DD for the next packet. Writing data 0x0 into the EP DMA Disable register will cause only resetting of the DMA_PROCEED flag without disabling DMA for any endpoint.

15.4.3.1.5 Ending the packet transfer

The DMA engine will write back the DD with an updated status to the same memory location from where it was read. The ‘dma_buffer_start_addr’, ‘present_dma_count’ and the status bits field in the DD get updated. Only words 2 and 3 are updated by hardware in this mode.

A DD can have the following types of completion:

Normal completion: If the current packet is fully transferred and the ‘dma_count’ field equals the ‘dma_buffer_length’ defined in the descriptor, the DD has a normal completion. The DD will be written back to memory with ‘DD_retired’ bit set. END_OF_TRANSFER interrupt is raised for this endpoint. DD_Status bits are updated for ‘normal_completion’ code.

Transfer end completion: If the current packet is fully transferred, its size is less than the ‘max_packet_size’ defined in the descriptor, and the end of the buffer is still not reached the transfer end completion occurs. The DD will be written back to the memory with ‘DD_retired’ bit set and DD_Status bits showing ‘data under run’ completion code. Also, the ‘END_OF_TRANSFER’ interrupt for this endpoint is raised.

Error completion: If the current packet is partially transferred i.e. 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_status ‘data over run’ and ‘DD_retired’ bit is set. The DMA engine will raise the End of Transfer interrupt and reset the corresponding bit for this endpoint in the ‘DMA_ENABLE’ register. This packet will be re-transmitted to the memory fully when DMA_ENABLE bit is set again by writing into the EP DMA Enable register.

15.4.3.1.6 No_Packet DD

For IN transfers, it can happen that for a DMA request the system does not have any data to send for a long time. The system can suppress this request by programming a no_packet DD. This is done by setting the ‘Maxpacketsize’ and ‘dma_buffer_length’ in the DD control field to 0. No packets will be sent to the host in response to the no_packet DD.

15.4.3.2 Concatenated transfer (ATLE) mode operation

Some host drivers like ‘NDIS’ (Network Driver Interface Standard) are capable of concatenating small transfers (delta transfers) to form a single large transfer. The device hardware should be able to break up this single transfer back into delta transfers and transfer them to different DMA buffers. This is achieved in the ATLE mode operation. This is applicable only for Bulk endpoints.

In ATLE mode, the Host driver can concatenate various transfer lengths, which correspond to different DMA descriptors on Device side. And these transfers have to be done on USB without breaking the packet. This is the primary difference between the Normal Mode and ATLE mode of DMA operation, wherein one DMA transfer length ends with either a full USB packet or a short packet and the next DMA transfer length starts with a new USB packet in Normal mode. These two transfers may be concatenated in the last USB packet of the first DMA transfer in ATLE mode.

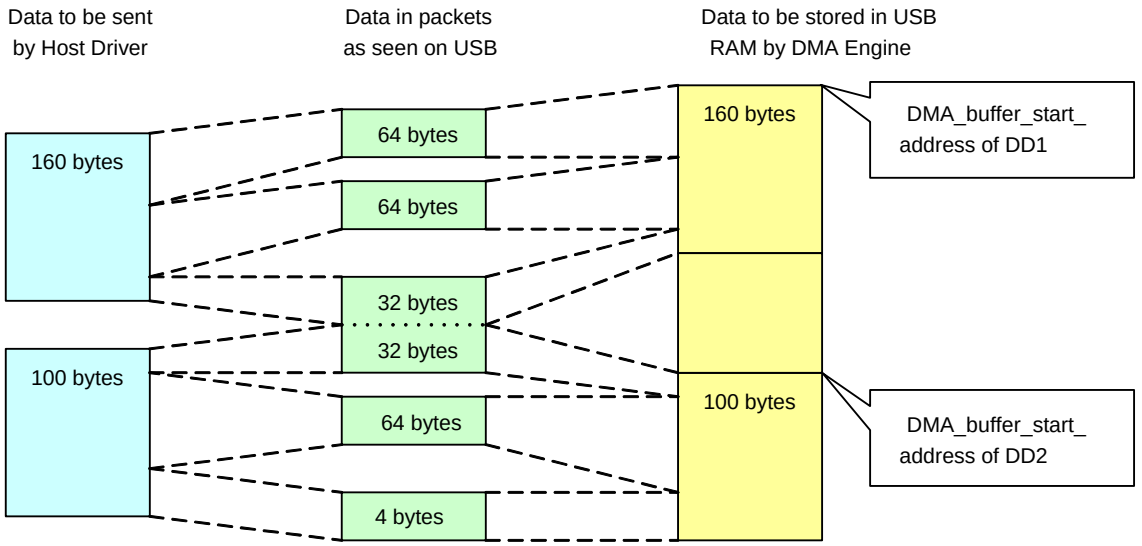


Fig 58. Data transfer in ATLE mode

15.4.3.2.1 OUT transfer in ATLE mode

Figure 58 shows a typical OUT transfer, where the host concatenates two DMA transfer lengths of 160 bytes and 100 bytes respectively. As seen on USB, there would be four packets of 64 bytes (MPS=64) and a short packet of 4 bytes in ATLE mode unlike Normal mode with five packets of 64, 64, 32, 64, 36 bytes in the given order.

It is now responsibility of the DMA engine to separate these two transfers and put them in proper memory locations as pointed by the "DMA_buffer_start_address" field of DMA Descriptor 1 (DD1) and DMA Descriptor 2 (DD2).

There are two things in OUT transfer of ATLE mode which differentiate it from the OUT transfer in Normal mode of DMA operation. The first one is that the Device software does not know the "DMA_buffer_length" of the incoming transfer and hence this field in DD is programmed to 0. But by the NDIS protocol, the device driver knows at which location in the incoming data transfer the transfer length will be stored. This value is programmed in the field "Message_length_position" of the DD.

It is responsibility of the hardware to read the two byte wide "DMA_buffer_length" at the offset (from start of transfer) specified by "Message_length_position", from incoming data and write it in "DMA_buffer_length" field of the DD. Once this information is extracted from the incoming data and updated in the DD, the transfer continues as in Normal mode of operation.

It may happen that the message length position points to the last byte in the USB packet, which means that out of two bytes of buffer length, first (LS) byte is available in the current packet and the second (MS) byte would follow in the next packet. To deal with such situations, the flags "LS_byte_extracted" and "MS_byte_extracted" are used by hardware. When the hardware reads the LS byte (which is the last byte of USB packet), it writes the contents of LS byte in position [23:16] of "DMA_buffer_length" field, sets the flag "LS_byte_extracted" to 1, and updates the DD in System memory (since the packet transfer is over).

On reception of the next packet, looking at "LS_byte_extracted" field 1 and "MS_byte_extracted" field 0, hardware knows that it has to read the first incoming byte as MS byte of buffer length, update the position (31:24) of "DMA_buffer_length" with the read contents and set the flag "MS_byte_extracted". After the extraction of MS byte of DMA buffer length, the transfer continues as in Normal mode of operation.

The second thing, which differentiates the ATLE mode OUT transfer from Normal mode OUT transfer, is the behavior in case when DD is retired in between a USB packet transfer.

As can be seen in Figure 58, the first 32 bytes of the 3rd packet correspond to DD1 and the remaining 32 bytes correspond to DD2. In such a situation, on reception of first 32 bytes, the first DD (i.e. DD1) is retired and updated in the system memory, the new DD (pointed by "next_DD_pointer") is fetched and the remaining 32 bytes are transferred to the location in system memory pointed by "DMA_buffer_start_address" of new DD (i.e. DD2).

It should be noted that in ATLE mode, the software will always program the "LS_byte_extracted" and "MS_byte_extracted" fields to 0 while preparing a DD, and hence on fetching the DD2 in above situation, the Buffer Length Extraction process will start again as described earlier.

If the first DD is retired in between the packet transfer and the next DD is not programmed, i.e. "next_DD_valid" field in DD1 is 0, then the first DD is retired with the status "data over run" (DD_status = 1000), which has to be treated as an error condition and the DMA channel for that particular endpoint is disabled by the hardware. Otherwise the first DD is retired with status "normal completion" (DD_status = 0010).

Please note that in this mode the last buffer length to be transferred would always end with a short packet or empty packet indicating that no more concatenated data is coming on the way. If the concatenated transfer lengths are such that the last transfer ends on a packet boundary, the (NDIS) host will send an empty packet to mark the End Of Transfer.

15.4.3.2.2 IN transfer in ATLE mode

The operation in IN transfers is relatively simple compared to the OUT transfer in ATLE mode since device software knows the buffer length to be transferred and it is programmed in "DMA_buffer_length" field while preparing the DD, thus avoiding any transfer length extraction mechanism.

The only difference for IN transfers between ATLE mode and Normal mode of DMA operation is that the DDs can get retired in mid of the USB packet transfer. In such a case, the hardware will update the first DD in system memory, fetch the new DD pointed by "next_DD_pointer" field of the first DD, and fetch the remaining bytes from system memory pointed by "DMA_buffer_start_address" of second DD to complete the packet before sending it on USB.

In the above situation, if the next DD is not programmed, i.e. "next_DD_valid" field in DD is 0, and the buffer length for current DD has completed before the packet boundary, then the available bytes from current DD are sent as a short packet on USB, which marks the End Of Transfer for the Host.

In cases where the intended buffer lengths are already transferred and the last buffer length has completed on the USB packet boundary, it is responsibility of Device software to program the next DD with "DMA_buffer_length" field 0, after which an empty packet is sent on USB by the hardware to mark the End Of Transfer for the Host.

15.4.3.2.3 Setting up the DMA transfer

There is an additional field in the descriptor called ‘message_length_position’ which has to be set for the OUT endpoints. This indicates the start location of the message length in the incoming data packet. Also the software will set the ‘dma_buffer_length’ field to ‘0’ for OUT endpoints as this field has to be updated by hardware.

For IN endpoints, descriptors are to be set in the same way as the normal mode operation.

Since a single packet can have two transfers which have to be transferred or collected from different DMA buffers, the software should keep two buffers ready always, except for the last delta transfer which ends with a short packet.

15.4.3.2.4 Finding the DMA descriptor

DMA descriptors are found in the same way as in the normal mode operation.

15.4.3.2.5 Transferring the data

For OUT end points 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. The ‘dma_buffer_length’ field derived from this information is 2 bytes long. Once the extraction is complete, both the ‘LS_byte_extracted’ and ‘MS_byte_extracted’ bits will be set.

For IN endpoints transfer proceeds like the normal mode and continues till the number of bytes transferred equals the ‘dma_ buffer_length’.

15.4.3.2.6 Ending the packet transfer

DMA engine proceeds with the transfer till the number of bytes specified in the field ‘dma_buffer_length’ gets transferred to or from the USB RAM. An END_OF_TRANSFER 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.

For an OUT endpoint if the linked DD is not valid and the packet is partially transferred to memory, the DD ends with data_over_run status set and DMA will be disabled for this endpoint. Otherwise DD_status will be updated with ‘normal completion’.

For an IN endpoint if the linked DD is not valid and the packet is partially transferred to USB, DD ends with ‘normal completion’ and the packet will be sent as a short packet (since this situation is the end of transfer). Also, when the linked DD is valid and buffer length is 0, a short packet will be sent.

15.4.4 Isochronous endpoint operation

In case of isochronous endpoint operation the packet size can vary on each and every packet. There will be one packet per isochronous endpoint at every frame.

15.4.4.1 Setting up the DMA transfer

For the Isochronous DMA descriptor, the DMA length equals the number of frames for the transfer rather than the number of bytes. The DMA count is also updated in terms of the number of frames

15.4.4.1.1 Finding the DMA descriptor

Finding the descriptor is done in the same way as that for a non isochronous endpoint.

DMA descriptor has a bit field in the word 1 (isochronous_endpoint) to indicate that the descriptor belongs to an isochronous endpoint. Also, isochronous DD has a fifth word showing where the packet length for the frame has to be put (for OUT endpoint) or from where it has to be read.

A DMA request will be placed for DMA enabled isochronous endpoints on every frame interrupt. For a DMA request the DMA engine will fetch the descriptor, and if it identifies that the descriptor belongs to an Isochronous endpoint, it will fetch the fifth word of the DD which will give the location from where the packet length has to be placed or fetched.

15.4.4.2 Transferring the data

The data is transferred to or from the memory location pointed by the dma_buffer_start_addr. After the end of the packet transfer the dma_count value is incremented by 1.

For an OUT transfer a word is formed by combining the frame number and the packet length such that the packet length appears at the least significant 2 bytes (15 to 0). Bit 16 shows whether the packet is valid or not (set when packet is valid i.e. it was received without any errors). The frame number appears in the most significant 2 bytes (bit 31 to 17). The frame number is available from the USB device. This word is then transferred to the address location pointed by the variable Isochronous_packet_size_memory_address. The Isochronous_packet_size_memory_address is incremented by 4 after receiving or transmitting an Isochronous data packet. The Isochronous_packet_size memory buffer should be big enough to hold information of all packets sent by the host.

For an IN endpoint only the bits from 15 to 0 are applicable. An Isochronous data packet of size specified by this field is transferred from the USB device to the Host in each frame. If the size programmed in this location is zero an empty packet will be sent by the USB device.

The Isochronous endpoint works only in the normal mode DMA operation.

An Isochronous endpoint can have only ‘normal completion’ since there is no short packet on Isochronous endpoint and the transfer continues infinitely till a system error occurs. Also, there is no data_over_run detection.

15.4.4.2.1 Isochronous OUT endpoint operation example

For example assume that an isochronous endpoint is programmed for the transfer of 10 frames. After transferring four frames with packet size 10,15, 8 and 20 bytes: the descriptors and memory map looks as shown in Figure 59, assuming that the transfer starts when the internal frame number was 21.

The total number of bytes transferred = 0xA + 0xF + 0x8 + 0x14 = 0x35.

The sixteenth bit for all the words in the packet length memory will be set to 1.

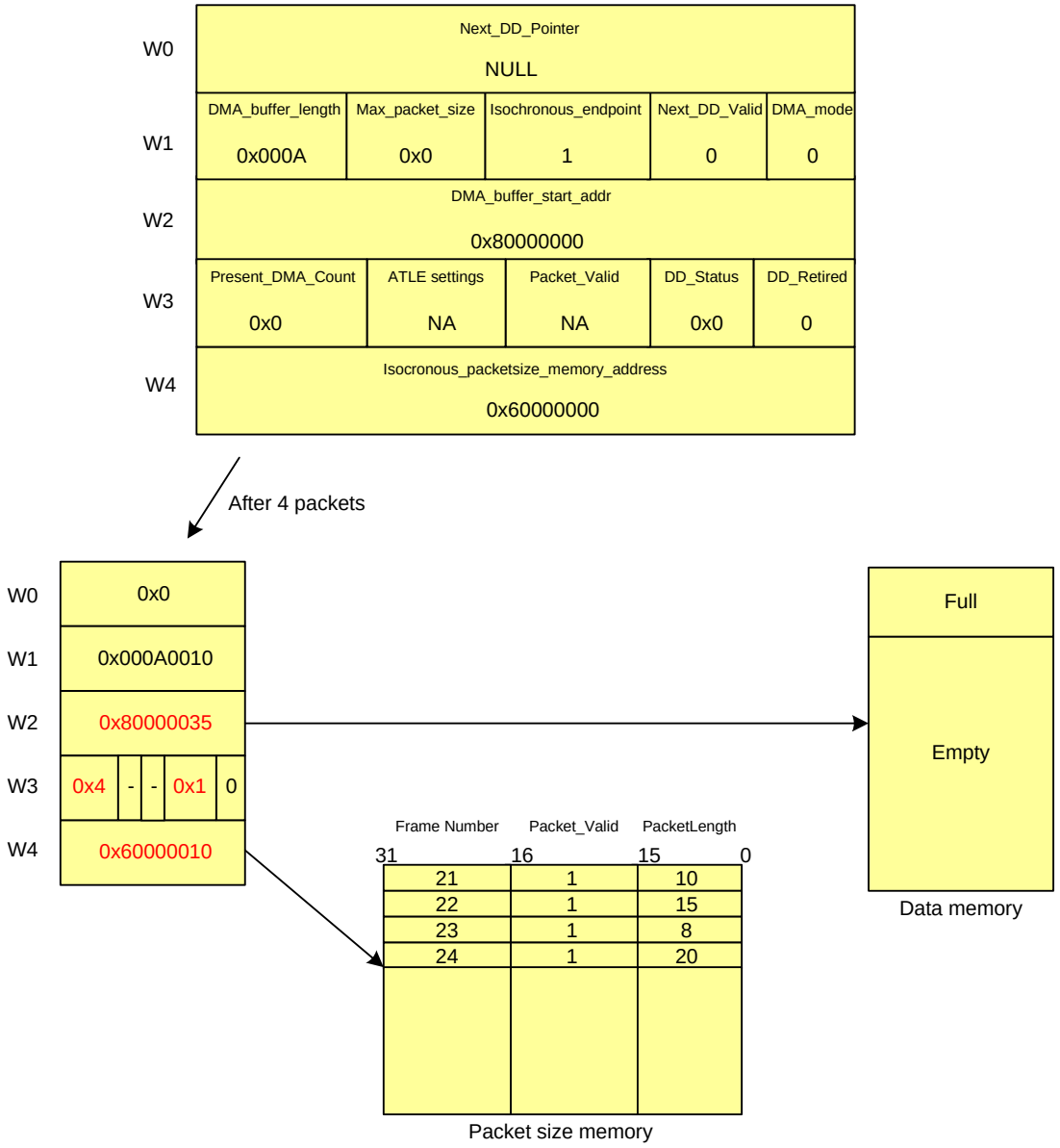


Fig 59. Isochronous OUT endpoint operation example

16.1 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 I2C. The I2C 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 391. 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

16.1.1 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.

16.1.2 Architecture

The architecture of the USB host controller is shown below in Figure 60.

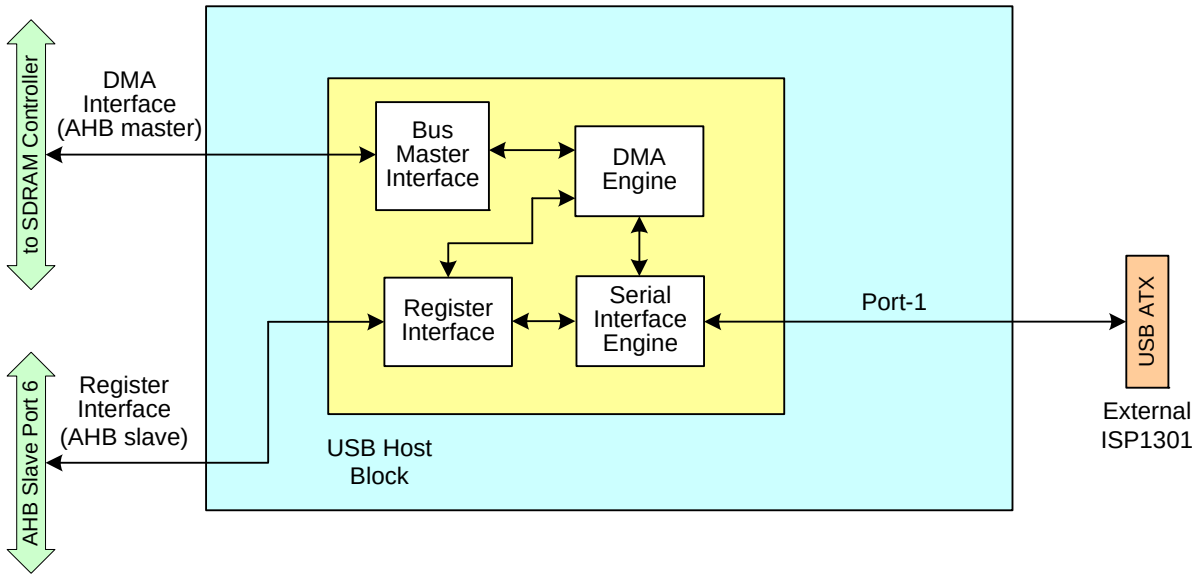


Fig 60. USB Host Controller Block Diagram

16.2 Interfaces

16.2.1 Pin description

Table 392. USB external interface

Name	Direction	Description
USB_I2C_SDA	I/OT	I^2C serial bus data[1]
USB_I2C_SCL	I/OT	I^2C serial bus clock[1]
USB_ATX_INT_N	I	Interrupt from transceiver
USB_OE_TP_N	I/O	Transmit enable for DAT/SE0
USB_DAT_VP	I/O	TX data / D+ receive
USB_SE0_VM	I/O	S. E. Zero transmit / D− receive

[1] Open drain pin requiring an external pull-up resistor

16.2.2 Software interface

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

16.2.2.1 Register map

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bit wide and aligned in the word address boundaries.

Table 393. USB Host register address definitions

Name	Address	R/W[1]	Function	Reset value
HcRevision	0x3102 0000	R	BCD representation of the version of the HCI specification that is implemented by the Host Controller.	0x10
HcControl	0x3102 0004	R/W	Defines the operating modes of the HC.	0x0
HcCommandStatus	0x3102 0008	R/W	This register is used to receive the commands from the Host Controller Driver (HCD). It also indicates the status of the HC.	0x0
HcInterruptStatus	0x3102 000C	R/W	Indicates the status on various events that cause hardware interrupts by setting the appropriate bits.	0x0
HcInterruptEnable	0x3102 0010	R/W	Controls the bits in the HcInterruptStatus register and indicates which events will generate a hardware interrupt.	0x0
HcInterruptDisable	0x3102 0014	R/W	The bits in this register are used to disable corresponding bits in the HCInterruptStatus register and in turn disable that event leading to hardware interrupt.	0x0
HcHCCA	0x3102 0018	R/W	Contains the physical address of the host controller communication area.	0x0
HcPeriodCurrentED	0x3102 001C	R	Contains the physical address of the current isochronous or interrupt endpoint descriptor.	0x0
HcControlHeadED	0x3102 0020	R/W	Contains the physical address of the first endpoint descriptor of the control list.	0x0
HcControlCurrentED	0x3102 0024	R/W	Contains the physical address of the current endpoint descriptor of the control list	0x0
HcBulkHeadED	0x3102 0028	R/W	Contains the physical address of the first endpoint descriptor of the bulk list.	0x0
HcBulkCurrentED	0x3102 002C	R/W	Contains the physical address of the current endpoint descriptor of the bulk list.	0x0
HcDoneHead	0x3102 0030	R	Contains the physical address of the last transfer descriptor added to the ‘Done’ queue.	0x0
HcFmInterval	0x3102 0034	R/W	Defines the bit time interval in a frame and the full speed maximum packet size which would not cause an overrun.	0x2EDF
HcFmRemaining	0x3102 0038	R	A 14-bit counter showing the bit time remaining in the current frame.	0x0
HcFmNumber	0x3102 003C	R	Contains a 16-bit counter and provides the timing reference among events happening in the HC and the HCD.	0x0
HcPeriodicStart	0x3102 0040	R/W	Contains a programmable 14-bit value which determines the earliest time HC should start processing a periodic list.	0x0
HcLSThreshold	0x3102 0044	R/W	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.	0x628h
HcRhDescriptorA	0x3102 0048	R/W	First of the two registers which describes the characteristics of the root hub.	0xFF000902
HcRhDescriptorB	0x3102 004C	R/W	Second of the two registers which describes the characteristics of the Root Hub.	0x60000h

Table 393. USB Host register address definitions *…continued*

Name	Address	R/W[1]	Function	Reset value
HcRhStatus	0x3102 0050	R/W	This register is divided into two parts. The lower D-word represents the hub status field and the upper word represents the hub status change field.	0x0
HcRhPortStatus[1]	0x3102 0054	R/W	Controls and reports the port events on a per-port basis.	0x0
HcRhPortStatus[2]	0x3102 0058	R/W	Controls and reports the port events on a per port basis.	0x0

[1] The R/W column in Table 393 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.

16.2.2.2 USB Host Register Definitions

Refer to the *OHCI specification document* for register definitions.

17.1 Introduction

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.org website.

17.1.1 Features

- Fully compliant with On-The-Go supplement to the USB Specification 2.0 Revision 1.0.
- Supports Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) for dual-role devices under software control. HNP is partially implemented in hardware.
- Provides programmable timers required for HNP and SRP.
- Supports slave mode operation through AHB slave interface.
- Supports the OTG ATX from NXP (ISP 1301) or any external CEA-2011OTG specification compliant ATX.

17.1.1.1 Architecture

The architecture of the USB OTG controller is shown below in Figure 61.

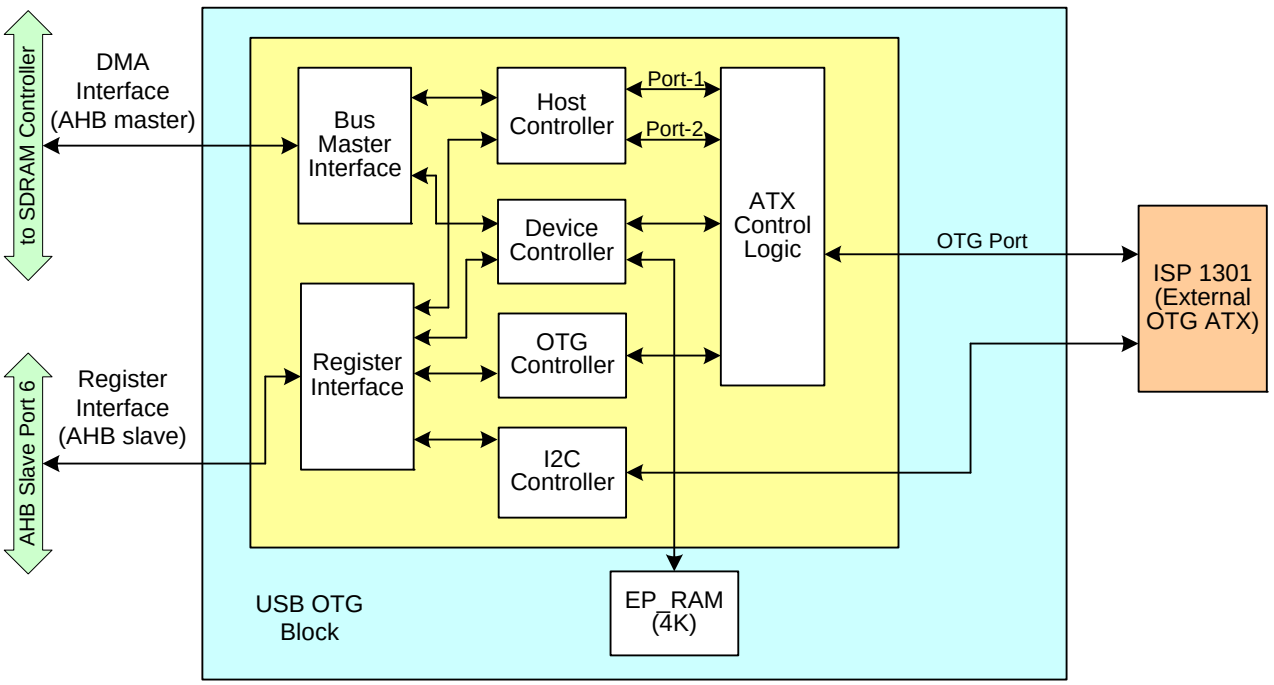


Fig 61. USB OTG controller block diagram

17.2 Modes of operation

Under software commands, the OTG controller is capable of operating in the following modes:

- USB OTG dual role device
- One port OHCI host (FS and LS)
- One port host or one port device

17.2.1 Pin description

Table 394. USB external interface

Name	Direction	Description
USB_I2C_SDA	I/OT	I^2C serial bus data[1]
USB_I2C_SCL	I/OT	I^2C serial bus clock[1]
USB_ATX_INT_N	I	Interrupt from transceiver
USB_OE_TP_N	I/O	Transmit enable for DAT/SE0
USB_DAT_VP	I/O	TX data / D+ receive
USB_SE0_VM	I/O	S. E. Zero transmit / D− receive

[1] Open drain pin requiring an external pull-up resistor

17.2.2 Software interface

The USB OTG controller contains a number of registers that are software programmable from the AHB slave system bus to determine configuration, control and status. All the registers are placed in the word aligned boundary. These are described as Device, Host, OTG and I^2C registers. The Device and Host registers are explained in the *USB device controller* and *USB host (OHCI) controller* chapters.

17.2.3 Interrupts

The USB OTG controller has seven interrupt output lines. The interrupts usb_dev_Ip_int and usb_dev_hp_int facilitate the transfer of data in slave mode. These two interrupt lines are provided to allow two different priority (high/low) levels in slave mode transfer. Each of the individual endpoint interrupts can be routed to either high priority or low priority levels using corresponding bits in the endpoint interrupt priority register. The interrupt level is triggered with active HIGH polarity. The external interrupt generation takes place only if the necessary ‘enable’ bits are set in the device interrupt enable register. Otherwise, they will be registered only in the status registers. The usb_dev_dma_int is raised when an end_of_transfer or a system error has occurred. DMA data transfer is not dependent on this interrupt. The interrupt usb_host_int is from the host block. The interrupt usb_i2c_int is from the I^2C block. The interrupt usb_otg_atx_int_n is from the external transceiver. The interrupt USB_otg_timer_int is from the timer block. Device and Host interrupts also contribute to the USB_INT which can act as a start source in STOP mode. usb_i2c_int, usb_otg_atx_int_n, and USB_otg_timer_int can also act as a start source in STOP mode.

17.2.3.1 Register map

The following registers are located in the AHB clock domain. They can be accessed directly by the CPU. All registers are 32 bit wide and aligned on word address boundaries.

USB OTG registers are located in the address region 0x3102 0100 to 0x3102 0114. OTG Clock Control registers are located in the address region 0x3102 0FF4 to 0x3102 0FFC. I^2C registers are located in the address region 0x3102 0300 to 0x3102 0310.

Table 395. USB OTG and I^2C register address definitions

Name	Address	R/W[1]	Function
OTG registers			
OTG_int_status	0x3102 0100	R	This register holds the status of the OTG interrupts
OTG_int_enable	0x3102 0104	R/W	This register is used for enabling the OTG interrupts
OTG_int_set	0x3102 0108	S	This register is used for setting the interrupts
OTG_int_clear	0x3102 010C	C	This register is used for clearing the interrupts
OTG_status	0x3102 0110	R/W	This register is used to monitor and control the operation of the OTG controller
OTG_timer	0x3102 0114	R/W	Timer to be used for various OTG time-out activities
I^2C registers			
I2C_RX	0x3102 0300	R	Receive FIFO
I2C_TX	0x3102 0300	W	Transmit FIFO
I2C_STS	0x3102 0304	R	Status
I2C_CTL	0x3102 0308	R/W	Control
I2C_CLKHI	0x3102 030C	R/W	Clock division high, set to run min frequency
I2C_CLKLO	0x3102 0310	W	Clock division low, set to run min frequency
Clock control registers			
OTG_clock_control	0x3102 0FF4	R/W	Controls clocking of the OTG controller
OTG_clock_status	0x3102 0FF8	R	Clock availability status

[1] The R/W column in Table 395 lists the accessibility of the register:

a) Registers marked ‘R’ for access will return their current value when read.

b) Registers marked ‘S’ for access allows individual bits to be set to ‘1’ for each corresponding register bit. Bits set to ‘0’ will not affect the value of the corresponding register bit. Reading an ‘S’ marked register will return an invalid value.

c) Registers marked ‘C’ for access allows individual bits to be cleared by writing a value that has bits set to ‘1’ for each corresponding register bit that needs to be set to ‘0’. Bits set to ‘0’ will not affect the value of the corresponding register bit. Reading a ‘C’ marked register will return invalid value.

d) Registers marked ‘R/W’ allow both read and write.

17.2.3.2 USB OTG Register Definitions

17.2.3.2.1 OTG interrupt status register[20] - (OTG_int_status - 0x3102 0100, R)

Table 396. OTG interrupt status register - (OTG_int_status - 0x3102 0100, R)

Bits	Name	Function	Reset value
31:4	-	Reserved	-
3	hnp_success	Set by the hardware when the interrupt event occurs. When software writes a value '1' into the OTG_int_clear register bit '3', this will be cleared by the hardware. If 'hnp_success_en' bit is set to '1', then the value in this register will be reflected on to the interrupt line. Refer to Section 17.2.3.3 "OTG switching" for details.	0
2	hnp_failure	Set by the hardware when the interrupt event occurs. When software writes a value '1' into the OTG_int_clear register bit '3', this will be cleared by the hardware. If 'hnp_failure_en' bit is set to '1', then the value in this register will be reflected on to the interrupt line. Refer to Section 17.2.3.3 "OTG switching" for details.	0
1	remove_pullup	Set by the hardware when the interrupt event occurs. When software writes a value '1' into the OTG_int_clear register bit '1', this will be cleared by the hardware. If 'Remove_pullup_en' bit is set to '1', then the value in this register will be reflected on to the interrupt line. Refer to Section 17.2.3.3 "OTG switching" for details.	0
0	timer_interrupt_status	Set by the hardware when the interrupt event occurs. When software writes a value '1' into the OTG_int_clear register bit '0', this will be cleared by the hardware. If "timer_interrupt_en" bit is set to '1', then the value in this register will be reflected on to the interrupt line.	0

17.2.3.2.2 OTG interrupt enable register - (OTG_int_enable - 0x3102 0104, R/W)

If the Interrupt Enable bit value is set, an external interrupt is generated (on OTG_timer_int interrupt line) when the corresponding bit in the interrupt status register is set. If it is not set, no external interrupt is generated but interrupt will still be held in the interrupt status register.

Table 397. OTG interrupt enable register - (OTG_int_enable - 0x3102 0104, R/W)

Bits	Name	Function	Reset value
31:4	-	Reserved	-
3	hnp_success_en	Enable/ disable timer interrupt. A value '1' in this register will enable the interrupt due to hnp_success.	0
2	hnp_failure_en	Enable/ disable timer interrupt. A value '1' in this register will enable the interrupt due to hnp_failure.	0
1	remove_pullup_en	Enable/ disable timer interrupt. A value '1' in this register will enable the interrupt due to remove_pullup.	0
0	timer_interrupt_en	Enable/ disable timer interrupt. A value '1' in this register will enable the interrupt due to timer.	0

20. Some of the interrupt status bits may carry different meanings, based on the context of operation (whether the switching is from 'B' to 'A' or 'A' to 'B')

17.2.3.2.3 OTG interrupt set register - (OTG_int_set - 0x3102 020C, S)

Setting a particular bit to ‘1’ in this register will set the corresponding bit in the OTG_interrupt_status register. Writing a ‘0’ will not have any influence.

Table 398. OTG interrupt set register - (OTG_int_set - 0x3102 020C, S)

Bits	Name	Function	Reset value
31:4	-	Reserved	-
3	hnp_success_set	If software writes a value ’1’ into this register, then the "hnp_success" bit will be set to ’1’ in OTG_int_status register.	-
2	hnp_failure_set	If software writes a value ’1’ into this register, then the "hnp_failure" bit will be set to ’1’ in OTG_int_status register.	-
1	remove_pullup_set	If software writes a value ’1’ into this register, then the "remove_pullup" bit will be set to ’1’ in OTG_int_status register.	-
0	timer_interrupt_set	If software writes a value ’1’ into this register, then the "timer_interrupt" bit will be set to ’1’ in OTG_int_status register.	-

17.2.3.2.4 OTG interrupt clear register - (OTG_int_clear - 0x3102 010C, C)

Setting a particular bit to ‘1’ in this register causes the clearing of the interrupt by resetting the corresponding bit in the OTG_interrupt_status register. Writing a ‘0’ will not have any influence.

Table 399. OTG interrupt clear register - (OTG_int_clear - 0x3102 010C, C)

Bits	Name	Function	Reset value
31:4	-	Reserved	-
3	hnp_success_clear	If software writes a value ’1’ into this register, then the "hnp_success" bit will be reset to ’0’ in OTG_int_status register.	-
2	hnp_failure_clear	If software writes a value ’1’ into this register, then the "hnp_failure" bit will be reset to ’0’ in OTG_int_status register.	-
1	remove_pullup_clear	If software writes a value ’1’ into this register, then the "remove_pullup" bit will be reset to ’0’ in OTG_int_status register.	-
0	timer_interrupt_clear	If software writes a value ’1’ into this register, then the "timer_interrupt" bit will be reset to ’0’ in OTG_int_status register.	-

17.2.3.2.5 OTG status and control register - (OTG_status - 0x3102 0110, R/W)

Table 400. OTG status and control register - (OTG_status - 0x3102 0110, R/W)

Bits	Name	Function	Reset value
31:16	Timer count status	The present count value of the timer is reflected here.	0x0
15:11	-	Reserved.	-
10	Pullup_removed	During a ’B’ to ’A’ hand over, when the software removes the D+ pull-up, this bit also should be set by the software. This is an indication to the hardware that, from now onwards, it can look for a connection from the ’A’ device. Hardware will clear this bit, either when hnp_success or hnp_failure get reported, or when b_to_a_hnp_track bit is cleared by the software.	0

Table 400. OTG status and control register - (OTG_status - 0x3102 0110, R/W) *…continued*

Bits	Name	Function	Reset value
9	a_to_b_hnp_track	Based on the context of OTG switching ('A' device to 'B' device switching), software can set this bit, so that OTG controller hardware can track the HNP related activities and can inform the software (OTG stack) through interrupt mechanism. All time critical activities are handled by the hardware and non-time critical ones are handled at the software level. Refer to Section 17.2.3.3 "OTG switching". Hardware will clear this bit on time-out or on hnp_failure.	0
8	b_to_a_hnp_track	Based on the context of OTG switching ('B' device to 'A' device switching), software can set this bit, so that OTG controller hardware can track the HNP related activities and can inform the software (OTG stack) through interrupt mechanism. All time critical activities are handled by the hardware and non-time critical ones are handled at the software level. Refer to Section 17.2.3.3 "OTG switching". Hardware will clear this bit on hnp_success or hnp_failure.	0
7	Transparent_I2C_en	This bit should be used only when the ISP 1301 is used in transparent I^2C mode. This will 3-state the OE pad and enables the internal pull-up for the pad. The interrupt source is also shifted from the USB_ATX_INT_N pin to the USB_OE_TP_N.	0
6	Timer_reset	Reset the Timer. Writing'1' to this register will reset the timer. This provides a single bit control for the software to restart the timer, when the timer is active.	0
5	Timer_enable	Start timer. A value'1' in this register will start the timer. If this bit is set to'0' while the timer is active, then the timer will get restarted, when the timer is enabled again.	0
4	Timer_mode	0=> monoshot, 1=> free running In monoshot mode, an interrupt will be generated at the end of the time-out count and the timer will be disabled. In free running mode, an interrupt will be generated when the time-out count is reached and the timer value will be reloaded into the counter. The timer is not disabled here.	0
3:2	Timer_scale	Timer granularity selection 0x00: 10 us (100 kHz) 0x01: 100 us (10 kHz) 0x10: 1000us (1 kHz) 0x11: Reserved	0x0
1	-	Reserved	-
0	Host_En	USB Host or Device selection 0x0 Device enabled 0x1 Host enabled	0

17.2.3.2.6 UART mode

The OTG Transceiver Interface Specification v0.92a specifies a "UART Mode", where SE0_VM and DAT_VP can be used as UART TX and RX (U5_RX and U5_TX) respectively. The differential transmitter, receiver, and the single ended receivers are not functional. By muxing U5_RX and otg_rx_data to the USB_DAT_VP pin and U5_TX and otg_tx_se0 to the USB_SE0_VM pin of the transceiver, it is possible to use the D+ and D-

pins for UART traffic (Rx to D+ and Tx to D-). The “UART Mode” must be set in the UART block as well as the in external transceiver. The correct UART signaling level (2.8 V) must also be set in the external transceiver.

The UART block is aware of the UART mode by setting the uart5_mode bit in the UART_CTRL[0] register. The uart5_rx will be ‘H’ default both in DAT/SE0 mode and UART mode. The UART block will see ‘H’ all the time during DAT/SE0 (and initial VP/VM) mode. The transceiver is put into UART Mode, by setting transceiver register ModeControl1[uart_en] = ‘1’. Additionally, the transceivers internal pull-up resistors should be enabled and pull-down resistors disabled (in that order) by first setting OTGControlSet[dp_pullup, dm_pullup] = ‘11’ and then OTGControlClear[dp_pulldown, dm_pulldown] = ‘11’.

All registers in the transceiver are accessed over I^2C.

17.2.3.2.7 OTG timer register - (OTG_timer - 0x3102 0114, R/W)

Table 401. OTG timer register - (OTG_timer - 0x3102 0114, R/W)

Bits	Name	Function	Reset value
31:16	-	Reserved.	-
15:0	Timer Value	16-bit timer value to be counted. When the timer is enabled, the internal counter will be incremented based on the timer granularity. In mono mode, when the counter reaches timer value, an interrupt will be generated, and the timer will be disabled. In free running mode, when the counter reaches timer value, an interrupt will be generated, and counter will get a reset. The timer will not be disabled in this instance.	0xFFFF

17.2.3.2.8 OTG clock control register - (OTG_clock_control - 0x3102 0FF4, R/W)

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 this exercise for every register access, provided that the corresponding OTG_clock_control bits are already set.

Table 402. OTG clock control register - (OTG_clock_control - 0x3102 0FF4, R/W)

Bits	Name	Function	Reset value
31:5	-	Reserved.	-
4	AHB_CLK_ON	AHB clock control. 0: Disable the AHB clock. 1: Enable the AHB clock.	0
3	OTG_CLK_ON	OTG clock control. 0: Disable the OTG clock. 1: Enable the OTG clock.	0

Table 402. OTG clock control register - (OTG_clock_control - 0x3102 0FF4, R/W) *…continued*

Bits	Name	Function	Reset value
2	I2C_CLK_ON	I^2C clock control. 0: Disable the I^2C clock. 1: Enable the I^2C clock.	0
1	DEV_CLK_ON	Device clock control. 0: Disable the Device clock. 1: Enable the Device clock.	0
0	HOST_CLK_ON	Host clock control. 0: Disable the Host clock. 1: Enable the Host clock.	0

17.2.3.2.9 OTG clock status register - (OTG_clock_status - 0x3102 0FF8, R/W)

This register holds the clock availability status. The software should poll the otg_clock_status for the corresponding bit. If it is set, then software can go ahead with the register access. Software does not have to do this exercise for every access. If the otg_clock_control is already set before and the clock status information is already available to the software, then software can go ahead with normal register access, provided that the otg_clock_control content (respective bits) are not disturbed.

Table 403. OTG clock status register - (OTG_clock_status - 0x3102 0FF8, R/W)

Bits	Name	Function	Reset value
31:5	-	Reserved.	-
4	AHB_CLK_OK	AHB clock status. 0: AHB clock is not available. 1: AHB clock is available.	0
3	OTG_CLK_ON	OTG clock status. 0: OTG clock is not available. 1: OTG clock is available.	0
2	I2C_CLK_ON	I^2C clock status. 0: I^2C clock is not available. 1: I^2C clock is available.	0
1	DEV_CLK_ON	Device clock status. 0: Device clock is not available. 1: Device clock is available.	0
0	HOST_CLK_ON	Host clock status. 0: Host clock is not available. 1: Host clock is available.	0

17.2.3.2.10 I2C RX register - (I2C_RX - 0x3102 0300, R)

The I2C_RX 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 404. I2C RX register - (I2C_RX - 0x3102 0300, R)

Bits	Name	Function	Reset value
7:0	RX Data	Receive data.	-

17.2.3.2.11 I2C TX register - (I2C_TX - 0x3102 0300, W)

The TX 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 (CTL bit 7), or if an arbitration failure occurs (STS bit 3). Data writes to a full FIFO are ignored.

The 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 405. I2C TX register - (I2C_TX - 0x3102 0300, W)

Bits	Name	Function	Reset value
9	STOP	1 = Issue a STOP condition after transmitting this byte.	-
8	START	1= Issue a START condition before transmitting this byte.	-
7:0	TX Data	Transmit data.	-

17.2.3.2.12 I2C STS register - (I2C_STS - 0x3102 0304, R)

The I2C_STS register provides status information on the Tx and Rx blocks as well as the current state of the external buses.

Table 406. I2C STS register - (I2C_STS - 0x3102 0304, R)

Bits	Name	Function	Reset value
31:12	-	Reserved	-
11	TFE	Transmit FIFO Empty. 0: TX FIFO contains valid data. 1: TX FIFO is empty TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data.	1
10	TFF	Transmit FIFO Full. 0: TX FIFO is not full. 1: TX FIFO is full TFF is set when the TX FIFO is full and is cleared when the TX FIFO is not full.	0
9	RFE	Receive FIFO Empty. 0: RX FIFO contains data. 1: RX FIFO is empty RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data.	1
8	RFF	Receive FIFO Full (RFF). 0: RX FIFO is not full 1: RX FIFO is full 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
7	SDA	The current value of the SDA signal.	-

Table 406. I2C STS register - (I2C_STS - 0x3102 0304, R) *…continued*

Bits	Name	Function	Reset value
6	SCL	The current value of the SCL signal.	-
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
4	DRSI	Slave Data Request (DRSI). 0: Slave transmitter does not need data. 1: Slave transmitter needs data. 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
3	DRMI	Master Data Request. 0: Master transmitter does not need data. 1: Master transmitter needs data. 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
2	NAI	No Acknowledge. 0: Last transmission received an acknowledge. 1: Last transmission did not receive an acknowledge. 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
1	AFI	Arbitration Failure. 0: No arbitration failure on last transmission. 1: Arbitration failure occurred on last transmission. When transmitting, if the SDA is low when SDAOUT is high, then this I^2C 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 '1' to bit 1 of the status register.	0
0	TDI	Transaction Done. 0: Transaction has not completed. 1: Transaction completed. This flag is set if a transaction completes successfully. It is cleared by writing a '1' to bit 0 of the status register. It is unaffected by slave transactions.	0

17.2.3.2.13 I2C CTL Register - (I2C_CTL - 0x3102 0308, R/W)

The I2C_CTL register is used to enable interrupts and reset the I^2C state machine.

Table 407. I2C CTL Register - (I2C_CTL - 0x3102 0308, R/W)

Bits	Name	Function	Reset value
31:9	-	Reserved.	-
8	SRST	Soft reset. 1: Reset the I^2C to idle state. Self clearing. 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 I^2C 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
7	TFFIE	Transmit FIFO Not Full Interrupt Enable. 0: Disable the TFFI. 1: Enable the TFFI. This enables the Transmit FIFO Not Full interrupt to indicate that 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 I^2C block only when there is room in the FIFO and do this without polling the status register.	0
6	RFDAIE	Receive Data Available Interrupt Enable. 0: Disable the DAI. 1: Enable the DAI. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty).	0
5	RFFIE	Receive FIFO Full Interrupt Enable. 0: Disable the RFFI. 1: Enable the RFFI. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data.	0
4	DRSIE	Slave Transmitter Data Request Interrupt Enable. 0: Disable the DRSI interrupt. 1: Enable the DRSI interrupt. 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
3	DRMIE	Master Transmitter Data Request Interrupt Enable. 0: Disable the DRMI interrupt. 1: Enable the DRMI interrupt. 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

Table 407. I2C CTL Register - (I2C_CTL - 0x3102 0308, R/W) *...continued*

Bits	Name	Function	Reset value
2	NAIE	Transmitter No Acknowledge Interrupt Enable. 0: Disable the NAI. 1: Enable the NAI. This enables the NAI interrupt signalling that transmitted byte was not acknowledged.	0
1	AFIE	Transmitter Arbitration Failure Interrupt Enable. 0: Disable the AFI. 1: Enable the AFI. 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	TDIE	Transmit Done Interrupt Enable. 0: Disable the TDI interrupt. 1: Enable the TDI interrupt. This enables the TDI interrupt signalling that this I^2C issued a STOP condition.	0

17.2.3.2.14 I2C CLock High register - (I2C_CLKHI - 0x3102 030C, R/W)

The CLK register holds a terminal count for counting PERIPH_CLK clock cycles to create the high period of the slower I^2C serial clock, SCL.

Table 408. I2C CLock High register - (I2C_CLKHI - 0x3102 030C, R/W)

Bits	Name	Function	Reset value
7:0	CDHI	Clock divisor high. This value is the number of PERIPH_CLK clocks the serial clock (SCL) will be high.	0x41

17.2.3.2.15 I2C Clock Low register - (I2C_CLKLO - 0x3102 0310, R/W)

The CLK register holds a terminal count for counting PERIPH_CLK clock cycles to create the low period of the slower I^2C serial clock, SCL.

Table 409. I2C Clock Low register - (I2C_CLKLO - 0x3102 0310, R/W)

Bits	Name	Function	Reset value
7:0	CDLO	Clock divisor low. This value is the number of PERIPH_CLK clocks the serial clock (SCL) will be low.	0x41

17.2.3.3 OTG switching

The context of OTG controller operation is described in Figure 62. The Host controller consist of a communication interface with the OHCI stack using a set of control and status registers as well as interrupts. The OTG stack with the OTG control block and device stack with device controller block are similar. The OTG stack also contains an interface to ISP 1301 (external OTG ATX) using the I^2C interface as well as interrupts. During the SRP, the protocol events are handled by the OTG stack and the ISP 1301. During the HNP hand over sequence, some controlling events are time critical. It is better to handle them in hardware if system interrupt latency is fairly high (of the order of few milliseconds). The hardware required for doing these operations are incorporated inside the OTG control block. The software (OTG stack) has a well defined interface to this hardware using a set of control and interrupt status registers as well as interrupts. Here the software has the option of switching on the OTG control logic to track the time critical activities or to do it

entirely in software (achieved through b_to_a_hnp_track and a_to_b_hnp_track register bits). If the hardware option is used, then during the handover sequence, the OTG control block will generate specific interrupt events to the OTG stack to do appropriate actions (with sufficient time granularity on the interrupt latency). In most cases, it could be around 20 ms.).

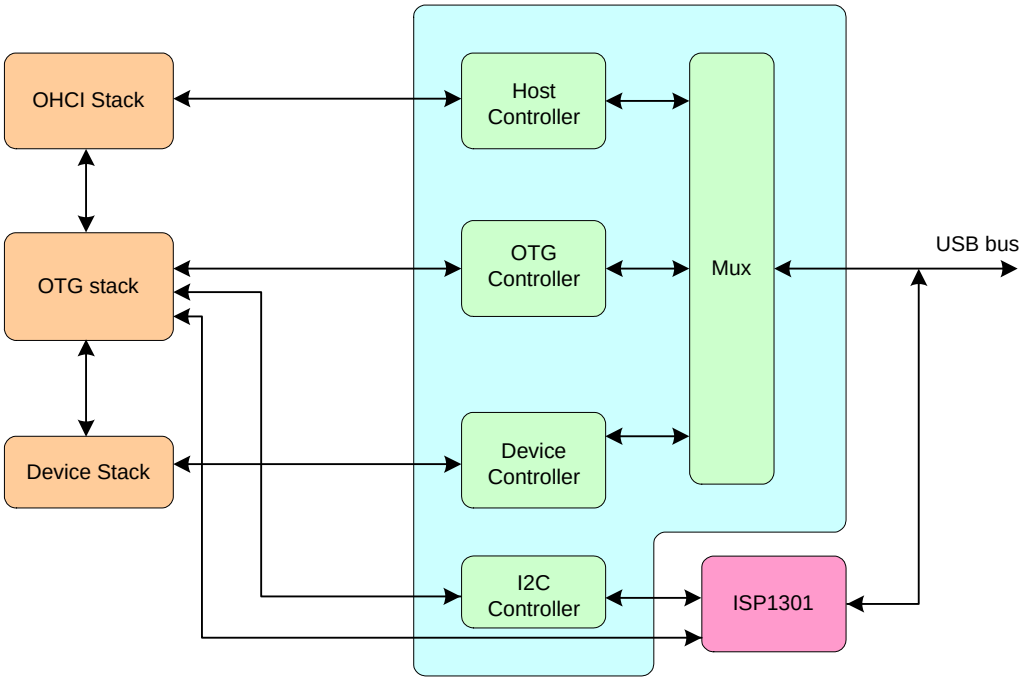


Fig 62. USB OTG controller with software stack

17.2.3.3.1 B to A HNP switching

The following are the actions (Table 410) required by the OTG hardware and the OTG stack during B (device) to A (host) handover sequence. The necessary steps are numbered in the sequence they appear.

Table 410. B to A HNP switching

No.	Controlling event	Action by hardware[1]	Action by software	Remarks
1.	Set feature command is executed by 'A' device.		When the 'B' device decides to become a host (B-> A) then software should set the b_to_a_hnp_track bit in the OTG_status and control register.	
2.	b_to_a_hnp_track bit is set.	Track the USB receive lines for suspend (Rx D+ and Rx D-).	-	Here the hardware will track the receive lines for 'J' condition for > 3ms.
3.	Receive line is idle for more than 3 ms (suspend condition).	An interrupt will be raised to the OTG stack, and remove_pullup bit will be set in the OTG_int_status register. The Rx D+ and Rx D- interface to Device controller will be isolated and will be placed under 'J' state. Continue monitoring the receive lines.	Once the interrupt is received, then OTG stack need to do the following actions in order. 1. Remove the D+ PULLUP (through I^2C and ISP1301). 2. Set the Pullup_removed bit in the OTG_status and control register. 3. At the end of the I^2C transaction (TDI bit set in I^2C), check whether the HNP failure bit is set or not. If set, then add the pull-up through I^2C. The action mentioned in '3' is required to avoid any race condition because of a resume or reset being generated form the 'A' device.	An interrupt will be raised by the Device controller indicating the suspend condition. This can be used by the Device controller stack for further processing. There is no interrupt latency issue. The 'B' device has around 150 ms time to respond with a removal of PULLUP. The Pullup_removed bit is used by the hardware if a 'J' condition is detected to determine whether it is due to the addition of pull-up by the 'A' device. Once the remove_pullup interrupt is generated, and before the actual removal of the pull-up, the 'A' device can cancel the hand over by placing a resume condition on the bus. The action mentioned as no '3' (HNP failure) in the previous column will take care of this.

Table 410. B to A HNP switching *...continued*

No.	Controlling event	Action by hardware[1]	Action by software	Remarks
4.	Resume event detected or bus reset detected (on the receive lines) before the removal of PULLUP.	Remove isolation from the Host receive port. (OTG port connected back to device controller). Wait for Pullup_removed bit to get set. Once the Pullup_removed bit is set, then raise an interrupt to the OTG stack, and HNP failure bit will be set in the interrupt status register. Clear b_to_a_hnp_track bit. Clear Pullup_removed bit.	Once the interrupt is received, then OTG stack should add the D+ PULLUP (through I^2C and ISP1301). Go back to the peripheral state.	Resume detected become active, when 'K' condition exists on the bus for more than 25 μs. Bus reset detected become active when SE0 detected on the bus for more than 3.2 ms. Once the interrupt is raised, the add pullup actions should be completed within ~17 ms. When the port is connected back to device controller, a suspend change event bit will be set in device controller. The device controller stack should be ready within ~17 ms to receive tokens from the 'A' device.
5.	Pullup_removed bit is set (no resume event detected).	Start a timer for 25 μs.		The hardware will wait here for the timer to expire, to avoid any residual effects of the PULLUP removal.
6.	25 us timer expired.	Continue polling the receive lines for status change		The maximum time the hardware wait here is ~3.125 ms. Within this time, if any status change reported in USB receive lines, then appropriate actions are to be taken.

Table 410. B to A HNP switching *...continued*

No.	Controlling event	Action by hardware[1]	Action by software	Remarks
7.	Resume detected or bus reset detected (after removal of PULLUP).	An interrupt will be raised to the OTG stack, and HNP failure bit will be set in the interrupt status register. Clear b_to_a_hnp_track bit. Clear Pullup_removed bit. Reset timer. OTG port is connected back to device controller.	Once the interrupt is received, then OTG stack should add the D+ PULLUP (through I^2C and ISP1301). Go back to the peripheral state.	Once the interrupt is raised, the add pullup actions should be completed in ~ 17 ms. When the port is connected back to device controller, a suspend change event bit will be set in device controller. This can be used by the Device controller stack for further processing.
8.	'J' detected (after removal of PULLUP).	An interrupt will be raised to the OTG stack, and HNP success bit will be set in the interrupt status register. Set the host_en bit. Clear pullup_removed bit. Change the Host controller internal receive port status to 'J' (SE0 to J transition). Start sending reset on the USB bus. Monitor the Host controller transmit lines (Tx D+, Tx D- and tx_en_n) for SE0 condition.	Once the Interrupt is received, change the status to b_host. Once this is happened, the OHCI driver should reset the port (10 ms).	The status change in Host controller port will be treated as a new full speed connection by the Host root hub function. This will result in a root hub status change interrupt being raised to OHCI stack. The 'J' detection must be sensed through a debounce circuit(2.5 μs).
9.	Host starts sending bus reset.	Connect the OTG port to Host controller. Stop driving bus reset on USB bus. Clear b_to_a_hnp_track bit Reset timer. Clear Pullup_removed bit.		From this point onwards, Host controller will send bus reset on USB bus for another 10 ms.

[1] In many instances, the same event causes two levels of interrupts being generated from the hardware. This may not be a desirable situation, unless it is properly handled. One possible solution could be, when the OTG stack is active for a hand over, then all actions based on interrupt are initiated by the OTG stack. This means, the OTG stack provides the communication to device and OHCI stack. All Device and Host specific interrupts are ignored.

17.2.3.3.2 A to B HNP Switching

The following are the actions (Table 411) required by the OTG controller hardware and the OTG stack during a "A" (host) to "B" (device) hand over sequence.

Table 411. A to B HNP switching

No.	Controlling event	Action by hardware	Action by software	Remarks
1.			OTG stack generates the set feature command through the Host function (software).	
2.			Set the "BDIS_ACON_EN" bit in the ISP 1301.	This will be set by the software once it is ready for a hand over. When this bit is set, the software should make sure that, the other side present 'B' device will not respond back with any USB packet (only SOF on the USB bus).
3.			Set the "a_to_b_hnp_track" bit in OTG_status and control register. Suspend the traffic on the OTG port. Load and enable the timer.	The timer value is loaded by software and should correspond to at least 150 ms interval.
4.	a_to_b_hnp_track bit set.	Poll the status on the Receive USB lines for suspend.	-	This bit is set by software before the OTG port get into suspend state. Software enables a timer.
5.	USB bus goes into suspend state.	Isolate Internal Host receive port (drive 'J'). Poll the status on the Receive USB lines.		
6.	Resume detected, when the timer is active	Connect the Host port to OTG port. Clear a_to_b_hnp_track bit. Disable timer. Set HNP failure bit in the interrupt status register. Generate Interrupt to OTG stack.	Clear "BDIS_ACON_EN" bit in ISP 1301. Go back to the a_host state.	A J-> K transition is treated as resume here, and can come from the downstream device (reflected on the receive D+ and D- lines).

Table 411. A to B HNP switching *...continued*

No.	Controlling event	Action by hardware	Action by software	Remarks
7.	Timer expired.	Connect the Host port to OTG port. Clear a_to_b_hnp_track bit. Disable timer. Set hnp_failure and timer_interrupt_status bit in the OTG_int_status register. Generate Interrupt to OTG stack.	Clear "BDIS_ACON_EN" bit in ISP 1301. If HNP failure bit and the timer_interrupt_status bit are set, then go to a_wait_vfall state.	
8.	ISP 1301 generates BDIS_ACON_EN interrupt		No action	
9.	Bus reset detected on USB lines.	Set HNP success interrupt. Set host_en to '0'. Clear a_to_b_hnp_track bit. Disable timer. go back to IDLE state.	Transition to peripheral state.	When the host_en bit is set to'0'. Internal Host port will see "SE0" condition. This will signal a disconnect event to the OHCI software.

17.2.4 External transceiver interface

Figure 63 shows physical connectivity of ISP_1301 external ATX device with OTG controller.

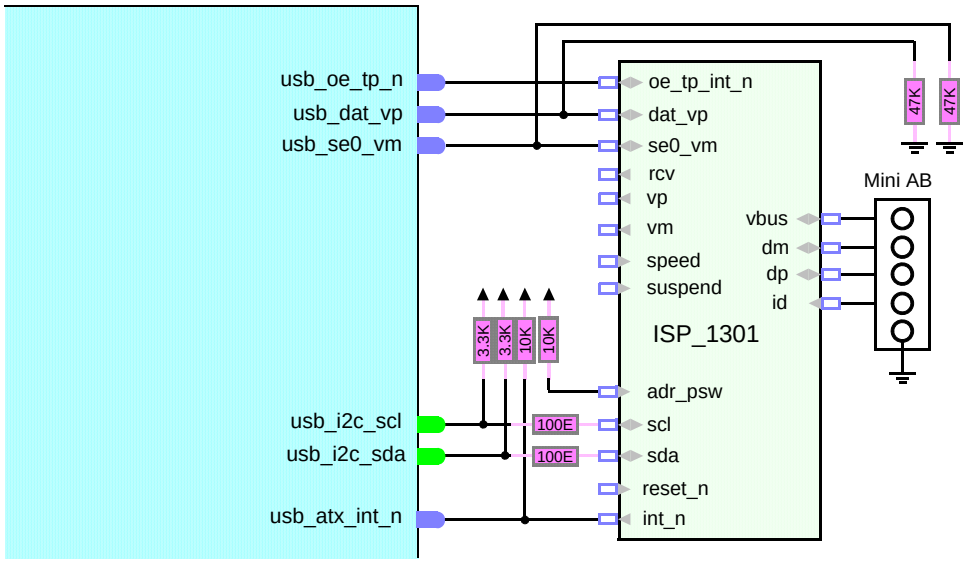


Fig 63. ISP_1301 interface example

18.1 Introduction

The Secure Digital interface allows access to external Secure Digital memory cards. The SD Card interface conforms to the *SD Memory Card Specification Version 1.01*. The SD block interfaces to slave port 5 on the AHB Matrix. The SD Card interface uses an APB interface and is interfaced to the AHB bus through an AHB-APB bridge.

18.2 Features

- Conformance to the SD Memory Card Specification Version 1.01.
- DMA is supported through the system DMA Controller.
- Provides all functions specific to the secure digital memory card. These include the clock generation unit, power management control, command and data transfer.
- APB interface provides access to the SD Card Interface registers, and generates interrupt and DMA request signals.

18.3 Pin description

Table 412. SD card interface pin description

Pin name	Type	Description
MS_SCLK	Output	SD card clock output.
MS_BS	Input	SD card command input/output.
MS_DIO[3:0]	Output	SD card data lines.

18.4 Functional description

Figure 64 shows the connection of the SD Card Interface to an external Secure Digital memory card. If other pins are required for a specific SD Card arrangement, they would be implemented by software using GPIO or GPO pins.

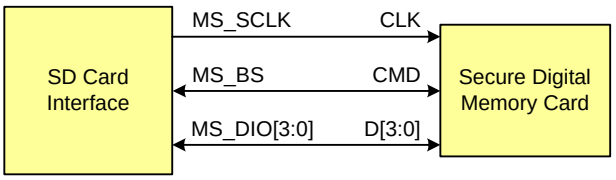


Fig 64. Secure digital memory card connection

Figure 65 shows a simplified block diagram of the SD Card Interface.

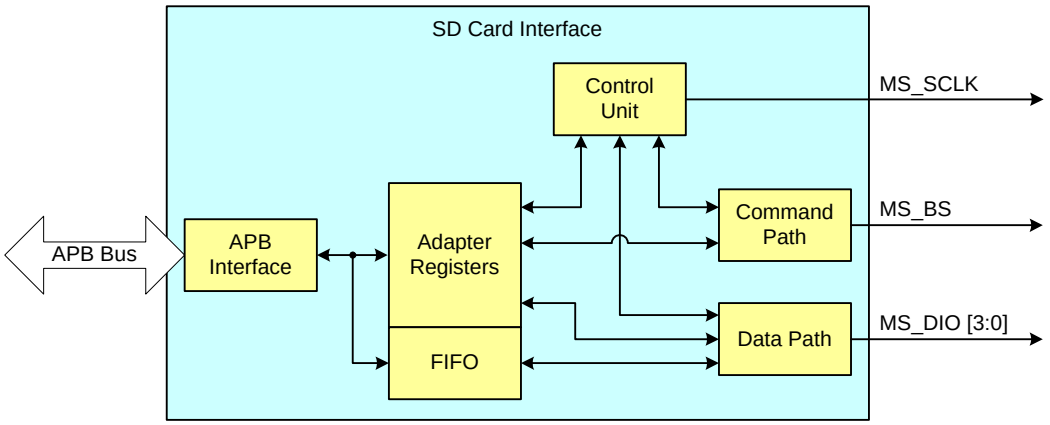


Fig 65. Block diagram of the SD Card Interface

The SD Card Interface consists of five subunits:

- Adapter register block.
- Control unit.
- Command path.
- Data path.
- Data FIFO.

18.4.1 Adapter register block

The register block contains all of the SD Card interface registers. This block also generates the signals that clear the static flags in the SD card. The clear signals are generated when 1 is written into the corresponding bit location of the SD_Clear register.

18.4.2 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; and Power-on.

Note that a GPIO or GPO is used to control external SD card power. SD_POWER[CTRL] should be set to power_up until power is stable, then to power_on. During Power-off and Power-up phases the interface output pins are disabled.

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 the interface is 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).

18.4.3 Command path

The command path subunit sends commands to and receives responses from the SD card.

The CPU controls command transfers. The SD_FIFO can be read or written as a 32 bit wide register. The FIFO contains 16 entries on 16 sequential addresses. This allows the CPU to use load-store multiple operands for accessing the FIFO.

18.4.3.1 Command path state machine

When the command register is written 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, the state machine waits for the response. When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set. The state machine uses a fixed time-out of 64 SDCLKs when waiting for a response from the SD Card. Figure 66 gives details of the CPSM.

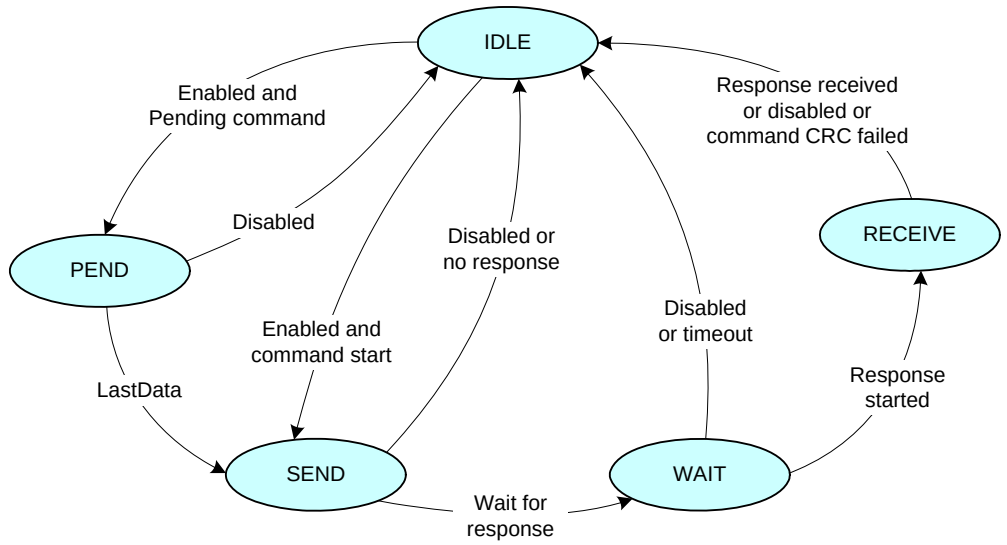


Fig 66. Command path state machine

When the WAIT state is entered, the command timer starts running. If the time-out is reached before the CPSM moves to the RECEIVE state, the time-out flag is set and the IDLE state is entered. The time-out period has a fixed value of 64 SD_CLK clock periods.

If the interrupt bit is set in the command register, the fixed time-out of 64 SDCLKs is disabled, and the CPSM waits forever for an interrupt request response from the SD card.

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. The CPSM remains in the IDLE state for at least eight SD_CLK periods to meet Ncc and Nrc timing constraints in the SD card specification.

Figure 67 shows the command transfer.

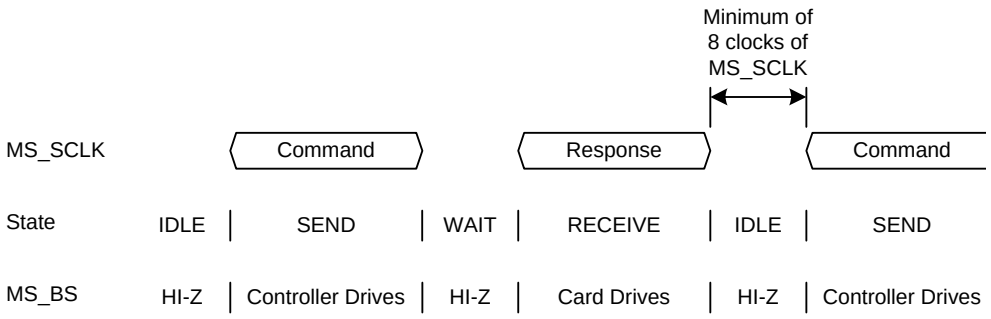


Fig 67. Command transfer

18.4.3.2 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 MS_BS output is in HI-Z state, as shown in Figure 67. Data on MS_BS is synchronous to the rising SD_CLK edge. All commands have a fixed length of 48 bits. Table 413 shows the command format.

Table 413. Command format

Bit position	Width	Value	Description
47	1	0	Start bit.
46	1	1	Transmission bit.
[45:40]	6	-	Command index.
[39:8]	32	-	Argument.
[7:1]	7	-	CRC7.
0	1	1	End bit.

The SD Card Interface supports two response types. Both use CRC error checking:

- 48 bit short response (see Table 414).
- 136 bit long response (see Table 415).

Note: If the response does not contain a CRC (CMD1 response), the device driver must ignore the CRC failed status.

Table 414. Short response format

Bit position	Width	Value	Description
47	1	0	Start bit.
46	1	0	Transmission bit.
[45:40]	6	-	Command index.
[39:8]	32	-	Argument.
[7:1]	7	-	CRC7 (or 1111111).
0	1	1	End bit.

Table 415. Long response format

Bit position	Width	Value	Description
135	1	0	Start bit.
134	1	1	Transmission bit.
[133:128]	6	111111	Reserved.
[127:1]	127	-	CID or CSD (including internal CRC7).
0	1	1	End 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 (refer to the SD_Command register description for more information). The command path implements the status flags shown in Table 416 (refer to the SD_Status register description for more information).

Table 416. 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:

$$CRC[6:0] = \text{Remainder } [(M(x) \times x^7) / G(x)]$$

$$G(x) = x^7 + x^3 + 1$$

$$M(x) = (\text{start bit}) \times x^{39} + \ldots + (\text{last bit before CRC}) \times x^0, \text{ or}$$

$$M(x) = (\text{start bit}) \times x^{119} + \ldots + (\text{last bit before CRC}) \times x^0$$

18.4.4 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 (MS_DIO[3:0]). If the wide bus mode is not enabled, only one bit per clock cycle is transferred over MS_DIO[0].

18.4.4.1 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.

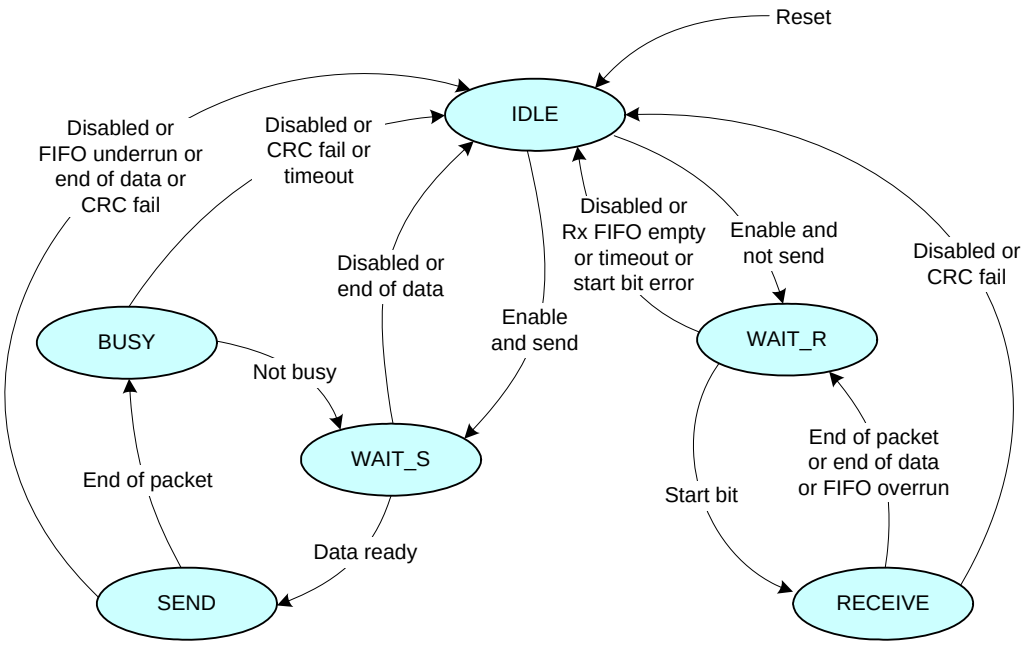


Fig 68. Data path state machine

18.4.4.1.1 IDLE

The data path is inactive, and the MS_DIO[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.

18.4.4.1.2 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 MS_DIO.

The DPSM moves to the RECEIVE state if it receives a start bit before a time-out, and loads the data block counter. If it reaches a time-out before it detects a start bit, or a start bit error occurs, the DPSM moves to the IDLE state and sets the time-out status flag.

18.4.4.1.3 RECEIVE

Serial data received from the SD 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.

18.4.4.1.4 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 the Nwr timing constraints in the SD card specification.

18.4.4.1.5 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.

18.4.4.1.6 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 MS_DIO[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.

18.4.4.1.7 Data timer

The data timer is enabled when the DPSM is in the WAIT_R or BUSY state, and generates the data time-out error:

- When transmitting data, the time-out occurs if the DPSM stays in the BUSY state for longer than the programmed time-out period.
- When receiving data, the time-out 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 time-out period.

18.4.4.2 Data counter

The data counter has two functions:

- To stop a data transfer when it reaches zero. This is the end of the data transfer.
- To start transferring a pending command (see Figure 69). This is used to send the stop command for a stream data transfer.

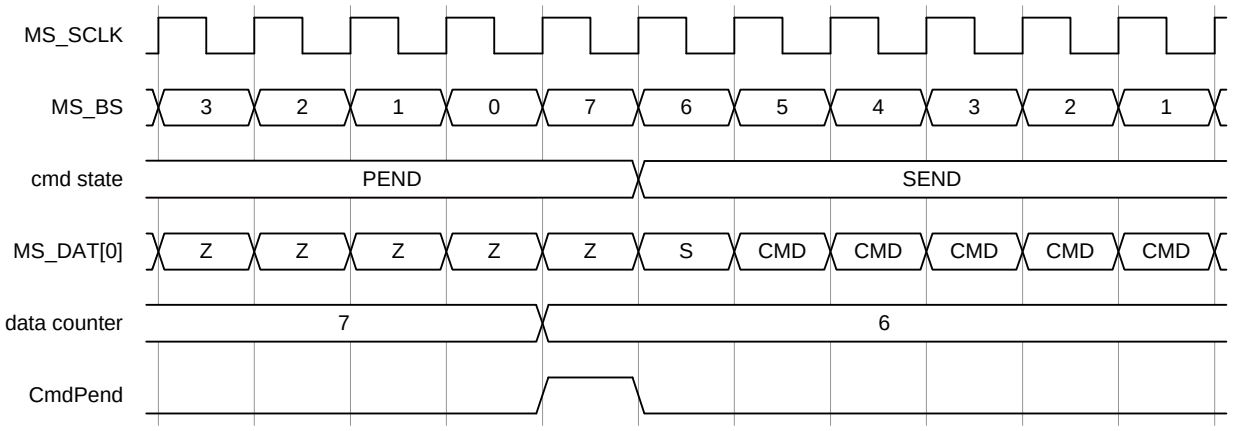


Fig 69. Pending command start

The data block counter determines the end of a data block. If the counter is zero, the end-of-data condition is TRUE (refer to the SD_DataCtrl register description for more information).

18.4.4.3 Bus mode

In wide bus mode, all four data signals (MS_DIO[3:0]) are used to transfer data, and the CRC code is calculated separately for each data bit. While transmitting data blocks to a card, only MS_DIO[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, MS_DIO[3:0] are in the HI-Z state.

Data on these signals is synchronous to the rising edge of the clock period.

18.4.4.4 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 MS_DIO[0] LOW. Table 417 shows the CRC token status values.

Table 417. CRC token status

Token	Description
010	Card has received error-free data block.
101	Card has detected a CRC error.

18.4.4.5 Status flags

Table 418 lists the data path status flags (refer to the SD_Status register description for more information).

Table 418. 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.

18.4.4.6 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] = Remainder [(M(x) × x15) / G(x)]

G(x) = x16 + x12 + x5 + 1

M(x) = (first data bit) × xn + ... + (last data bit) × x0

18.4.5 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, in addition to transmit and receive logic.

18.4.5.1 Transmit FIFO

Data is written to the transmit FIFO through the APB interface once the SD Card Interface is enabled for transmission. When a write occurs, data is written into the FIFO location specified by the current value of the data pointer. The pointer is incremented after every FIFO write.

The transmit FIFO contains a data output register. This holds the data word pointed to by the read pointer. If the transmit FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. Table 419 lists the transmit FIFO status flags.

Table 419. 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 SD_Clear register.

18.4.5.2 Receive FIFO

When the data path subunit receives a word of data, it is written to the receive FIFO. The write pointer is incremented after the write is completed. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer are available for reading by the CPU. The read pointer is incremented when the data is read via the APB bus interface.

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. Table 420 lists the receive FIFO status flags.

Table 420. Receive FIFO status flags

Flag	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.
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 SD_Clear register.

18.4.6 APB interface

The APB interface generates 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 system DMA Controller.

18.5 Register description

This section describes the SD registers and provides programming details. The SD Card Interface registers are shown in Table 421.

Table 421. Secure Digital card interface register summary

Address offset	Name	Description	Reset value	Type
0x4000 4080	MS_CTRL	SD Card interface clock and pad control	0x0000 0000	R/W
0x2009 8000	SD_Power	Power Control Register	0x0000 0000	R/W
0x2009 8004	SD_Clock	Clock Control Register	0x0000 0000	R/W

Table 421. Secure Digital card interface register summary *…continued*

Address offset	Name	Description	Reset value	Type
0x2009 8008	SD_Argument	Argument register	0x0000 0000	R/W
0x2009 800C	SD_Command	Command register	0x0000 0000	R/W
0x2009 8010	SD_Respcmd	Command response register	0x0000 0000	RO
0x2009 8014	SD_Response0	Response register 0	0x0000 0000	RO
0x2009 8018	SD_Response1	Response register 1	0x0000 0000	RO
0x2009 800C	SD_Response2	Response register 2	0x0000 0000	RO
0x2009 8020	SD_Response3	Response register 3	0x0000 0000	RO
0x2009 8024	SD_DataTimer	Data Timer	0x0000 0000	R/W
0x2009 8028	SD_DataLength	Data Length register	0x0000 0000	R/W
0x2009 802C	SD_DataCtrl	Data Control register	0x0000 0000	R/W
0x2009 8030	SD_DataCnt	Data counter	0x0000 0000	RO
0x2009 8034	SD_Status	Status register	0x0000 0000	RO
0x2009 8038	SD_Clear	Clear register	0x0000 0000	WO
0x2009 803C	SD_Mask0	Interrupt mask register 0	0x0000 0000	R/W
0x2009 8040	SD_Mask1	Interrupt mask register 1	0x0000 0000	R/W
0x2009 8048	SD_FIFOCnt	FIFO counter	0x0000 0000	RO
0x2009 8080 to 0x2009 80BC	SD_FIFO	Data FIFO register	0x0000 0000	R/W

18.5.1 Memory Card Control register (MS_CTRL - 0x4000 4080)

The MS_CTRL register selects whether the SD card interface is enabled. It also controls pad pull-up and pull-down and clocks to the related peripheral blocks.

Table 422. Memory Card Control register (MS_CTRL - 0x4000 4080)

Bit	Function	Reset value
31:11	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
10	Disable SD_card pins. If the SD Card interface is not used, this bit (10) and bit 9 should be programmed to 0, and bits 6 through 8 should be programmed to 1. 0 = enable SD_card interface pins. 1 = disable SD_card interface pins and enable Peripheral MUX registers see Table 650.	0
9	Enables clock and pull-ups to MSSDIO pins. If the SD Card interface is not used, this bit should be programmed to 0, and bits 6 through 8 should be programmed to 1. To enable clocking, the SD_card controller requires both bits 5 and 9 be set to 1. 0 = MSSDIO pull-up and clock disabled. 1 = MSSDIO pull-up and clock enable.	0
8	MSSDIO2 and MSSDIO3 pad control. 0 = MSSDIO2 and 3 pad has pull-up enabled. 1 = MSSDIO2 and 3 pad has no pull-up.	0
7	MSSDIO1 pad control. 0 = MSSDIO1 pad has pull-up enabled. 1 = MSSDIO1 pad has no pull-up.	0

Table 422. Memory Card Control register (MS_CTRL - 0x4000 4080) *...continued*

Bit	Function	Reset value
6	MSSDIO0/MSBS pad control. 0 = MSSDIO0 pad has pull-up enable. 1 = MSSDIO0 pad has no pull-up.	0
5	SD Card clock control. This bit controls MSSDCLK to the SD Card block. The registers in the peripheral block cannot be accessed if the clock is stopped. To enable clocking, the SD_card controller requires both bits 5 and 9 be set to 1. 0 = Clocks disabled. 1 = Clocks enabled.	0
4	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3:0	These register bits control the divider ratio when generating the clock from the ARM PLL output clock. Software must insure that the maximum clock frequency of the targeted device is not exceeded. 0000 = MSSDCLK stopped. Divider in low power mode. 0001 = MSSDCLK equals ARM PLL output clock divided by 1. …… 1110 = MSSDCLK equals ARM PLL output clock divided by 14. 1111 = MSSDCLK equals ARM PLL output clock divided by 15.	0

18.5.2 Power control register (SD_Power - 0x2009 8000)

The SD_Power register controls an external power supply. Power can be switched on and off. 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. The card bus outlets are disabled during both phases. Note that after a data write, data cannot be written to this register for three MS_SCLK clock periods plus two HCLK periods. Table 423 shows the bit assignment of the SD_Power register.

Table 423. Power control register (SD_Power - 0x2009 8000)

Bit	Function	Description	Reset value
31:7	Not used	-	-
6	OpenDrain	SDCMD output control 0=MS_BS (MS_BS pin) is push-pull type (default) 1=MS_BS (MS_BS pin) is open drain type	0
5:2	Not used	-	-
1:0	Ctrl	Power mode control. 00 = Power Off. 01 = Reserved. 10 = Power up. Disables output pins. The SD_PWR function may be implemented in software by using a GPO pin. 11 = Power on. Enables output pins. The SD_PWR function may be implemented in software by using a GPO pin.	0

18.5.3 Clock control register (SD_Clock - 0x2009 8004)

The SD_Clock register controls the SD_CLK output. While the SD Card Interface is in identification mode, the maximum SD_CLK frequency is 400 kHz. Note that after a data write, data cannot be written to this register for three SD_CLK clock periods plus two HCLK periods. Table 424 shows the bit assignment of the clock control register.

Table 424. Clock control register (SD_Clock - 0x2009 8004)

Bit	Function	Description	Reset value
31:12	Not used	-	-
11	WideBus	Enables the wide bus mode. 0 = Standard bus mode (only MS_DIO[0] used). 1 = Wide bus mode (MS_DIO[3:0] used).	0
10	Bypass	Enables bypassing the clock divide logic. 0 = No bypass. 1 = Bypass SDCLK divider. (SD_CLK = SDCLK)	0
9	PwrSave	Disables the SD card clock output when the bus is idle. 0 = SD_CLK is always enabled. 1 = SD_CLK is disabled when SD bus is idle.	0
8	Enable	Enables the SD card clock. 0 = SD_CLK disabled. 1 = SD_CLK enabled.	0
7:0	ClkDiv	Controls the SD card clock period. Set SD_CLK output frequency SD_CLK = SDCLK / (2×(ClkDiv+1))	0

18.5.4 Argument register (SD_Argument - 0x2009 8008)

The SD_Argument register contains a 32-bit command argument, which is sent to a card as part of a command message. If a command contains an argument, it must be loaded into the SD_Argument register before writing a command to the SD_Command register. Table 425 shows the bit assignment of the SD_Argument register.

Table 425. Argument register (SD_Argument - 0x2009 8008)

Bit	Function	Description	Reset value
31:0	Argument	Command argument	0

18.5.5 Command register (SD_Command - 0x2009 800C)

The SD_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.

Note that after a data write, data cannot be written to this register for three SD_CLK clock periods plus two HCLK periods. Table 426 shows the bit assignment of the SD_Command register.

Table 426. Command register (SD_Command - 0x2009 800C)

Bit	Function	Description	Reset value
31:11	Not used	-	-
10	Enable	0 = Disable the Command Path State Machine. 1 = Enable the Command Path State Machine.	0
9	Pending	0 = Do not wait before sending the command. 1 = Wait for CmdPend before sending the command.	0
8	Interrupt	0 = No interrupt. 1 = Disable the command timer and wait for a card interrupt request without timeout.	0
7	LongRsp	0 = Expect a normal response. 1 = Expect a 136-bit long command response if a response is required.	0
6	Response	0 = No response is required. 1 = A response to the command is required.	0
5:0	CmdIndex	Command Index. This is sent to the card as part of the command message.	0

Table 427 shows the response types.

Table 427. 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.

18.5.6 Command response register (SD_Respcmd - 0x2009 8010)

The SD_Respcmd register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long response), the RespCmd field is unknown. Table 428 shows the bit assignment of the SD_Respcmd register.

Table 428. Command response register (SD_Respcmd - 0x2009 8010)

Bit	Function	Description	Reset value
31:6	Not used	-	-
5:0	RespCmd	Response command index.	0

18.5.7 Response registers (SD_Response0-3 - 0x2009 8014, 018, 01C, 020)

The SD_Response0-3 registers contain the status of a card, which is part of the received response. Table 429 shows the bit assignment of the SD_Response0-3 registers.

Table 429. Response registers (SD_Response0-3 - 0x2009 8014, 018, 01C, 020)

Bit	Function	Description	Reset value
31:0	Status	Card status.	0

The most significant bit of the card status is received first. The SD_Response3 register LSB is always 0. The card status size can be 32 or 127 bits, depending on the response type, as shown in Table 430.

Table 430. Response register type

Description	Short response	Long response
SD_Response0	Card status [31:0].	Card status [127:96].
SD_Response1	Unused.	Card status [95:64].
SD_Response2	Unused.	Card status [63:32].
SD_Response3	Unused.	Card status [31:1].

18.5.8 Data timer register (SD_DataTimer - 0x2009 8024)

The SD_DataTimer register contains the data time-out period in card bus clock periods. 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 time-out 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. Table 431 shows the bit assignment of the SD_DataTimer register.

Table 431. Data timer register (SD_DataTimer - 0x2009 8024)

Bit	Function	Description	Reset value
31:0	DataTime	Data time-out period.	0

18.5.9 Data length register (SD_DataLength - 0x2009 8028)

The SD_DataLength register indicates the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts. For a block data transfer, the value in the data length register must be a multiple of the block size (see Data control register, SD_DataCtrl). A data transfer must be written to the data timer register and the data length register before being written to the data control register. Table 432 shows the bit assignment of the SD_DataLength register.

Table 432. Data length register (SD_DataLength - 0x2009 8028)

Bit	Function	Description	Reset value
31:16	Not used	-	-
15:0	DataLength	Data length value.	0

18.5.10 Data control register (SD_DataCtrl - 0x2009 802C)

The SD_DataCtrl register controls the DPSM. 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. Note that after a data write, data cannot be written to this register for three SD_CLK clock periods plus two HCLK periods. Table 433 shows the bit assignment of the SD_DataCtrl register.

Table 433. Data control register (SD_DataCtrl - 0x2009 802C)

Bit	Function	Description	Reset value
31:8	Not used	-	-
7:4	BlockSize	0000 = 1 byte. 0001 = 2 bytes. 0010 = 4 bytes. ..... 1011 = 2048 bytes (maximum). Any value above 1011 is reserved.	0
3	DMAEnable	0 = DMA is disabled. 1 = DMA is enabled.	0
2	Mode	0 = Block data transfer. 1 = Stream data transfer.	0
1	Direction	0 = From controller to Card (transmit). 1 = From Card to Controller (receive).	0
0	Enable	0 = Data transfer disabled. 1 = Data transfer enabled.	0

18.5.11 Data counter register (SD_DataCnt - 0x2009 8030)

The SD_DataCnt register loads the value from the data length register (see Data length register, SD_DataLength) 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. This register should be read only when the data transfer is complete. Table 434 shows the bit assignment of the SD_DataCnt register.

Table 434. Data counter register (SD_DataCnt - 0x2009 8030)

Bit	Function	Description	Reset value
31:16	Not used	-	-
15:0	DataCount	Indicates the number of bytes remaining to transfer.	0

18.5.12 Status register (SD_Status - 0x2009 8034)

The SD_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 Clear register, SD_Clear).
- 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 is written to the FIFO).

Table 435 shows the bit assignment of the SD_Status register.

Table 435. Status register (SD_Status - 0x2009 8034)

Bit	Function	Description	Reset value
31:22	Not used	-	-
21	RxDataAvlbl	1 = Data available in receive FIFO.	0
20	TxDataAvlbl	1 = Data available in transmit FIFO.	0
19	RxFifoEmpty	1 = Receive FIFO empty.	0
18	TxFifoEmpty	1 = Transmit FIFO empty.	0
17	RxFifoFull	1 = Receive FIFO full.	0
16	TxFifoFull	1 = Transmit FIFO full.	0
15	RxFifoHalfFull	1 = Receive FIFO half full.	0
14	TxFifoHalfEmpty	1 = Transmit FIFO half empty.	0
13	RxActive	1 = Data receive in progress.	0
12	TxActive	1 = Data transmit in progress.	0
11	CmdActive	1 = Command transfer in progress.	0
10	DataBlockEnd	1 = Data block sent/received (CRC check passed).	0
9	StartBitErr	1 = Start bit not detected on all data signals in wide bus mode.	0
8	DataEnd	1 = Data end (Data counter is zero).	0
7	CmdSent	1 = Command sent (No response required).	0
6	CmdRespEnd	1 = Command Response received (CRC check passed).	0
5	RxOverrun	1 = Receive FIFO overrun.	0
4	TxUnderrun	1 = Transmit FIFO underrun.	0
3	DataTimeOut	1 = Data Timeout.	0
2	CmdTimeOut	1 = Command Response Timeout.	0
1	DataCrcFail	1 = Data block sent/received (CRC check failed).	0
0	CmdCrcFail	1 = Command response received (CRC check failed).	0

18.5.13 Clear register (SD_Clear - 0x2009 8038)

The SD_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 436 shows the bit assignment of the SD_Clear register.

Table 436. Clear register (SD_Clear - 0x2009 8038)

Bit	Function	Description	Reset value
31:11	Not used	-	-
10	DataBlockEndClr	Clears the DataBlockEnd flag.	0
9	StartBitErrClr	Clears the StartBitErr flag.	0
8	DataEndClr	Clears the DataEnd flag.	0
7	CmdSentClr	Clears the CmdSent flag.	0
6	CmdRespEndClr	Clears the CmdRespEnd flag.	0
5	RxOverrunClr	Clears the RxOverrun flag.	0
4	TxUnderrunClr	Clears the TxUnderrun flag.	0

Table 436. Clear register (SD_Clear - 0x2009 8038) *...continued*

Bit	Function	Description	Reset value
3	DataTimeOutClr	Clears the DataTimeOut flag.	0
2	CmdTimeOutClr	Clears the CmdTimeOut flag.	0
1	DataCrcFailClr	Clears the DataCrcFail flag.	0
0	CmdCrcFailClr	Clears the CmdCrcFail flag.	0

18.5.14 Interrupt mask registers (SD_Maskx - 0x2009 803C, 040)

The interrupt SD_Mask0 register masks interrupts to the MIC SD0_INT interrupt signal from the SD card interface, and SD_Mask1 masks interrupts to the MIC SD1_INT interrupt signal. The bits in these registers determine which status flags generate an interrupt request by setting the corresponding bit to 1, the two interrupts together can be used as a first level filter to select interrupts as desired and direct them to two separate interrrupt service routines. Table 437 shows the bit assignment of the SD_Maskx registers.

Table 437. Interrupt mask registers (SD_Maskx - 0x2009 803C, 040)

Bit	Function	Description	Reset value
31:22	Not used	-	-
21	Mask21	1 = enable interrupt for the RxDataAvlbl flag.	0
20	Mask20	1 = enable interrupt for the TxDataAvlbl flag.	0
19	Mask19	1 = enable interrupt for the RxFifoEmpty flag.	0
18	Mask18	1 = enable interrupt for the TxFifoEmpty flag.	0
17	Mask17	1 = enable interrupt for the RxFifoFull flag.	0
16	Mask16	1 = enable interrupt for the TxFifoFull flag.	0
15	Mask15	1 = enable interrupt for the RxFifoHalfFull flag.	0
14	Mask14	1 = enable interrupt for the TxFifoHalfEmpty flag.	0
13	Mask13	1 = enable interrupt for the RxActive flag.	0
12	Mask12	1 = enable interrupt for the TxActive flag.	0
11	Mask11	1 = enable interrupt for the CmdActive flag.	0
10	Mask10	1 = enable interrupt for the DataBlockEnd flag.	0
9	Mask9	1 = enable interrupt for the StartBitErr flag.	0
8	Mask8	1 = enable interrupt for the DataEnd flag.	0
7	Mask7	1 = enable interrupt for the CmdSent flag.	0
6	Mask6	1 = enable interrupt for the CmdRespEnd flag.	0
5	Mask5	1 = enable interrupt for the RxOverrun flag.	0
4	Mask4	1 = enable interrupt for the TxUnderrun flag.	0
3	Mask3	1 = enable interrupt for the DataTimeOut flag.	0
2	Mask2	1 = enable interrupt for the CmdTimeOut flag.	0
1	Mask1	1 = enable interrupt for the DataCrcFail flag.	0
0	Mask0	1 = enable interrupt for the CmdCrcFail flag.	0

18.5.15 FIFO counter register (SD_FIFOCnt - 0x2009 8048)

The SD_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 Data length register, SD_DataLength) when the Enable bit is set in the data control register. If the data length is not word aligned (a multiple of 4), the remaining 1 to 3 bytes are regarded as a word. Table 438 shows the bit assignment of the SD_FIFOCnt register.

Table 438. FIFO counter register (SD_FIFOCnt - 0x2009 8048)

Bit	Function	Description	Reset value
31:15	Not used	-	-
14:0	DataCount	Remaining data words to transfer.	0

18.5.16 Data FIFO register (SD_FIFO - 0x2009 8080 to 0x2009 80BC)

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 CPU to use load and store multiple operands to read and write to the FIFO. Table 439 shows the bit assignment of the SD_FIFO register.

Table 439. Data FIFO register (SD_FIFO - 0x2009 8080 to 0x2009 80BC)

Bit	Function	Description	Reset value
31:0	Data	Read or write FIFO data.	0

19.1 Introduction

The LPC32x0 contains seven UARTs, four of which are standard UARTs, downwards compatible with the INS16Cx50. The standard UARTs are described in this chapter. The remaining three 14-clock UARTs are described in the 14-clock UART chapter.

19.2 Features

- Each standard UART has 64 byte Receive and Transmit FIFOs.
- Receiver FIFO trigger points at 16, 32, 48, and 60 bytes.
- Transmitter FIFO trigger points at 0, 4, 8, and 16 bytes.
- Register locations conform to ‘550 industry standard.
- Each standard UART has a fractional rate pre-divider and an internal baud rate generator.
- The standard UARTs support 3 clocking modes: on, off, and auto-clock. The auto-clock mode shuts off the clock to the UART when it is idle.
- UART 6 includes an IrDA mode to support infrared communication.
- The standard UARTs are designed to support data rates of 2400; 4800; 9,600; 19,200; 38,400; 57,600; 115,200; 230,400; and 460,800 bps.
- Each UART includes an internal loopback mode.

19.3 Pin description

Table 440. Standard UART Pin Description

Pin name	Type	Function	Description
Un_RX	Input	RXD	**Receive data input.** Serial data to the UART is input on this pin for UART 3, 4, and 5.
Un_TX	Output	TXD	**Transmit data output.** Serial data from the UART is output on this pin for UARTs 3, 4, and 5.
U3_RTS	Output	$\overline{\text{RTS}}$	**Request To Send.** U3_RTS informs the modem or data set that UART3 is ready to transmit data. U3_RTS is set to its active state by setting bit 0 in the modem control register (U3_MCR), and is changed to inactive state by clearing bit 1 (RTS) in the U3_MCR. U3_RTS also goes inactive as a result of a master reset or during loop mode operation. The complement of this signal is stored in U3_MSR[1].

Table 440. Standard UART Pin Description

Pin name	Type	Function	Description
U3_CTS	Input	$\overline{\text{CTS}}$	**Clear to send.** U3_CTS is a modem status signal. The state of this pin can be checked by reading bit 4 (CTS) of the modem status register (U3_MSR). Bit 0 (ΔCTS) of the U3_MSR indicates that this signal has changed state since the last read from the modem status register. If the modem status interrupt (U3_IER[3]) is enabled when U3_CTS changes state, an interrupt is generated.
U3_DSR	Input	$\overline{\text{DSR}}$	**Data Set Ready.** U3_DSR is a modem status signal. Its state of this pin can be checked by reading bit 5 (DSR) of the modem status register (U3_MSR). Bit 1 (ΔDSR) of the modem status register indicates that this signal has changed state since the last read from the modem status register. If the modem status interrupt (U3_IER[3]) is enabled when U3_DSR changes state, an interrupt is generated.
U3_DTR	Output	$\overline{\text{DTR}}$	**Data Terminal Ready.** U3_DTR is an active low signal that informs a modem or data set that UART3 is ready to establish communication. U3_DTR is set to its the active state by setting bit 0 (DTR Control) in the modem control register (U3_MCR), and is changed to inactive by clearing bit 0 in the U3_MCR. U3_DTR also enters the inactive state as a result of a master reset or during loop mode operation. The complement of this signal is stored in U3_MSR[0].
U3_DCD	Input	$\overline{\text{DCD}}$	**Data Carrier Detect.** U3_DCD is an active low modem status signal that indicates if the external modem has established a communication link with the UART3 and data may be exchanged. Its state can be checked by reading bit 7 (DCD) of the modem status register (U3_MSR). Bit 3 (ΔDCD) of modem status register indicates that this signal has changed states since the last read from the modem status register. If the modem status interrupt is enabled (U3_IER[3]) when DCD changes state, an interrupt is generated
U3_RI	Input	$\overline{\text{RI}}$	**Ring Indicator.** U3_RI is an active low and modem status signal that indicates a telephone ringing signal has been detected by the modem. Its condition can be checked by reading bit 6 (RI) of the modem status register (U3_MSR). Bit 2 (TERI) of the modem status register indicates that the RI input has transitioned from a low to a high state since the last read from the modem status register. If the modem status interrupt is enabled when this transition occurs, an interrupt is generated
U6_IRRX	Input		**Receive data input.** Serial data to UART6 is input on this pin. UART 6 supports a selectable IrDA mode.
U6_IRTX	Output		**Transmit data output.** Serial data from UART6 is output on this pin. UART 6 supports a selectable IrDA mode.

19.4 Functional description

The architecture of the standard UARTs is shown in the block diagram, Figure 70. The UART receiver monitors the serial input line for valid input. The UART Rx Shift Register (RSR) accepts valid characters via the Un_RX pin. After a valid character is assembled in the RSR, it is passed to the UART Rx Buffer Register FIFO to await access by the CPU.

The UART transmitter accepts data written by the CPU or host and buffers the data in the UART Tx Holding Register FIFO (THR). The UART Tx Shift Register (TSR) reads the data stored in the THR and assembles the data to transmit via the serial output pin, Un_TX.

UART 3 includes hardware modem control. However, it does not support auto-CTS or auto-RTS. Note that without auto-CTS and using the TX FIFO, the transmitter will send any data present in the transmit FIFO and a receiver overrun error may result.

The UART Baud Rate Generator block generates the timing used by the UART transmitter and receiver. The interrupt interface contains registers IER and IIR. Status information from the Tx and Rx lines is stored in the LSR register. Control information for the Tx and Rx lines is stored in the LCR register. The FCR register controls the FIFOs for the Rx and Tx lines.

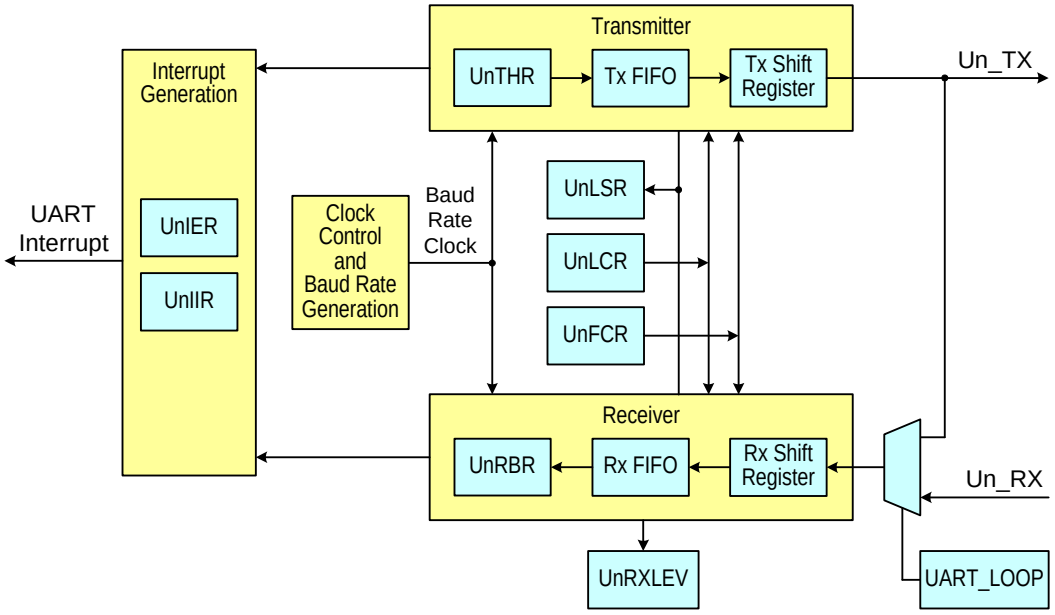


Fig 70. Standard UART block diagram

19.4.1 UART clock modes

Each UART has three clock modes, on, off and autoclock mode. In the on mode, the clock to the UART is always on, in off mode the clock is always off.

In the autoclock mode, the clock is normally switched off but is automatically switched on by hardware when required. The automatic function works for both transmit and receive. An incoming start bit is detected and turns the receiver clock on. The clock is switched off

again when the receiver goes idle. When data is written to the transmit buffer, the clock is switched on, and remains on until the last data is transmitted. The clock control mechanism is applied to all parts of the UART and may be used to save power.

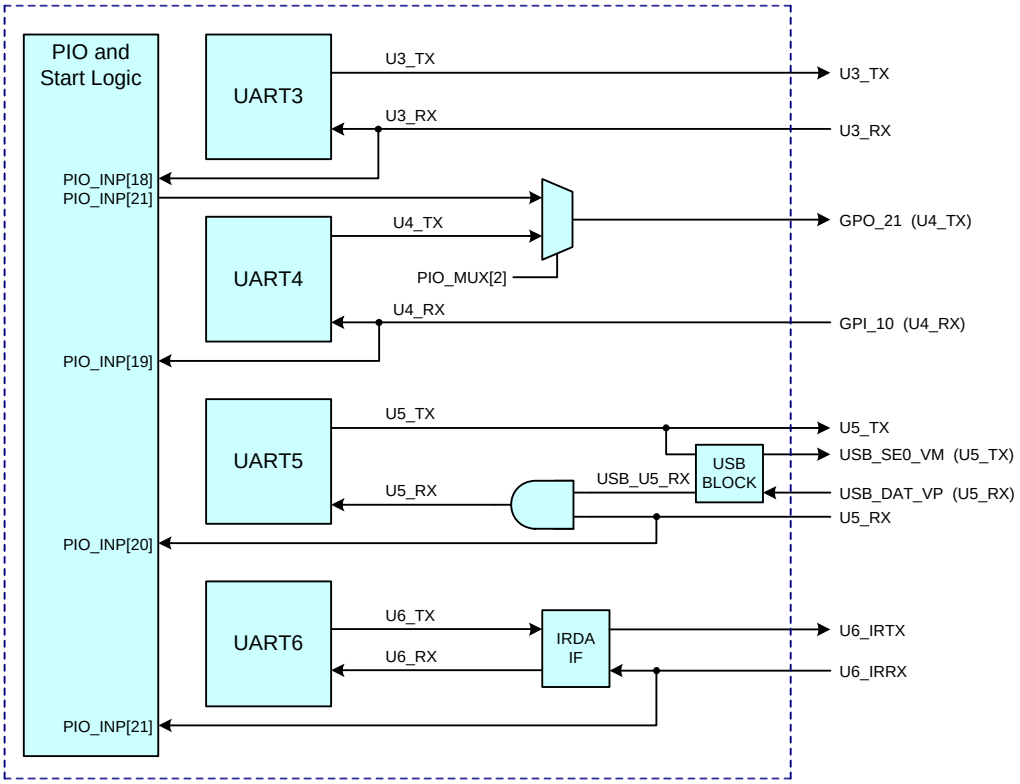


Fig 71. UART pin connections

19.5 UART base addresses

Table 441. Standard UART base addresses

UART	Base address
3	0x4008 0000
4	0x4008 8000
5	0x4009 0000
6	0x4009 8000

19.6 Register description

19.6.1 Primary UART control registers

Each standard UART contains registers as shown in Table 442. Address offsets are shown in the first column. Each UART contains that register at the base address from Table 441 plus the offset value from Table 442.

Table 442. Registers for each standard UART

Address Offset	Name	Description	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Type	Reset value
0x00 (DLAB = 0)[1]	UnRBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	-
0x00 (DLAB = 0) [1]	UnTHR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	-
0x00 (DLAB = 1) [1]	UnDLL	Divisor Latch Lower Byte	MSB							LSB	R/W	0x01
0x04 (DLAB = 1) [1]	UnDLM	Divisor Latch Upper Byte	MSB							LSB	R/W	0
0x04 (DLAB = 0) [1]	UnIER	Interrupt Enable Register	Reserved				Modem Status IE	Rx Line Status IE	THRE IE	Rx Data Avail. IE	R/W	0
0x08	UnIIR	Interrupt ID Register	(2 copies of FCR[0])		Reserved		Interrupt ID			Interrupt Pending	RO	0x01
0x08	UnFCR	FIFO Control Register	Rx Trigger Level Select		Tx Trigger Level Select		DMA Mode Select	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0
0x0C	UnLCR	Line Control Register	Divisor Latch Access Bit (DLAB)	Break Control	Parity Select	Even Parity	Parity Enable	Stop Bit Select	Word Length Select		R/W	0
0x10	U3MCR	Modem Control Register	Reserved (write 0)	Reserved (write 0)	Reserved (write 0)	Loop back	Reserved (write 0)		RTSC	DTRC	R/W	0
0x14	UnLSR	Line Status Register	Rx FIFO Error	Transmitter Empty	Transmit Holding Reg. Empty (THRE)	Break Interrupt (BI)	Framing Error (FE)	Parity Error (PE)	Overrun Error	Receiv Data Ready	RO	0x60
0x18	U3MSR	Modem Status Register	DCD	RI	DSR	CTS	ΔDCD	TERI	ΔDSR	ΔCTS	RO	0x60
0x1C	UnRXLEV	Receive FIFO Level Register	Reserved	Rx Level							RO	0

[1] The Divisor Latch Access Bit (DLAB) is contained in UnLCR[7]. When DLAB = 1, the Divisor Latches are accessible and the Receiver Buffer Register, Transmit Holding Register, and Interrupt Enable Register are not accessible.

19.6.2 Additional UART control registers

There are additional registers in a separate address space that control other aspects of the standard UARTs. These are shown in Table 443.

Table 443. Additional control registers for standard UARTs

Address	Name	Description	Reset value	Access
0x4000 40D0	U3CLK	UART 3 Clock Control Register	0	R/W
0x4000 40D4	U4CLK	UART 4 Clock Control Register	0	R/W
0x4000 40D8	U5CLK	UART 5 Clock Control Register	0	R/W
0x4000 40DC	U6CLK	UART 6 Clock Control Register	0	R/W
0x4000 40E0	IRDACLK	IrDA Clock Control Register	0	R/W
0x4005 4000	UART_CTRL	UART Clock Control Register	0	R/W
0x4005 4004	UART_CLKMODE	UART Clock Mode Register	0	R/W
0x4005 4008	UART_LOOP	UART Loopback Control Register	0	R/W

19.6.3 UART Receiver Buffer Register (UnRBR - 0x4008 0000, 0x4008 8000, 0x4009 0000, 0x4009 8000)

The UnRBR register allows reading the top byte of the Receiver FIFO of UARTn. The top byte of the Rx FIFO contains the oldest character received. Bit 0 always contains the first 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 the UnLCR register must be zero in order to access the UnRBR register. UnRBR is a Read Only register.

Since PE, FE and BI bits in the UnLSR register correspond to the byte sitting on the top of the RBR FIFO (i.e. the byte that will be provided in the next read from UnRBR), the approach for fetching the valid pair of received byte and its associated status bits is to first read the status from UnLSR, and then to read a data byte from UnRBR.

Table 444. UART Receiver Buffer Register (UnRBR - 0x4008 0000, 0x4008 8000, 0x4009 0000, 0x4009 8000)

UnRBR	Function	Description	Reset value
7:0	Receiver Buffer Register	The UARTn Receiver Buffer Register contains the oldest received byte in the UARTn Rx FIFO.	0

19.6.4 UARTn Transmitter Holding Register (UnTHR - 0x4008 0000, 0x4008 8000, 0x4009 0000, 0x4009 8000)

The UnTHR register accesses the top byte of the Transmit FIFO of UARTn. The top byte is the newest character in the Tx FIFO. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in the UnLCR register must be zero in order to access the UnTHR register. UnTHR is a Write Only register.

Table 445. UARTn Transmitter Holding Register (UnTHR - 0x4008 0000, 0x4008 8000, 0x4009 0000, 0x4009 8000)

UnTHR	Function	Description	Reset value
7:0	Transmit Holding Register	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.	N/A

19.6.5 UARTn Divisor Latch LSB Register (UnDLL - 0x4008 0000, 0x4008 8000, 0x4009 0000, 0x4009 8000); UARTn Divisor Latch MSB Register (UnDLM - 0x4008 0004, 0x4008 8004, 0x4009 0004, 0x4009 8004)

The UARTn Divisor Latch is part of the UARTn Baud Rate Generator and holds the value used to divide the UART clock 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. If both registers together contain a 0, it is treated as a 1 value in order to prevent division by zero. Refer to the Baud Rate Calculation section of this chapter for complete information on rate programming.

The Divisor Latch Access Bit (DLAB) in the UnLCR register must be one in order to access the UARTn Divisor Latches.

Table 446. UARTn Divisor Latch LSB Register (UnDLL - 0x4008 0000, 0x4008 8000, 0x4009 0000, 0x4009 8000)

UnDLL	Function	Description	Reset value
7:0	Divisor Latch LSB Register	The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.	0x01

Table 447. UARTn Divisor Latch MSB Register (UnDLM - 0x4008 0004, 0x4008 8004, 0x4009 0004, 0x4009 8004)

UnDLM	Function	Description	Reset value
7:0	Divisor Latch MSB Register	The UARTn Divisor Latch MSB Register, along with the UnDLL register, determines the baud rate of the UARTn.	0

19.6.6 UARTn Interrupt Enable Register (UnIER - 0x0x4008 0004, 0x4008 8004, 0x4009 0004, 0x4009 8004)

The UnIER register is used to enable the four interrupt sources available in each UART, as shown in Table 448.

The Divisor Latch Access Bit (DLAB) in the UnLCR register must be one in order to access the UARTn Divisor Latches.

Table 448. UARTn Interrupt Enable Register (UnIER - 0x0x4008 0004, 0x4008 8004, 0x4009 0004, 0x4009 8004)

UnIER	Function	Description	Reset value
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3[1]	Modem Status Interrupt Enable	This bit enables the UART3 modem interrupt. The status of this interrupt can be read from U3_MSR[3:0]. 0: Disable the modem interrupt. 1: Enable the modem interrupt.	0
2	Rx Line Status Interrupt Enable	This bit enables the UARTn Receiver Line Status interrupt. This interrupt reflects Overrun Error, Parity Error, Framing Error, and Break conditions. The status of this interrupt can be read from UnLSR[4:1]. 0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts.	0
1	THRE Interrupt Enable	This bit enables the Transmit Holding Register Empty (THRE) interrupt for UARTn. The status of this interrupt can be read from UnLSR[5]. 0: Disable the THRE interrupt. 1: Enable the THRE interrupt.	0
0	RDA Interrupt Enable	This bit enables the Receive Data Available (RDA) interrupt for UARTn. 0: Disable the RDA interrupt. 1: Enable the RDA interrupt.	0

[1] This bit is only available in UART3, for UART4 and UART5 this bit is reserved and user software should not write a one to this bit. The value read from this bit is not defined.

19.6.7 UARTn Interrupt Identification Register (UnIIR - 0x4008 0008, 0x4008 8008, 0x4009 0008, 0x4009 8008)

The UnIIR is a read-only register that provides a status code that denotes the source of a pending interrupt. The interrupts are frozen during an access to UnIIR. If an interrupt occurs during an UnIIR access, the interrupt is recorded for the next UnIIR access.

Table 449. UARTn Interrupt Identification Register (UnIIR - 0x4008 0008, 0x4008 8008, 0x4009 0008, 0x4009 8008)

UnIIR	Function	Description	Reset value
7:6	FIFO Enable	These bits contain the same value as UnFCR[0].	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3:1	Interrupt Identification	If the Interrupt Pending flag (bit 0 of this register) = 0, then the value of this field identifies the cause of the interrupt. The encoding of UnIIR[3:0] is shown in Table 450.	0
0	Interrupt Pending	This flag indicates when there are no UARTn related interrupts pending. Note that this bit is active LOW. The pending interrupt can be determined by evaluating UnIIR[3:0]. 0: At least one interrupt is pending. 1: No pending interrupts.	1

Interrupts are identified as described in Table 450. Given the value 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[3:0] = 0110) 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 by an UnLSR read.

The UARTn RDA interrupt (UnIIR[3:0] = 0100) shares the second level priority with the CTI interrupt (UnIIR[3:0] = 1100). When the receive FIFO is enabled, 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. When the receive FIFO is disabled, the RDA is activated when any received data is available.

The CTI interrupt (UnIIR[3:0] = 1100) 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 4 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 450. UARTn interrupt handling

UnIIR [3:0]	Priority	Interrupt type	Interrupt source	Method of clearing interrupt
0x1	-	none	none	-
0x6	1 (High)	Receiver Line Status (RLS)	OE (Overrun Error), PE (Parity Error), FE (Framing Error), or BI (Break Indication). Note that an RLS interrupt is asserted immediately rather than waiting for the corresponding character to reach the top of the FIFO.	Read of UnLSR.
0x4	2	Receiver Data Available (RDA)	When the FIFO is turned off (UnFCR[0] = 0), this interrupt is asserted when receive data is available. When the FIFO is turned on (UnFCR[0] = 1), this interrupt is asserted when the receive trigger level (as specified by UnFCR[7:6]) has been reached in the FIFO.	Read of UnRBR when UnFCR[0] = 0, or UARTn FIFO contents go below the trigger level when UnFCR[0] = 1.
0xC	2	Character Time-out Indication (CTI)	This case occurs when there is at least one character in the Rx FIFO and no character has been received or removed from the FIFO during the last 4 character times.	Read of UnRBR, or a Stop bit is received.
0x2	3	Transmit Holding Register Empty (THRE)	When the FIFO is turned off (UnFCR[0] = 0), this interrupt is asserted when the transmit holding register is empty. When the FIFO is turned on (UnFCR[0] = 1), this interrupt is asserted when the transmit trigger level (as specified by UnFCR[5:4]) has been reached in the FIFO.	Read of UnIIR or write to THR.
0x0	4(low)	Modem Status	Clear to send, data set ready, ring indicator, or data carrier detect	Read of the U3_MSR.

The UARTn THRE interrupt (UnIIR[3:0] = 0010) is a third level interrupt and is activated when the UARTn THR FIFO empties out to a specific level. When the FIFO is enabled, the THRE interrupt occurs when the Tx FIFO level is below the threshold set in the Tx Trigger Level Select field in the UnFCR register (described later in this chapter). When the FIFO is disabled, the THRE interrupt occurs when the Tx FIFO 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).

The UART3 Modem Status interrupt (UnIIR[3:0] = 0000) is a forth level interrupt and is activated when a clear to send, data set ready, ring indicator, or data carrier detect interrupt has been generated. Reading the U3_MSR will clear the interrupt.

19.6.8 UARTn FIFO Control Register (UnFCR - 0x4008 0008, 0x4008 8008, 0x4009 0008, 0x4009 8008)

The UnFCR controls the operation of the UARTn Rx and Tx FIFOs. Bits in UnFCR allow enabling the FIFOs, resetting the FIFOs, and selecting the FIFO trigger levels. Details are shown in Table 451.

Table 451. UARTn FIFO Control Register (UnFCR - 0x4008 0008, 0x4008 8008, 0x4009 0008, 0x4009 8008)

UnFCR	Function	Description	Reset value
7:6	Receiver Trigger Level Select	These two bits determine how many receiver UARTn FIFO characters must be present before an interrupt is activated. 00: trigger level = 16 01: trigger level = 32 10: trigger level = 48 11: trigger level = 60	0
5:4	Transmitter Trigger Level Select	These two bits determine the level of the UARTn transmitter FIFO causes an interrupt. 00: trigger level = 0 01: trigger level = 4 10: trigger level = 8 11: trigger level = 16	0
3	FIFO Control	Internal UARTn FIFO control. This bit must be set to 1 for proper FIFO operation.	0
2	Transmitter FIFO Reset	Writing a logic 1 to UnFCR[2] will clear all bytes in UARTn Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
1	Receiver FIFO Reset	Writing a logic 1 to UnFCR[1] will clear all bytes in UARTn Rx FIFO and reset the pointer logic. This bit is self-clearing. **Remark:** If an Rx data or timeout interrupt is pending after the UARTn Rx FIFO is cleared, a single read of the Rx FIFO will clear the interrupt. It is recommended to always perform a read of the UARTn Rx FIFO after it is flushed.	0
0	FIFO Enable	UARTn transmit and receive FIFO enable. Any transition on this bit will automatically clear the UARTn FIFOs. 0: UARTn Rx and Tx FIFOs disabled. 1: UARTn Rx and Tx FIFOs enabled and other UnFCR bits activated.	0

19.6.9 UART3 Modem Control Register (U3_MCR - 0x4008 0010)

The U3_MCR enables the modem loopback mode and controls the modem output signals.

Table 452: UART3 Modem Control Register (U3_MCR - address 0x4008 0000) bit description

Bit	Symbol	Description	Reset value
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
4	Loopback Mode Select	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, U3RXD, has no effect on loopback and output pin, TXD is held in marking state. The four modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the four modem outputs are connected to the four modem inputs. As a result of these connections, the upper four bits of the U3_MSR will be driven by the lower four bits of the U3_MCR rather than the four modem inputs in normal mode. This permits modem status interrupts to be generated in loopback mode by writing the lower four bits of U3_MCR. 0: Disable modem loopback mode. 1: Enable modem loopback mode.	0
3-2	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
1	RTS Control	Control for modem output pin U3_RTS. 0: Set RTS to output 1 (default). 1: Clear RTS to output 0. This bit reads as 0 when modem loopback mode is active.	0
0	DTR Control	Control for modem output pin U3_DTR. 0: Set DTR to output 1 (default). 1: Clear DTR to output 0. This bit reads as 0 when modem loopback mode is active.	0

19.6.10 UARTn Line Control Register (UnLCR - 0x4008 000C, 0x4008 800C, 0x4009 000C, 0x4009 800C)

The UnLCR determines the format of the data character that is to be transmitted or received.

Table 453. UARTn Line Control Register (UnLCR - 0x4008 000C, 0x4008 800C, 0x4009 000C, 0x4009 800C)

UnLCR	Function	Description	Reset value
7	Divisor Latch Access Bit	Allows access to the alternate registers at address offsets 0 and 4. 0: Disable access to baud rate Divisor Latches, enable access to UnRBR, UnTHR, and UnIER. 1: Enable access to baud rate Divisor Latches, disable access to UnRBR, UnTHR, and UnIER.	0
6	Break Control	Allows forcing the Un_TX output low in order to generate a break condition. 0: Disable break transmission 1: Enable break transmission.	0
5:4	Parity Select	If bit UnLCR[3] = 1, selects the type of parity used by the UART. 00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
3	Parity Enable	Selects the whether or not the UART uses parity. 0: Disable parity generation and checking 1: Enable parity generation and checking	0
2	Stop Bit Select	Selects the number of stop bits used by the UART. 0: 1 stop bit 1: 2 stop bits (1.5 if UnLCR[1:0] = 00)	0
1:0	Word Length Select	Selects the character length (in bits) used by the UART. 00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0

19.6.11 UARTn Line Status Register (UnLSR - 0x4008 0014, 0x4008 8014, 0x4009 0014, 0x4009 8014)

The UnLSR is a read-only register that provides status information on the UARTn Tx and Rx blocks.

Table 454. UARTn Line Status Register (UnLSR - 0x4008 0014, 0x4008 8014, 0x4009 0014, 0x4009 8014)

UnLSR	Function	Description	Reset value
7	FIFO Rx Error	FIFO Rx Error. This bit is set when a character with a receive 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: UnRBR contains no UARTn Rx errors or UnFCR[0] = 0. 1: UnRBR contains at least one UARTn Rx error.	0
6	TEMT	Transmitter Empty. This bit is set when the last character has been transmitted from the Transmit Shift Register. TEMT is cleared when another character is written to UnTHR. 0: UnTHR and/or the UnTSR contains valid data. 1: UnTHR and the UnTSR are empty.	1
5	THRE	Transmitter Holding Register Empty. This bit is set when the transmitter FIFO reaches the level selected in UnFCR. THRE is cleared on a UnTHR write. 0: UnTHR contains valid data. 1: UnTHR is empty.	1
4	BI	Break Interrupt. When the Un_RX pin is held low 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 the Un_RX pin goes high. A read of UnLSR clears this status bit. 0: Break interrupt status is inactive. 1: Break interrupt status is active.	0
3	FE	Framing Error. When the stop bit of a received character is a logic 0, a framing error occurs. A read of UnLSR clears this bit. A framing error is associated with the character at the top of the UARTn RBR FIFO. Upon detection of a framing error, the receiver 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. 0: Framing error status is inactive. 1: Framing error status is active.	0

Table 454. UARTn Line Status Register (UnLSR - 0x4008 0014, 0x4008 8014, 0x4009 0014, 0x4009 8014) *…continued*

UnLSR	Function	Description	Reset value
2	PE	Parity Error. When the parity bit of a received character is in the wrong state, a parity error occurs. A read of UnLSR clears this bit. A parity error is associated with the character at the top of the UARTn RBR FIFO. 0: Parity error status is inactive. 1: Parity error status is active.	0
1	OE	Overrun Error. This bit is set when the 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. The overrun error condition is set as soon as it occurs. A read of UnLSR clears the OE flag. 0: Overrun error status is inactive. 1: Overrun error status is active.	0
0	RDR	Receiver Data Ready. This bit is set when the UnRBR holds an unread character and is cleared when the UARTn RBR FIFO is empty. 0: UnRBR is empty. 1: UnRBR contains valid data.	0

19.6.12 UART3 Modem Status Register (U3_MSR - 0x4008 0018)

The U3_MSR is a read-only register that provides status information on the modem input signals. U3_MSR[3:0] is cleared on U3_MSR read. Note that modem signals have no direct affect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 455: UART1 Modem Status Register (U3_MSR - address 0xE001 0018) bit description

Bit	Symbol	Description	Reset Value
0	ΔCTS	Set upon state change of input CTS. Cleared on an U3MSR read. 0 = No change detected on modem input, CTS. 1 = State change detected on modem input, CTS.	0
1	ΔDSR	Set upon state change of input DSR. Cleared on an U3_MSR read. 0 = No change detected on modem input, DSR. 1 = State change detected on modem input, DSR.	0
2	TERI	Set on a low to high transition of input pin RI. Cleared on an U3_MSR read. 0 = No change detected on modem input, RI. 1 = Low-to-high transition detected on RI.	0
3	ΔDCD	Set upon state change of input DCD. Cleared on an U3_MSR read. 0 = No change detected on modem input, DCD. 1 = State change detected on modem input, DCD.	0
4	CTS	Clear To Send State. Complement of input signal CTS. This bit is connected to U3_MCR[1] in modem loopback mode.	0

Table 455: UART1 Modem Status Register (U3_MSR - address 0xE001 0018) bit description

Bit	Symbol	Description	Reset Value
5	DSR	Data Set Ready State. Complement of input signal DSR. This bit is connected to U3_MCR[0] in modem loopback mode.	0
6	RI	Ring Indicator State. Complement of input RI. This bit is connected to U3_MCR[2] in modem loopback mode.	0
7	DCD	Data Carrier Detect State. Complement of input DCD. This bit is connected to U3_MCR[3] in modem loopback mode.	0

19.6.13 UARTn Rx FIFO Level Register (UnRXLEV - 0x4008 001C, 0x4008 801C, 0x4009 001C, 0x4009 801C)

The UnRXLEV register is a read-only register that provides the current level of the receiver FIFO for UARTn. This allows software to have more information about UART activity than provided by the FIFO level interrupt.

Table 456. UARTn Rx FIFO Level Register (UnRXLEV - 0x4008 001C, 0x4008 801C, 0x4009 001C, 0x4009 801C)

UnRXLEV	Function	Description	Reset value
6:0	RXLEV	Current receiver FIFO level.	0

19.6.14 UARTn Clock Select Registers (Un_CLK - 0x4000 40D0; 0x4000 40D4; 0x4000 40D8; 0x4000 40DC)

Each of the standard (not 14-clock) UARTs have a fractional rate pre-divider, which creates the clock used by the UART as the input to the baud rate generator for transmit and receive functions. If the pre-divider is set to generate the desired baud rate, the UART baud rate generator may not be needed. The Un_CLK registers control these rate generators. When using the fractional divider, we strongly recommend selecting an X/Y ratio with the smallest denominator that meets the requirements for baud rate, for example a ratio of 19/67 is better than 38/134. For details of baud rate generation, see the Baud Rate Calculation section.

Table 457. UARTn Clock Select Registers (Un_CLK - 0x4000 40D0; 0x4000 40D4; 0x4000 40D8; 0x4000 40DC)

Un_CLK	Function	Description	Reset value
16	Clock source select	0: Use PERIPH_CLK as input clock to the X/Y divider. 1: Use HCLK as input to the X/Y divider.	0
15:8	X divider value	If this value is set to 0, the output clock is stopped and the divider is put in a low power mode. See the description of the Y divider value below.	0
7:0	Y divider value	If this value is set to 0, the output clock is stopped and the divider is put in a low power mode. The X/Y divider divides the selected input clock using an X/Y divider. The output should be set to either 16 times the UART bit rate to be used, or a higher frequency if the UART baud rate generator (using the UnDLM and UnDLL registers) divides further down. Dividing directly down to 16 times the required bit rate is the most power efficient method. Note that the X/Y divider cannot multiply the clock rate. The X value must be less than or equal to the Y value. The smallest possible value of Y that meets the baud rate timing requirements is also recommended.	0

19.6.15 IrDA Clock Control Register (IRDACLK - 0x4000 40E0)

The IRDACLK register controls the IrDA X/Y clock divider. This divider takes PERIPH_CLK as input and outputs a divided IRDA_CLK. This clock is used by the IrDA block associated with UART6 when the IrDA block is configured to operate in fixed 3/16 of 115.2 kbps mode. This configuration is done by UART_CTRL[2:1]. The IRDA_CLK should be stopped in order to save power when the IrDA block is set to run at the UART6 bit rate and not a fixed 115.2 kbps rate. For PERIPH_CLK = 13 MHz, the value to program is 0x1386. (X=19 and Y=134). This outputs a 1.8432 MHz clock to the UART (16 times oversampling). The IrDA clocking scheme is shown in Figure 72.

Table 458. IrDA Clock Control Register (IRDACLK - 0x4000 40E0)

IRDACLK	Function	Description	Reset value
15:8	X divider value	If this value is set to 0, the output clock is stopped and the divider is put in a low power mode. See the description of the Y divider value below.	0
7:0	Y divider value	If this value is set to 0, the output clock is stopped and the divider is put in a low power mode. The X/Y divider divides the selected input clock using an X/Y divider. The output should be set to either 16 times the UART bit rate to be used, or a higher frequency if the UART baud rate generator (using the UnDLM and UnDLL registers) divides further down. Dividing directly down to 16 times the required bit rate is the most power efficient method.	0

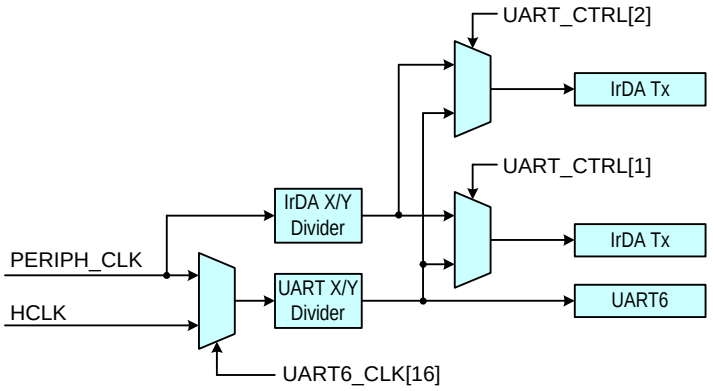


Fig 72. UART6 IrDA clocking

19.6.16 UART Control Register (UART_CTRL - 0x4005 4000)

The UART_CTRL register controls various details of the UART6 IrDA feature, as well as the connection of UART3 and UART5 pins.

Table 459. UART Control Register (UART_CTRL - 0x4005 4000)

UART_CTRL	Function	Description	Reset value
11	UART3_MD_CTRL	0: UART3 does not use modem control pins. 1: UART3 uses modem control pins	0
10	HDPX_INV	This inversion comes in addition to the IRRX6_INV controlled inversion. 0: IRRX6 is not inverted. 1: IRRX6 is inverted.	0
9	HDPX_EN	This is used for stopping IRRXD6 data received from the IrDA transceiver while transmitting (optical reflection suppression). 0: IRRX6 is not disabled by TXD. 1: IRRX6 is masked while TXD is low.	0
8:6	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
5	UART6_IRDA	modulator/demodulator. 0: UART6 uses the IrDA modulator/demodulator. 1: UART6 bypasses the IrDA	0
4	IRTX6_INV	0:The IRTX6 pin is not inverted. 1:The IRTX6 pin is inverted.	0
3	IRRX6_INV	0:The IRRX6 pin is not inverted. 1:The IRRX6 pin is inverted.	0

Table 459. UART Control Register (UART_CTRL - 0x4005 4000) *…continued*

UART_CTRL	Function	Description	Reset value
2	IR_RxLength	0:The IRDA expects Rx pulses 3/16 of the selected bit period. 1:The IRDA expects Rx pulses 3/16 of a 115.2 kbps bit period.	0
1	IR_TxLength	0:The IRDA Tx uses 3/16 of the selected bit period. 1:The IRDA Tx uses 3/16 of a 115.2 kbps bit period.	0
0	UART5_MODE	0:The UART5 TX/RX function is only routed to the U5_TX and U5_RX pins. 1:The UART5 TX/RX function is also routed to the USB D+ and D- pins.	0

19.6.17 UART Clock Mode Register (UART_CLKMODE - 0x4005 4004)

The UART_CLKMODE register selects the clocking mode for standard UARTs, and also provides status information about the clocking for all UARTs (including 14-clock UARTs).

Table 460. UART Clock Mode Register (UART_CLKMODE - 0x4005 4004)

UART_CLKMODE	Function	Description	Reset value
22:16	CLK_STATX	This read-only field provides the Individual status of all UART clocks. 0000000: No UART clocks are running xxxxxx1: The UART 1 clock is running. Refer to the 14-clock UART chapter. xxxxx1x: The UART 2 clock is running. Refer to the 14-clock UART chapter. xxxx1xx: The UART 3 clock is running. xxx1xxx: The UART 4 clock is running. xx1xxxx: The UART 5 clock is running. x1xxxxx: The UART 6 clock is running. 1xxxxxx: The UART 7 clock is running. Refer to the 14-clock UART chapter.	0
15	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	
14	CLK_STAT	This read-only bit indicates whether any UARTs (including 14-clock UARTs) are currently being clocked. This is useful when all UARTs are in the autoclock mode, as a check to determine if it is safe to enter stop mode. 0: No UART clocks are running. All UARTs are either turned off or in the auto-off state. 1: One or more UART clocks are running.	0
13:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 460. UART Clock Mode Register (UART_CLKMODE - 0x4005 4004) *...continued*

UART_CLKMODE	Function	Description	Reset value
11:10	UART6_CLK	Selects the clock mode for UART6. 00: Clock off mode (default) 01: Clock on mode 10: Auto clock mode 11: Not used	0
9:8	UART5_CLK	Selects the clock mode for UART5. The bit coding is the same as for UART6.	0
7:6	UART4_CLK	Selects the clock mode for UART4. The bit coding is the same as for UART6.	0
5:4	UART3_CLK	Selects the clock mode for UART3. The bit coding is the same as for UART6.	0
3:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
1:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

19.6.18 UART Loopback Control Register (UART_LOOP - 0x4005 4008)

The UART_LOOP register allows any of the seven UARTs to have the transmit output internally connected back to the receive input. This is generally done for testing purposes.

Table 461. UART Loopback Control Register (UART_LOOP - 0x4005 4008)

UART_LOOP	Function	Description	Reset value
6	LOOPBACK7	0 = UART7 loopback is turned off. 1 = UART7 is set to loopback mode.	0
5	LOOPBACK6	0 = UART6 loopback is turned off. 1 = UART6 is set to loopback mode. Note: The IRTX6 pin outputs a low in loopback mode when IrDA is enabled. When IrDA is bypassed the IRTX6 pin outputs a high as long as the IRTX6_INV bit is 1.	0
4	LOOPBACK5	0 = UART5 loopback is turned off. 1 = UART5 is set to loopback mode.	0
3	LOOPBACK4	0 = UART4 loopback is turned off. 1 = UART4 is set to loopback mode.	0
2	LOOPBACK3	0 = UART3 loopback is turned off. 1 = UART3 is set to loopback mode.	0
1	LOOPBACK2	0 = UART2 loopback is turned off. 1 = UART2 is set to loopback mode.	0
0	LOOPBACK1	0 = UART1 loopback is turned off. 1 = UART1 is set to loopback mode.	0

19.7 Baud rate calculation

Baud rates for the standard UARTs are based on either HCLK or PERIPH_CLK, and are divided by the fractional pre-divider (if used) and the UART baud rate generator (if used). The clocking scheme is shown in Figure 73.

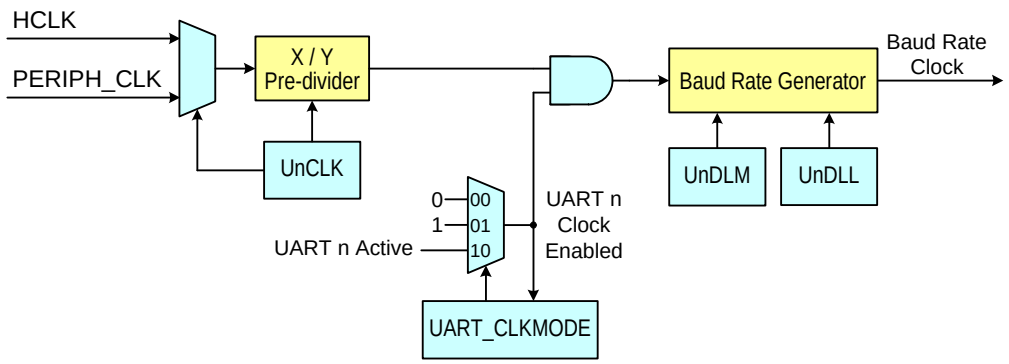


Fig 73. Baud rate generation for standard UARTs

The baud rate for one of the standard UARTs can be calculated from the equation:

$$\text{Baud Rate} = \text{Source Clock} \times \frac{X}{Y} \times \frac{1}{16 \times (256 \times \text{UnDLM} + \text{UnDLL})} \quad (13)$$

Source Clock in Equation 13 is either HCLK or PCLK as selected in the U[n]_CLK registers.

If power usage is an issue in an application, the preferred method of generating baud rates is to use only the pre-divider to create the desired rate clock for the UART. Alternatively, only the UART baud rate generator may be used to create the desired UART clock.

If power usage is not critical, and multiple baud rates are to be supported, the highest baud rate required may be generated by the pre-divider, while the UART baud rate generator divides that clock down to the actual desired rate.

19.7.1 Examples of calculating baud rate values

19.7.1.1 Rates generated using only the pre-divider

Table 462 shows examples of baud rate calculations using only the pre-divider. (Source Clock = PCLK, UnDLM = 0 and UnDLL = 1 in Equation 13)

Table 462. Baud rates generated using only the pre-divider

Desired baud rate	Desired clock (baud x 16)	Source clock	Rate adjustment value		Actual Baud rate	Rate error
	(MHz)	(MHz)	(DesiredCLK/SourceCLK) (Decimal)	(X / Y) (Fraction)		(%)
2400	0.0384	13	0.002954	1 / 255	3186.3	32.761
4800	0.0768	13	0.005908	1 / 169	4807.7	0.160
9600	0.1536	13	0.011816	3 / 254	9596.5	-0.037
19200	0.3072	13	0.023631	3 / 127	19192.9	-0.037
38400	0.6144	13	0.047262	6 / 127	38385.8	-0.037
57600	0.9216	13	0.070892	9 / 127	57578.7	-0.037
115200	1.8432	13	0.141785	19 / 134	115205.2	0.005
230400	3.6864	13	0.283569	19 / 67	230410.4	0.005
460800	7.3728	13	0.567138	38 / 67	460820.9	0.005

19.7.1.2 Rates generated using only the UART baud rate generator

Table 463 shows examples of baud rate calculations using only the UART baud rate generator. (Source Clock = PCLK, X = 1 and Y = 1 in Equation 13)

Table 463. Baud rates generated using only the UART baud rate generator

Desired baud rate	Source clock (MHz)	UnDLM : UnDLL (Raw)	UnDLM : UnDLL (Rounded)	Actual baud rate $\frac{\text{Source Clock}}{16 \times (\text{UnDLM:UnDLL})}$	Rate error %
2400	13	338.54	339	2396.76	–0.14
4800	13	169.27	169	4807.69	0.16
9600	13	84.64	85	9558.82	–0.43
19200	13	42.32	42	19345.24	0.76
38400	13	21.16	21	38690.48	0.76
57600	13	14.11	14	58035.71	0.76
115200	13	7.05	7	116071.43	0.76
230400	13	3.53	4	203125.00	–11.84
460800	13	1.76	2	406250.00	–11.84

19.8 IRDA encoding and decoding

The IrDA block associated with UART6 includes an encoder and decoder for the IrDA standard protocol. When in this mode, UART6 will communicate with an external IrDA transmitter/receiver module, and supports a maximum performance of up to 115.2 kbps. The connections that are unique to UART6 are shown in Figure 74.

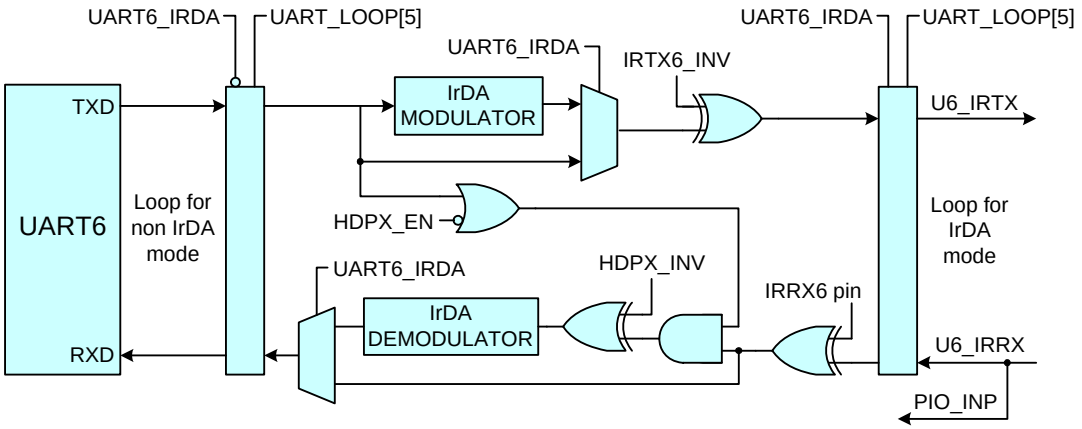


Fig 74. UART6 connections

The IrDA encoder generates the transformation from UART frame to IrDA frame according to Figure 75. It is possible to select the IrDA pulse width to be 3/16 of the actual bit rate or 3/16 of a 115.2 kbps pulse width according to Table 464.

When using the IrDA UART in the 3/16 of 115.2 kbps pulse mode, the IrDA clock will be generated by the special IrDA clock generator, controlled by the IRDACLK register. The IrDA clock generator uses the PERIPH_CLK as input and outputs a 1.8432 MHz IRDA_CLK. There is no restriction that the UART_CLK must be 7.3728 MHz. The IrDA clock generator is automatically enabled when IrDA is selected with fixed pulse width. (UART_CTRL[5] = 0 and either UART_CTRL[2] = 1 or UART_CTRL[1] = 1)

The receive path must also be configured to accept the short pulses. This is done by setting bit 2 (IR_RxLength) in the UART_CTRL register.

If an IrDA transceiver module with pulse-shaping on the receiver for short pulses is used, the IR_RxLength bit must always be set.

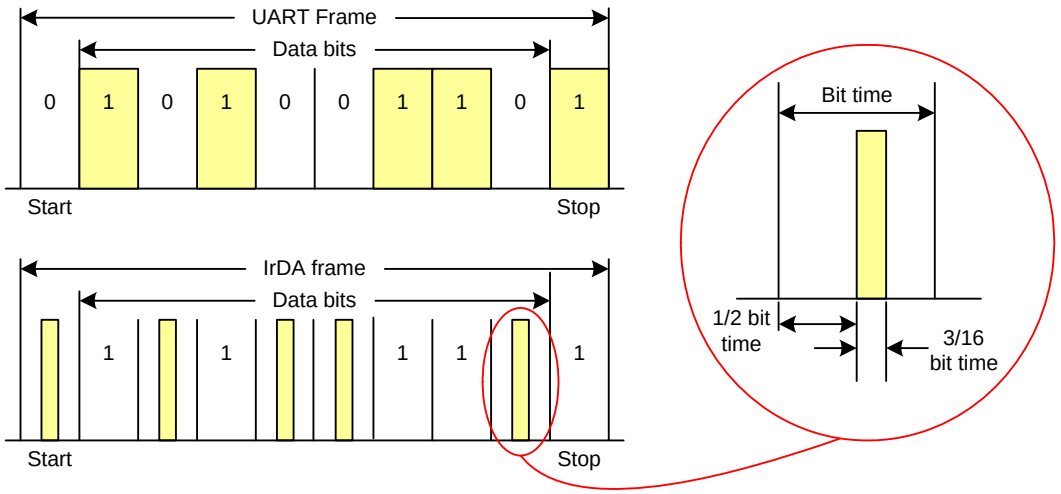


Fig 75. UART6 connections

Table 464. IrDA pulse timing

Bit rate (kbps)	Bit rate tolerance (% of bit rate)	Nominal pulse width (115-mode) (in µs)	Nominal pulse width (µs)		
			Min.	Nom.	Max.
4.8	± 0.87	1.63		88.55	
9.6	± 0.87	1.63		19.53	22.13
19.2	± 0.87	1.63		9.77	11.07
38.4	± 0.87	1.63		4.88	5.96
57.6	± 0.87	1.63		3.26	4.34
115.2	± 0.87	1.63	1.41	1.63	2.32

20.1 Introduction

The LPC32x0 contains seven UARTs, three of which are referred to as 14-clock UARTs. The 14-clock UARTs are described in this chapter. The four standard UARTS are described in the standard UART chapter.

20.2 Features

- Each 14-clock UART has 64 byte Receive and Transmit FIFOs.
- Receiver FIFO trigger points at 1, 4, 8, 16, 32, and 48 bytes.
- Transmitter FIFO trigger points at 0, 4, 8, and 16 bytes.
- Each 14-clock UART has an internal rate generator.
- The 14-clock UARTs are designed to support data rates of 2400; 4800; 9,600; 19,200; 38,400; 57,600; 115,200; 230,400; 460,800; and 921,600 bps.
- The three 14-clock UARTs only support a the communications format of (8N1) 8-bit data word length, 1-stop bit, no parity, and no flow control.
- The 14-clock UARTs always run in auto-clock mode, which shuts off the clock to the UART when it is idle.
- Each UART includes an internal loopback mode.

20.3 Pin description

Table 465. UART1, 2, and 7 pin description

Pin name	Type	Description
Un_Rx	Input	Receive data input. Serial data to the UART is input on this pin.
Un_Tx	Output	Transmit data output. Serial data from the UART is output on this pin.
Un_HCTS	Input	Clear To Send. Active LOW input signal indicates that an external device is ready to accept transmitted data from the associated UART. Available for UARTs 2 and 7 only.
Un_HRTS	Output	Request To Send. Active LOW output signal indicates that the associated UART wishes to transmit data to an external device. Available for UARTs 2 and 7 only.

20.4 14-clock UART base addresses

Table 466. 14-clock UART base addresses

UART	Base address
1	0x4001 4000
2	0x4001 8000
7	0x4001 C000

20.5 Functional description

The three 14-clock UARTs use 14 times over-sampling instead of 16 times, which is the typical over-sampling rate for a UART. With an input clock running at 13 MHz this gives a maximum standard bit rate of 921,600 bps.

The architecture of the 14-clock UARTs is shown in the block diagram, Figure 76. The UART receiver monitors the serial input line for valid input. The UART Rx Shift Register accepts valid characters via the Un_RX pin. After a valid character is assembled in the Rx Shift Register, it is passed to the receive FIFO to await access by the CPU. The receiver for UARTs 2 and 7 can be configured to generate a Request To Send (RTS) signal as a handshake to control pacing of incoming data.

The UART transmitter accepts data written by the CPU or host and buffers the data in the transmit FIFO. The UART Tx Shift Register reads the data stored in the transmit FIFO and assembles the data to transmit via the serial output pin, Un_TX. The transmitter for UARTs 2 and 7 can be configured to accept a Clear To Send (CTS) signal as a handshake to control pacing of outgoing data.

The Rate Generator block generates the timing used by the UART transmitter and receiver. The interrupt interface is controlled by bits in the HSUn_CTRL register and provides status information via the HSUn_IIR register. Control information for the transmitter and receiver is stored in additional bits in HSUn_CTRL and additional status information is provided via bits in the HSUn_IIR register.

20.5.1 UART clock mode

The 14-clock UARTs always run in autoclock mode. In autoclock mode, the clock is normally switched off but is automatically switched on by hardware when required. The automatic function works for both transmit and receive. An incoming start bit is detected and turns the receiver clock on. The clock is switched off again when the receiver goes idle. When data is written to the transmit buffer, the clock is switched on, and remains on until the last data is transmitted. The clock control mechanism is applied to all parts of the UART and may be used to save power.

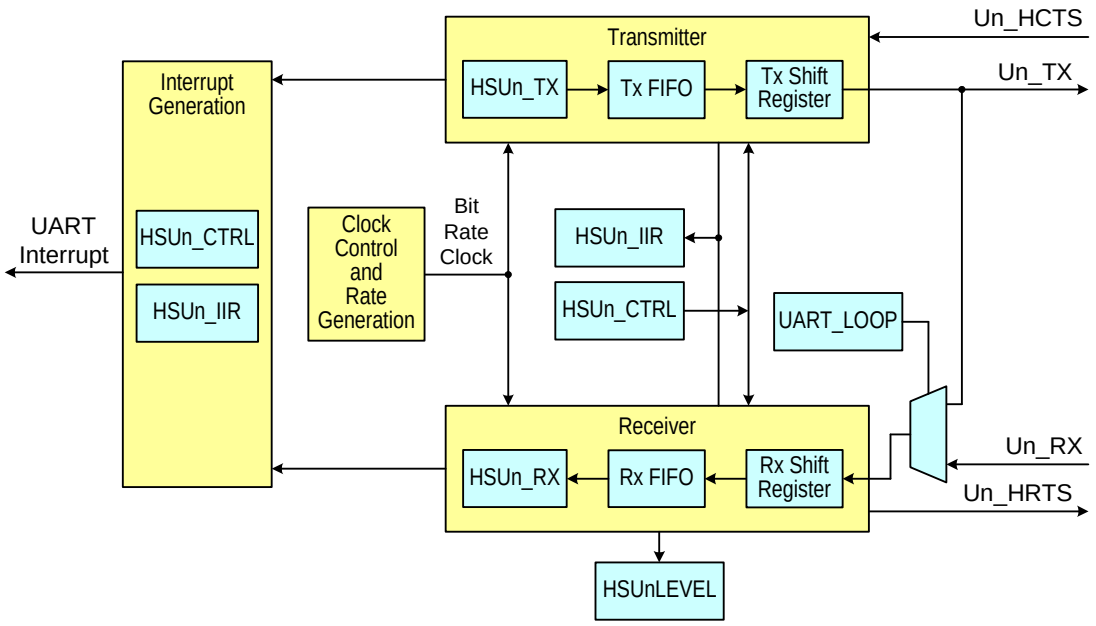


Fig 76. 14-clock UART block diagram

The details of how the 14-clock UARTs connect to device pins is shown in Figure 77.

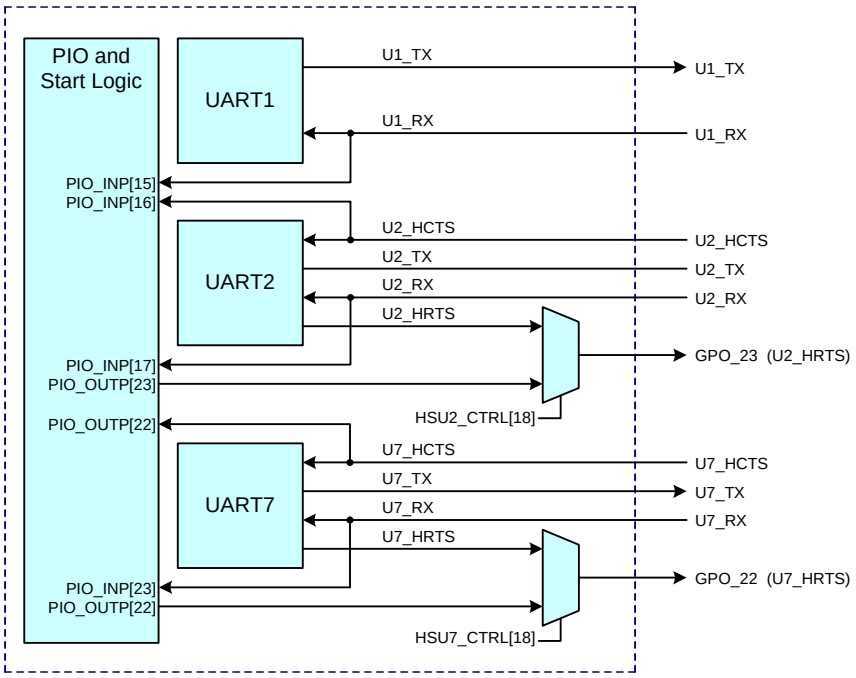


Fig 77. 14-clock UART pin connections

20.5.2 DMA support

The 14-clock UARTs have support for DMA. This is implemented by using the TX/RX Interrupts as request signals to the DMA controller. In addition, there are signals from the DMA controller that clear the DMA requests when the operation completes. The burst size of the DMA channel must match the trigger level set in HSU_CTRL register.

20.6 Register description

Each 14-clock UART contains registers as shown in Table 467. Address offsets are shown in the first column. Each UART contains that register at the base address from Table 466 plus the offset value from Table 467.

Table 467. 14-clock UART register summary

Address offset	Name	Description	Reset value	Type
0x00	HSUn_RX	14-clock UARTn Receiver FIFO	0x1XX	RO
0x00	HSUn_TX	14-clock UARTn Transmitter FIFO	-	WO
0x04	HSUn_LEVEL	14-clock UARTn FIFO Level Register	0	RO
0x08	HSUn_IIR	14-clock UARTn Interrupt Identification Register	0	R/W
0x0C	HSUn_CTRL	14-clock UARTn Control Register	0x0000 2800	R/W
0x10	HSUn_RATE	14-clock UARTn Rate Control Register	0	R/W

20.6.1 14-clock UARTn Receiver FIFO Register (HSUn_RX - 0x4001 4000, 0x4001 8000, 0x4001 C000)

The read-only HSUn_RX register allows reading the top byte of the Receiver FIFO of 14-clock UARTn. The top byte of the Rx FIFO contains the oldest character received. Bit 0 always contains the first received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

Table 468. 14-clock UARTn Receiver FIFO Register (HSUn_RX - 0x4001 4000, 0x4001 8000, 0x4001 C000)

HSUn_RX	Function	Description	Reset value
10	HSU_BREAK	This bit indicates whether a break condition has been encountered. This bit is the same as HSU_IIR[4] 0: No break. 1: A break condition has been received.	0
9	HSU_ERROR	This bit reads indicates whether a framing error or an overflow error has been detected. This bit applies to the attached character in bits [7:0]. 0: No errors 1: An error has occurred	0
8	HSU_RX_EMPTY	This bit gives the status of the receiver FIFO. 0: One or more data bytes is available in the receiver FIFO. 1: The receiver FIFO is empty.	1
7:0	HSU_RX_DATA	Received data from the UARTn receiver FIFO.	-

20.6.2 14-clock UARTn Transmitter FIFO Register (HSUn_TX - 0x4001 4000, 0x4001 8000, 0x4001 C000)

The write-only HSUn_TX register accesses the top byte of the Transmit FIFO of UARTn. The top byte is the newest character in the Tx FIFO. The LSB represents the first bit to transmit.

Table 469. 14-clock UARTn Transmitter FIFO Register (HSUn_TX - 0x4001 4000, 0x4001 8000, 0x4001 C000)

HSUn_TX	Function	Description	Reset value
7:0	HSU_TX_DATA	Writing to the UARTn Transmit Data 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.	-

20.6.3 14-clock UARTn Level Register (HSUn_LEVEL - 0x4001 4004, 0x4001 8004, 0x4001 C004)

The HSUn_Level register is a read-only register that provides the current level of the receiver FIFO for UARTn. This allows software to have more information about UART activity than provided by the FIFO level interrupt.

Table 470. 14-clock UARTn Level Register (HSUn_LEVEL - 0x4001 4004, 0x4001 8004, 0x4001 C004)

HSUn_LEVEL	Function	Description	Reset value
15:8	HSU_TX_LEV	Current transmitter FIFO level.	0
7:0	HSU_RX_LEV	Current receiver FIFO level.	0

20.6.4 14-clock UARTn Interrupt Identification Register (HSUn_IIR - 0x4001 4008, 0x4001 8008, 0x4001 C008)

The HSUn_IIR register provides status of pending interrupt in the corresponding UART. HSUn_IIR also provides a means to clear most interrupts.

Table 471. 14-clock UARTn Interrupt Identification Register (HSUn_IIR - 0x4001 4008, 0x4001 8008, 0x4001 C008)

HSUn_IIR	Function	Description	Reset value
6	HSU_TX_INT_SET	This write-only bit allows forcing a transmit interrupt as a method of starting a DMA transfer. 0: Writing a 0 has no effect. 1: Set the transmit interrupt flag.	0
5	HSU_RX_OE	This bit allows checking and clearing the overrun error flag. Read: 0: No overflow interrupt. 1: An overrun condition has occurred. Write: 0: Writing a 0 has no effect. 1: Clears the overrun interrupt.	0
4	HSU_BRK	This bit allows checking and clearing the break flag. Read: 0: No break interrupt. 1: A break condition has occurred (the stop-bit is zero, and all data bits are zero). Write: 0: Writing a 0 has no effect. 1: Clears the break interrupt.	0
3	HSU_FE	This bit allows checking and clearing the framing error flag. Read: 0: No framing error interrupt. 1: A framing error has occurred (the stop-bit is zero). Write: 0: Writing a 0 has no effect. 1: Clears the framing error interrupt.	0

Table 471. 14-clock UARTn Interrupt Identification Register (HSUn_IIR - 0x4001 4008, 0x4001 8008, 0x4001 C008)

HSUn_IIR	Function	Description	Reset value
2	HSU_RX_TIMEOUT	This read-only bit allows checking the receiver timeout flag. This bit is only set if the timeout interrupt is enabled in the HSUX_CONTROL register. This bit may be cleared by reading data from HSUn_RX. 0: A receiver timeout has not occurred. 1: The receiver timeout interrupt is active.	0
1	HSU_RX_TRIG	This read-only bit allows checking the receiver trigger level. This bit may be cleared by reading data from HSUn_RX until the receiver FIFO is below the trigger level. 0: The receiver FIFO is below the trigger level. 1: The receiver FIFO is above the trigger level.	0
0	HSU_TX	This read-only bit allows checking and clearing the transmitter interrupt. The transmit interrupt can also be cleared by writing data to the transmit FIFO. Read: 0: The transmit interrupt is inactive. 1: The transmit interrupt is active. Write: 0: Writing a 0 has no effect. 1: Clear the transmit interrupt.	0

20.6.5 14-clock UARTn Control Register (HSUn_CTRL - 0x4001 400C, 0x4001 800C, 0x4001 C00C)

The HSUn_CTRL register controls various details of 14-clock UART operation. These include the FIFO trigger depths, handshake enable, polarity of handshake signals, and interrupt enables.

Table 472. 14-clock UARTn Control Register (HSUn_CTRL - 0x4001 400C, 0x4001 800C, 0x4001 C00C)

HSUn_CTRL	Function	Description	Reset value
21	HRTS_INV	This bit controls the polarity of the Un_HRTS signal. The polarity applies only to the UART, not the IO signal. Supported only by UARTs 2 and 7. 0: HRTS is not inverted. 1: HRTS is inverted.	0
20:19	HRTS_TRIG	This field controls the hardware RTS flow control trigger level. The Un_HRTS pin is set low when the RX FIFO level is above the specified value. Supported only by UARTs 2 and 7. 00: 8 bytes. 01: 16 bytes. 10: 32 bytes. 11: 48 bytes.	0
18	HRTS_EN	Controls the enabling of hardware RTS flow control. Supported only by UARTs 2 and 7. When enabled, the Un_HRTS pin is set low when the RX FIFO level is above the specified value in HSUn_CTRL[20:19]. 0: Hardware RTS flow control is disabled. The HRTS pin is controlled by the PIO block. 1: Hardware RTS flow control is enabled.	0
17:16	TMO_CONFIG	Configures the receiver timeout interrupt. 00: The receiver timeout interrupt is disabled. Use this configuration with DMA. 01: The timeout is set when the receiver is inactive for 4 character times. 10: The timeout is set when the receiver is inactive for 8 character times. 11: The timeout is set when the receiver is inactive for 16 character times.	0
15	HCTS_INV	This bit controls the polarity of the Un_HCTS signal. The polarity applies only to the UART, not the PIO signal. Supported only by UARTs 2 and 7. 0: HCTS is not inverted. 1: HCTS is inverted.	0
14	HCTS_EN	Controls the enabling of hardware CTS flow control. If this bit is set to one the transmit shift register will stop sending more data when it has finished the current character when Un_HCTS pin is low. Supported only by UARTs 2 and 7. 0: Transmit flow control is disabled. 1: Transmit flow is controlled by the Un_HCTS pin.	0

Table 472. 14-clock UARTn Control Register (HSUn_CTRL - 0x4001 400C, 0x4001 800C, 0x4001 C00C) *...continued*

HSUn_CTRL	Function	Description	Reset value
13:9	HSU_OFFSET	Sets the first bit sampling point. The first bit will be sampled the number of clocks specified by HSU_OFFSET after the start bit is detected. This allows adjusting the sampling time to compensate for delays at high bit rates. 00000: 0 clocks offset. 00001: 1 clocks offset. 00010: 2 clocks offset. : 10100: 20 clocks offset (value after reset). : 11111: 31 clocks offset.	0x14
8	HSU_BREAK	This bit controls the generation of a break on the transmit data line. When break is enabled, the Un_TX line is driven low. When break is disabled, Un_TX is driven by the transmitter state machine. 0: Disable break. 1: Enable break.	0
7	HSU_ERR_INT_EN	This bit controls the generation of an interrupt when there is an error detected in received data. The interrupt reflects framing error, break, and overrun error conditions. 0: Disable the UARTn error interrupt. 1: Enable the UARTn error interrupt.	0
6	HSU_RX_INT_EN	This bit controls the generation of a receive interrupt. 0: The receive interrupt is disabled. 1: The receive interrupt is enabled.	0
5	HSU_TX_INT_EN	This bit controls the generation of a transmit interrupt. 0: The transmit interrupt is disabled. 1: The transmit interrupt is enabled.	0
4:2	HSU_RX_TRIG	This field selects the receiver FIFO trigger level. 000: 1 byte. 001: 4 bytes. 010: 8 bytes. 011: 16 bytes. 100: 32 bytes. 101: 48 bytes. 110 to 111: Reserved.	0
1:0	HSU_TX_TRIG	This field selects the transmitter FIFO trigger level 00: 0 (empty). 01: 4 bytes. 10: 8 bytes. 11: 16 bytes.	0

20.6.6 14-clock UARTn Rate Control Register (HSUn_RATE - 0x4001 4010, 0x4001 8010, 0x4001 C010)

The HSUn_RATE register holds the value used to divide the UART clock in order to produce the bit rate clock. 14-clock UARTs use 14x oversampling, so the rate equation includes this value. Refer to the Rate Calculation section for more details on bit rate generation.

Table 473. 14-clock UARTn Rate Control Register (HSUn_RATE - 0x4001 4010, 0x4001 8010, 0x4001 C010)

HSUn_RATE	Function	Description	Reset value
7:0	HSU_RATE	Controls the 14-clock UART clock divider, setting the UART rate. The UART rate = PERIPH_CLK / ((HSU_RATE+1) × 14) 0: UARTn bit rate is PERIPH_CLK divided by 1 × 14. 1: UARTn bit rate is PERIPH_CLK divided by 2 × 14. 255: 1: UARTn bit rate is PERIPH_CLK divided by 256 × 14.	0

20.6.7 Other relevant registers

Some registers described in the Standard UART chapter also have relevance to 14-clock UARTs.

20.6.7.1 Clock status

The status of clocks to all seven UARTs is reflected in the CLK_STATX and CLK_STAT fields of the UART_CLKMODE register, which is described in the Standard UART chapter. Using these fields, it can be determined whether clocks to a particular 14-clock UART have been turned off by the autoclocking feature.

20.6.7.2 Loopback mode

Any of the 14-clock UARTs can be put into loopback mode by using bits in the UART_LOOP register. This register controls all 7 UARTS and is described in the Standard UART chapter.

20.7 Rate calculation for the 14-clock UARTs

The bit rates of the 14-clock UARTs are based PERIPH_CLK. A simple divider allows selecting the desired bit rate. In order to use high bit rates with the 14-clock UARTs, the frequency of PERIPH_CLK must be close to an even multiple of 14 times the desired rate. The nominal frequency of PERIPH_CLK is considered to be 13 MHz.

The UART rate is given by the equation: PERIPH_CLK / ((HSU_RATE+1) × 14)

Examples of 14-clock UART bit rates are shown in Table 474.

Table 474. Examples of 14-clock UART bit rates

PERIPH_CLK Frequency (MHz)	Desired bit rate	Divide value (Raw)	Divide value (Rounded)	HSUn_RATE value	Actual bit rate	Rate error %
13	2400	386.90	387	386	2399.41	−0.02
13	4800	193.45	193	192	4811.25	0.23
13	9600	96.73	97	96	9572.90	−0.28

Table 474. Examples of 14-clock UART bit rates

PERIPH_CLK Frequency (MHz)	Desired bit rate	Divide value (Raw)	Divide value (Rounded)	HSUn_RATE value	Actual bit rate	Rate error %
13	19200	48.36	48	47	19345.24	0.76
13	38400	24.18	24	23	38690.48	0.76
13	57600	16.12	16	15	58035.71	0.76
13	115200	8.06	8	7	116071.43	0.76
13	230400	4.03	4	3	232142.86	0.76
13	460800	2.02	2	1	464285.71	0.76
13	921600	1.01	1	0	928571.43	0.76

20.8 UART timing

Figure 78 shows the timing details of the 14 bit over-sampling used by the 14-clock UARTs, as well as how the data sampling offset operates. The offset feature allows compensation for timing issues at high bit rates. Refer to the description of the HSUn_CTRL register for details.

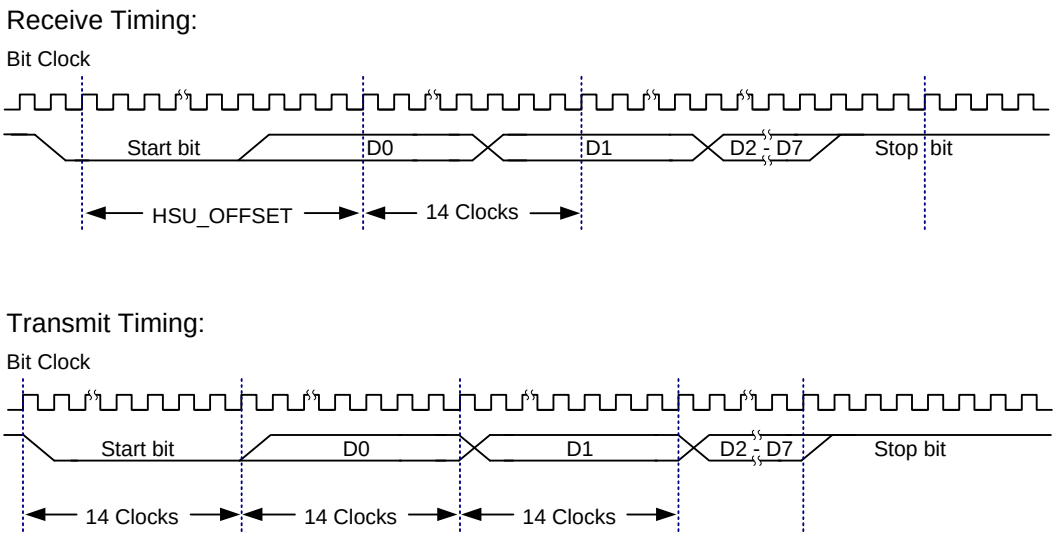


Fig 78. 14-clock UART timing

21.1 Introduction

The LPC32x0 has two Serial Peripheral Interfaces (SPI). The SPI is a 3-wire serial interface designed to interface with a large range of serial peripheral or memory devices (SPI mode 0 to 3 compatible slave devices). The SPI does not support operation as a slave.

21.2 Features

- Supports slaves compatible with SPI modes 0 to 3.
- Half duplex synchronous transfers.
- DMA support for data transmit and receive.
- 1 to 16 bit word length.
- Choice of LSB or MSB first data transmission.
- 64 x 16-bit input or output FIFO.
- Bit rates up to 52 Mbits per second.
- Busy input function.
- DMA time out interrupt to allow detection of end of reception when using DMA.
- Timed interrupt to facilitate emptying the FIFO at the end of a transmission.
- SPI clock and data pins may be used as general purpose pins if the SPI is not used.

In the following sections, the term SPIn refers to both of the SPI interfaces, essentially replacing the "n" with "1" or "2" in order to apply to SPI1 or SPI2.

21.3 Pin description

Table 475. SPI pin description

Pin name	Type	Description
SPIn_CLK	Input/ Output	SPIn_CLK is a clock signal used to synchronize the transfer of data across an SPI bus. The SPI is always driven by the master and received by the slave.
SPIn_DATIN	Input	The SPIn_DATIN pin inputs data.
SPIn_DATIO	Output	The SPIn_DATIO pin outputs data.
GPI_4 / SPI1_BUSY and GPI_8 / SPI2_BUSY	Input	SPIn_BUSY is an optional input that allows a slave to indicate that data transfer should pause.

21.4 Functional description

Following reset, the SPI pins are connected as GPIOs. To use each SPI interface the pins must be enabled in the SPIn_CON register.

The 3-wire serial interface consists of a serial data output (SPIn_DATIO), a serial data input (SPIn_DATIN), and a serial clock signal (SPIn_CLK). A fourth pin (SPIn_BUSY) may optionally be used to allow a slave to pause an SPI transfer. Single-master operations are supported by the interface. It is also possible to program SPIn_DATIO to be a bidirectional line.

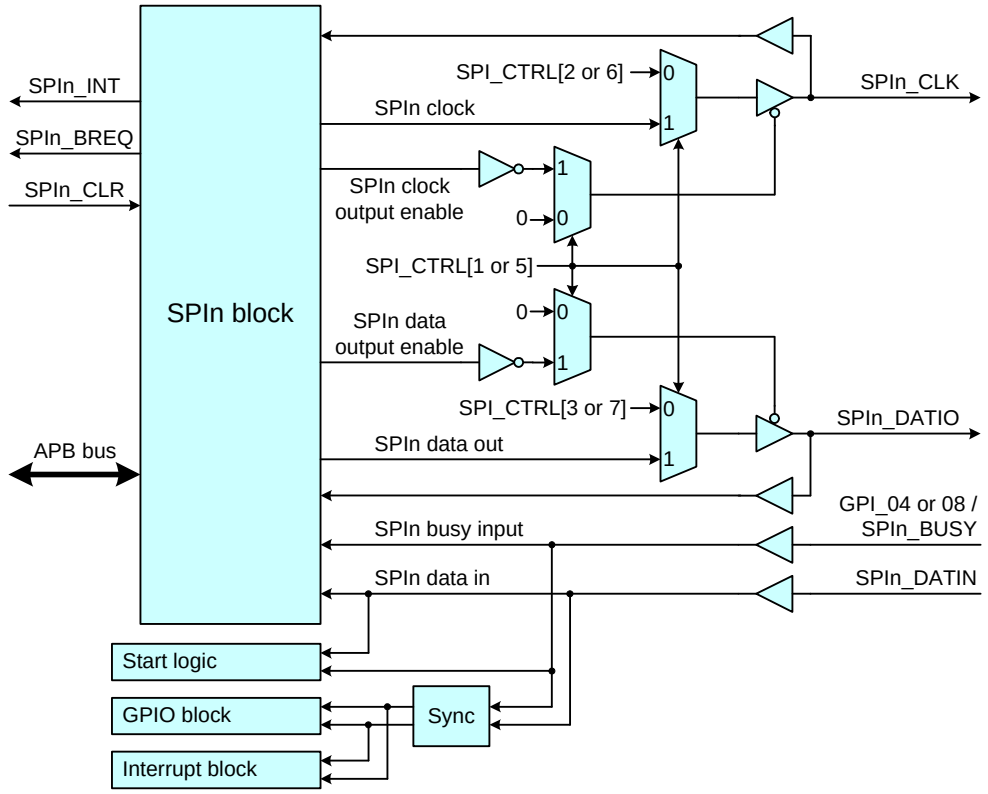


Fig 79. SPI pin connections and output logic

The SPIn_CLK pin is used to output the clock used for SPI data transfers. The master clock is generated by the internal SPI clock generator. The SPIn_DATIN pin and the SPIn_DATIO pin are the SPI data I/O-lines. For each slave device connected to the SPI master, a chip select signal must be generated, typically using a GPO or GPIO pin.

In order to use the interface, the enable bit in SPIn_GLOBAL must be set. Using the rst bit in SPIn_GLOBAL, a software controlled reset of the SPI interface can be initiated. The reset is only executed if the enable bit has been set.

SPI modes 0 to 3 are supported. The integrated FIFO allows continuous data transfers up to a programmed number of SPI frames. The frame length can be configured between 1 and 16 bits.

The moment the SPI_FIFO is accessed (either reading from it or writing to it), SPI clock activity will occur (assuming the shift_off bit in the SPIn_CON register is set to 1, see Table 479).

21.4.1 Single frame transfers

Transfers of a single SPI frame can be executed when the SPIn_FRM register is set to zero and the FIFO is empty. Data coming from or going to the external device can be accessed via the SPIn_DAT register. This register consists of a separate read and write register. The read part of the register provides incoming data from the shift register; the write part of the register delivers data to the shift register. The size of the data frame to be transferred is determined by the bitnum field in the SPIn_CON register. The clock of the shift register is based on the output of the SPI clock generator. The bits of the SPIn_DAT write register are shifted out on SPIn_DATIO or the bits on SPIn_DATIN are shifted into the shift register (depending on the rxtx bit in SPIn_CON).

Sending data is accomplished by writing the data to be sent to the SPIn_DAT register, which is copied to the shift register. Data is clocked out on SPIn_DATIO. SPIn_DAT may be written with new data while the previous data is being shifted out. An end of transfer interrupt is always generated after bitnum+1 SPIn_CLK clock cycles, if the interrupt is enabled via the inteot bit in the SPIn_IER register.

To read data, the SPI can be started by doing a dummy read or dummy write to SPIn_DAT. Received data may be read from SPIn_DAT after bitnum+1 SPI clock cycles.

A data read sequence can be stopped by setting the shift_off bit in the SPIn_CON register. Setting shift_off is not necessary if the SPI interface is turned off prior to reading the SPIn_DAT register for the last time.

21.4.2 Block transfers

A block transfer on the SPI can be used to transfer a number of frames. For examples, these could be pages of data to or from an external SPI memory device. The SPIn_FRM register contains the number of consecutive SPI frames to transfer. The threshold interrupt should be enabled if the number of frames exceeds the FIFO size. The FIFO pointer controls the read or write actions to the FIFO and generates an interrupt request if the number of entries falls below the threshold in transmit mode or rises above the threshold in receive mode.

The data transfer is only stopped if the receive FIFO is full during receive mode or the transmit FIFO is empty during transmit mode. In this case the transfer will continue after software reads or writes data, until the frame count reaches zero. At that point, the end of transfer interrupt is flagged.

A block transfer can be configured as follows.

- Check that the FIFO is empty and no transfer is running (SPIn_STAT).
- Define the SPI frame size (the bitnum field in SPIn_CON).
- Load the SPIn_FRM register with the number of SPI frames to be transferred.
- Enable the threshold interrupt if required.

The master can initiate the transfer as follows.

- Read the SPIn_DAT register if a block should be transferred from the external device.
- Write the data word of the first SPI frame to the SPIn_DAT register if a block should be transferred to the external device. This frame is included in the value of SPIn_FRM.

Every time a threshold interrupt is asserted, software must read or write data from SPIn_DAT to maintain a continuous data transfer on the interface. After the complete block is transferred, the end of transfer interrupt is asserted. If there are still entries in the FIFO at that point, they should be read by the software.

The status of the block transfer may be checked by reading the shiftact bit in SPIn_STAT. This bit is set when the first frame is being transferred, and cleared when all frames defined in SPIn_FRM have been transmitted. Writing a value of zero to SPIn_FRM clears the shiftact status flag.

21.4.3 DMA mode

During reception, DMA requests are generated if the FIFO is not empty. During transmit, DMA requests are generated if the FIFO is not full. The DMA controller must be configured to respond appropriately to these requests. The DMA burst size should always be programmed to 1 on the SPI side.

In DMA master receive mode, an initial dummy read or write to SPIn_DAT is needed to start the reception. After that, DMA transfer requests are generated when the FIFO is not empty.

The FIFO threshold interrupt must be disabled, and for block transfer either the DMA Controller terminal count or the SPI end of transfer interrupt may be used.

Note: In reception, the SPI end of transfer interrupt (inteot) may be asserted while some data remains inside the DMA Controller buffers.

21.4.4 Busy signal

The SPIn_BUSY signal may be necessary for some slave devices which need time to complete internal operations. When enabled, the SPI interface will stop generation of SPIn_CLK pulses while SPIn_BUSY is asserted. If SPIn_BUSY becomes active the current transfer will stall regardless the state of the transfer (i.e. the transfer can be stopped in a middle of a word).

21.4.5 Single-master multiple-slave support

Multiple chip select output pins are needed to select between multiple slaves. GPIO or GPO signals may be used for this purpose. Software must ensure that only one chip select output is active at any time.

21.5 Register description

The registers in Table 476 give control over the SPI interfaces.

Table 476. Summary of SPI registers

Address	Name	Description	Reset state	Access
0x4000 40C4	SPI_CTRL	SPI1 and SPI2 clock and pin control	0	R/W
0x2008 8000 0x2009 0000	SPI1_GLOBAL; SPI2_GLOBAL	SPIn Global Control Register. Controls resetting and enabling of SPI1 and SPI2.	0	R/W
0x2008 8004 0x2009 0004	SPI1_CON; SPI2_CON	SPIn Control Register. Controls many details of SPI operation.	0x0E08	R/W

Table 476. Summary of SPI registers *…continued*

Address	Name	Description	Reset state	Access
0x2008 8008 0x2009 0008	SPI1_FRM; SPI2_FRM	SPIn Frame Count Register. Selects the number of SPI frames to be transferred.	0	R/W
0x2008 800C 0x2009 000C	SPI1_IER; SPI2_IER	SPIn Interrupt Enable Register. Enables or disables the 3 types of interrupts that may be generated by the SPI.	0	R/W
0x2008 8010 0x2009 0010	SPI1_STAT; SPI2_STAT	SPIn Status Register. Provides information on conditions in the SPI interface.	0x01	R/W
0x2008 8014 0x2009 0014	SPI1_DAT SPI2_DAT	SPIn Data Buffer Register. Provides access to the transmit and receive FIFO buffers.	0	R/W
0x2008 8400 0x2009 0400	SPI1_TIM_CTRL SPI2_TIM_CTRL	SPIn Timer Control Register. Controls the generation of timed interrupts.	0x02	R/W
0x2008 8404 0x2009 0404	SPI1_TIM_COUNT SPI2_TIM_COUNT	SPIn Timer Counter Register. This is the counter for timed interrupts.	0	R/W
0x2008 8408 0x2009 0408	SPI1_TIM_STAT SPI2_TIM_STAT	SPIn Timer Status Register. Contains the timed interrupt pending flag.	0	R/W

21.5.1 SPI Control register (SPI_CTRL - 0x4000 40C4)

The SPI_CTRL register controls some aspects of the two SPI interfaces: enabling and disabling clocks; connecting the interface to the related pins; and controlling pin output values if the SPI interface is not used (i.e. used as a GPO). The SPI clock enable bit (0 and 1) must be enabled to use the peripheral, this includes the use of the SPI as GPO.

Table 477. SPI Control register (SPI_CTRL - 0x4000 40C4)

Bit	Function	Reset value
7	SPI2_DATIO output level. 0 = The pin drives low, (if bit 5 is 0). 1 = The pin drives high, (if bit 5 is 0).	0
6	SPI2_CLK output level. 0 = The pin drives low, (if bit 5 is 0). 1 = The pin drives high, (if bit 5 is 0).	0
5	Output pin control. By default, the SPI2_DATIO and SPI2_CLK pins are driven to the values set in bits 7 and 6. In order to use the SPI2 block, this bit must be written to a 1. 0 = SPI2_DATIO and SPI2_CLK outputs the level set by bit 6 and 7. 1 = SPI2_DATIO and SPI2_CLK are driven by the SPI2 block.	0
4	SPI2 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
3	SPI1_DATIO output level. 0 = The pin drives low, (if bit 1 is 0). 1 = The pin drives high, (if bit 1 is 0).	0

Table 477. SPI Control register (SPI_CTRL - 0x4000 40C4) *…continued*

Bit	Function	Reset value
2	SPI1_CLK output level. 0 = The pin drives low, (if bit 1 is 0). 1 = The pin drives high, (if bit 1 is 0).	0
1	Output pin control. By default, the SPI1_DATIO and SPI1_CLK pins are driven to the values set in bits 3 and 2. In order to use the SPI1 block, this bit must be written to a 1. 0 = SPI1_DATAIO and SPI1_CLK outputs the level set by bit 2 and 3. 1 = SPI1_DATIO and SPI1_CLK are driven by the SPI1 block.	0
0	SPI1 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

21.5.2 SPIn Global Control register (SPIn_GLOBAL - 0x2008 8000, 0x2009 0000)

The SPIn_GLOBAL register contains control bits that allow enabling and/or resetting the related SPI interface.

Table 478. SPIn Global Control register (SPIn_GLOBAL - 0x2008 8000, 0x2009 0000)

SPIn_GLOBAL	Function	Description	Reset value
1	rst	Allows software reset of the SPI interface. 0: No action. 1: The SPI interface is reset. The interface must be enabled to be active.	0
0	enable	Enables the SPI interface. 0: The SPI interface is disabled. 1: The SPI interface is enabled.	0

21.5.3 SPIn Control register (SPIn_CON - 0x2008 8004, 0x2009 0004)

The SPIn_CON register contains bits that control many aspects of SPI operation. These include selection of how the pins are used, the frame size, the SPI mode (how data and clocks are related), and the FIFO thresholds.

Table 479. SPIn Control register (SPIn_CON - 0x2008 8004, 0x2009 0004)

SPIn_CON	Function	Description	Reset value
23	unidir	Selects bidirectional or unidirectional usage of the SPIn_DATIO pin. 0: The SPI operates with the bidirectional data line SPIn_DATIO 1: The SPI operates with unidirectional input and output pins SPIn_DATIN and SPIn_DATIO reset	0
22	bhalt	Busy halt. Determines whether the SPIn_BUSY affects SPI operation. 0: The SPIn_BUSY pin is ignored during master operation. 1: Data transfer is halted if SPIn_BUSY is active during master operation.	0
21	bpol	Busy polarity. Controls the polarity of the SPIn_BUSY signal. 0: SPIn_BUSY is active LOW. 1: SPIn_BUSY is active HIGH.	0
20	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
19	msb	Controls the order in which data bits are transferred. 0: Data is transferred MSB first. 1: Data is transferred LSB first.	0
18	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
17:16	mode	SPI mode selection. 00: SPI mode 0. Clock starts low, data is sampled at the clock rising edge. 01: SPI mode 1. Clock starts low, data is sampled at the clock falling edge. 10: SPI mode 2. Clock starts high, data is sampled at the clock falling edge. 11: SPI mode 3. Clock starts high, data is sampled at the clock rising edge.	0
15	rxtx	Controls the direction of data transfer. 0: data is shifted into the SPI (receive) 1: data is shifted out the SPI (transmit)	0
14	thr	Controls the FIFO threshold. This determines the operation of the FIFO threshold interrupt flag in the SPIn_STAT register. For receive (the rxtx bit = 0): 0: The FIFO threshold is disabled, threshold = 1 entry in FIFO. 1: The FIFO threshold is enabled, threshold = 56 entries in FIFO. For transmit (the rxtx bit = 1): 0: The FIFO threshold is disabled. 1: The FIFO threshold is enabled, threshold=8 entries in FIFO.	0

Table 479. SPIn Control register (SPIn_CON - 0x2008 8004, 0x2009 0004) *…continued*

SPIn_CON	Function	Description	Reset value
13	shift_off	Controls generation of clock pulses on SPIn_CLK. 0: enables the generation of clock pulses on SPIn_CLK. 1: disables the generation of clock pulses on SPIn_CLK.	0
12:9	bitnum	Defines the number of bits to be transmitted or received in one block transfer (transmit or receive operation). The value is the number of bits - 1. The reset value gives 8 data bits.	0x7
8	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
7	ms	SPI master mode enable. 0: Not supported. 1: SPI is operating as a master.	0
6:0	rate	SPI transfer rate. SPIn_CLK = HCLK / (rate + 1) × 2). Refer to the Rate Calculation section for more details.	0x08

21.5.4 SPIn Frame Count register (SPIn_FRM - 0x2008 8008, 0x2009 0008)

The SPIn_FRM register specifies the number of SPI frames to be transferred when a block transfer is used.

Table 480. SPIn Frame Count register (SPIn_FRM - 0x2008 8008, 0x2009 0008)

SPIn_FRM	Function	Description	Reset value
15:0	spif	SPI frame count. This field specifies the number of SPI frames to be transferred (one frame is the number of bits specified in field bitnum in register SPIn_CON). Writing a zero to this field clears the shiftact status flag.	0

21.5.5 SPIn Interrupt Enable register (SPIn_IER - 0x2008 800C, 0x2009 000C)

The SPIn_IER register allows selection of which SPI interrupts are enabled.

Table 481. SPIn Interrupt Enable register (SPIn_IER - 0x2008 800C, 0x2009 000C)

SPIn_IER	Function	Description	Reset value
2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
1	inteot	End of transfer interrupt enable. 0: The end of transfer interrupt is disabled. 1: The end of transfer interrupt is enabled.	
0	intthr	FIFO threshold interrupt enable. 0: The FIFO threshold interrupt is disabled. 1: The FIFO threshold interrupt is enabled.	

21.5.6 SPIn Status Register (SPIn_STAT - 0x2008 8010, 0x2009 0010)

The SPIn_STAT register provides information on the activities of the SPI interface.

Table 482. SPIn Status Register (SPIn_STAT - 0x2008 8010, 0x2009 0010)

SPIn_STAT	Function	Description	Reset value
8	intclr	SPI interrupt clear. Writing a one to this bit clears the SPI interrupt. Writing a zero to this bit has no effect.	0
7	eot	End of transfer interrupt flag.[1][2] This flag is cleared by writing a 1 to bit 8. 0: The end of transfer has not been reached. 1: The end of transfer has been reached.	0
6	busylev	SPIn_BUSY level.[2] This bit gives the current level of the SPIn_BUSY input.	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3	shiftact	Shift active. Indicates when the SPI is transferring data.[2] 0: No SPI data transfer is in progress. 1: An SPI data transfer is in progress.	0
2	bf	FIFO full interrupt flag.[2] 0: The FIFO is not full. 1: The FIFO is full.	0
1	thr	FIFO threshold interrupt flag.[1][2] For rxtx=0: (receive) 0: The number of entries in the FIFO is below the threshold. 1: The number of entries in the FIFO is at or above the threshold. For rxtx=1: (transmit) 0: The number of entries in the FIFO is above the threshold. 1: The number of entries in the FIFO is at or below the threshold.	0
0	be	FIFO empty interrupt flag. [2] 0: The FIFO is not empty. 1: The FIFO is empty.	1

[1] These flags trigger the SPI interrupt.

[2] The hardware flag goes inactive after 2 HCLK cycles, but the SPI interrupt will remain active.

21.5.7 SPIn Data Buffer register (SPIn_DAT - 0x2008 8014, 0x2009 0014)

The SPIn_DAT register is the means to read incoming data and write outgoing data. Note that when a data size of less than 16 bits is selected in the bitnum field of SPIn_CON, the data is right-justified in SPIn_DAT for both reading and writing.

Table 483. SPIn Data Buffer register (SPIn_DAT - 0x2008 8014, 0x2009 0014)

SPIn_DAT	Function	Description	Reset value
15:0	spid	SPI data. When SPIn_DAT is read, an entry is read from the FIFO and deleted. When SPIn_DAT is written, an entry is added to the FIFO.	0

21.5.8 SPIn Timer Control register (SPIn_TIM_CTRL - 0x2008 8400, 0x2009 0400)

The SPIn_TIM_CTRL register controls the activities of the time out timer. The timer can generate interrupts when the FIFO contains data for longer than a predetermined time, or when an SPI DMA request is not serviced for longer than a predetermined time. Details of timer operation may be found in the Timed Interrupt and DMA Time-out Modes section following the register descriptions. The peripheral interrupt and timed interrupt are ORed together when both are enabled.

Table 484. SPIn Timer Control register (SPIn_TIM_CTRL - 0x2008 8400, 0x2009 0400)

SPIn_TIM_CTRL	Function	Description	Reset value
2	tirqe	Timed interrupt enable. 0: Timed interrupt is disabled. 1: Timed interrupt is enabled.	0x02
1	pirqe	Peripheral interrupt enable. 0: SPI status interrupt input disabled. 1: SPI status interrupt input enabled.	
0	mode	The mode bit determines how the timer is used. 0: Timed interrupt mode. 1: DMA time out mode.	

21.5.9 SPIn Timer Counter register (SPIn_TIM_COUNT - 0x2008 8404, 0x2009 0404)

The SPIn_TIM_COUNT register contains the clock count value that is used in both the timed interrupt and DMA Time-out modes. Writing a value to this register starts the timed interrupt counter from 0. Details of timer operation may be found in the Timed Interrupt and DMA Time Out Modes section following the register descriptions.

Table 485. SPIn Timer Counter register (SPIn_TIM_COUNT - 0x2008 8404, 0x2009 0404)

SPIn_TIM_COUNT	Function	Description	Reset value
15:0	count	This field contains the timed interrupt period.	0

21.5.10 SPIn Timer Status register (SPIn_TIM_STAT - 0x2008 8408, 0x2009 0408)

The SPIn_TIM_STAT register allows checking the status of the time-out timer and distinguishing a time-out interrupt from SPI peripheral interrupts.

Table 486. SPIn Timer Status register (SPIn_TIM_STAT - 0x2008 8408, 0x2009 0408)

SPIn_TIM_STAT	Function	Description	Reset value
15	tirqstat	Timed interrupt status flag. Write 1 to this bit to clear the flag. 0: no timed interrupt pending 1: timed interrupt pending	
14:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

21.6 Timed interrupt and DMA time-out modes

The interrupt generator of the SPI handles several interrupt sources like the FIFO full interrupt bf, the FIFO empty interrupt be, and the FIFO threshold interrupt thr. In addition to these interrupt sources, the SPI may also generate a timed interrupt based on counts occurring at the frequency of SPIn_CLK.

21.6.1 Timed interrupt mode

The timed interrupt mode can generate an interrupt if data remains in the FIFO for a predetermined time.

- When SPIn_TIM_COUNT is written, the time out counter starts from 0.
- The time out counter re-starts from 0 when the value written in SPIn_TIM_COUNT is reached, generating on output pulse.
- If the FIFO depth is higher than 0 during the rising edge of the counter output pulse, a timed interrupt is generated.
- When the software clears the tirqstat bit in SPIn_TIM_STAT, the SPI interrupt is cleared regardless of counter value.

21.6.2 DMA time-out mode

The DMA time out mode can generate an interrupt if the SPI DMA request is not serviced within a predetermined time.

- When SPIn_TIM_COUNT is written, the time out counter is enabled, but does not start incrementing.
- The time out counter is started by the first DMA burst request from the SPI block (the signal SPIn_BREQ).
- Every rising edge of SPIn_BREQ clears the time out counter.
- When the time out counter reaches the value in SPIn_TIM_COUNT, the timed interrupt is set, and the time out counter is disabled.
- When software clears the tirqstat bit, the SPI interrupt output is cleared regardless of the level of SPIn_BREQ.

21.7 Rate calculation

The bit transfer rate of the SPI is defined by the formula:
SPIn_CLK = HCLK / ((rate + 1) $\times$ 2).

The maximum bit rate is 52 MHz. Table 487 gives some examples of SPI rates.

Table 487. Examples of SPI bit rates

Rate selection SPIn_CON[6:0]	Clock division	SPI bit rate for HCLK = 104 MHz (Mbit/s)
0000000	2	52
0000001	4	26
0000010	6	17.33
...	...	...
0011001	52	2
...	...	...
1111111	256	0.406

22.1 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.

LPC32x0 has two Synchronous Serial Port controllers -- SSP0 and SSP1.

22.2 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 bits frame.
- DMA transfers supported by GPDMA.

22.3 Pin descriptions

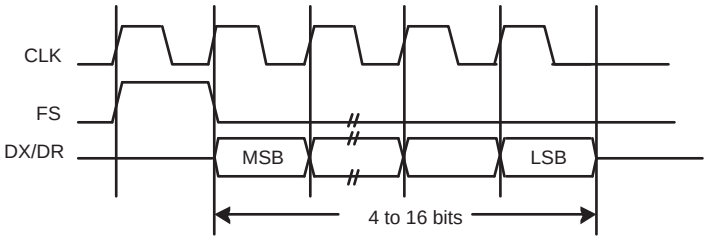
Table 488. SSP pin descriptions

Pin Name	Type	Interface pin name/function			Pin Description
		SPI	SSI	Microwire	
SCK0/1	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 SPI interface is used the clock is programmable to be active high or active low, otherwise it is always active high. SCK1 only switches during a data transfer. Any other time, the SSPn either holds it in its inactive state, or does not drive it (leaves it in high impedance state).
SSEL0/1	I/O	SSEL	FS	CS	**Frame Sync/Slave Select.** When the SSPn is a bus master, it drives this signal from shortly before the start of serial data, to shortly after the end of serial data, to signify a data transfer as appropriate for 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	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	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.

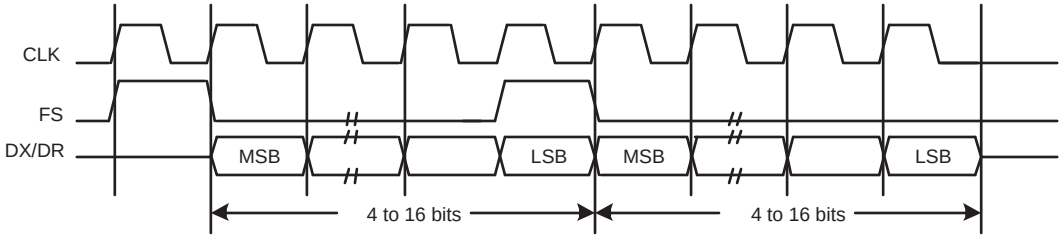
22.4 Bus description

22.4.1 Texas Instruments synchronous serial frame format

Figure 80 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 80. 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 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.

22.4.2 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.

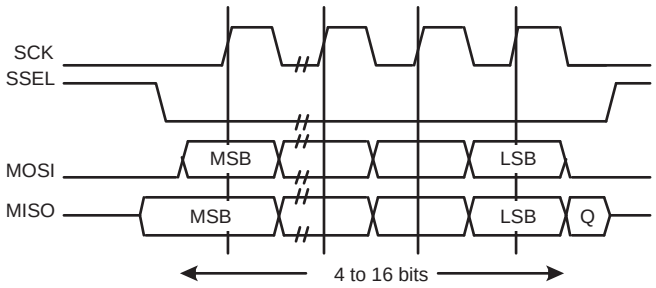
22.4.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

When the CPOL clock polarity control bit is LOW, it produces a steady state low value on the SCK pin. If the CPOL clock polarity control bit is HIGH, a steady state high value is placed on the CLK pin when data is not being transferred.

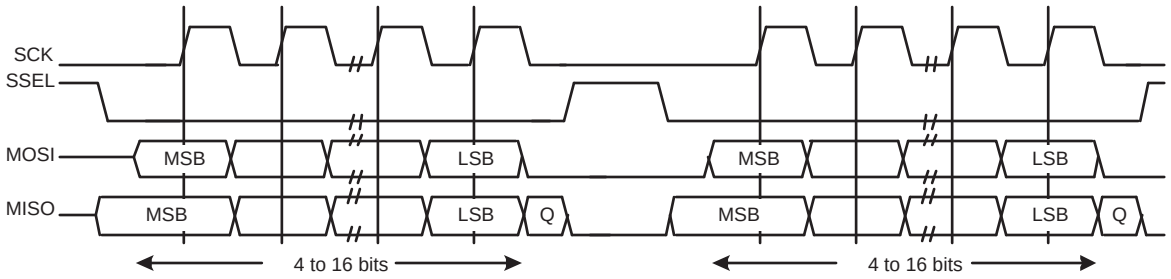
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is LOW, data is captured on the first clock edge transition. If the CPHA clock phase control bit is HIGH, data is captured on the second clock edge transition.

22.4.2.2 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 81.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 81. 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.

22.4.2.3 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 82, which covers both single and continuous transfers.

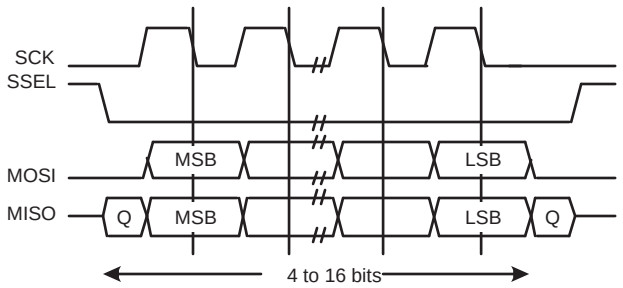


Fig 82. SPI frame format with CPOL=0 and CPHA=1

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.

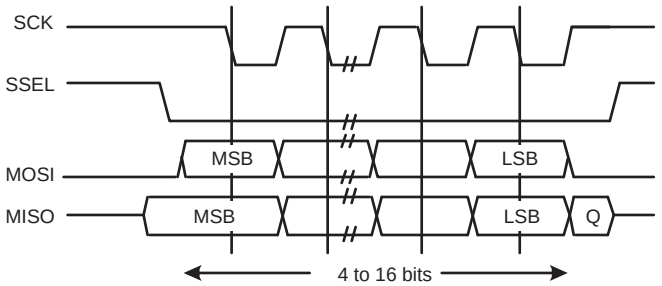
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

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.

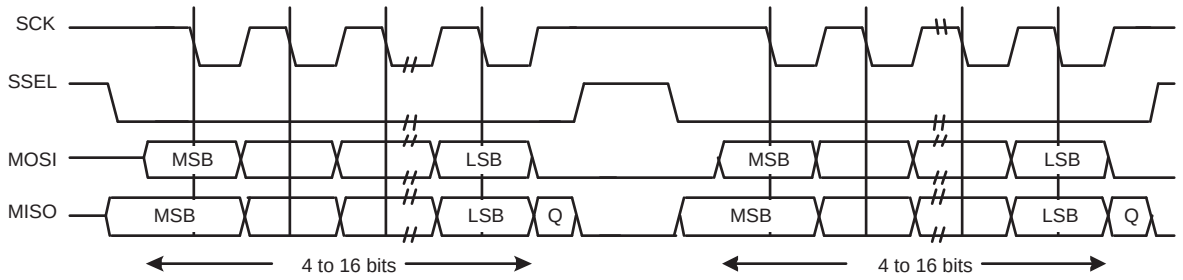
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

22.4.2.4 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 83.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 83. 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.

22.4.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 84, which covers both single and continuous transfers.

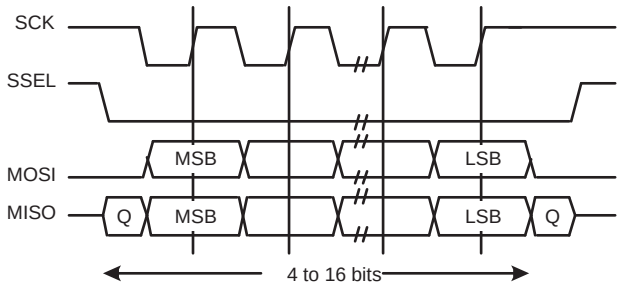


Fig 84. SPI Frame Format with CPOL = 1 and CPHA = 1

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.

22.4.3 Semiconductor Microwire frame format

Figure 85 shows the Microwire frame format for a single frame. Figure 86 shows the same format when back-to-back frames are transmitted.

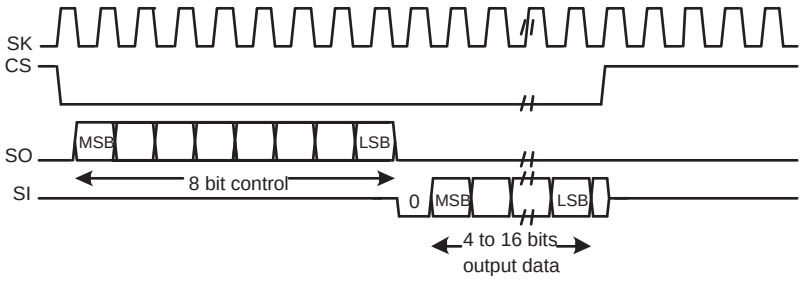


Fig 85. Microwire frame format (single transfer)

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.

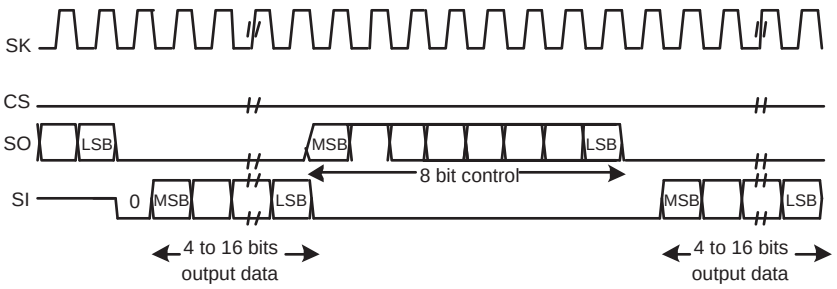


Fig 86. Microwire frame format (continuous transfers)

22.4.3.1 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 87 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.

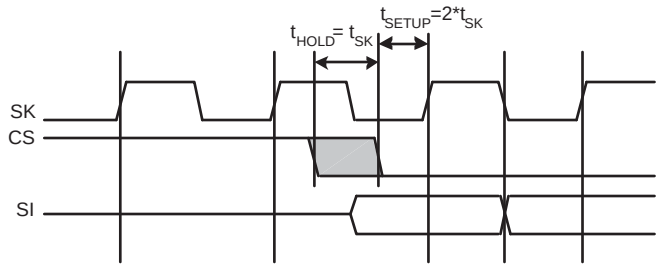


Fig 87. Microwire frame format setup and hold details

22.5 Register description

The register offsets from the SSP controller base addresses are shown in the Table 489.

Table 489. SSP Register Map

Generic Name	Description	Access	Reset Value[1]	SSPn Register Name & Address
SSP_CTRL	SSP0 and SSP1 clock control	R/W	0	SSP_CTRL - 0x4000 4078
CR0	Control Register 0. Selects the serial clock rate, bus type, and data size.	R/W	0	SSP0CR0 - 0x2008 4C00 SSP1CR0 - 0x2008 C000
CR1	Control Register 1. Selects master/slave and other modes.	R/W	0	SSP0CR1 - 0x2008 4004 SSP1CR1 - 0x2008 C004

Table 489. SSP Register Map

Generic Name	Description	Access	Reset Value[1]	SSPn Register Name & Address
DR	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	R/W	0	SSP0DR - 0x2008 4008 SSP1DR - 0x2008 C008
SR	Status Register	RO		SSP0SR - 0x2008 400C SSP1SR - 0x2008 C00C
CPSR	Clock Prescale Register	R/W	0	SSP0CPSR - 0x2008 4010 SSP1CPSR - 0x2008 C010
IMSC	Interrupt Mask Set and Clear Register	R/W	0	SSP0IMSC - 0x2008 4014 SSP1IMSC - 0x2008 C014
RIS	Raw Interrupt Status Register	R/W		SSP0RIS - 0x2008 4018 SSP1RIS - 0x2008 C018
MIS	Masked Interrupt Status Register	R/W	0	SSP0MIS - 0x2008 401C SSP1MIS - 0x2008 C01C
ICR	SSPICR Interrupt Clear Register	R/W	NA	SSP0ICR - 0x2008 4020 SSP1ICR - 0x2008 C020
DMACR	DMA Control Register	R/W	0	SSP0DMACR - 0x2008 4024 SSP1DMACR - 0x2008 C024

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

22.5.1 SSP Control register (SSP_CTRL - 0x4000 4078)

The SSP_CTRL register controls some aspects of the two SSP interfaces: enabling and disabling clocks and switching between the interface to the GPDMA Controller

Table 490. SSP Control register (SSP_CTRL - 0x4000 4078)

Bit	Function	Reset value
31:6	Reserved	0
5	SSP1 RX DMA connection control 0 = SSP1 RX is not connected to DMA and SPI2 is connected to DMA. 1 = SSP1 RX is connected to DMA and SPI2 is not connected to DMA	0
4	SSP1 TX DMA connection control 0 = SSP1 TX is not connected to DMA and SPI1 is connected to DMA. 1 = SSP1 TX is connected to DMA and SPI1 is not connected to DMA.	0
3	SSP0 RX DMA connection control 0 = SSP0 RX is not connected to DMA. 1 = SSP0 RX is connected to DMA.	0
2	SSP0 TX DMA connection control 0 = SSP0 TX is not connected to DMA. 1 = SSP0 TX is connected to DMA.	0
1	SSP1 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
0	SSP0 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

22.5.2 SSPn Control Register 0 (SSP0CR0 - 0x2008 4000, SSP1CR0 - 0x2008 C000)

This register controls the basic operation of the SSP controller.

Table 491. SSPn Control Register 0 (SSP0CR0 - address 0x2008 4000, SSP1CR0 - 0x2008 C000) 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 CPSDVR is the prescale divider, and the AHB clock HCLK clocks the prescaler, the bit frequency is HCLK / (CPSDVSR × [SCR+1]).	0x00

22.5.3 SSPn Control Register 1 (SSP0CR1 - 0x2008 4004, SSP1CR1 - 0x2008 C004)

This register controls certain aspects of the operation of the SSP controller.

Table 492. SSPn Control Register 1 (SSP0CR1 - address 0x2008 4004, SSP1CR1 - 0x2008 C004) 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
7:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

22.5.4 SSPn Data Register (SSP0DR - 0x2008 4008, SSP1DR - 0x2008 C008)

Software can write data to be transmitted to this register, and read data that has been received.

Table 493. SSPn Data Register (SSP0DR - address 0x2008 4008, SSP1DR - 0x2008 C008) 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.	0x0000

22.5.5 SSPn Status Register (SSP0SR - 0x2008 400C, SSP1SR - 0x2008 C00C)

This read-only register reflects the current status of the SSP controller.

Table 494. SSPn Status Register (SSP0SR - address 0x2008 400C, SSP1SR - 0x2008 C00C) 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
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

22.5.6 SSPn Clock Prescale Register (SSP0CPSR - 0x2008 4010, SSP1CPSR - 0x2008 C010)

This register controls the factor by which the Prescaler divides the AHB clock HCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in SSPnCR0, to determine the bit clock.

Table 495. SSPn Clock Prescale Register (SSP0CPSR - address 0x2008 4010, SSP1CPSR - 0x2008 C010) bit description

Bit	Symbol	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which HCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0

Important: the SSPnCPSR value must be properly initialized or the SSP controller will not be able to transmit data correctly. In case of an SSP operating in the master mode, the $CPSDVSR_{min}$ = 2, while in case of the slave mode $CPSDVSR_{min}$ = 12.

22.5.7 SSPn Interrupt Mask Set/Clear Register (SSP0IMSC - 0x2008 4014, SSP1IMSC - 0x2008 C014)

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 496. SSPn Interrupt Mask Set/Clear register (SSP0IMSC - address 0x2008 4014, SSP1IMSC - 0x2008 C014) 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 no has not been read for a "timeout period".	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
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

22.5.8 SSPn Raw Interrupt Status Register (SSP0RIS - 0x2008 4018, SSP1RIS - 0x2008 C018)

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 497. SSPn Raw Interrupt Status register (SSP0RIS - address 0x2008 4018, SSP1RIS - 0x2008 C018) 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 "timeout period".	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
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

22.5.9 SSPn Masked Interrupt Status Register (SSP0MIS - 0x2008 401C, SSP1MIS - 0x2008 C01C)

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 498. SSPn Masked Interrupt Status register (SSPnMIS -address 0x2008 401C, SSP1MIS - 0x2008 C01C) 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 "timeout period", 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
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

22.5.10 SSPn Interrupt Clear Register (SSP0ICR - 0x2008 4020, SSP1ICR - 0x2008 C020)

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 499. SSPn interrupt Clear Register (SSP0ICR - address 0x2008 4020, SSP1ICR - 0x2008 C020) 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 timeout period" interrupt.	NA
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

22.5.11 SSPn DMA Control Register (SSP0DMACR - 0x2008 4024, SSP1DMACR - 0x2008 C024)

The SSPnDMACR register is the DMA control register.It is a read/write register. Table 500 shows the bit assignments of the SSPnDMACR register.

Table 500. SSPn DMA Control Register (SSP0DMACR - address 0x2008 4024, SSP1DMACR - 0x2008 C024) bit description

Bit	Symbol	Description	Reset Value
0	Receive DMA Enable (RXDMAE)	When this bit is set to one 1, DMA for the receive FIFO is enabled, otherwise receive DMA is disabled.	0
1	Transmit DMA Enable (TXDMAE)	When this bit is set to one 1, DMA for the transmit FIFO is enabled, otherwise transmit DMA is disabled	0
15:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

23.1 Features

- The two I^2C blocks are standard I^2C compliant bus interfaces that may be used in Master, Multi-master and slave modes.
- Programmable clock to allow adjustment of I^2C transfer rates.
- Bidirectional data transfer.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.

23.2 Applications

Interfaces to external I^2C standard parts, such as serial RAMs, LCDs, tone generators, diagnostic ports, etc.

23.3 Description

There are two I^2C interfaces in the UM10326 of controllers. These I^2C blocks can be configured as a master, multimaster or slave supporting clock rates up to 400 kHz. The I^2C blocks also support 7 or 10 bit addressing. Each has a four word FIFO for both transmit and receive. An interrupt signal is available from each block.

There is a separate slave transmit FIFO. The slave transmit FIFO (TXS) and its level are only available when the controller is configured as a Master/Slave device and is operating in a multi-master environment. Separate TX FIFOs are needed in a multi-master because a controller might have a message queued for transmission when an external master addresses it to be come a slave-transmitter, a second source of data is needed.

Note that the I^2C clock must be enabled in the I2CCLK_CTRL register before using the I^2C. The I^2C clock can be disabled between communications, if used as a single master I^2C interface, software has full control of when I^2C communication is taking place on the bus.

A typical I^2C-bus configuration is shown in Figure 89. Depending on the state of the direction bit (r/$\overline{w}$), 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.

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 I2C-bus will not be released.

Each of the I2C interfaces on the LPC32x0 contains a four byte FIFO, allowing more data to be transferred before additional software attention is needed.

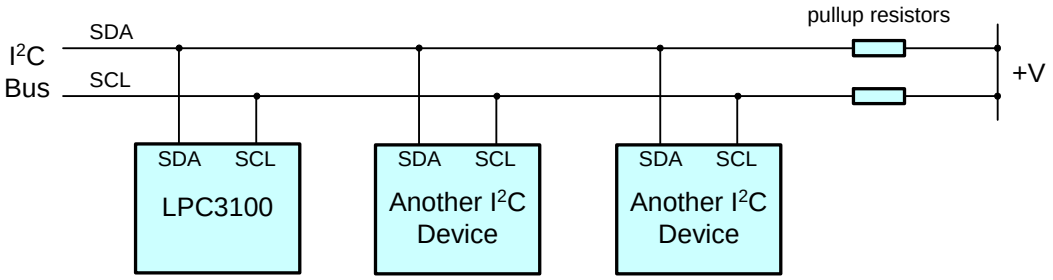


Fig 88. I2C-bus configuration

23.4 Pin description

The I2C1 interface pins are powered by the VDD_IOB domain.

The I2C2 interface pins are powered by the VDD_IOC domain.

Table 501. I2C-bus pin description

Pin name	Type	Description
I2C1_SDA, I2C2_SDA	Input/Output	I2C Serial Data.
I2C1_SCL, I2C2_SCL	Input/Output	I2C Serial Clock.

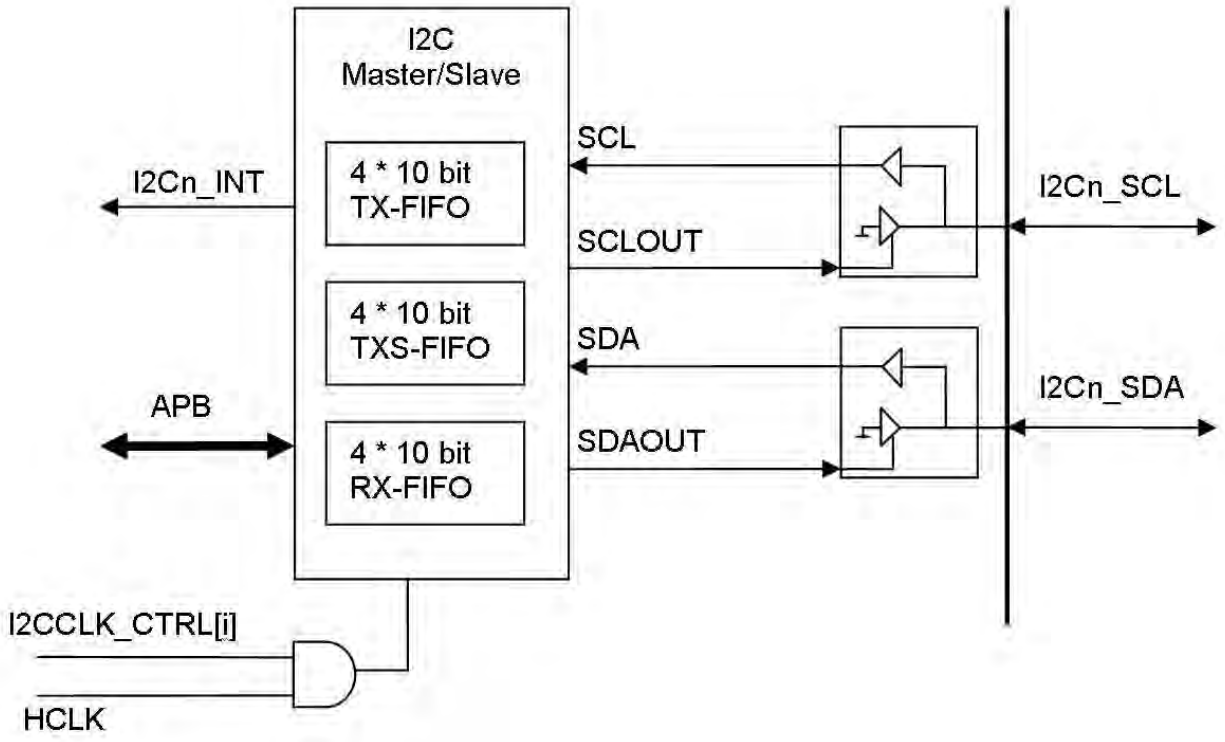


Fig 89. I²C-bus configuration

23.5 Register description

The two I²C blocks are located at the base addresses shown in Table 502.

Table 502. Standard I2C base addresses

I²C block	Base address
1	0x400A 0000
2	0x400A 8000

Each I²C interface contains the registers shown in Table 503. More detailed descriptions follow.

Table 503. I²C registers

Address offset	Name	Description	Access
0x00	I2Cn_RX	I2Cn RX Data FIFO	RO
0x00	I2Cn_TX	I2Cn TX Data FIFO	WO
0x04	I2Cn_STAT	I2Cn Status Register	RO
0x08	I2Cn_CTRL	I2Cn Control Register	R/W
0x0C	I2Cn_CLK_HI	I2Cn Clock Divider high	R/W
0x10	I2Cn_CLK_LO	I2Cn Clock Divider low	R/W
0x14	I2Cn_ADR	I2Cn Slave Address	R/W

Table 503. I²C registers

Address offset	Name	Description	Access
0x18	I2Cn_RXFL	I2Cn Rx FIFO level	RO
0x1C	I2Cn_TXFL	I2Cn Tx FIFO level	RO
0x20	I2Cn_RXB	I2Cn Number of bytes received	RO
0x24	I2Cn_TXB	I2Cn Number of bytes transmitted	RO
0x28	I2Cn_S_TX	Slave Transmit FIFO	WO
0x2C	I2Cn_S_TXFL	Slave Transmit FIFO level	RO

23.5.1 I2Cn RX Data FIFO (I2Cn_RX - 0x400A 0000, 0x400A 8000)

The RX FIFO may be cleared via a soft reset, by setting bit 8 in the I2Cn_CTRL register.

When operating as a master-receiver, the TX_FIFO must be written for proper operation. When operating as a master-receiver the controller ignores bits [7:0] in the TX_FIFO register. The master-receiver must write a (dummy) byte to the TX_FIFO for each byte it expects to receive in the RX_FIFO. The first dummy byte triggers the clock sequence to transfer data from the slave transmitter. Each dummy byte that follows has a dual purpose, it acknowledges reception of the previous byte and triggers the clock sequence to transfer the next byte form the slaver transmitter. If the master-receiver sets the STOP_bit (9) (to signal the end of a receive) or sets the START_bit (8) (to cause a RESTART condition), then the last byte read from the slave is not acknowledged. That is, the last byte of a master-receiver does not acknowledge by writing a dummy value to the TX_FIFO register.

If the RX FIFO is read while empty, a DATA ABORT exception is generated.

Table 504. I2Cn RX Data FIFO (I2Cn_RX - 0x400A 0000, 0x400A 8000)

I2Cn_RX	Function	Description	Reset value
7:0	RxData	Receive FIFO data bits 7:0	N/A

23.5.2 I2Cn TX Data FIFO (I2Cn_TX - 0x400A 0000, 0x400A 8000)

The TX FIFO may be cleared via a soft reset, by setting bit 8 in the I2Cn_CTRL register.

If the controller is configured as a Master/Slave and is operating in a multi-master environment, then only master-transmit data should be written to I2Cn_TX, slave transmit data should be written to I2Cn_S_TX.

If the TX FIFO is written to while full a DATA ABORT exception is generated.

Table 505. I2Cn TX Data FIFO (I2Cn_TX - 0x400A 0000, 0x400A 8000)

I2Cn_TX	Function	Description	Reset value
9	STOP	0 = Do not issue a STOP condition after transmitting this byte 1 = Issue a STOP condition after transmitting this byte.	NA
8	START	0 = Do not Issue a START condition before transmitting this byte 1 = Issue a START condition before transmitting this byte.	NA
7:0	TxData	Transmit FIFO data bits 7:0	NA

23.5.3 I2Cn Status register (I2Cn_STAT - 0x400A 0004, 0x400A 8004)

The status is a read-only register that provides status information on the TX and RX blocks as well as the current state of the external buses. A soft reset will clear the status register with the exception of the TFE and RFE bits, which will be set, and the SCL and SDA bits, which continue to reflect the state of the bus pins.

Table 506. I2Cn Status register (I2Cn_STAT - 0x400A 0004, 0x400A 8004)

I2Cn_STAT	Function	Description	Reset value
13	TFES	Slave Transmit FIFO Empty. Slave TFE is set when the slave TX FIFO is empty and is cleared when the slave TX FIFO contains valid data. 0 = TX FIFO is not empty. 1 = TX FIFO is empty	1
12	TFFS	Slave Transmit FIFO Full Slave TFF is set when the slave TX FIFO is full and is cleared when the slave TX FIFO is not full. 0 = Slave TX FIFO is not full. 1 = SlaveTX FIFO is full	0
11	TFE	Transmit FIFO Empty. TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data. 0 = TX FIFO contains valid data. 1 = TX FIFO is empty	1
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 = TX FIFO is not full. 1 = TX FIFO is full	0
9	RFE	Receive FIFO Empty. RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data. 0 = RX FIFO contains data. 1 = RX FIFO is empty	1
8	RFF	Receive FIFO Full. 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 ARM reads the RX FIFO and makes room for it. 0 = RX FIFO is not full 1 = RX FIFO is full	0
7	SDA	The current value of the SDA signal.	NA
6	SCL	The current value of the SCL signal.	NA
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

Table 506. I2Cn Status register (I2Cn_STAT - 0x400A 0004, 0x400A 8004) *…continued*

I2Cn_STAT	Function	Description	Reset value
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 ARM core writes another byte to transmit. This bit is cleared when a byte is written to the slave Tx FIFO. 0 = Slave transmitter does not need data. 1 = Slave transmitter needs data.	NA
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 ARM core writes another byte to transmit. This bit is cleared when a byte is written to the master TX FIFO. 0 = Master transmitter does not need data. 1 = Master transmitter needs data.	0
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 = Last transmission received an acknowledge. 1 = Last transmission did not receive an acknowledge.	0
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 '1' to bit 1 of the status register. 0 = No arbitration failure on last transmission. 1 = Arbitration failure occurred on last transmission.	NA
0	TDI	Transaction Done Interrupt. This flag is set if a transaction completes successfully. It is cleared by writing a '1' to bit 0 of the status register. It is unaffected by slave transactions. 0 = Transaction has not completed. 1 = Transaction completed.	0

23.5.4 I2Cn Control Register (I2Cn_CTRL - 0x400A 0008, 0x400A 8008)

The CTL register is used to enable interrupts and reset the I^2C state machine.

Table 507. I2Cn Control Register (I2Cn_CTRL - 0x400A 0008, 0x400A 8008)

I2Cn_CTRL	Function	Description	Reset value
10	TFFSIE	Slave Transmit FIFO Not Full Interrupt Enable. This enables the Slave Transmit FIFO Not Full interrupt to indicate that the more data can be written to the slave transmit FIFO. Note that this is *not* full. 0 = Disable the TFFSI. 1 = Enable the TFFSI.	0
9	SEVEN	Seven-bit slave address. This bit is selects 7-bit or 10-bit slave address operation. 0 = Use 7-bit slave addressing, bits [6:0] in address register. 1 = Use 10-bit slave address. Note: Performing a Soft Reset clears this bit to 0. Use 7-bit slave addressing.	0
8	RESET	Soft Reset. On a soft reset, the TX and RX FIFOs are flushed, STS register is cleared, and all internal state machines are reset to appear idle, and clears bit 9 forcing 7-bit slave addressing. The I2Cn_CLK_LO, I2Cn_CLK_HI, I2Cn_CTRL, and I2Cn_ADR registers are NOT modified by a soft reset. 0 = No effect 1 = Reset the I^2C to idle state. Self clearing.	0
7	TFFIE	Transmit FIFO Not Full Interrupt Enable. This enables the Transmit FIFO Not Full interrupt to indicate that more data can be written to the transmit FIFO. Note that this is not full. It is intended help the ARM Write to the I^2C block only when there is room in the FIFO to accept it and do this without polling the status register. 0 = Disable the TFFI. 1 = Enable the TFFI.	0
6	RFDAIE	Receive FIFO Data Available Interrupt Enable. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty). 0 = Disable the DAI. 1 = Enable the DAI.	0
5	RFFIE	Receive FIFO Full Interrupt Enable. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data. 0 = Disable the RFFI. 1 = Enable the RFFI.	0
4	DRSIE	Data Request Slave Transmitter 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 = Disable the DRSI interrupt. 1 = Enable the DRSI interrupt.	NA
3	DRMIE	Data Request Master Transmitter 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 = Disable the DRMI interrupt. 1 = Enable the DRMI interrupt.	0

Table 507. I2Cn Control Register (I2Cn_CTRL - 0x400A 0008, 0x400A 8008) *...continued*

I2Cn_CTRL	Function	Description	Reset value
2	NAIE	Transmitter No Acknowledge Interrupt Enable. This enables the NAI interrupt signalling that transmitted byte was not acknowledged. 0 = Disable the NAI. 1 = Enable the NAI.	0
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 = Disable the AFI. 1 = Enable the AFI.	NA
0	TDIE	Transmit Done Interrupt Enable. This enables the TDI interrupt signalling that this I^2C issued a stop condition. 0 = Disable the TDI interrupt. 1 = Enable the TDI interrupt.	0

23.5.5 I2Cn Clock Divider High (I2Cn_CLK_HI - 0x400A 000C, 0x400A 800C)

The I^2Cn_CLK_HI register holds a terminal count for counting HCLK cycles to create the high period of the slower I^2C serial clock, SCL. When reset, the clock divider will be set to run at its reset value.

Table 508. I2Cn Clock Divider High (I2Cn_CLK_HI - 0x400A 000C, 0x400A 800C)

I2Cn_CLK_HI	Function	Description	Reset value
9:0	CLK_DIV_HI	Clock Divisor High. This value sets the number of HCLK cycles SCL will be high. $F_{SCL}=F_{HCLK}/(CLK\_DIV\_HI + CLK\_DIV\_LO)$ This means that in order to get F_{SCL} =100 kHz set CLK_DIV_HIGH = CLK_DIV_LOW = 520. (F_{HCLK} =104 MHz). The lowest operating frequency is about 50 kHz.	0x2BA

23.5.6 I2Cn Clock Divider Low (I2Cn_CLK_LO - 0x400A 0010, 0x400A 8010)

The I2Cn_CLK_LO register holds a terminal count for counting HCLK cycles to create the low period of the slower I^2C serial clock, SCL. When reset, the clock divider will be set to run at its reset value frequency.

Table 509. I2Cn Clock Divider Low (I2Cn_CLK_LO - 0x400A 0010, 0x400A 8010)

I2cN_CLK_LO	Function	Description	Reset value
9:0	CLK_DIV_LO	Clock Divisor Low. This value sets the number of HCLK cycles SCL will be low. FSCL=FHCLK/(CLK_DIV_HI + CLK_DIV_LO) This means that in order to get FSCL =100 kHz set CLK_DIV_HIGH = CLK_DIV_LOW = 520. (FHCLK =104 MHz). The lowest operating frequency is about 50 kHz.	0x2BA

23.5.7 I2Cn Slave Address (I2Cn_ADR - 0x400A 0014, 0x400A 8014)

The I2Cn_ADR register holds the I2C bus slave address.

Table 510. I2Cn Slave Address (I2Cn_ADR - 0x400A 0014, 0x400A 8014)

I2Cn_ADR	Function	Description	Reset value
9:0	ADR	ADR is the I2C bus slave address. Bits 6:0 are enabled if the I2C is configured for slave mode. Bits 9:7 are enabled only if it is configured for slave mode and '10-bit addressing'. Note: A soft-reset disables Bits 9:7, see the description of Bits 8 and 9 in the I2Cn_CTRL registers.	0x06E

23.5.8 I2Cn Receive FIFO level (I2Cn_RXFL - 0x400A 0018, 0x400A 8018)

The I2Cn_RXFL is a read only register that contains the number of bytes in the RX FIFO.

Table 511. I2Cn RX FIFO level (I2Cn_RXFL - 0x400A 0018, 0x400A 8018)

I2Cn_RXFL	Function	Description	Reset value
1:0	RxFL	Receive FIFO level	0

23.5.9 I2Cn Transmit FIFO level (I2Cn_TXFL - 0x400A 001C, 0x400A 801C)

The I2Cn_TXFL is a read only register that contains the number of bytes in the TX FIFO.

Table 512. I2Cn TX FIFO level (I2Cn_TXFL - 0x400A 001C, 0x400A 801C)

I2Cn_TXFL	Function	Description	Reset value
1:0	TxFL	Transmit FIFO level	0

23.5.10 I2Cn Receive byte count (I2Cn_RXB - 0x400A 0020, 0x400A 8020)

The I2Cn_RXB contains the number of bytes received. The register is reset when I2C transitions from inactive to active.

Table 513. I2Cn RX byte count (I2Cn_RXB - 0x400A 0020, 0x400A 8020)

I2Cn_RXB	Function	Description	Reset value
15:0	RxB	Number of bytes received	N/A

23.5.11 I2Cn Transmit Byte count (I2Cn_TXB - 0x400A 0024, 0x400A 8024)

The I2Cn_TXB contains the number of bytes transmitted. The register is reset when I2C transitions from inactive to active.

Table 514. I2Cn TX byte count (I2Cn_TXB - 0x400A 0024, 0x400A 8024)

I2Cn_TXB	Function	Description	Reset value
15:0	TxB	Number of bytes sent	N/A

23.5.12 I2Cn Slave Transmit FIFO (I2Cn_S_TX - 0x400A 0028, 0x400A 8028)

The I2Cn_S_TX FIFO may be cleared via a soft reset, by setting bit 8 in the I2Cn_CTRL register.

If the controller is configured as a Master/Slave and is operating in a multi-master environment, then only master-transmit data should be written to I2Cn_TX, slave transmit data should be written to I2Cn_S_TX.

If the TX FIFO is written to while full a DATA ABORT exception is generated.

Table 515. I2Cn Slave TX Data FIFO (I2Cn_S_TX - 0x400A 0028, 0x400A 8028)

I2Cn_S_TX	Function	Description	Reset value
7:0	TXS	Slave Transmit FIFO data bits 7:0	NA

23.5.13 I2Cn Slave Transmit FIFO level (I2Cn_S_TXFL - 0x400A 002C, 0x400A 802C)

The I2Cn_S_TXFL is a read only register that contains the number of bytes in the Slave TX FIFO.

Table 516. I2Cn Slave TX FIFO level (I2Cn_S_TXFL - 0x400A 002C, 0x400A 802C)

I2Cn_S_TXFL	Function	Description	Reset value
1:0	TxFL	Slave Transmit FIFO level	0

23.6 I²C clock settings

Table 517 shows some examples of clock settings for various HCLK and I²C frequencies.

Table 517. Example I²C rate settings

HCLK frequency (MHz)	I²C clock rate (kHz)	I2Cn_CLK_HI + I2Cn_CLK_LO	I2Cn_CLK_HI	I2Cn_CLK_LO	Comment
104	100	1040	520	520	Symmetric clock (standard for 100 kHz I²C)
52	100	520	260	260	Symmetric clock (standard for 100 kHz I²C)
13	100	130	65	65	Symmetric clock (standard for 100 kHz I²C)
104	400	260	94	166	Asymmetric clock (per 400 kHz I²C spec)
52	400	130	47	83	Asymmetric clock (per 400 kHz I²C spec)
13	400	33 (rounded up)	12	21	Asymmetric clock (per 400 kHz I²C spec). Actual rate will be 393.9 kHz.

23.7 I²C Tutorial

To learn the details of I2C, one should read Phillips' specification. This is only a simple tutorial to learn the basics of I2C

23.7.1 Overview

I2C is a two-wire interface made up of a clock (SCL) and data (SDA). Each of these two wires has a pull-up (or current source). Any device on the bus can pull it to a logic zero or let it float to a logic one (Wire-ANDed). A bus master drives the clock line and is responsible for driving an address on the data wire. A bus slave is any device being addressed by a master. A bus transmitter writes data to the bus by driving the data line. A bus receiver reads data from the bus. A bus master can be either a transmitter or a receiver; likewise for a bus slave.

23.7.2 I²C Data

When sending data, the data line, SDA, must change when the clock line, SCL, is low. Data is sampled at the rising edge of SCL. Figure 90 shows the relative timing of SDA to SCL

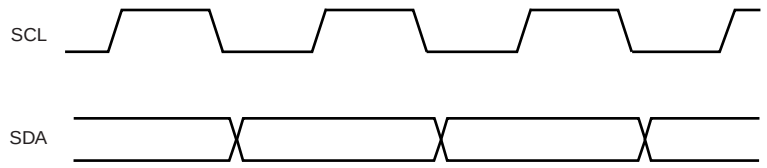


Fig 90. I²C Data

23.7.3 Start Condition

A start condition is signaled by a high-to-low transition of SDA while SCL is high. Figure 91 illustrates a start condition. A bus master can issue a start when the bus is idle or if it already has control of the bus.

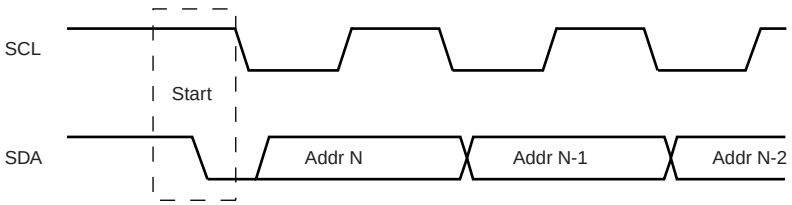


Fig 91. I²C Start Condition

23.7.4 Stop Condition

A stop condition is signaled by a low-to-high transition of SDA while SCL is high. Figure 92 illustrates a stop condition.

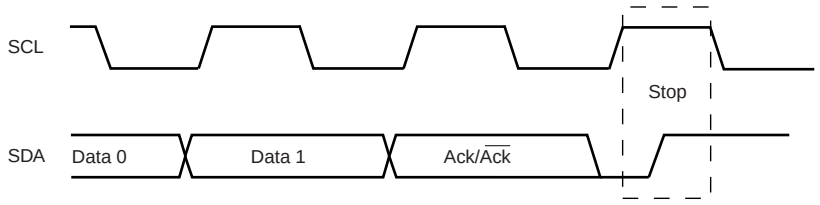


Fig 92. I²C Stop Condition

23.7.5 Acknowledge

After every byte is transferred the receiver must acknowledge receipt of the byte through an acknowledge bit. This is done by pulling SDA low for one cycle. The acknowledge is just like a data bit in that SDA must be changed when SCL is low and the acknowledge must be stable for the rising edge of SCL. If a bus master doesn't receive an acknowledge after sending a byte, the master issues a stop condition and the transfer is aborted.

When operating as a master-receiver, the slave-transmitter must release the bus at the end of the transfer so the master can generate a stop condition. To force the slave-transmitter to release, the master receiver does not acknowledge the last byte received. See Section 23.5.1 for additional information.

23.7.6 I²C Addresses

I²C devices can use 7-bit addressing or 10-bit addressing. The bus master must send out an address after issuing a start condition. The address is sent in MSB to LSB order followed by a read (not write) bit which controls the direction of the following transfer. An example of a 7-bit address is shown in Figure 93 where S is a start condition, Ax is bit x of the address, r/$\overline{w}$ is the read(not write) bit, Ack is the acknowledge, and P is a stop condition.

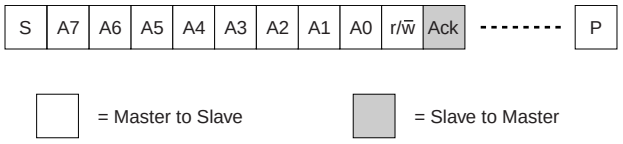


Fig 93. I²C 7-bit device addressing

10-bit addressing is accomplished by sending the address in two bytes. The First byte is 11110xx followed by the r/$\overline{w}$ bit where xx are the two MSBs of the address. The second byte is the eight LSBs of the address. Figure 94 shows 10-bit addressing. Slaves matching the first byte of the address must acknowledge. Slaves matching the second address byte also acknowledge that byte. Writing to a 10-bit address issues the two address bytes followed by the data.

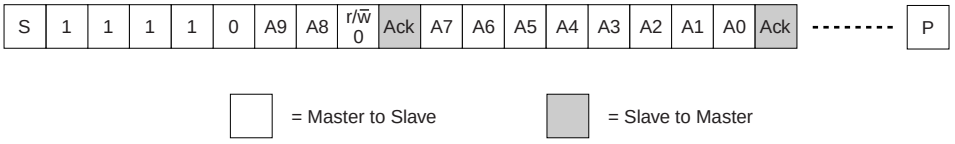


Fig 94. I²C 10-bit device addressing

23.7.7 I²C Write

To write data on the I²C bus using 7-bit addressing, the master sends out a slave address, a write bit, receives an acknowledge, then sends data bytes in MSB to LSB order receiving an acknowledge after each byte, and finally issues a stop. Figure 95 illustrates a write operation.

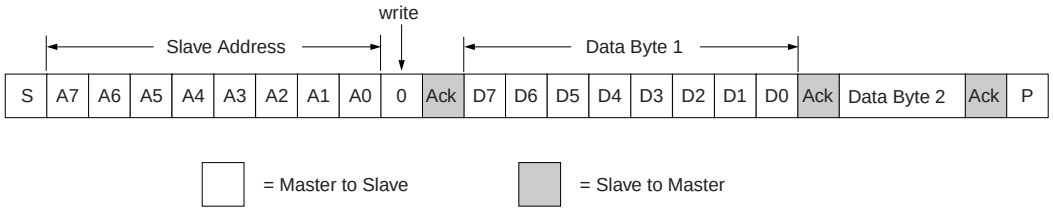


Fig 95. I²C 7-bit slave address write operation

To write data on the I²C bus using 10-bit addressing, the master sends out a the high slave address, a write bit, receives an acknowledge, then sends the low slave address, receives an acknowledge, then sends data bytes in MSB to LSB order receiving an acknowledge after each byte, and finally issues a stop. Figure 96 illustrates a 10-bit slave address write operation.

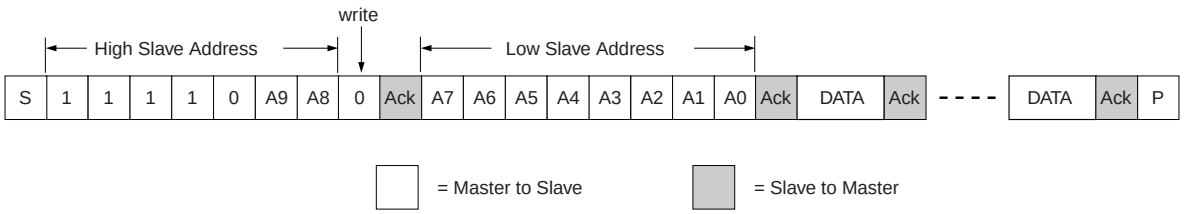


Fig 96. I²C 10-bit slave address write operation

23.7.8 I²C Read

To read data on the I²C bus, the master send out a slave address, a read bit, receives an acknowledge, then receives data bytes in MSB to LSB order sending an acknowledge after each byte, and finally issues a stop. Figure 97 illustrates a read operation. The master drives SCL for the entire operation.

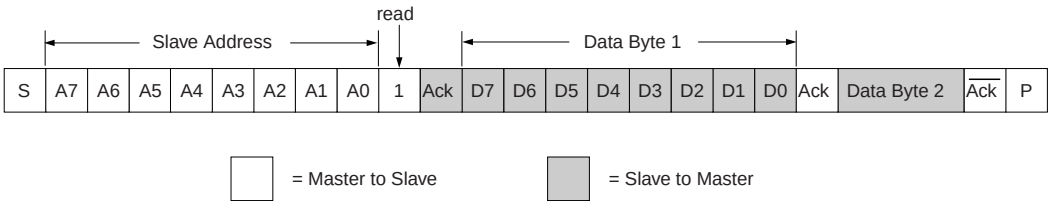


Fig 97. I²C Read Operation from a 7-bit slave address device

Reading from a 10-bit address is somewhat different; The master must write to the slave before restarting and reading from the same slave. The master must send the first byte with the r/w bit low (write) then second address byte, followed by a restart, then send only the first address byte with the r/w bit high (read), then read bytes from the slave. The last byte read from the slave is not acknowledged to signal the end of the read operation.

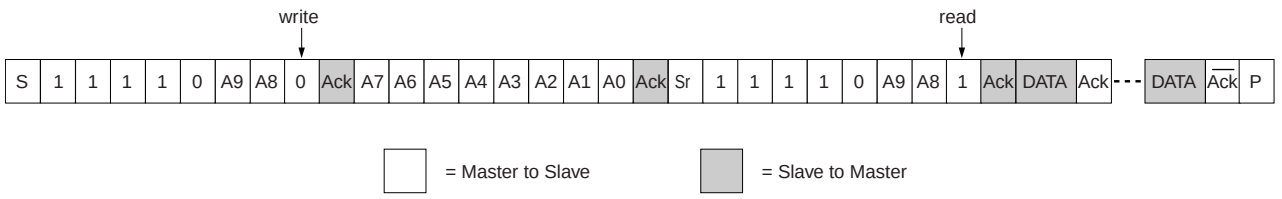


Fig 98. I²C Reading from a 10-bit slave address device

23.7.9 I²C Write/Read

A repeated start condition allows the bus master to reissue an address with the option of changing the r/w bit. The master can issue any number of start conditions before issuing a stop. Figure 99 is an example of a write followed by a read where Sr is a repeated start condition.

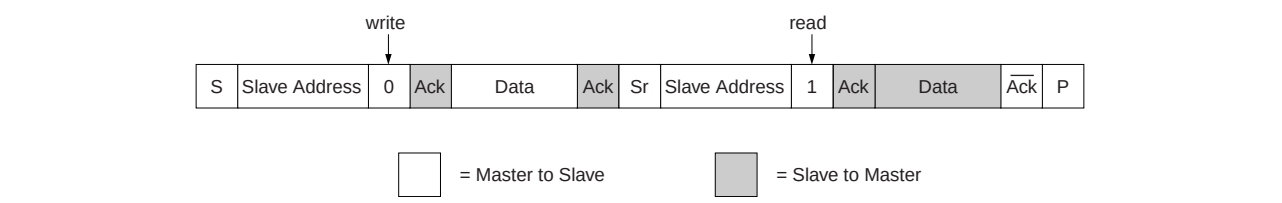


Fig 99. I²C 7-bit slave address write then read operation

23.7.10 Bus Arbitration

The I2C allows any bus master to start a transfer when the bus is idle. In a multi-master system, it is possible to have more than one master start a transfer at the same time. To arbitrate between masters, all I2C masters must monitor the state of the bus while they are driving it. If a master is trying to put a “1” on the bus while another is driving a “0”, the bus will be low (wire-AND) and the master trying to put a “1” on the bus must abort its operation. The master driving a “0” continues its operation unaware that another master aborted a transfer.

23.7.11 Clock Synchronization

Since different devices can drive SCL at different frequencies, masters must account for this by starting their clock-high timer when SCL actually goes high rather than when it tries to drive it high. Likewise, the master starts counting the clock-low time when SCL actually goes low. An example is shown in Figure 100.

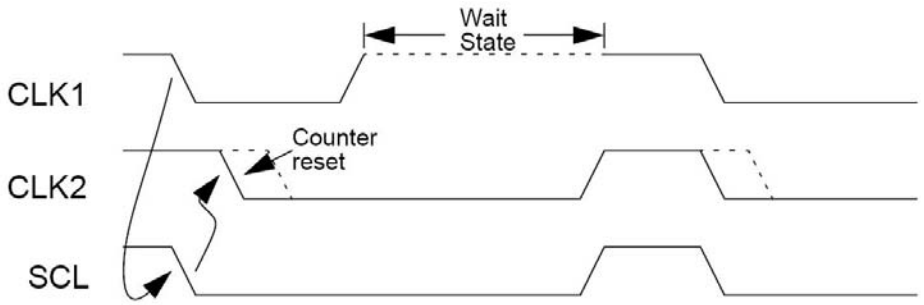


Fig 100. I^2C Clock Synchronization

24.1 Features

The I2S bus provides a standard communication interface for digital audio applications. The LPC32x0 includes two I2S interfaces; I2S0 and I2S1.

The I2S bus specification defines a 3-wire serial bus, having 1 data, 1 clock, and one word select signal. The basic I2S connection has one master, which is always the master, and one slave. The I2S interface on the LPC32x0 provides a separate transmit and receive channel, each of which can operate as either a master or a slave.

- The I2S input can operate in either master or slave mode.

  The I2S output can operate as either master a slave, independent of the I2S input.
- Supports independent TX and RX clocks, a common clock based on TX clock input, or a common clock based on the RX clock input.
- Capable of handling 8, 16, and 32 bit word sizes.
- Mono and stereo audio data supported.
- The sampling frequency can range (in practice) from 16 - 96 kHz. (16, 22.05, 32, 44.1, 48, 96 kHz) for audio applications.
- Word Select period in master mode is configurable (separately for I2S input and I2S output).
- Two 8 word FIFO data buffers are provided, one for transmit and one for 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.

24.2 Description

The I2S transfers serial data out via the transmit channel and serial data in via the receive channel. These transfers support the NXP Inter IC Audio format for 8, 16 and 32 bits audio data both for stereo and mono modes. Configuration, data access and control is performed by an APB register set. The data streams are buffered by FIFOs with a depth of 8 words. An I2S interface is comprised of three signals, SCK, WS, SD.

I2S Interface signal description

SCK — provides the clock needed to synchronize the transfer of data through the interface.

WS — provides the clock needed to separate left and right channels and establish the sampling frequency.

SD — provides the serial data sent through the interface.

The I2S receive and transmit stage can operate independently in either slave or master mode, In addition, configuration options are available which allow sharing the SCK and WS between the transmit and receive channels. One purpose of the shared signals is to reduce the number of external pins needed to connect to an external I2S device, there is a six pin configuration and a four pin configuration. The number of pins can be reduced if the SCK and WS clocks for the transmitter and receiver can be the same in the application.

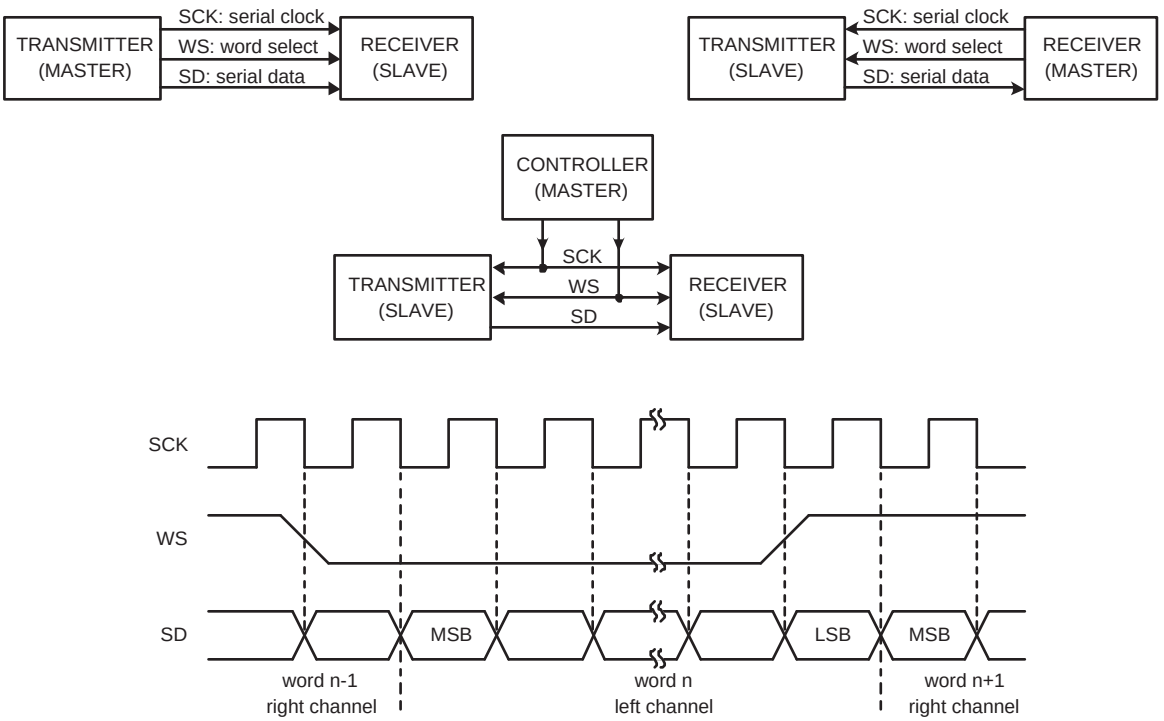


Fig 101. Simple I2S configurations and bus timing

Within the I2S module, the word select (WS) signal 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.

24.3 Pin descriptions

Table 518. Pin descriptions

Pin Name	Type	Description
I2S0RX_CLK I2S1RX_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.
I2S0RX_WS I2S1RX_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).
I2S0RX_SDA I2S1RX_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.
I2S0TX_CLK I2S1TX_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.
I2S0TX_WS I2S1TX_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).
I2S0TX_SDA I2S1TX_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.

24.4 Operation

When an I2S interface is active, the word select, receive clock and transmit clock signals are sent continuously by the I2S master, while data is sent continuously by the transmitter which can be either the master or slave. By definition, the device that provides the two clock signals SCK and WS is the I2S master. The assignment of master or slave is not dependent on the device that provides the data signal SD.

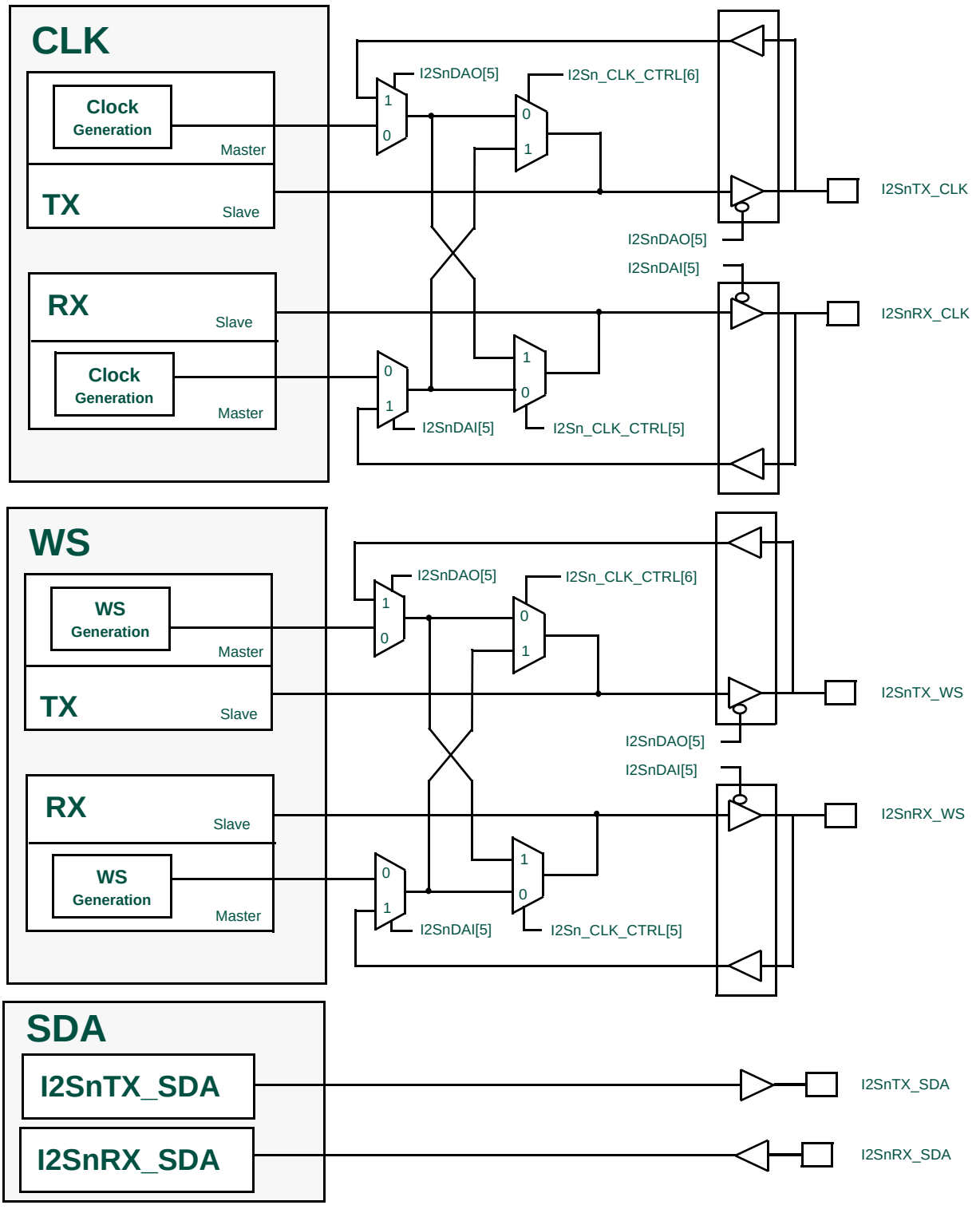


Fig 102. I2S Configuration

In master mode (I2SnDAI[5] = 0 and/or I2SnDAO[5] = 0), SCLK is generated internally using HCLK and a fractional divider, see Figure 104 for details of the clock generation block.

WS is generated from the active bitrate clock, see Figure 104 for details. The half period count value in these register (I2SnDAI or I2SDAO) defines the half period of the WS signal.

In slave mode (I2SnDAI[5] = 1 and/or I2SnDAO[5] = 1) SCLK and WS are input from the relevant input pin.

Table 519 provides a table that shows the possible I2S configurations.

Table 519. I2S Configuration Table

TX master select I2Sn_DAO[5]	RX master select I2Sn_DAI[5]	TX mode I2Sn_CLK_CTRL[6]	RX mode I2Sn_CLK_CTRL[5]	TX Bitrate CLK Drives	RX Bitrate CLK Drives	TX CLK Pin Drives	RX CLK Pin Drives	TX WS Generator Drives	RX WS Generator Drives	TX WS Pin Drives	RX WS Pin Drives
0	0	0	0	TX pin	RX pin	-	-	TX pin	RX pin	-	-
0	0	0	1	TX pin, RX pin	-	-	-	TX pin, RX pin	-	-	-
0	0	1	0	-	RX pin, TX pin	-	-	-	RX pin, Tx pin	-	-
0	0	1	1	RX pin	TX pin	-	-	RX pin	TX pin	-	-
0	1	0	0	TX pin	-	-	RXslave	TX pin	-	-	RXslave
0	1	0	1	TX pin, RXslave	-	-	-	TX pin, RXslave	-	-	-
0	1	1	0	-	-	-	TX pin, RXslave	-		-	TX pin, RXslave
0	1	1	1	RXslave	-	-	TXpin	RXslave		-	TX pin,
1	0	0	0	-	RX pin	TXslave	-	-	RX pin,	TXslave	-
1	0	0	1	-		TXslave, RX pin	-	-		TXslave, RX pin	-
1	0	1	0	-	RX pin, TXslave		-	-	RX pin, TXslave	-	-
1	0	1	1	-	TXslave	RX pin	-	-	TXslave	RX pin	-
1	1	0	0	-	-	TXslave	RXslave	-	-	TXslave	RXslave
1	1	0	1	-	-	TXslave, RXslave		-	-	TXslave, RXslave	
1	1	1	0	-	-	-	RXslave, TXslave	-	-	-	RXslave, TXslave
1	1	1	1	-	-	RXslave	TXslave	-	-	RXslave	TXslave

Disabling the I2S can be done using 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.

24.5 I²S transmit and receive interfaces

The I2S interface can transmit and receive 8, 16 or 32 bits 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 bits data words.
  - 1: word is considered to contain two 16 bits data words.
  - 3: word is considered to contain one 32 bits 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 FIFO's are 32 bits. Figure 103 shows the possible data sequences.

A data sample in the FIFO consists of:

- 1x32 bits in 8 or 16 bit stereo modes.
- 1x32 bits in mono modes.
- 2x32 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.

24.6 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 I2SSTATE status register.

Table 520. Conditions for FIFO level comparison

Level Comparison	Condition
dmareq_tx_0	tx_depth_dma0 >= tx_level
dmareq_rx_0	rx_depth_dma0 <= rx_level
dmareq_tx_1	tx_depth_dma1 >= tx_level
dmareq_rx_1	rx_depth_dma1 <= 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 521. DMA and interrupt request generation

System Signaling	Condition
irq	(irq_rx & rx_irq_enable) | (irq_tx & tx_irq_enable
dmareq[0]	(dmareq_tx_0 & tx_dma0_enable) | (dmareq_rx_0 & rx_dma0_enable)
dmareq[1]	(dmareq_tx_1 & tx_dma1_enable) | (dmareq_rx_1 & rx_dma1_enable)

Table 522. Status feedback in the I2SSTATE register

Status Feedback	Status
irq	irq_rx | irq_tx
dmareq0	(dmareq_tx_0 | dmareq_rx_0)
dmareq1	(dmareq_rx_1 | dmareq_tx_1)

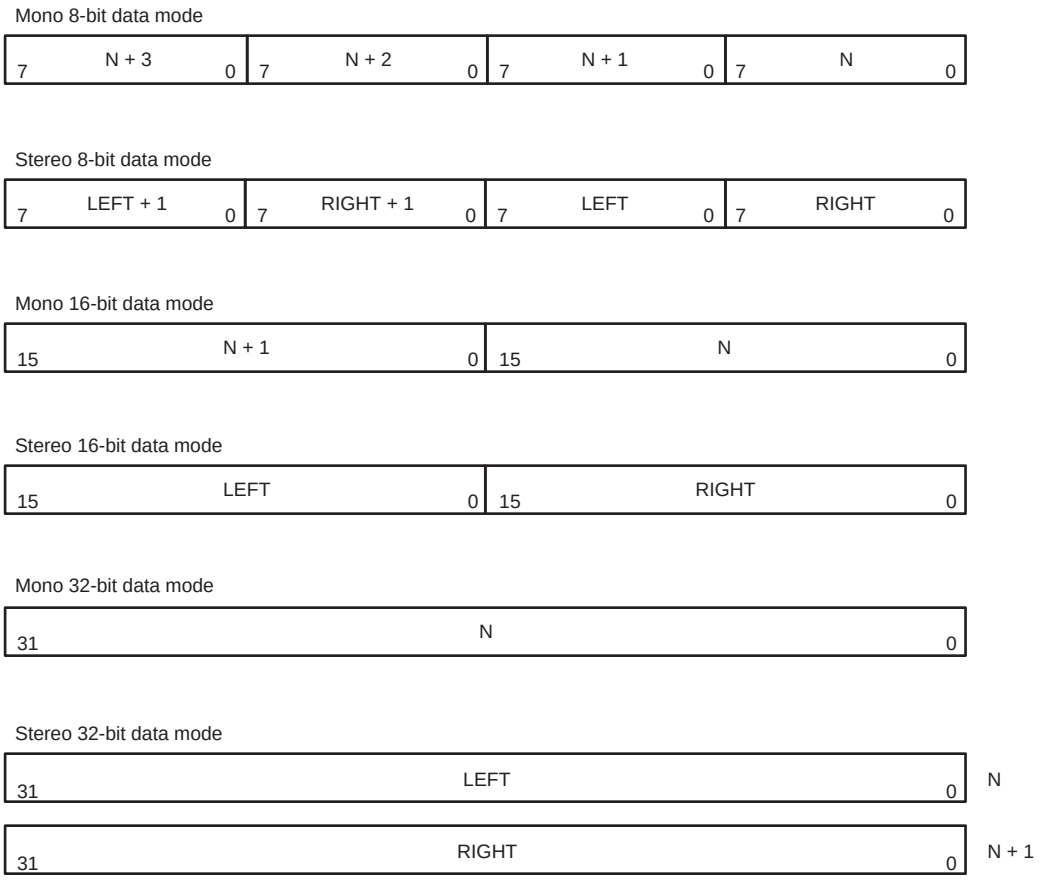


Fig 103. FIFO contents for various I²S modes

24.7 Register description

Table 523 shows the registers associated with the two I2S interface and a summary of their functions. Following the table are details for each register.

Table 523. I2S register map

Generic Name	Description	Access	Reset Value[1]	I2Sn Register Name & Address
I2S_CTRL	I2S control register. Contains control bits for the I2S clocking, DMA, and pin configuration.	R/W	0x00	I2S_CTRL 0x4000 407C
I2SDAO	Digital Audio Output Register. Contains control bits for the I2S transmit channel.	R/W	0xE1	I2S0DAO 0x2009 4000 I2S1DAO 0x2009 C000
I2SDAI	Digital Audio Input Register. Contains control bits for the I2S receive channel.	R/W	0xE1	I2S0DAI 0x2009 4004 I2S1DAI 0x2009 C004
I2STXFIFO	Transmit FIFO. Access register for the 8 ¥ 32 bit transmitter FIFO.	WO	-	I2S0TXFIFO 0x2009 4008 I2S1TXFIFO 0x2009 C008

Table 523. I2S register map

Generic Name	Description	Access	Reset Value[1]	I2Sn Register Name & Address
I2SRXFIFO	Receive FIFO. Access register for the 8 ¥ 32 bit receiver FIFO.	RO	-	I2S0RXFIFO 0x2009 400C I2S1RXFIFO 0x2009 C00C
I2SSTATE	Status Feedback Register. Contains status information about the I2S interface.	RO	0x07	I2S0STATE 0x2009 4010 I2S1STATE 0x2009 C010
I2SDMA0	DMA Configuration Register 0. Contains control information for DMA request 0.	R/W	0	I2S0DMA0 0x2009 4014 I2S1DMA0 0x2009 C014
I2SDMA1	DMA Configuration Register 1. Contains control information for DMA request 1.	R/W	0	I2S0DMA1 0x2009 4018 I2S1DMA1 0x2009 C018
I2SIRQ	Interrupt Request Control Register. Contains bits that control how the I2S interrupt request is generated.	R/W	0	I2S0IRQ 0x2009 401C I2S1IRQ 0x2009 C01C
I2STXRATE	Transmit bit rate divider. This register determines the I2S transmit bit rate by specifying the value to divide pclk by in order to produce the transmit bit clock.	R/W	0	I2S0TXRATE 0x2009 4020 I2S1TXRATE 0x2009 C020
I2SRXRATE	Receive bit rate divider. This register determines the I2S receive bit rate by specifying the value to divide pclk by in order to produce the receive bit clock.	R/W	0	I2S0RXRATE 0x2009 4024 I2S1RXRATE 0x2009 C024

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

24.7.1 I2S Block Control register (I2S_CTRL - 0x4000 407C)

The I2S_CTRL register controls some aspects of the two I2S interfaces (I2S0 and I2S1): enabling and disabling clocks, selecting DMA request line, and controlling source clock output or input configuration.

Table 524. I2S Block Control register (I2S_CTRL - 0x4000 407C)

Bit	Function	Reset value
0	I2S0_CLK enable. 0 = Disable clock. 1 = Enable clock.	0
1	I2S1_CLK enable. 0 = Disable clock. 1 = Enable clock.	0
2	I2S0_CLK_RX_MODE select. 0 = RX_CLK drives I2S RX timing. Applies to both Master (output) and Slave (input) modes 1 = TX_CLK drives I2S RX timing. Applies to both Master (output) and Slave (input) modes	0

Table 524. I2S Block Control register (I2S_CTRL - 0x4000 407C) *…continued*

Bit	Function	Reset value
3	I2S0_CLK_TX_MODE select. 0 = TX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes. 1 = RX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes.	0
4	I2S1 DMA 1 connection control 0 = UART7 RX is connected to DMA; (I2S1 DMA 1 is not connected to DMA) 1 = I2S1 DMA 1 is connected to DMA; (UART7 RX is not connected to DMA)	0
5	I2S1_CLK_RX_MODE select. 0 = RX_CLK drives I2S RX timing. Applies to both Master (output) and Slave (input) modes 1 = TX_CLK drives I2S RX timing. Applies to both Master (output) and Slave (input) modes	0
6	I2S1_CLK_TX_MODE select. 0 = TX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes. 1 = RX_CLK drives I2S TX timing. Applies to both Master (output) and Slave (input) modes.	0
31:7	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.2 Digital Audio Output Register (I2S0DAO - 0x2009 4000 I2S1DAO - 0x2009 C000)

The I2SDAO register controls the operation of the I2S transmit channel. The function of bits in DAO are shown in Table 525.

Table 525. Digital Audio Output register (I2S0DAO - 0x2009 4000 I2S1DAO - 0x2009 C000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	wordwidth		Selects the number of bytes in data as follows:	01
		00	8 bit data	
		01	16 bit data	
		10	Reserved, do not use this setting	
		11	32 bit data	
2	mono		When one, data is of monaural format. When zero, the data is in stereo format.	0
3	stop		Stops draining of TX FIFO, places the transmit channel in mute mode.	0
4	reset		Asynchronously reset the transmit channel and FIFO.	0
5	ws_sel		When 0 master mode, when 1 slave mode.	1
14:6	ws_halfperiod		Word select half period minus one, i.e. WS 64clk period -> ws_halfperiod = 31.	0x1F
15	mute		When true, the transmit channel sends only zeroes.	1
31:16	reserved		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.3 Digital Audio Input Register (I2S0DAI - 0x2009 4004 I2S1DAI - 0x2009 C004)

The I2SDAI register controls the operation of the I2S receive channel. The function of bits in DAI are shown in Table 526.

Table 526. Digital Audio Input register (I2S0DAI - 0x2009 4004 I2S1DAI - 0x2009 C004) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	wordwidth		Selects the number of bytes in data as follows:	01
		00	8 bit data	
		01	16 bit data	
		10	Reserved, do not use this setting	
		11	32 bit data	
2	mono		When one, data is of monaural format. When zero, the data is in stereo format.	0
3	stop		Stops filling of RX FIFO.	0
4	reset		Asynchronously reset the transmit channel and FIFO.	0
5	ws_sel		When 0 master mode, when 1 slave mode.	1
14:6	ws_halfperiod		Word select half period minus one, i.e. WS 64clk period -> ws_halfperiod = 31.	0x1F
15	Unused		Unused.	1
31:16	reserved		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.4 Transmit FIFO Register (I2S0TXFIFO - 0x2009 4008 I2S1TXFIFO - 0x2009 C008)

The I2STXFIFO register provides access to the transmit FIFO. The function of bits in I2STXFIFO are shown in Table 527.

Table 527. Transmit FIFO register (I2S0TXFIFO - 0x2009 4008 I2S1TXFIFO - 0x2009 C008) bit description

Bit	Symbol	Description	Reset Value
31:0	I2STXFIFO	8 ¥ 32 bits transmit FIFO.	Level = 0

24.7.5 Receive FIFO Register (I2S0RXFIFO - 0x2009 400C I2S1RXFIFO - 0x2009 C00C)

The I2SRXFIFO register provides access to the receive FIFO. The function of bits in I2SRXFIFO are shown in Table 528.

Table 528. Receive FIFO register (I2S0RXFIFO - 0x2009 400C I2S1RXFIFO - 0x2009 C00C) bit description

Bit	Symbol	Description	Reset Value
31:0	I2SRXFIFO	8 ¥ 32 bits transmit FIFO.	level = 0

24.7.6 Status Feedback Register (I2S0STATE - 0x2009 4010 I2S1STATE - 0x2009 C010)

The I2SSTATE register provides status information about the I2S interface. The meaning of bits in I2SSTATE are shown in Table 529.

Table 529. Status Feedback register (I2S0STATE - 0x2009 4010 I2S1STATE - 0x2009 C010) bit description

Bit	Symbol	Description	Reset Value
0	irq	This bit reflects the presence of Receive Interrupt or Transmit Interrupt.	1
1	dma0	This bit reflects the presence of Receive or Transmit DMA Request 0.	1
2	dma1	This bit reflects the presence of Receive or Transmit DMA Request 1.	1
7:3	Unused	Unused.	0
11:8	rx_level	Reflects the current level of the Receive FIFO. Only values 0 to 8 are supported because FIFO is 8 words.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	tx_level	Reflects the current level of the Transmit FIFO. Only values 0 to 8 are supported because FIFO is 8 words.	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.7 DMA Configuration Register 0 (I2S0DMA0 - 0x2009 4014 I2S1DMA0 - 0x2009 C014)

The I2SDMA0 register controls the operation of DMA request 0. The function of bits in I2SDMA0 are shown in Table 530. Refer to the General Purpose DMA Controller chapter for details of DMA operation.

Table 530. DMA Configuration register 0 (I2S0DMA0 - 0x2009 4014 I2S1DMA0 - 0x2009 C014) bit description

Bit	Symbol	Description	Reset Value
0	rx_dma0_enable	When 1, enables DMA0 for I2S receive.	0
1	tx_dma0_enable	When 1, enables DMA0 for I2S transmit.	0
7:2	Unused	Unused.	0
11:8	rx_depth_dma0	Set the FIFO level that triggers a receive DMA request on DMA0. Only values 0 to 8 are supported because FIFO is 8 words.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	tx_depth_dma0	Set the FIFO level that triggers a transmit DMA request on DMA0. Only values 0 to 8 are supported because FIFO is 8 words.	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.8 DMA Configuration Register 1 (I2S0DMA1 - 0x2009 4018 I2S1DMA1 - 0x2009 C018)

The I2SDMA1 register controls the operation of DMA request 1. The function of bits in I2SDMA1 are shown in Table 525.

Table 531. DMA Configuration register 1 (I2S0DMA1 - 0x2009 4018 I2S1DMA1 - 0x2009 C018) 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	Unused	Unused.	0
10:8	rx_depth_dma1	Set the FIFO level that triggers a receive DMA request on DMA1.	0
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
18:16	tx_depth_dma1	Set the FIFO level that triggers a transmit DMA request on DMA1.	0
31:19	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.9 Interrupt Request Control Register (I2S0IRQ - 0x2009 401C I2S1IRQ - 0x2009 C01C)

The I2SIRQ register controls the operation of the I2S interrupt request. The function of bits in I2SIRQ are shown in Table 525.

Table 532. Interrupt Request Control register (I2S0IRQ - 0x2009 401C I2S1IRQ - 0x2009 C01C) 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	Unused.	0
15:8	rx_depth_Irq	Set the FIFO level on which to create an irq request.	0
23:16	tx_depth_Irq	Set the FIFO level on which to create an irq request.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.7.10 Transmit Clock Rate Register (I2S0TXRATE - 0x2009 4020 I2S1TXRATE - 0x2009 C020)

The bit 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.

Note: If the value of X = 0, then no clock is generated.

$$I2STXRATE = \left((HCLK) \times \frac{X}{Y}\right) / 2 \quad (14)$$

For example, a 48 kHz sample rate for 16 bit stereo data requires a TXBitrate of (48,000 X 16 X 2) = 1.536 MHz. Using a 104MHz HCLK and dividing by the TXBitrate establishes the X/Y divider ratio as 0.0295384. The best fit setting occurs when X = 7 and Y = 237 which is a ratio of 0.0295358 an error of 0.088%

Table 533. Transmit Clock Rate register (I2S0TXRATE - 0x2009 4020 I2S1TXRATE - 0x2009 C020) bit description

Bit	Symbol	Description	Reset Value
7:0	Y_divider	I2S transmit bit rate denominator. This value is used to divide HCLK to produce the transmit bit clock. Eight bits of fractional divide supports a wide range of I2S rates over a wide range of HCLK rates. A value of zero stops the clock.	0
15:8	X_divider	I2S transmit bit rate numerator. This value is used to multiply HCLK by to produce the transmit bit clock. A value of zero stops the clock. Eight bits of fractional divide supports a wide range of I2S rates over a wide range of HCLK rates. note: the resulting ratio X/Y is divided by 2.	0
31:16	Unused	Unused.	0

See Figure 104 for a diagram showing clock generation in the I2S interface.

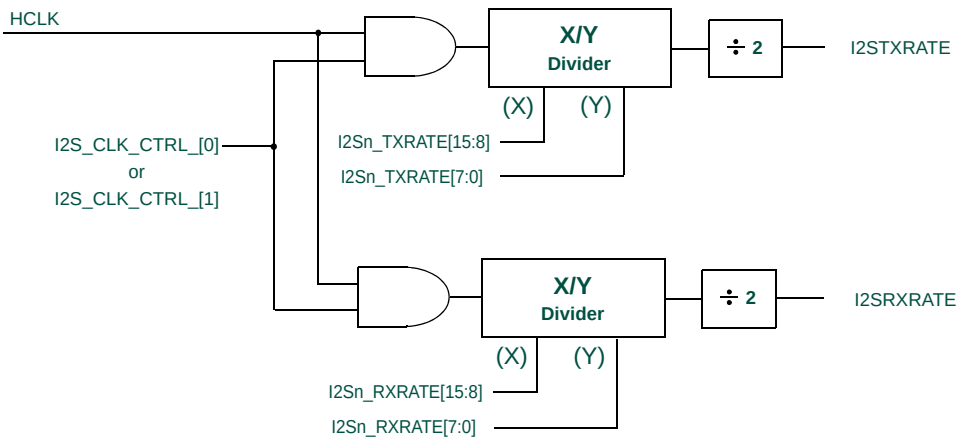


Fig 104. I2S Clock Generation

24.7.11 Receive Clock Rate Register (I2S0RXRATE - 0x2009 4024 I2S1RXRATE - 0x2009 C024)

The bit rate for the I2S receiver is determined by the value of the I2SRXRATE register. The settings depend on the audio sample rate desired, the format (stereo/mono) used, and the data size. The equation for I2SRXRATE is the same as for I2STXRATE.

Table 534. Receive Clock Rate register (I2S0RXRATE - 0x2009 4024 I2S1RXRATE - 0x2009 C024) bit description

Bit	Symbol	Description	Reset Value
7:0	Y_divider	I2S receive bit rate denominator. This value is used to divide HCLK by to produce the receive bit clock. Eight bits of fractional divide supports a wide range of I2S rates over a wide range of HCLK rates. A value of zero stops the clock	0
15:7	X_divider	I2S receive bit rate numerator. This value is used to multiply HCLK to produce the receive bit clock. A value of zero stops the clock. Eight bits of fractional divide supports a wide range of I2S rates over a wide range of HCLK rates. note: the resulting ratio X/Y is divided by 2.	0
31:15	Unused	Unused.	0

25.1 Features

Each of the six standard timers support the following features:

- A 32 bit Timer/Counter with a programmable 32 bit Prescaler.
- Counter or Timer operation
- Up to four 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.

Remark: The four Timer/Counters are created from identical IP blocks except for their peripheral base addresses and pin implementation. Match registers without pins are still support their internal features. Implementation details are summarized here:

- Timer 0 provides two capture input and four match output pins.
- Timer1, Timer2, Timer3, each provide one capture input and two match output pins,
- Timer4 provides one capture input and has no match output pins.
- Timer5 has no external pins.

25.2 Applications

- Interval Timer for counting internal events.
- Pulse Width Demodulator via capture inputs.
- Free running timer.

25.3 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.

25.4 Architecture

The block diagram for TIMER/COUNTER0 and TIMER/COUNTER1 is shown in Figure 105.

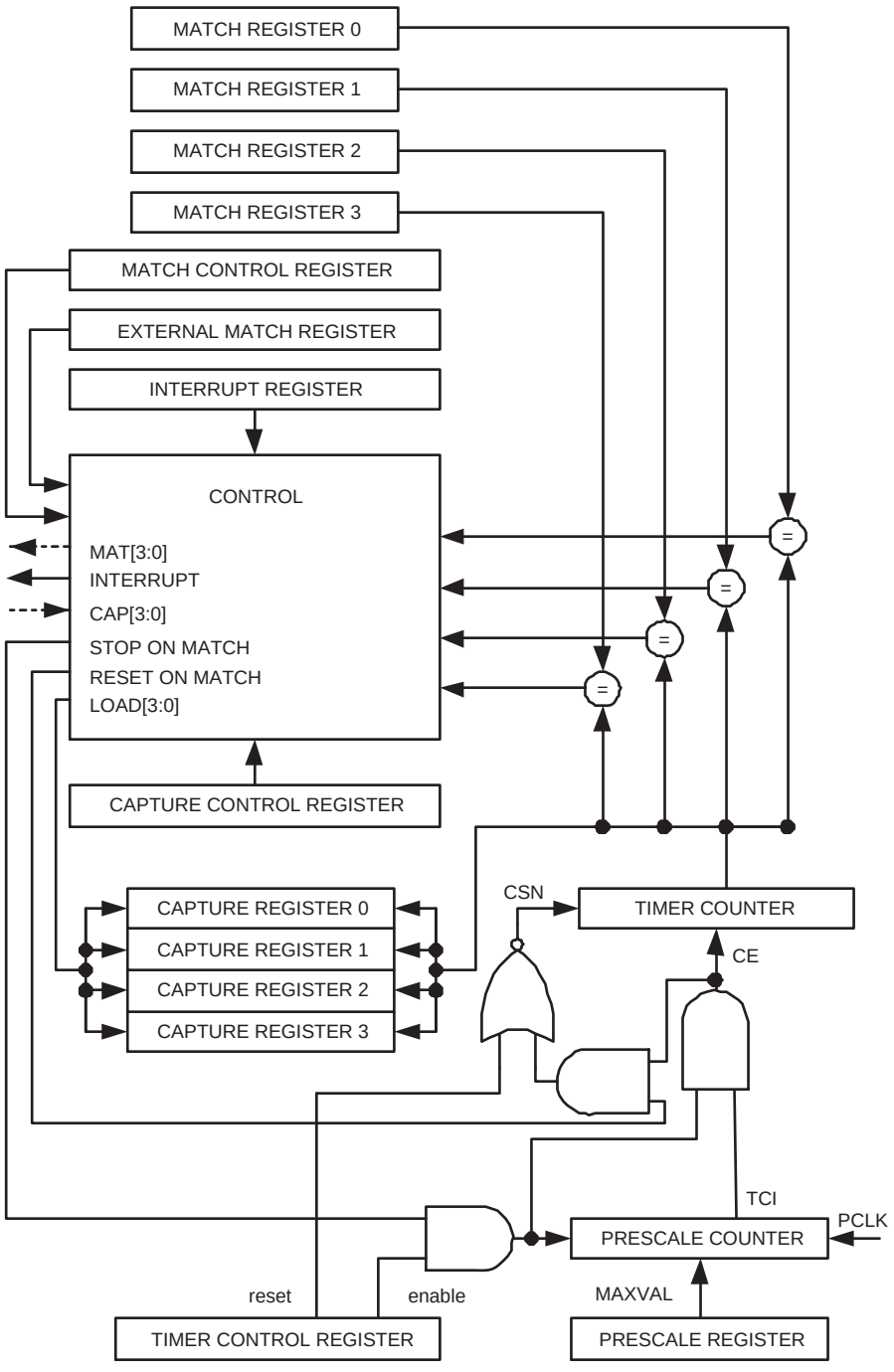


Fig 105. Timer block diagram

25.5 Pin description

Table 535 gives a brief summary of each of the Timer/Counter related pins. Note that TImer1, Timer2 and Timer3 have only one CAPn pin and two MATn pins, and even though no pins exist for these timers, the internal registers are still available for use.

Table 535. Timer/Counter pin description

Pin	Type	Description
CAP0.[1:0] CAP1.[0] CAP2.[0] CAP3.[0] CAP4.[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. Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Table 544.
MAT0.[3:0] MAT1.[1:0] MAT2.[1:0] MAT3.[1:0]	Output	External Match Output 0/1- When a match register n (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.

25.5.1 Multiple CAP and MAT pins

Software must enable pins for most of the CAP or MAT functions using the Pin Select registers, which are described in Section 33.1 to and including Section 33.4 of the pin multiplexing chapter.

25.6 Timer base addresses

Table 536. Addresses for Timers 0, 1, 2, 3, 4, and 5

Timer	Base addresses
0	0x4004 4000
1	0x4004 C000
2	0x4005 8000
3	0x4006 0000
4	0x4002 C000
5	0x4003 0000

25.7 Register description

Each Timer/Counter contains the registers shown in Table 537 ("Reset Value" refers to the data stored in used bits only; it does not include reserved bits content). More detailed descriptions follow.

Table 537. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
IR	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.	R/W	0	T0IR - 0x4004 4000 T1IR - 0x4004 C000 T2IR - 0x4005 8000 T3IR - 0x4006 0000 T4IR - 0x4002 C000 T5IR - 0x4003 0000
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	T0TCR - 0x4004 4004 T1TCR - 0x4004 C004 T2TCR - 0x4005 8004 T3TCR - 0x4006 0004 T4TCR - 0x4002 C004 T5TCR - 0x4003 0004
TC	Timer Counter. The 32 bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	T0TC - 0x4004 4008 T1TC - 0x4004 C008 T2TC - 0x4005 8008 T3TC - 0x4006 0008 T4TC - 0x4002 C008 T5TC - 0x4003 0008
PR	Prescale Register. When the Prescale Counter (described below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0x4004 400C T1PR - 0x4004 C00C T2PR - 0x4005 800C T3PR - 0x4006 000C T4PR - 0x4002 C00C T5PR - 0x4003 000C
PC	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.	R/W	0	T0PC - 0x4004 4010 T1PC - 0x4004 C010 T2PC - 0x4005 8010 T3PC - 0x4006 0010 T4PC - 0x4002 C010 T5PC - 0x4003 0010
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a match occurs.	R/W	0	T0MCR - 0x4004 4014 T1MCR - 0x4004 C014 T2MCR - 0x4005 8014 T3MCR - 0x4006 0014 T4MCR - 0x4002 C014 T5MCR - 0x4003 0014
MR0	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.	R/W	0	T0MR0 - 0x4004 4018 T1MR0 - 0x4004 C018 T2MR0 - 0x4005 8018 T3MR0 - 0x4006 0018 T4MR1 - 0x4002 C01C T5MR1 - 0x4003 001C
MR1	Match Register 1. See MR0 description.	R/W	0	T0MR1 - 0x4004 401C T1MR1 - 0x4004 C01C T2MR1 - 0x4005 801C T3MR1 - 0x4006 001C T4MR1 - 0x4002 C01C T5MR1 - 0x4003 001C

Table 537. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
MR2	Match Register 2. See MR0 description.	R/W	0	T0MR2 - 0x4004 4020 T1MR2 - 0x4004 C020 T2MR2 - 0x4005 8020 T3MR2 - 0x4006 0020 T4MR2 - 0x4002 C020 T5MR2 - 0x4003 0020
MR3	Match Register 3. See MR0 description.	R/W	0	T0MR3 - 0x4004 4024 T1MR3 - 0x4004 C024 T2MR3 - 0x4005 8024 T3MR3 - 0x4006 0024 T4MR3 - 0x4002 C024 T5MR3 - 0x4003 0024
CCR	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.	R/W	0	T0CCR - 0x4004 4028 T1CCR - 0x4004 C028 T2CCR - 0x4005 8028 T3CCR - 0x4006 0028 T4CCR - 0x4002 C028
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 (CAP0.0, CAP1.0, CAP2.0, CAP3.0 respectively) input.	RO	0	T0CR0 - 0x4004 402C T1CR0 - 0x4004 C02C T2CR0 - 0x4005 802C T3CR0 - 0x4006 002C T4CR0 - 0x4002 C02C
CR1	Capture Register 1. See CR0 description.	RO	0	T0CR1 - 0x4004 4030 T1CR1 - 0x4004 C030 T2CR1 - 0x4005 8030 T3CR1 - 0x4006 4030
CR2	Capture Register 2. See CR0 description.	RO	0	T0CR1 - 0x4004 4034 T1CR0 - 0x4004 C034 T2CR0 - 0x4005 8034 T3CR0 - 0x4006 0034
CR3	Capture Register 3. See CR0 description.	RO	0	T0CR1 - 0x4004 4038 T1CR0 - 0x4004 C038 T2CR0 - 0x4005 8038 T3CR0 - 0x4006 0038
EMR	External Match Register. The EMR controls the external match pins MATn.n.	R/W	0	T0EMR - 0x4004 403C T1EMR - 0x4004 C03C T2EMR - 0x4005 803C T3EMR - 0x4006 003C
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	T0CTCR - 0x4004 4070 T1CTCR - 0x4004 C070 T2CTCR - 0x4005 8070 T3CTCR - 0x4006 0070 T4CTCR - 0x4002 C070 T5CTCR - 0x4003 0070

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

25.7.1 Timer Interrupt Register (T0IR - 0x4004 4000, T1IR - 0x4004 C000, T2IR - 0x4005 8000, T3IR - 0x4006 0000, T4IR - 0x4002 C000 and T5IR 0x4003 0000)

The Interrupt Register consists of four bits for the match interrupts and four 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.

Table 538: Timer Interrupt Registers (T0IR - 0x4004 4000,T1IR - 0x4004 C000, T2IR - 0x4005 8000, T3IR - 0x4006 0000, T4IR - address 0x4002 C000 and T5IR address 0x4003 0000) bit description

Bit	Symbol	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
6	CR2 Interrupt	Interrupt flag for capture channel 2 event.	0
7	CR3 Interrupt	Interrupt flag for capture channel 3 event.	0

25.7.2 Timer Control Register (T0TCR - 0x4004 4004, T1TCR - 0x4004 C004, T2TCR - 0x4005 8004, T3TCR - 0x4006 0004, T4TCR - 0x4002 C004 and T5TCR 0x4003 0004)

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 539. Timer Control Register (T0TCR - 0x4004 4004, T1TCR - 0x4004 C004, T2TCR - 0x4005 8004, T3TCR - 0x4006 0004, T4TCR - address 0x4002 C004 T5TCR address 0x4003 0004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, 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
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

25.7.3 Timer Counter registers (T0TC - 0x4004 4008, T1TC - 0x4004 C008, T2TC - 0x4005 8008, T3TC - 0x4006 0008, T4TC - 0x4002 C008, T5TC - 0x4003 0008)

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.

25.7.4 Prescale register (T0PR - 0x4004 400C, T1PR - 0x4004 C00C, T2PR - 0x4005 800C, T3PR - 0x4006 000C, T4PR - 0x4002 C00C, T5PR - 0x4003 000C)

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

25.7.5 Prescale Counter register (T0PC - 0x4004 4010, T1PC - 0x4004 C010, T2PC - 0x4005 8010, T3PC - 0x4006 0010, T4PC - 0x4002 C010, T5PC - 0x4003 0010)

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.

25.7.6 Match Register MR0 (T0MR0 - 0x4004 4018, T1MR0 - 0x4004 C018, T2MR0 - 0x4005 8018, T3MR0 - 0x4006 0018, T4MR0 - 0x4002 C018, T5MR0 - 0x4003 0018)

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.

25.7.7 Match Register MR1 (T0MR1 - 0x4004 401C, T1MR1 - 0x4004 C01C, T2MR1 - 0x4005 801C, T3MR1 - 0x4006 001C, T4MR1 - 0x4002 C01C, T5MR1 - 0x4003 001C)

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.

25.7.8 Match Register MR2 (T0MR2 - 0x4004 4020, T1MR2 - 0x4004 C020, T2MR2 - 0x4005 8020, T3MR2 - 0x4006 0020, T4MR2 - 0x4002 C020, T5MR2 - 0x4003 0020)

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.

25.7.9 Match Register MR3 (T0MR3 - 0x4004 4024, T1MR3 - 0x4004 C024, T2MR3 - 0x4005 8024, T3MR3 - 0x4006 0024, T4MR3 - 0x4002 C024, T5MR3 - 0x4003 0024)

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.

25.7.10 Match Control Register (T0MCR - 0x4004 4014, T1MCR - 0x4004 C014, T2MCR - 0x4005 8014, T3MCR - 0x4006 0014, T4MCR - 0x4004 4014, T5MCR - 0x4004 C014)

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 540.

Table 540. Match Control Register (T0MCR - 0x4004 4014, T1MCR - 0x4004 C014, T2MCR - 0x4005 8014, T3MCR - 0x4006 0014, T4MCR - 0x4004 4014, T5MCR - 0x4004 C014)

Bit	Symbol	Description	Reset Value
0	MR0I	Interrupt on TnMR0: 1 = an interrupt is generated when TnMR0 matches the value in the TC. 0 = Feature disabled.	0
1	MR0R	Reset on TnMR0: 1 = the TC will be reset if TnMR0 matches it. 0 = Feature disabled.	0
2	MR0S	Stop on TnMR0: 1 = the TC and PC will be stopped and TCR[0] will be set to 0 if TnMR0 matches the TC. 0 = Feature disabled.	0
3	MR1I	Interrupt on TnMR1: 1 = an interrupt is generated when TnMR1 matches the value in the TC. 0 = Feature disabled.	0
4	MR1R	Reset on TnMR1: 1 = the TC will be reset if TnMR1 matches it. 0 = Feature disabled.	0
5	MR1S	Stop on TnMR1: 1 = the TC and PC will be stopped and TCR[0] will be set to 0 if TnMR1 matches the TC. 0 = Feature disabled.	0

Table 540. Match Control Register (T0MCR - 0x4004 4014, T1MCR - 0x4004 C014, T2MCR - 0x4005 8014, T3MCR - 0x4006 0014, T4MCR - 0x4004 4014, T5MCR - 0x4004 C014)

Bit	Symbol	Description	Reset Value
6	MR2I	Interrupt on TnMR2: 1 = an interrupt is generated when TnMR2 matches the value in the TC. 0 = Feature disabled.	0
7	MR2R	Reset on TnMR2: 1 = the TC will be reset if TnMR2 matches it. 0 = Feature disabled.	0
8	MR2S	Stop on TnMR2: 1 = the TC and PC will be stopped and TCR[0] will be set to 0 if TnMR2 matches the TC. 0 = Feature disabled.	0
9	MR3I	Interrupt on TnMR3: 1 = an interrupt is generated when TnMR3 matches the value in the TC. 0 = Feature disabled.	0
10	MR3R	Reset on TnMR3: 1 = the TC will be reset if TnMR3 matches it. 0 = Feature disabled.	0
11	MR3S	Stop on TnMR3: 1 = the TC and PC will be stopped and TCR[0] will be set to 0 if TnMR3 matches the TC. 0 = Feature disabled.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

25.7.11 Capture Registers CR0 (T0CR0 - 0x4004 402C, T1CR0 - 0x4004 C02C, T2CR0 - 0x4005 802C, T3CR0 - 0x4006 002C, T4CR0 - 0x4002 C02C)

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.

25.7.12 Capture Registers CR1 (T0CR1 - 0x4004 4030, T1CR1 - 0x4004 C030, T2CR1 - 0x4005 8030, T3CR1 - 0x4006 4030)

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.

25.7.13 Capture Registers CR2 (T0CR2 - 0x4004 4034, T1CR2 - 0x4004 C034, T2CR2 - 0x4005 8034, T3CR2 - 0x4006 0034)

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.

25.7.14 Capture Registers CR3 (T0CR3 - 0x4004 4038, T1CR3 - 0x4004 C038, T2CR3 - 0x4005 8038, T3CR3 - 0x4006 0038T0CR1 - 0x4004 4030)

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.

25.7.15 Capture Control Register (T0CCR - 0x4004 4028, T1CCR - 0x4004 C028, T2CCR - 0x4005 8028, T3CCR - 0x4006 0028)

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, 1, 2, or 3.

Note: Some capture inputs are not available as input pins, see Table 535

Note: If counter mode is selected for a particular CAPn 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 CAPn inputs.

Table 541. Capture Control Register (T0CCR - 0x4004 4028, T1CCR - 0x4004 C020, T2CCR - 0x4005 8028, T3CCR - 0x4006 0028, T4CCR - 0x4004 4028,)

Bit	Symbol	Description	Reset Value
0	CAP0RE	Capture on CAPn.0 rising edge: 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.	0
1	CAP0FE	Capture on CAPn.0 falling edge: 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.	0
2	CAP0I	Interrupt on CAPn.0 event: 1 = a CR0 load due to a CAPn.0 event will generate an interrupt. 0 = This feature is disabled.	0

Table 541. Capture Control Register (T0CCR - 0x4004 4028, T1CCR - 0x4004 C020, T2CCR - 0x4005 8028, T3CCR - 0x4006 0028, T4CCR - 0x4004 4028,)

Bit	Symbol	Description	Reset Value
3	CAP1RE	Capture on CAPn.1 rising edge: 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.	0
4	CAP1FE	Capture on CAPn.1 falling edge: 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.	0
5	CAP1I	Interrupt on CAPn.1 event: 1 = a CR1 load due to a CAPn.1 event will generate an interrupt. 0 = This feature is disabled.	0
6	CAP2RE	Capture on CAPn.2 rising edge: 1 = A sequence of 0 then 1 on CAPn.2 will cause CR2 to be loaded with the contents of TC. 0 = This feature is disabled.	0
7	CAP2FE	Capture on CAPn.2 falling edge: 1 = a sequence of 1 then 0 on CAPn.2 will cause CR2 to be loaded with the contents of TC. 0 = This feature is disabled.	0
8	CAP2I	Interrupt on CAPn.2 event: 1 = a CR2 load due to a CAPn.2 event will generate an interrupt. 0 = This feature is disabled.	0
9	CAP3RE	Capture on CAPn.3 rising edge: 1 = a sequence of 0 then 1 on CAPn.3 will cause CR3 to be loaded with the contents of TC. 0 = This feature is disabled.	0
10	CAP3FE	Capture on CAPn.3 falling edge: 1 = a sequence of 1 then 0 on CAPn.3 will cause CR3 to be loaded with the contents of TC 0 = This feature is disabled.	0
11	CAP3I	Interrupt on CAPn.3 event: 1 = a CR3 load due to a CAPn.3 event will generate an interrupt. 0 = This feature is disabled.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

25.7.16 External Match Register (T0EMR - 0x4004 403C, T1EMR - 0x4004 C03C, T2EMR - 0x4005 803C, T3EMR - 0x4006 003C)

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the Timer number 0, 1, 2, or 3, and “m” represent a Match number 0, 1, 2, or 3.

Table 542. External Match Register (T0EMR - 0x4004 403C, T1EMR - 0x4004 C03C, T2EMR - 0x4005 803C, T3EMR - 0x4006 003C) bit description

Bit	Symbol	Description	Reset Value
0	EM0	External Match 0. When a match occurs between the TC and TnMR0, 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 TnMR1, 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.2 pin, in a positive-logic manner (0 = low, 1 = high).	0
3	EM3	External Match 3. When a match occurs between the TC and TnMR3, 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.3 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. Table 543 shows the encoding of these bits.	00
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 543 shows the encoding of these bits.	00
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 543 shows the encoding of these bits.	00
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 543 shows the encoding of these bits.	00
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 543. External Match Control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

25.7.17 Count Control Register (T0CTCR - 0x4004 4070, T1CTCR - 0x4004 C070, T2CTCR - 0x4005 8070, T3CTCR - 0x4006 0070, 4CTCR - 0x4002 C070, T5CTCR 0x4003 0070)

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 corresponds to the one selected by bits 1:0 in the CTCR register, the Timer Counter register will be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input can not exceed one quarter of the PCLK clock. Consequently, duration of the high/low levels on the same CAP input in this case can not be shorter than 1/(2 PCLK).

Table 544. Count Control Register (T0CTCR - 0x4004 4070, T1CTCR - 0x4004 C070, T2CTCR - 0x4005 8070, T3CTCR - 0x4006 0070, T4CTCR - 0x4002 C004, T5CTCR - 0x4003 0004) bit description

Bit	Symbol	Description	Reset Value
1:0	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 = Timer Mode: every rising PCLK edge 01 = Counter Mode: TC is incremented on rising edges on the CAPn input selected by bits 3:2. 10 = Counter Mode: TC is incremented on falling edges on the CAPn input selected by bits 3:2. 11 = Counter Mode: TC is incremented on both edges on the CAPn input selected by bits 3:2.	00
3:2	Count Input Select	When bits 1:0 in this register are not 00, these bits select which CAPn pin is sampled for clocking: 00 = CAPn.0 for TIMERn 01 = CAPn.1 for TIMERn 10 = CAPn.2 for TIMERn 11 = CAPn.3 for TIMERn **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.	00
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

25.8 Example timer operation

Figure 106 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 107 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.

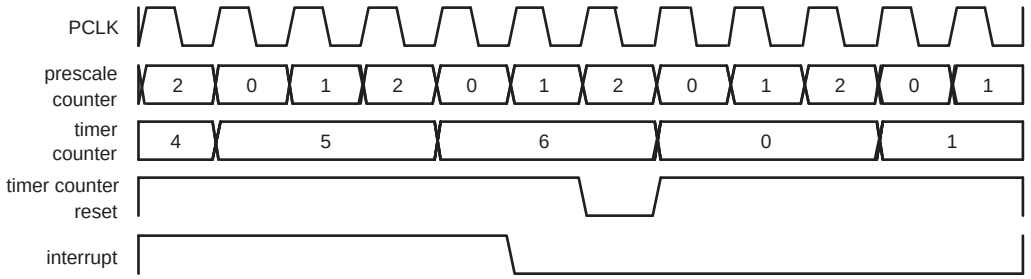


Fig 106. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

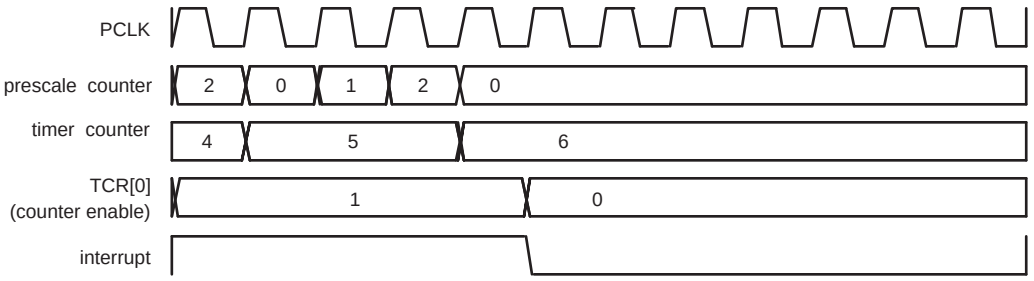


Fig 107. A timer Cycle in Which PR=2, MRx=6, and both interrupt and stop on match are enabled

26.1 Features

- 32-bit Timer/Counter with programmable 16-bit Prescaler.
- Counter or Timer operation.
- Two 32-bit capture channels which take a snapshot of the timer value when a signal transitions. A capture event may also optionally generate an interrupt.
- Three 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.
- Pause control to stop counting when core is in debug state.

26.2 Pin description

Table 545. High speed timer pin description

Pin name	Type	Description
GPI_6 (HSTIM_CAP)/ENET_RXD2	Input	External capture input

26.3 Description

The high speed timer block is clocked by PERIPH_CLK. The clock is first divided down in a 16 bit programmable prescale counter which clocks a 32 bit Timer/counter. The prescaler is typically set to output a suitable frequency e.g. 1 kHz. There are 3 match registers comparing value against the Timer/counter. A match in one of the match registers can generate an interrupt and the Timer/counter can be set to either continue to run, stop, or be reset. The high speed timer has only one interrupt connected to the main interrupt controller. The timer can be disabled in the TIMCLK_CTRL register. A time-out is typically handled by reading the Timer/counter, adding the number of clocks for the time-out and storing the new value into one of the Match registers. The overflow in the add operation (carry) can be discarded. The high-speed timer/counter also supports debug functionality, by setting the pause_en bit in the Timer control register the counter will not count while the ARM core is in debug state.

The block diagram of the High-speed timer is shown in Figure 108.

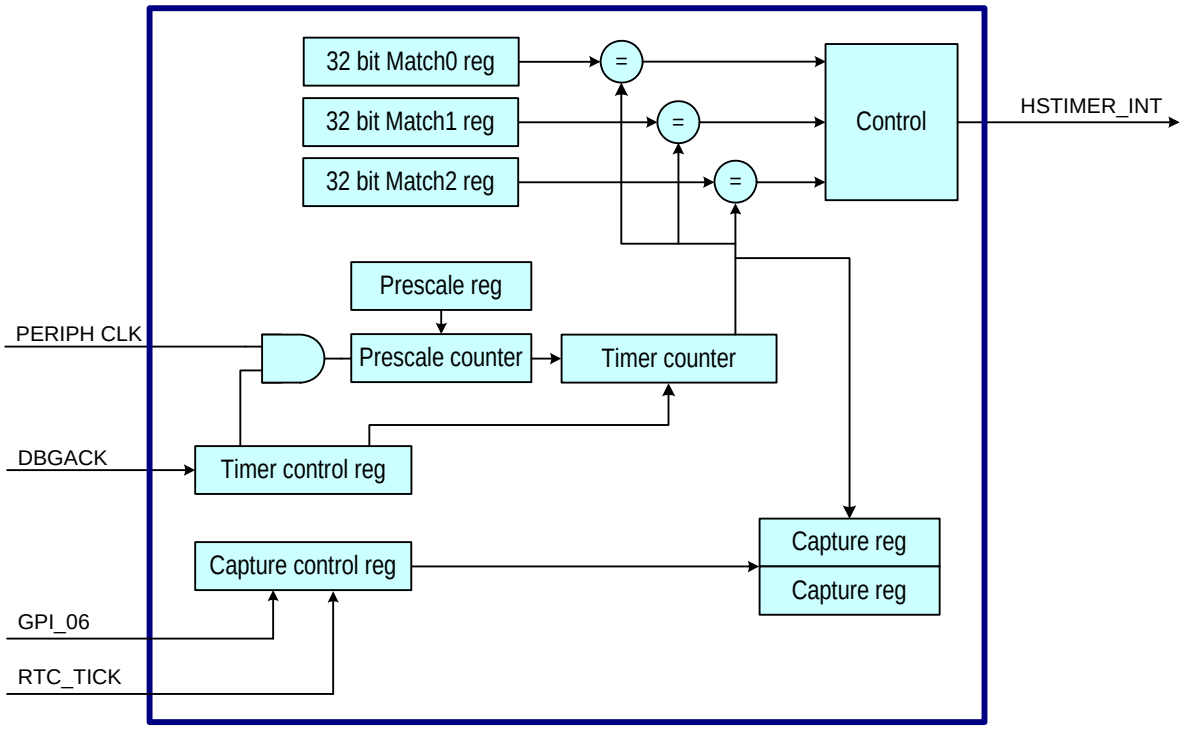


Fig 108. High speed timer block

26.4 Register description

The High Speed timer includes the registers shown in Table 546. Detailed descriptions of the registers follow.

Table 546. High speed timer registers

Address offset	Name	Description	Reset value	Type
0x4003 8000	HSTIM_INT	High Speed timer interrupt status register	0	R/W
0x4003 8004	HSTIM_CTRL	High Speed timer control register	0	R/W
0x4003 8008	HSTIM_COUNTER	High Speed timer counter value register	0	R/W
0x4003 800C	HSTIM_PMATCH	High Speed timer prescale counter match register	0	R/W
0x4003 8010	HSTIM_PCOUNT	High Speed Timer prescale counter value register	0	R/W
0x4003 8014	HSTIM_MCTRL	High Speed timer match control register	0	R/W
0x4003 8018	HSTIM_MATCH0	High Speed timer match 0 register	0	R/W
0x4003 801C	HSTIM_MATCH1	High Speed timer match 1 register	0	R/W
0x4003 8020	HSTIM_MATCH2	High Speed timer match 2 register	0	R/W
0x4003 8028	HSTIM_CCR	High Speed timer capture control register	0	R/W
0x4003 802C	HSTIM_CR0	High Speed timer capture 0 register	0	RO
0x4003 8030	HSTIM_CR1	High Speed timer capture 1 register	0	RO

26.4.1 High Speed Timer Interrupt Status register (HSTIM_INT, RW - 0x4003 8000)

Table 547. High Speed Timer Interrupt Status register (HSTIM_INT, RW - 0x4003 8000)

Bits	Description	Reset value
7:6	Not used. Write is don't care, Read returns random value.	0
5	RTC_TICK Reading a 1 indicates an active RTC tick capture status. Write 1 to clear this status. Note that this status can not generate an ARM interrupt. The pins can however generate PIO interrupts directly.	0
4	HSTIM_CAP Same description as for bit 5.	0
3	Not used. Write is don't care, Read returns random value	0
2	MATCH2_INT Reading a 1 indicates an active MATCH 2 interrupt. Writing a 1 clears the active interrupt status. Writing 0 has no effect. Note: Remove active match status by writing a new match value before clearing the interrupt. Otherwise this a new match interrupt may be activated immediately after clearing the match interrupt since the match may still be valid.	0x0
1	MATCH1_INT Same description as bit 2	0x0
0	MATCH0_INT Same description as bit 1	0x0

26.4.2 High Speed Timer Control register (HSTIM_CTRL, RW - 0x4003 8004)

Table 548. High Speed Timer Control register (HSTIM_CTRL, RW - 0x4003 8004)

Bits	Description	Reset value
7:3	Not used. Write is don't care, Read returns random value.	0x0
2	PAUSE_EN (DEBUG_EN) 0 = Timer counter continues to run if ARM enters debug mode. 1 = Timer counter is stopped when the core is in debug mode (DBGACK high).	0
1	RESET_COUNT 0 = Timer counter is not reset. (Default) 1 = Timer counter will be reset on next PERIPH_CLK edge. Software must write this bit back to low to release the reset.	0
0	COUNT_ENAB 0 = Timer Counter is stopped. (Default) 1 = Timer Counter is enabled	

26.4.3 High Speed Timer Counter Value register (HSTIM_COUNTER, RW - 0x4003 8008)

Table 549. High Speed Timer Counter Value register (HSTIM_COUNTER, RW - 0x4003 8008)

Bits	Description	Reset value
31:0	HSTIM_COUNTER A read reflects the current value of the High Speed Timer counter. A write loads a new value into the Timer counter.	0x0

26.4.4 High Speed Timer Prescale Counter Match register (HSTIM_PMATCH, RW - 0x4003 800C)

Table 550. High Speed Timer Prescale Counter Match register (HSTIM_PMATCH, RW - 0x4003 800C)

Bits	Description	Reset value
31:16	Not used. Write is don't care, Read returns random value.	0
15:0	PMATCH Holds the current match value for the prescale counter. The timer counter will be incremented every PMATCH+1 cycles of PERIPH_CLK. For example: If PERIPH_CLK is 13MHz, set the PMATCH value to 1299 (decimal) in order to increment the Timer Counter every 100 ms.	0x0

26.4.5 High Speed Timer Prescale Counter register (HSTIM_PCOUNT, RW - 0x4003 8010)

Table 551. High Speed Timer Prescale Counter register (HSTIM_PCOUNT, RW - 0x4003 8010)

Bits	Description	Reset value
31:16	Not used. Write is don't care, Read returns random value.	0
15:0	HSTIM_PCOUNT A read reflects the current value of the Prescale Counter. A write loads a new value into the counter.	0x0

26.4.6 High Speed Timer Match Control register (HSTIM_MCTRL, RW - 0x4003 8014)

Table 552. High Speed Timer Match Control register (HSTIM_MCTRL, RW - 0x4003 8014)

Bits	Description	Reset value
15:9	Not used. Write is don't care, Read returns random value.	0
8	STOP_COUNT2 0 = Disable the stop functionality on Match 2. (Default) 1 = Enable the Timer Counter to be stopped on Match 2.	0
7	RESET_COUNT2 0 = Disable reset of Timer Counter on Match 2. (Default) 1 = Enable reset of Timer Counter on Match 2.	0

Table 552. High Speed Timer Match Control register (HSTIM_MCTRL, RW - 0x4003 8014) *…continued*

Bits	Description	Reset value
6	MR2_INT 0 = Disable interrupt on the Match 2 register. (Default) 1 = Enable internal interrupt status generation on the Match 2 register	0
5	STOP_COUNT1 0 = Disable the stop functionality on Match 1. (Default) 1 = Enable the Timer Counter to be stopped on Match 1.	0
4	RESET_COUNT1 0 = Disable reset of Timer Counter on Match 1. (Default) 1 = Enable reset of Timer Counter on Match 1	0
3	MR1_INT 0 = Disable interrupt on the Match 1 register. (Default) 1 = Enable internal interrupt status generation on the Match 1 register	0
2	STOP_COUNT0 0 = Disable the stop functionality on Match 0. (Default) 1 = Enable the Timer Counter to be stopped on Match 0.	0
1	RESET_COUNT0 0 = Disable reset of Timer Counter on Match 0. (Default) 1 = Enable reset of Timer Counter on Match 0	0
0	MR0_INT 0 = Disable interrupt on the Match 0 register. (Default) 1 = Enable internal interrupt status generation on the Match 0 register	0

26.4.7 High Speed Timer Match 0 register (HSTIM_MATCH0, RW - 0x4003 8018)

Table 553. High Speed Timer Match 0 register (HSTIM_MATCH0, RW - 0x4003 8018)

Bits	Description	Reset value
31:0	MATCH0 Holds the match values for the Timer Counter. Accesses to this register have no other effect than reading the current register value or loading a new value.	0

26.4.8 High Speed Timer Match 1 register (HSTIM_MATCH1, RW - 0x4003 801C)

Table 554. High Speed Timer Match 1 register (HSTIM_MATCH1, RW - 0x4003 801C)

Bits	Description	Reset value
31:0	MATCH1 Holds the match values for the Timer Counter. Accesses to this register have no other effect than reading the current register value or loading a new value.	0

26.4.9 High Speed Timer Match 2 register (HSTIM_MATCH2, RW - 0x4003 8020)

Table 555. High Speed Timer Match 2 register (HSTIM_MATCH2, RW - 0x4003 8020)

Bits	Description	Reset value
31:0	MATCH2 Holds the match values for the Timer Counter. Accesses to this register have no other effect than reading the current register value or loading a new value.	0

26.4.10 High Speed Timer Capture Control Register (HSTIM_CCR, RW - 0x4003 8028)

Table 556. High Speed Timer Capture Control Register (HSTIM_CCR, RW - 0x4003 8028)

Bits	Description	Reset value
15:6	Not used. Write is don't care, Read returns random value.	0
5	RTC_TICK_EVENT If this bit is set, an interrupt status will be set in the HSTIM_INT register on a capture.	0
4	RTC_TICK _FALL If this bit is set, the timer counter will be loaded into the CR1 if a falling edge is detected on RTC_TICK.	0
3	RTC_TICK_RISE If this bit is set, the timer counter will be loaded into the CR1 if a rising edge is detected on RTC_TICK.	0
2	HSTIM_CAP_EVENT If this bit is set, an interrupt status will be set in the HSTIM_INT register on a capture.	0
1	HSTIM_CAP _FALL If this bit is set, the timer counter will be loaded into the CR0 if a falling edge is detected on HSTIM_CAP.	0
0	HSTIM_CAP_RISE If this bit is set, the timer counter will be loaded into the CR0 if a rising edge is detected on HSTIM_CAP	0

Remark: Selecting both falling and rising edge on a capture signal is valid.

26.4.11 High Speed Timer Capture 0 Register (HSTIM_CR0, RO - 0x4003 802C)

Table 557. High Speed Timer Capture 0 Register (HSTIM_CR0, RO - 0x4003 802C)

Bits	Description	Reset value
31:0	CAPT_VALUE Holds the captured value from the Timer Counter when rise or fall event happens on HSTIM_CAP.	0

26.4.12 High Speed Timer Capture 1 Register (HSTIM_CR1, RO - 0x4003 8030)

Table 558. High Speed Timer Capture 1 Register (HSTIM_CR1, RO - 0x4003 8030)

Bits	Description	Reset value
31:0	CAPT_VALUE Holds the captured value from the Timer Counter when rise or fall event happens on RTC_TICK.	0

26.5 Examples of timer operation

Figure 109 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 110 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.

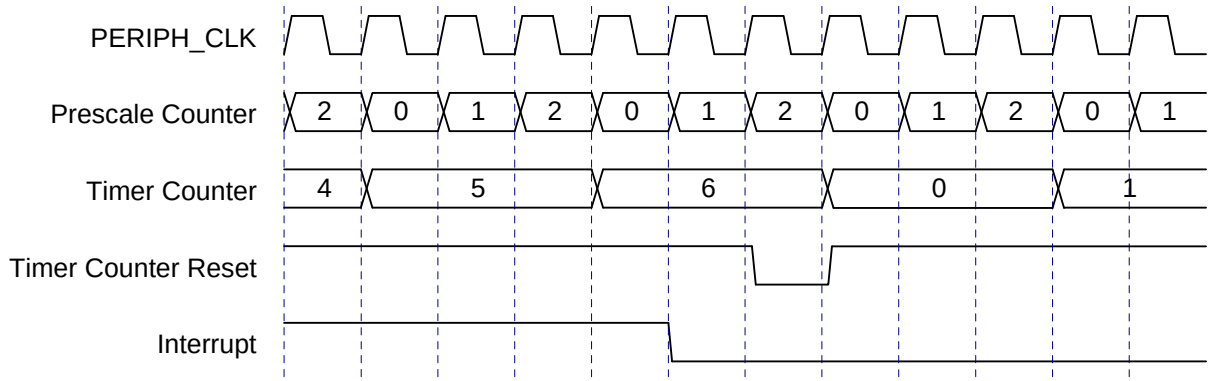


Fig 109. Timer cycle with PR=2, MRx=6, and both interrupt and reset on match enabled

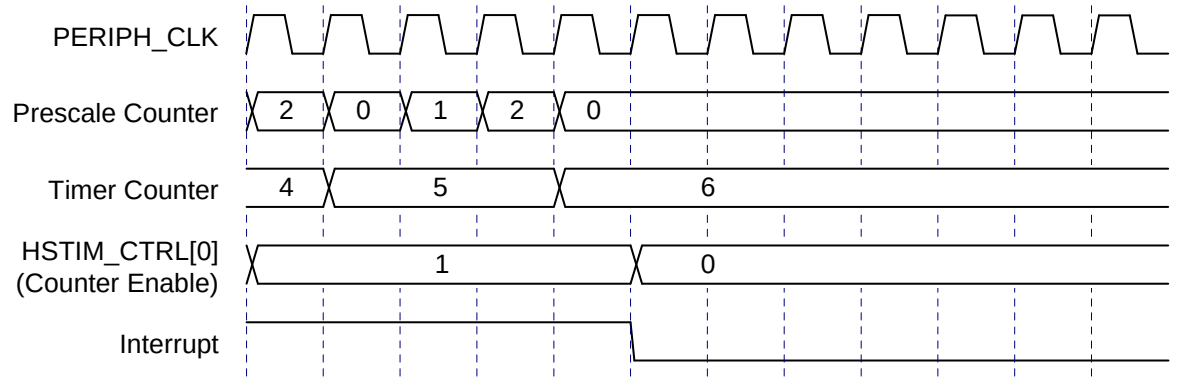


Fig 110. Timer cycle with PR=2, MRx=6, and both interrupt and stop on match enabled

27.1 Features

- 32-bit Timer/Counters.
- Counter or Timer operation.
- Two 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.
- Pause control to stop counting when core is in debug state.

27.2 Description

The Millisecond timer is clocked by the 32 kHz RTC clock. The registers are accessed on a different clock domain while the counter is counting on. This solution speeds up accesses to the Millisecond timer. Reads and writes to registers in the Millisecond timer are clocked by the HCLK. It takes a maximum of three HCLK before the writes are performed to the register.

There are two match registers comparing values against the Timer/counter. A match on one of the match registers can generate an interrupt and the Timer/counter can be set to either continue to run, stop, or be reset. The Millisecond timer has one interrupt connected to the main interrupt controller. The timer can be disabled in the MSTIM_CTRL register. A time-out is typically handled by reading the Timer/counter, adding the number of clocks for the time-out and storing the new value into one of the Match registers. The overflow in the add operation (carry) can be discarded. The millisecond timer also supports debug functionality: By setting the pause_en bit in the Timer/control register the counter will not count while the ARM core is in debug state.

The block diagram of the millisecond timer is shown below.

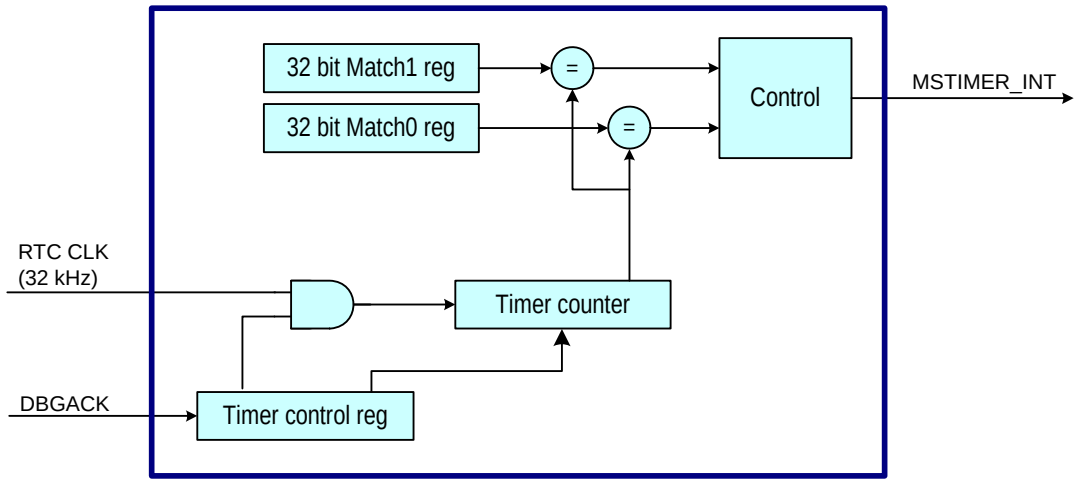


Fig 111. Millisecond timer block diagram

27.3 Register description

The Millisecond timer includes the registers shown in Table 559. Detailed descriptions of the registers follow.

Table 559. Millisecond timer registers

Address offset	Name	Description	Reset value	Type
0x4003 4000	MSTIM_INT	Millisecond timer interrupt status register	0	R/W
0x4003 4004	MSTIM_CTRL	Millisecond timer control register	0	R/W
0x4003 4008	MSTIM_COUNTER	Millisecond timer counter value register	0	R/W
0x4003 4014	MSTIM_MCTRL	Millisecond timer match control register	0	R/W
0x4003 4018	MSTIM_MATCH0	Millisecond timer match 0 register	0	R/W
0x4003 401C	MSTIM_MATCH1	Millisecond timer match 1 register	0	R/W

27.3.1 Millisecond Timer Interrupt Status register (MSTIM_INT, RW - 0x4003 4000)

Table 560. Millisecond Timer Interrupt Status register (MSTIM_INT, RW - 0x4003 4000)

Bits	Description	Reset value
7:2	Not used. Write is don't care, Read returns random value.	0
1	MATCH1_INT: Reading a 1 indicates an active MATCH 1 interrupt. Writing a 1 clears the active interrupt status. Writing 0 has no effect. Note: Remove active match status by writing a new match value before clearing the interrupt. Otherwise this a new match interrupt may be activated immediately after clearing the match interrupt since the match may still be valid.	0x0
0	MATCH0_INT See MATCH1_INT.	0x0

27.3.2 Millisecond Timer Control register (MSTIM_CTRL, RW - 0x4003 4004)

Table 561. Millisecond Timer Control register (MSTIM_CTRL, RW - 0x4003 4004)

Bits	Description	Reset value
7:3	Not used. Write is don't care, Read returns random value.	0x0
2	PAUSE_EN (DEBUG_EN) 0 = Timer counter continues to run if ARM enters debug mode. 1 = Timer counter is stopped when the core is in debug mode (DBGACK high).	0
1	RESET_COUNT 0 = Timer counter is not reset. (Default) 1 = Timer counter will be reset on next PERIPH_CLK edge. Software must write this bit back to low to release the reset.	0
0	COUNT_ENAB 0 = Timer Counter is stopped. (Default) 1 = Timer Counter is enabled	

27.3.3 Millisecond Timer Counter Value register (MSTIM_COUNTER, RW - 0x4003 4008)

Table 562. Millisecond Timer Counter Value register (MSTIM_COUNTER, RW - 0x4003 4008)

Bits	Description	Reset value
31:0	MSTIM_COUNTER A read reflects the current value of the Millisecond Timer counter. A write loads a new value into the Timer counter.	0x0

27.3.4 Millisecond Timer Match Control register (MSTIM_MCTRL, RW - 0x4003 4014)

Table 563. Millisecond Timer Match Control register (MSTIM_MCTRL, RW - 0x4003 4014)

Bits	Description	Reset value
15:6	Not used. Write is don't care, Read returns random value.	0
5	STOP_COUNT1 0 = Disable the stop functionality on Match 1. (Default) 1 = Enable the Timer Counter to be stopped on Match 1.	0
4	RESET_COUNT1 0 = Disable reset of Timer Counter on Match 1. (Default) 1 = Enable reset of Timer Counter on Match 1	0
3	MR1_INT 0 = Disable interrupt on the Match 1 register. (Default) 1 = Enable internal interrupt status generation on the Match 1 register	0

Table 563. Millisecond Timer Match Control register (MSTIM_MCTRL, RW - 0x4003 4014) *...continued*

Bits	Description	Reset value
2	STOP_COUNT0 0 = Disable the stop functionality on Match 0. (Default) 1 = Enable the Timer Counter to be stopped on Match 0.	0
1	RESET_COUNT0 0 = Disable reset of Timer Counter on Match 0. (Default) 1 = Enable reset of Timer Counter on Match 0	0
0	MR0_INT 0 = Disable interrupt on the Match 0 register. (Default) 1 = Enable internal interrupt status generation on the Match 0 register	0

27.3.5 Millisecond Timer Match 0 register (MSTIM_MATCH0, RW - 0x4003 4018)

Table 564. Millisecond Timer Match 0 register (MSTIM_MATCH0, RW - 0x4003 4018)

Bits	Description	Reset value
31:0	MATCH0 Holds the match values for the Timer Counter. Accesses to this register have no other effect than reading the current register value or loading a new value. Note: The match interrupt is generated at the end of the matching 32 kHz cycle. The delay will subsequently be up to one cycle longer than MATCH - COUNTER indicates.	0

27.3.6 Millisecond Timer Match 1 register (MSTIM_MATCH1, RW - 0x4003 401C)

Table 565. Millisecond Timer Match 1 register (MSTIM_MATCH1, RW - 0x4003 401C)

Bits	Description	Reset value
31:0	MATCH1 Holds the match value for the Timer Counter. Accesses to this register have no other effect than reading the current register value or loading a new value. Note: The match interrupt is generated at the end of the matching 32 kHz cycle. The delay will subsequently be up to one cycle longer than MATCH - COUNTER indicates.	0

28.1 Features

- Measures the passage of time in seconds.
- 32 bit up and down seconds counter
- Ultra Low Power design to support battery powered systems.
- Dedicated 32 KHz oscillator.
- Dedicated power supply pins can be connected to a battery or 1.2 V power supply.
- An ONSW output pin is included to assist in waking up when the chip has had power removed to all functions except the RTC and Battery RAM.
- Two 32 bit match registers with interrupt option.
- 128 bytes of very low voltage static RAM.
- RTC and Battery RAM power supply is isolated from the rest of the chip.

28.2 Description

28.2.1 RTC counter

The RTC is running at 32768 Hz using a very low power oscillator. The main purpose of the RTC is to clock the RTC Timers and to generate alarm interrupts which also can power up the device. The RTCCLK can also clock the 397x PLL, the Millisecond Timer, the A/D converter, the Keyboard Scanner, and the PWMs. The 32 bit RTC counters run continuously at 1 Hz and are never reset provided that the RTC has power. The RTC up-counter value represents a number of seconds elapsed since second 0, which is an application determined time. The RTC counter will reach maximum value after about 136 years. The RTC down-counter is initiated with all 1's. Software may use this timer to verify that the RTC block contains valid information. The two timers will have the same date and time, but using different algorithms for the mapping.

The suggested rule for verifying the RTC information is: Use sum of the counters equal to 0xFFFF FFFE which is different from the default values. If software finds that the two timers maps to different dates or times, the RTC should be reset by writing the RTC_CTRL[4] high and low. When the 1 Hz clock is running, the up/down counters should be stopped when writing to the counters. This will ensure that there are no clock edges during the writes. All RTC registers should be validated by software at boot time, if possible. Invalid RTC values can occur if the VDD_RTC has dropped below the minimum value.

The two 32 bit Match registers are readable and writable by the processor, and a match will result in an active interrupt provided that the interrupt is enabled in the Interrupt register. A match is true when the RTC up counter holds a value equal to the match register. Interrupts are by default disabled. In order to prevent an active RTC_ONSW signal when the RTC is powered up, the signal is gated with a pattern checker. If the RTC_KEY register does not contain the correct value, the RTC_ONSW signal will be

stopped from propagating onto the ONSW pin. The ONSW pin will only be driven as long as the match is active. After 1s with active match and the software has not accessed the RTC block, the ONSW pin will go low when the match is no longer active.

The RTC block resides in a separate voltage domain. The block is supplied via a separate supply pin with power either from the battery or a regulator. The RTC block will always be powered by a nominal voltage when there is any activity on the signals going to other blocks. This means that any ARM accesses to registers or SRAM will only take place when nominal voltage is present. No accesses are allowed to the RTC in the 0.9 V VddCore mode. However, the RTC oscillator and counter still work down to the minimum voltage. Also, the contents of the SRAM and all registers/ latches etc.are preserved down to the minimum voltage. To minimize power consumption in the RTC power domain, the RESET_N pin must be held low while power is removed from other power domains.

Note that if the RTC_KEY is correctly initiated, the RTC Timer registers are not reset by the device reset signal issued at power up. (Except for bits RTC_CTRL[7:4]). There are no register values that are set to default when the RTC voltage drops below the minimum value. This means that the default values shown in the register descriptions are valid only after the ARM core has written the RTC_CTRL[4] bit high in order to issue a hardware reset to the RTC block.

28.2.2 RTC SRAM

The RTC block also contains 32 words of very low voltage SRAM. This SRAM is able to hold its contents down to the minimum RTC operating voltage. Note that there will be no ARM accesses as long as the voltage is below normal operating conditions. The SRAM is accessible in 32-bit word only. The software should keep an inverted checksum in the SRAM for content validation. (Inverted to avoid a good checksum if all cells are zero). Care must be taken by software to not access this RAM too often in order to keep the average RTC current low enough.

28.2.3 RTC ONSW output

The ONSW output pin can be used to trigger the external power supply to turn on all the operating voltages. The source for the ONSW pin is an RTC-match event. The ONSW output goes to a positive-edge triggered power device. It is important that another external source does not hold an active high ONSW for an indefinite time. The RTC-alarm is by nature a 1s pulse. During power-up of any voltage domain, there will be no false pulses on the ONSW pin.

An RTC match can drive the ONSW pin active (high) for as long as the match lasts (1s). This may be disabled by the ONSW control register. The match status is latched so that software can find the source which generated the match and ONSW pulse even after the match has finished. The ONSW match latch is cleared by writing a 1 to RTC_INSTAT[2]. The ONSW pin can also be driven active directly by software. Software can also drive the ONSW pin high by writing a 1 to the RTC_CTRL[7] register bit provided that the RTC_KEY is correctly initiated.

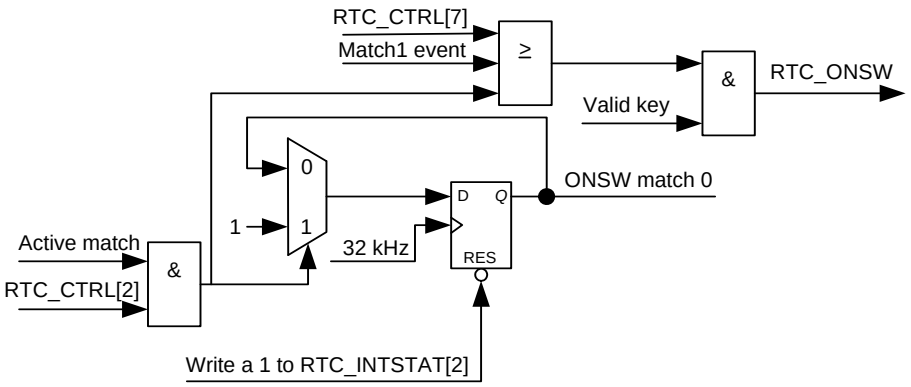


Fig 112. ONSW logic shown for Match 0 only

28.2.4 RTC oscillator

The crystal for the RTC is connected as shown in Figure 113. Capacitors are also required, as shown. Capacitor values may be found in Table 566. Values in the table do not take package capacitance into account.

Table 566. Recommended values for the RTC external 32 kHz oscillator $C_{X1/X2}$ components

Crystal load capacitance	Maximum crystal series resistance R_S	External load capacitors
11 pF	< 100 kΩ	18 pF
13 pF	< 100 kΩ	22 pF
15 pF	< 100 kΩ	27 pF

28.3 Architecture

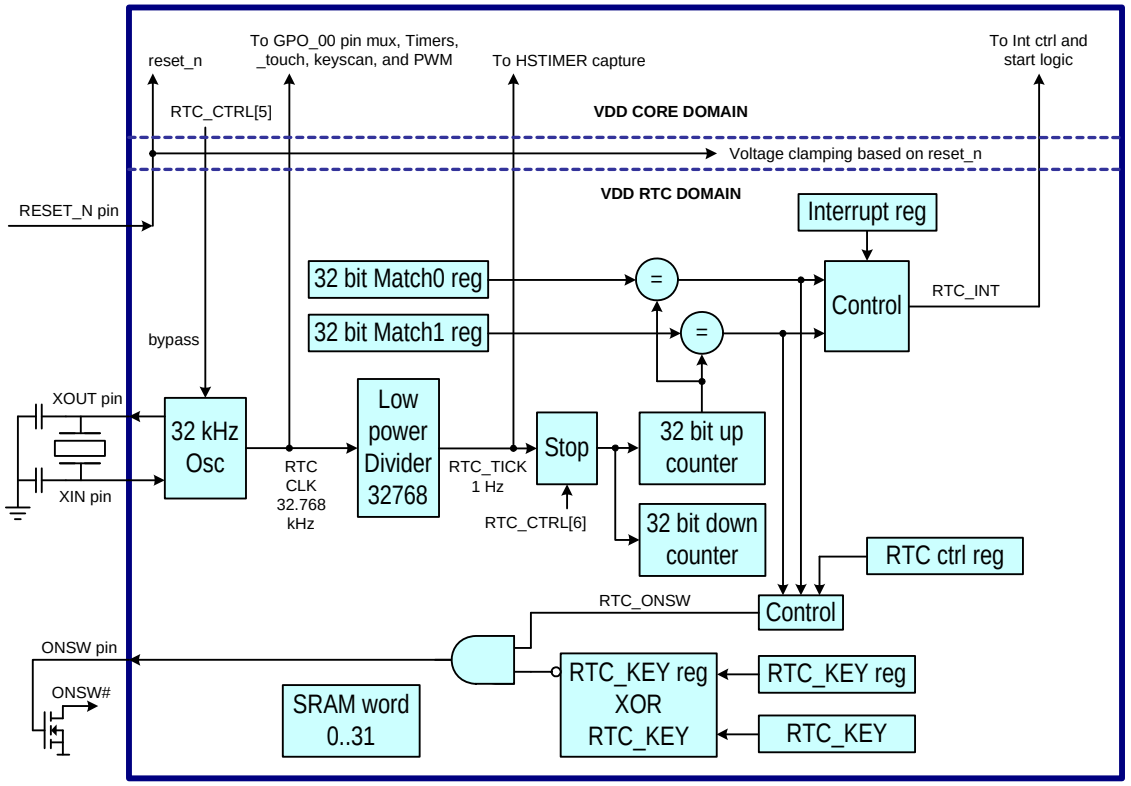


Fig 113. RTC block diagram

28.4 Register description

The Real Time Clock includes the registers shown in Table 567. Detailed descriptions of the registers follow.

Table 567. RTC registers

Address offset	Name	Description	Reset value	Type
0x4002 4000	RTC_UCOUNT	RTC up counter value register	0	R/W
0x4002 4004	RTC_DCOUNT	RTC down counter value register	0	R/W
0x4002 4008	RTC_MATCH0	RTC match 0 register	0	R/W
0x4002 400C	RTC_MATCH1	RTC match 1 register	0	R/W
0x4002 4010	RTC_CTRL	RTC control register	0	R/W
0x4002 4014	RTC_INTSTAT	RTC Interrupt status register	0	R/W
0x4002 4018	RTC_KEY	RTC Key register	0	R/W
0x4002 4080	RTC_SRAM	Battery RAM	0	R/W
…				
0x4002 40FF				

Note: The RTC registers take their default value after a software reset in RTC_CTRL[4]. There is no power-on reset signal to this block except RTC_CTRL register.

28.4.1 RTC Up Counter Value register (RTC_UCOUNT, RW - 0x4002 4000)

Table 568. RTC Up Counter Value register (RTC_UCOUNT, RW - 0x4002 4000)

Bits	Description	Reset value
31:0	A read reflects the current value of the RTC counter. A write loads a new value into the RTC counter. The counter value should never be written except when setting the correct time and date. The counter expires after about 136 years and has 1s resolution. The counter must be stopped (RTC_CTRL[6] = 1) when writing to this register. In order to read back the value just written, there must be minimum a 32 kHz clock period time delay since the write was done.	0x0

28.4.2 RTC Down Counter Value register (RTC_DCOUNT, RW - 0x4002 4004)

Table 569. RTC Down Counter Value register (RTC_DCOUNT, RW - 0x4002 4004)

Bits	Description	Reset value
31:0	A read reflects the current value of the RTC counter. A write loads a new value into the RTC counter. The counter value should never be written except when setting the correct time and date. The counter expires after about 136 years and has 1s resolution. The counter must be stopped (RTC_CTRL[6] = 1) when writing to this register. In order to read back the value just written, there must be minimum a 32 kHz clock period time delay since the write was done.	0x0

28.4.3 RTC Match 0 register (RTC_MATCH0, RW - 0x4002 4008)

Table 570. RTC Match 0 register (RTC_MATCH0, RW - 0x4002 4008)

Bits	Description	Reset value
31:0	Holds the current match value for RTC timer 0. Writing to this register with a different value than the RTC_UCOUNT holds, will set the match status inactive. Interrupt on a match can be enabled in the RTC_INT register. A match can also drive the ONSW pin active, powering on the microcontroller. This is controlled in the RTC_CTRL register. A match event can only happen on the 1 Hz edge. When setting up a match asynchronously to the RTC tick, the match will occur within a +/- 0.5s of the specified setting.	0x0

28.4.4 RTC Match 1 Register (RTC_MATCH1, RW - 0x4002 400C)

Table 571. RTC Match 1 Register (RTC_MATCH1, RW - 0x4002 400C)

Bits	Description	Reset value
31:0	Holds the current match value for RTC timer 1. See the description for RTC_MATCH0 register.	0x0

28.4.5 RTC Control register (RTC_CTRL, RW - 0x4002 4010)

Table 572. RTC Control register (RTC_CTRL, RW - 0x4002 4010)

Bits	Description	Reset value
31:11	Not used. Write is don't care, Read returns random value.	-
10	Read only: Output from the 32 kHz oscillator. A read returns the current level of the RTC_CLK.	
8:9	Not used. Write is don't care, Read returns random value.	-
7	RTC force ONSW[1] 0 = Do not force the ONSW signal. (Default). See Note below. 1 = Force ONSW high	0
6	RTC counter clock disable[1] 0 = up/down counters running. (Default) See Note below. 1 = 1 Hz clock stopped. Used when writing to the counters.	0
5	Not used. Write is don't care, Read returns random value.	-
4	Software controlled RTC reset[1] 0 = RTC running. (Default) See Note below. 1 = RTC in hardware reset. Resets all registers to default value. Note that software must write first 1, then 0. Do not attempt to set any bits in this register at the same time as releasing the reset.	0
3	Match 1 ONSW control. 0 = Match 1 is disabled from driving the ONSW pin. (Default) 1 = Match 1 will drive the ONSW pin active in 1s on a match as long as the RTC_KEY is correct.	0
2	Match 0 ONSW control. 0 = Match 0 is disabled from driving the ONSW pin. (Default) 1 = Match 0 will drive the ONSW pin active in 1s on a match as long as the RTC_KEY is correct.	0
1	Match 1 interrupt enable bit. Note that an active Match interrupt status and RTC_IRQ may still be set after writing 0 to this bit. The RTC_INTSTAT register also need clearing after setting this bit to 0. 0 = interrupt disabled (default) 1 = interrupt enabled	0
0	Match 0 interrupt enable bit. Note that an active Match interrupt status and RTC_IRQ may still be set after writing 0 to this bit. The RTC_INTSTAT register also need clearing after setting this bit to 0. 0 = interrupt disabled (default) 1 = interrupt enabled	0

[1] Note: RTC_CTRL[7:4] is reset to 0 on an active RESET_N (Device reset). These are the only register bits in the RTC block that will be reset by the device reset. The reset of the RTC is reset by RTC_CTRL[4] which is software controlled.

28.4.6 RTC Interrupt Status Register (RTC_INTSTAT, RW - 0x4002 4014)

Table 573. RTC Interrupt Status Register (RTC_INTSTAT, RW - 0x4002 4014)

Bits	Description	Reset value
31:3	Not used. Write is don't care, Read returns random value.	0x0
2	ONSW Match status Reading a 1 indicates that the ONSW latch is set from a match event. If the match is no longer active, the ONSW pin will be driven low. Writing a 1 clears the latch and ONSW will not be driven active. Writing 0 has no effect.	0
1	Match 1 interrupt status Reading a 1 indicates an active MATCH 1 interrupt. Writing a 1 clears the active interrupt status. Writing 0 has no effect. Always remove the active match condition by writing a new value to the RTC_MATCH1 register before clearing this status.	0
0	Match 0 interrupt status Reading a 1 indicates an active MATCH 0 interrupt. Writing a 1 clears the active interrupt status. Writing 0 has no effect. Always remove the active match condition by writing a new value to the RTC_MATCH0 register before clearing this status.	0

28.4.7 RTC Key Register (RTC_KEY, RW - 0x4002 4018)

Table 574. RTC Key Register (RTC_KEY, RW - 0x4002 4018)

Bits	Description	Reset value
31:0	The software must write the value 0xB5C13F27 into this register before the RTC_ONSW signal can pass onto the ONSW pin and prevent the hardware from clearing of the RTC registers by the bootloader on reset. The correct value is checked by logic in the RTC block.	0x0

28.4.8 Battery RAM (RTC_SRAM, RW - 0x4002 4080 - 40FF)

Table 575. Battery RAM (RTC_SRAM, RW - 0x4002 4080 - 40FF)

Bits	Description	Reset value
31:0	32 words of General purpose SRAM. This RAM can be treated as nonvolatile memory as long as the RTC block has valid supply voltage. See start of this chapter how to verify a valid RTC content after power up. The SRAM can only be accessed as 32 bit words. (No byte or halfword access is supported). Software needs to be careful how often this RAM is accessed because there is a maximum power consumption restriction on the external RTC power supply.	0x0

29.1 Features

- Internally resets chip if not periodically reloaded.
- Flag to indicate Watchdog reset.
- Programmable 32-bit timer.
- Can be used as a standard timer if watchdog is not used.
- Pause control to stop counting when core is in debug state.
- Programmable watchdog pulse output on RESOUT_N pin.

29.2 Description

The watchdog timer block is clocked by PERIPH_CLK, which clocks a 32 bit counter. The clock to the watchdog can be stopped using the TIMCLK_CTRL register. There is one match register value that is compared against the Timer counter. If configured for watchdog functionality, when the timer counter register matches the match register the watchdog timer will drive the match output low. The match output is gated with an enable signal from WDTIM_MCTRL[4:3] that can generate two types of reset signals. WDOG_RESET1_N that only resets the chip internally, and WDOG_RESET2 that goes through a programmable pulse generator before it goes to the external pin RESOUT_N and to the internal chip reset.

The watchdog timer can be used as a standard high speed timer when not used as watchdog. It also supports interrupt and debug functionality: Setting the pause_en bit in the WDTIM_CTRL register ensures that the counter will not count while the core is in debug state.

The block diagram of the watchdog is shown in Figure 114.

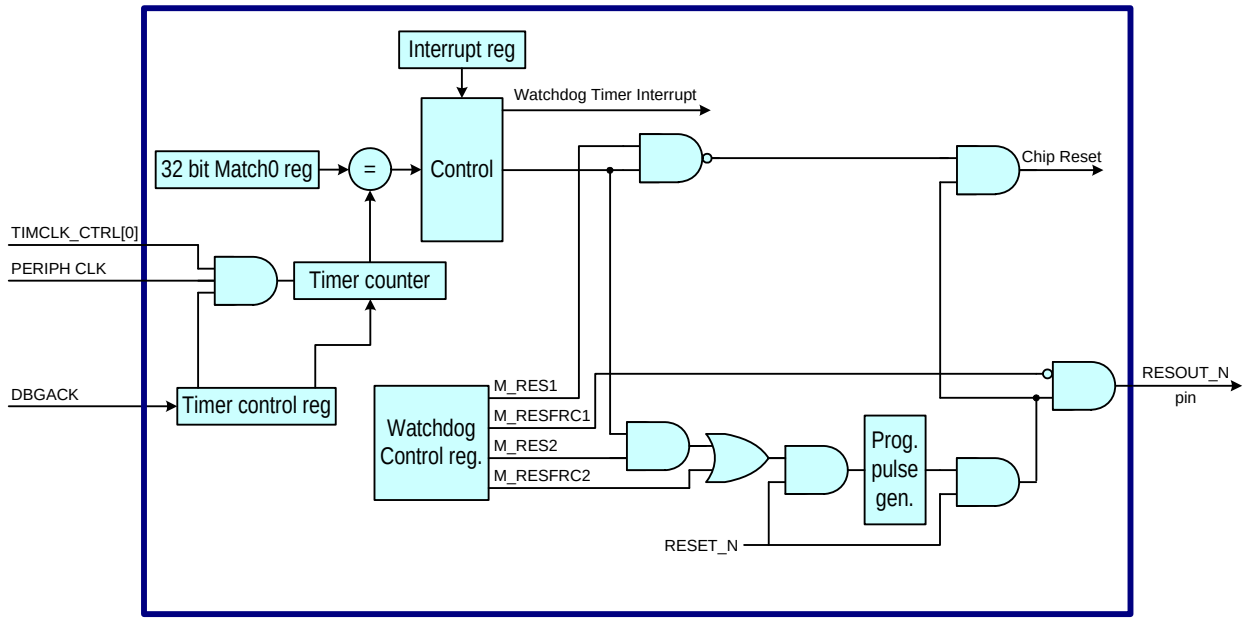


Fig 114. Watchdog timer block diagram

29.2.1 Reset examples

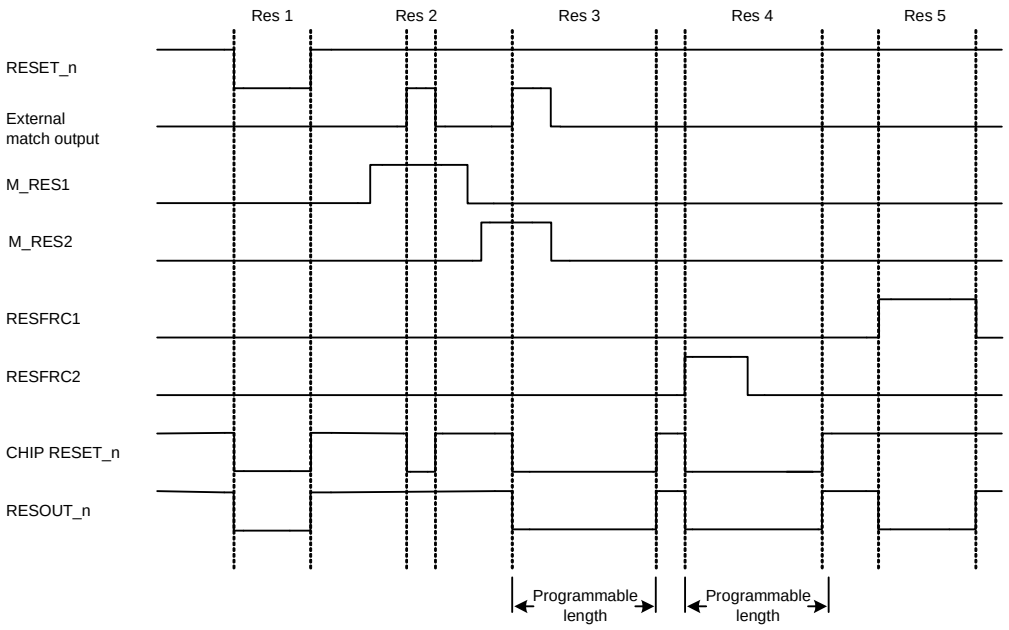


Fig 115. Reset examples

Res 1: An external reset gives an internal chip reset and an output on RESOUT_N.

Res 2: If there is an External Match output (WDOG match) and M_RES1 is High, then there will be an internal chip reset.

Res 3: If there is an External Match output (WDOG match) and M_RES2 is High, then there will be an internal chip reset and a RESOUT_N reset pulse with a programmable length.

Res 4: If RESFRC2 is set high, then there will be an internal chip reset and a RESOUT_N reset pulse with a programmable length.

Res 5 If RESFRC1 is set high, the RESOUT_N will go active (low).

29.2.2 Programmable pulse generator

The programmable pulse generator is triggered with a positive edge on the input and it generates a negative reset pulse with the length set in the WDOG_PULSE register. The pulse generator is not reset by an internal chip reset. The WDTIM_PULSE register must be programmed after reset like the rest of the watchdog.

29.2.3 WDTIM_RES register

The WDTIM_RES register bit is cleared on external reset and set on internal reset. Software can read the WDTIM_RES register to find out whether the most recent reset was due to the watchdog or external reset.

29.3 Register description

The Watchdog includes the following registers. Detailed descriptions of the registers follow. The Watchdog timer clock must be enabled in the TIMCLK_CTRL register to get read/write access to any registers.

Table 576. Watchdog timer registers

Address	Name	Description	Reset value	Type
0x4003 C000	WDTIM_INT	Watchdog timer interrupt status register	0	R/W
0x4003 C004	WDTIM_CTRL	Watchdog timer control register	0	R/W
0x4003 C008	WDTIM_COUNTER	Watchdog timer counter value register	0	R/W
0x4003 C00C	WDTIM_MCTRL	Watchdog timer match control register	0	R/W
0x4003 C010	WDTIM_MATCH0	Watchdog timer match 0 register	0	R/W
0x4003 C014	WDTIM_EMR	Watchdog timer external match control register	0	R/W
0x4003 C018	WDTIM_PULSE	Watchdog timer reset pulse length register	0	R/W
0x4003 C01C	WDTIM_RES	Watchdog timer reset source register	0	RO

29.3.1 Watchdog Timer Interrupt Status Register (WDTIM_INT, RW - 0x4003 C000)

Table 577. Watchdog Timer Interrupt Status Register (WDTIM_INT, RW - 0x4003 C000)

Bits	Description	Reset value
7:1	Not used. Write is don’t care, Read returns random value.	0
0	MATCH_INT Reading a 1 indicates an active MATCH 0 interrupt. Writing a 1 clears the active interrupt status. Writing 0 has no effect	0x0

29.3.2 Watchdog Timer Control Register (WDTIM_CTRL, RW - 0x4003 C004)

Table 578. Watchdog Timer Control Register (WDTIM_CTRL, RW - 0x4003 C004)

Bits	Description	Reset value
7:3	Not used. Write is don't care, Read returns random value.	0x0
2	PAUSE_EN (DEBUG_EN) 0 = Timer counter continues to run if ARM enters debug mode. 1 = Timer counter is stopped when the core is in debug mode (DBGACK high).	0
1	RESET_COUNT 0 = Timer counter is not reset. (Default) 1 = Timer counter will be reset on next PERIPH_CLK edge. Software must write this bit back to low to release the reset.	0
0	COUNT_ENAB 0 = Timer Counter is stopped. (Default) 1 = Timer Counter is enabled	

29.3.3 Watchdog Timer Counter Value Register (WDTIM_COUNTER, RW - 0x4003 C008)

Table 579. Watchdog Timer Counter Value Register (WDTIM_COUNTER, RW - 0x4003 C008)

Bits	Description	Reset value
31:0	WDTIM_COUNTER A read reflects the current value of the Watchdog Timer counter. A write loads a new value into the Timer counter.	0x0

29.3.4 Watchdog Timer Match Control Register (WDTIM_MCTRL, RW - 0x4003 C00C)

Table 580. Watchdog Timer Match Control Register (WDTIM_MCTRL, RW - 0x4003 C00C)

Bits	Description	Reset value
7	Not used. Write is don't care, Read returns random value.	0x0
6	RESFRC2 0 = No force. (Default) 1 = Force WDOG_RESET2 signal active	0
5	RESFRC1 0 = No force. (Default) 1 = Force RESOUT_N signal active	0
4	M_RES2 0 = Match output does not affect the WDOG_RESET2 signal. (Default) 1 = Enable Match output to generate active WDOG_RESET2 signal. This gives a chip Reset and RESOUT_N reset pulse.	0
3	M_RES1 0 = Match output does not generate device reset. (Default) 1 = Enable Match output to generate internal device reset. NB This does not generate a reset pulse on RESOUT_N.	0

Table 580. Watchdog Timer Match Control Register (WDTIM_MCTRL, RW - 0x4003 C00C) *…continued*

Bits	Description	Reset value
2	STOP_COUNT0 0 = Disable the stop functionality on Match 0. (Default) 1 = Enable the Timer Counter and Prescale counter to be stopped on Match 0.	0
1	RESET_COUNT0 0 = Disable reset of Timer Counter on Match 0. (Default) 1 = Enable reset of Timer Counter on Match 0	0
0	MR0_INT 0 = Disable interrupt on the Match 0 register. (Default) 1 = Enable internal interrupt status generation on the Match 0 register.	0

29.3.5 Watchdog Timer Match 0 Register (WDTIM_MATCH0, RW - 0x4003 C010)

Table 581. Watchdog Timer Match 0 Register (WDTIM_MATCH0, RW - 0x4003 C010)

Bits	Description	Reset value
31:0	MATCH0 Holds the match value for the Timer Counter. Accesses to this register have no other effect than reading the current register value or loading a new value.	0

29.3.6 Watchdog Timer External Match Control Register (WDTIM_EMR, RW - 0x4003 C014)

Table 582. Watchdog Timer External Match Control Register (WDTIM_EMR, RW - 0x4003 C014)

Bits	Description	Reset value
7:6	Not used. Write is don't care, Read returns random value.	0x0
5:4	MATCH_CTRL 00 = Do nothing with the match output on match. (Default) 01 = Do not use this. 10 = Set Match output high on match. Use this setting when watchdog shall generate device reset. 11 = Do not use this.	0
3:1	Not used. Write is don't care, Read returns random value.	0
0	EXT_MATCH0 Holds current value of the match output signal. Match output can be set/reset by writing directly to this bit.	0

29.3.7 Watchdog Timer Reset Pulse Length Register (WDTIM_PULSE, RW - 0x4003 C018)

Table 583. Watchdog Timer Reset Pulse Length Register (WDTIM_PULSE, RW - 0x4003 C018)

Bits	Description	Reset value
15:0	PULSE[1] This register gives the RESET pulse length. 0x0: Reset pulse length is 5 PERIPH_CLKs 0x1: Reset pulse length is 6 PERIPH_CLKs . . . . . . 0xFFFF: Reset pulse length is 65541 PERIPH_CLKs	0x0

[1] The default value is set by external RESET_N only, not internal watchdog reset.

29.3.8 Watchdog Timer Reset Source Register (WDTIM_RES, RO - 0x4003 C01C)

Table 584. Watchdog Timer Reset Source Register (WDTIM_RES, RO - 0x4003 C01C)

Bits	Description	Reset value
31:1	Not used. Write is don't care, Read returns random value.	0x0
0	0: The last reset was a RESET_N reset	0: external reset
	1: The last Reset was an internal chip reset (Watchdog reset)	1: internal reset

30.1 Basic configuration

The following registers configure The Motor Control PWM:

1. The motor controller uses Peripheral clock and is enabled in the TIMCLK_CTRL1 register Section 30.7.2.
2. Interrupts: For motor control PWM related interrupts, reference Section 30.7.5. Interrupts are enabled in the MIC Enable Register using Interrupt Enable bit 3, reference Section 5.4.1.
3. The following conditions must be true for all MCPWM signals to connect to external pins
   - TIMCLK_CTRL1[6] = 1 (enable MCPWM peripheral, select MCI1, select MCI2, select $\overline{\text{MCABORT}}$)
   - P2_MUX_DIR_STATE[30] = 0 (select MCI0)
   - P3_MUX_STATE[10] = 1 (select MCOB2)
   - P3_MUX_STATE[12] = 1 (select MCOA2)
   - P3_MUX_STATE[13] = 1 (select MCOB1)
   - P3_MUX_STATE[15] = 1 (select MCOA1)
   - P3_MUX_STATE[16] = 1 (select MCOB0)
   - P3_MUX_STATE[18] = 1 (select MCOA0)

30.2 Introduction

The Motor Control PWM (MCPWM) includes features that support three-phase AC and DC motor control applications. Additionally, the MCPWM is highly configurable for many other applications that need generalized timing, capture, and comparison.

30.3 Description

The MCPWM contains three independent channels, each channel includes:

- a 32-bit Timer (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 feedback capture register (CAP)
- Two modulated outputs (MCOA and MCOB) with opposite polarities
- a period interrupt, a pulse-width interrupt, and a capture interrupt

30.4 Pin description

Table 585 lists the MCPWM pins.

Table 585. Pin summary

Pin	Alternate Pin Name[1]	Notes	Type	Description
MC0A	MCOA0		O	Channel 0, ouput A.
MC0B	MCOB0		O	Channel 0, ouput B.
MC1A	MCOA1		O	Channel 1, ouput A.
MC1B	MCOB1		O	Channel 1, ouput B.
MC2A	MCOA2		O	Channel 2, ouput A.
MC2B	MCOB2		O	Channel 2, ouput B.
$\overline{\text{MCABORT}}$	$\overline{\text{MCABORT}}$	[2]	I	LOW-active Fast Abort
MCFB0	MCI0	[3]	I	Channel 0, input.
MCFB1	MCI1	[3]	I	Channel 1, input.
MCFB2	MCI2	[3]	I	Channel 2, input.

[1] Pin names are referenced in this chapter using this naming convention.

[2] The global MCABORT_N input forces all of channels into “passive” state and asserts an interrupt.

[3] Input pins MCFB0, MCFB1, MCFB2 trigger TIM capture events.

30.5 Block Diagram

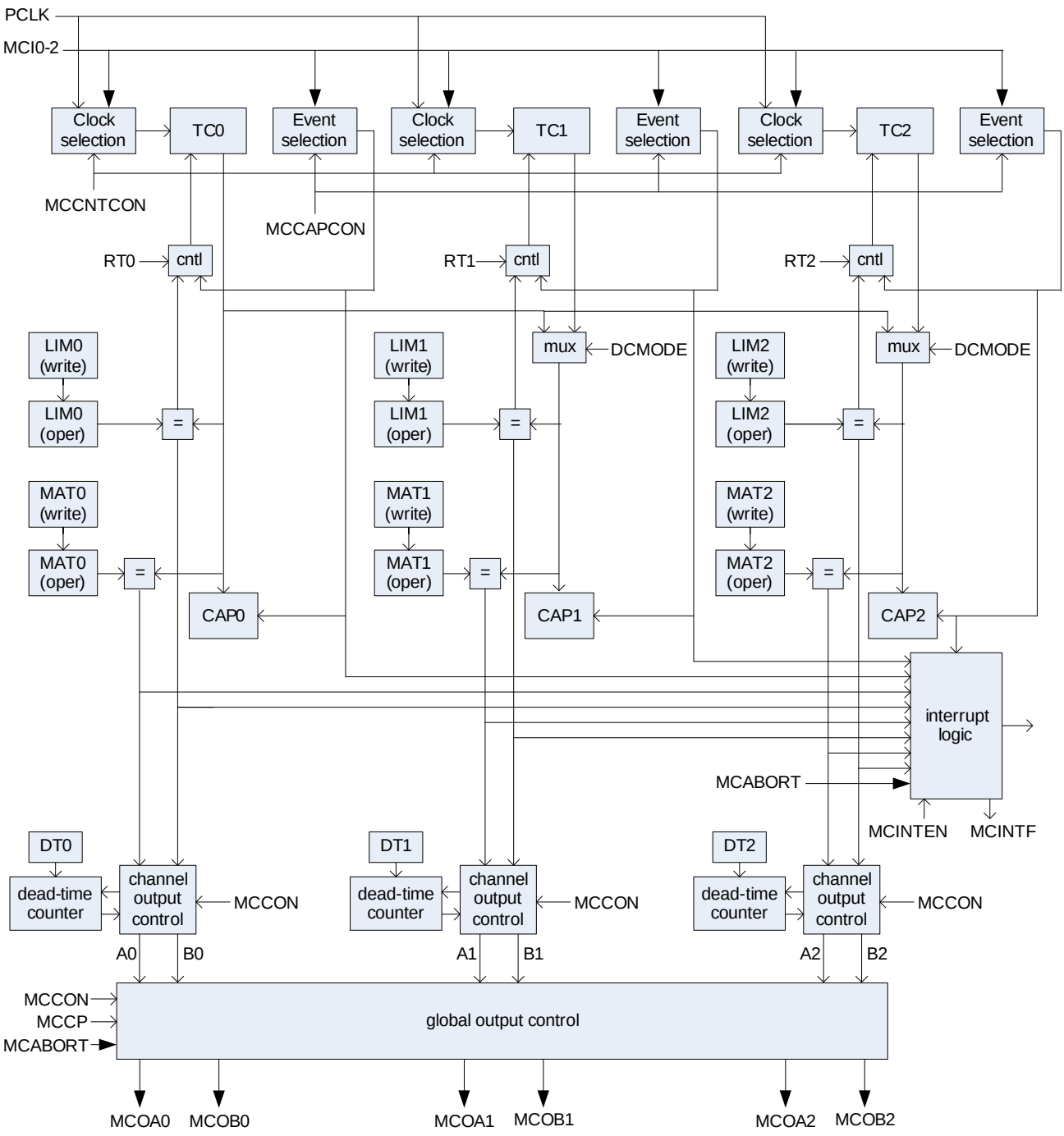


Fig 116. MCPWM Block Diagram

30.6 General Operation

Section 30.8 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 (TC) register that is incremented by a processor clock (PCLK).

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 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.

30.7 Register description

30.7.1 Introduction

The two Control registers (MCCON, MCCAPCON) use separate read, set, and clear register addresses for configuration. The Interrupt registers (MCINTEN, MCINTFLAG) also use separate read, set, and clear register addresses for configuration. For example, reading the control register (e.g. MCCON) yields the status of each control bit. Writing a one to the register’s set address (e.g. MCCON_SET) sets this register bit, writing a one to the register’s clear address (e.g. MCCON_CLR) clears this register bit.

The Capture registers (MCCAP) are read-only, and the write-only MCCAP_CLR address clears one or more of MCCAP registers.

All the other MCPWM registers (MCTC, MCLIM, MCMAT, MCDEADTIME, and MCCP) are normal read-write registers.

Table 586. Pulse Width Modulator (PWM) register map

Name	Description	Access	Reset value	Address
TIMCLK_CTRL1	Motor Controller clock control	R/W	0x0	0x4000 40C0
MCCON	PWM Control read address	RO	0	0x400E 8000
MCCON_SET	PWM Control set address	WO	-	0x400E 8004
MCCON_CLR	PWM Control clear address	WO	-	0x400E 8008
MCCAPCON	Capture Control read address	RO	0	0x400E 800C
MCCAPCON_SET	Capture Control set address	WO	-	0x400E 8010
MCCAPCON_CLR	Capture Control clear address	WO	-	0x400E 8014
MCTC0	Timer Counter register, channel 0	R/W	0	0x400E 8018
MCTC1	Timer Counter register, channel 1	R/W	0	0x400E 801C
MCTC2	Timer Counter register, channel 2	R/W	0	0x400E 8020
MCLIM0	Limit register, channel 0	R/W	0xFFFF FFFF	0x400E 8024
MCLIM1	Limit register, channel 1	R/W	0xFFFF FFFF	0x400E 8028
MCLIM2	Limit register, channel 2	R/W	0xFFFF FFFF	0x400E 802C
MCMAT0	Match register, channel 0	R/W	0xFFFF FFFF	0x400E 8030
MCMAT1	Match register, channel 1	R/W	0xFFFF FFFF	0x400E 8034
MCMAT2	Match register, channel 2	R/W	0xFFFF FFFF	0x400E 8038
MCDT	Dead time register	R/W	0x3FFF FFFF	0x400E 803C
MCCP	Communication Pattern register	R/W	0	0x400E 8040
MCCAP0	Capture register, channel 0	RO	0	0x400E 8044
MCCAP1	Capture register, channel 1	RO	0	0x400E 8048
MCCAP2	Capture register, channel 2	RO	0	0x400E 804C
MCINTEN	Interrupt Enable read address	RO	0	0x400E 8050
MCINTEN_SET	Interrupt Enable set address	WO	-	0x400E 8054
MCINTEN_CLR	Interrupt Enable clear address	WO	-	0x400E 8058
MCINTFLAG	Interrupt flags read address	RO	0	0x400E 8068
MCINTFLAG_SET	Interrupt flags set address	WO	-	0x400E 806C
MCINTFLAG_CLR	Interrupt flags clear address	WO	-	0x400E 8070
MCCAP_CLR	Capture clear address	WO	-	0x400E 8074

30.7.2 Timer Clock Control1 register (TIMCLK_CTRL1 - 0x4000 40C0)

The TIMCLK_CTRL1 register enables or disables the clocks to several peripheral device blocks: The Motor controller PWM and Timer[5,4, 3,2,1,0] blocks. This register is shown here for reference, because bit 6 enables the clock to the Motor control block.

Table 587. Timer Clock Control register (TIMCLK_CTRL1 - 0x4000 40C0)

Bit	Function	Reset value
6	Motor Control clock enable control. 0 = Disable clock. 1 = Enable clock.	0
5	Timer 3 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
4	Timer 2 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
3	Timer 1 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
2	Timer 0 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
1	Timer 5 clock enable control. 0 = Disable clock. 1 = Enable clock.	0
0	Timer 4 clock enable control. 0 = Disable clock. 1 = Enable clock.	0

30.7.3 MCPWM Control registers

30.7.3.1 MCPWM control register read address (MCCON - 0x400E 8000)

The MCCON register controls the operation of all channels of the PWM.

The MCCON register read address provides read-only status information. It shows the settings for general and motor specific applications of the PWM. The bits in this register are set and cleared by writing ones to the corresponding bits at register addresses MCCON_SET and MCCON_CLR.

Table 588. MCPWM control register read address (MCCON - 0x400E 8000) bit description

Bit	Symbol	Description	Reset value
31	DCMODE	3-phase DC mode select (see Section 30.8.5). 0 = 3-phase DC mode off: PWM channels are independent (unless bit ACMODE = 1) 1 = 3-phase DC mode on: The MC0A output is routed through the MCCP (i.e. a mask) register to all six PWM outputs.	0
30	ACMODE	3-phase AC mode select (see Section 30.8.6). 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.	0
29	INVBDC	Invert MCOB outputs for channels 0 to 2. This bit is used in 3-phase DC mode only. 0 = MCOB outputs not inverted. 1 = MCOB outputs inverted.	
28:21	-	Reserved.	

Table 588. MCPWM control register read address (MCCON - 0x400E 8000) bit description

Bit	Symbol	Description	Reset value
20	DISUP2	Enable/disable updates of functional registers, channel 2 (see Section 30.8.2). 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.	0
19	DTE2	Dead-time, channel 2 0 = Dead-time disabled. 1 = Dead-time enabled.	0
18	POLA2	Selects polarity of the MCOA2 and MCOB2 pins. 0 = Passive state is LOW, active state is HIGH. 1 = Passive state is HIGH, active state is LOW.	0
17	CENTER2	Edge/center aligned operation, channel 2. 0 = Edge-aligned. 1 = Center-aligned	0
16	RUN2	Stops/starts the timer, channel 2. 0 = Stop. 1 = Run.	0
15:13	-	Reserved.	0
12	DISUP1	Enable/disable updates of functional registers, channel 1 (see Section 30.8.2, Section 30.7.7.1). 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.	0
11	DTE1	Controls the dead-time feature for channel 1. 0 = Dead-time disabled. 1 = Dead-time enabled.	0
10	POLA1	Selects polarity of the MCOA1 and MCOB1 pins. 0 = Passive state is LOW, active state is HIGH. 1 = Passive state is HIGH, active state is LOW.	0
9	CENTER1	Edge/center aligned operation, channel 1. 0 = Edge-aligned. 1 = Center-aligned	0
8	RUN1	Stops/starts the timer, channel 1. 0 = Stop. 1 = Run.	0
7:5	-	Reserved.	
4	DISUP0	Enable/disable updates of functional registers, channel 0 (see Section 30.8.2, Section 30.7.7.1). 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.	0
3	DTE0	Controls the dead-time feature for channel 0. 0 = Dead-time disabled. 1 = Dead-time enabled.	0

Table 588. MCPWM control register read address (MCCON - 0x400E 8000) bit description

Bit	Symbol	Description	Reset value
2	POLA0	Selects polarity of the MCOA0 and MCOB0 pins. 0 = Passive state is LOW, active state is HIGH. 1 = Passive state is HIGH, active state is LOW.	0
1	CENTER0	Edge/center aligned operation, channel 0. 0 = Edge-aligned. 1 = Center-aligned	0
0	RUN0	Stops/starts the timer, channel 0. 0 = Stop. 1 = Run.	0

30.7.3.2 MCPWM control register set address (MCCON_SET - 0x400E 8004)

Writing a one to the bits at this register address sets the corresponding bits in the MCCON register. This register address is write only.

Table 589. PWM control register set address (MCCON_SET - 0x400E 8004) bit description

Bit	Description
31:0	Writing a one to any of the bits at the MCCON_SET register address sets the corresponding bit in the MCCON register. See Table 588 for bit allocation.

30.7.3.3 MCPWM control register clear address (MCCON_CLR - 0x400E 8008)

Writing a one to the bits at this register address clears the corresponding bits in the MCCON register. This register is write-only.

Table 590. MCPWM control clear register address (MCCON_CLR - address 0x400E 8008) bit description

Bit	Description
31:0	Writing a one to any of the bits at the MCCON_CLR register address clears the corresponding bit in the MCCON register. See Table 588 for bit allocation.

30.7.4 MCPWM Capture Control register

30.7.4.1 MCPWM Capture control register read address (MCCAPCON - 0x400E 800C)

The MCCAPCON register controls the detection of events for all MCPWM input channels. Any of the three MCPWM inputs can be used to trigger a capture event on any or all of the three channels.

The MCCAPCON register read address is read-only, but the underlying register can be modified by writing to register addresses MCCAPCON_SET and MCCAPCON_CLR.

Table 591. MCPWM Capture control register read address (MCCAPCON - read address 0x400E 800C) bit description

Bit	Symbol	Description	Reset Value
31:24	-	Reserved.	-
23	HNFCAP2	Enable hardware noise filter. Capture events on channel 2 are delayed as described in Section 30.8.4 1 = Hardware noise filter enabled. 0 = Hardware noise filter disabled.	0
22	HNFCAP1	Enable hardware noise filter. Capture events on channel 1 are delayed as described in Section 30.8.4 1 = Hardware noise filter enabled. 0 = Hardware noise filter disabled.	0
21	HNFCAP0	Enable hardware noise filter. Capture events on channel 0 are delayed as described in Section 30.8.4. 1 = Hardware noise filter enabled. 0 = Hardware noise filter disabled.	0
20	RT2	Reset MCTC2 register for any enabled capture event on channel 2. 1 = Timer resets on enabled capture event. 0 = Timer not affected by capture event.	0
19	RT1	Reset MCTC1 register for any enabled capture event on channel 1. 1 = Timer resets on enabled capture event. 0 = Timer not affected by capture event.	0
18	RT0	Reset MCTC0 register for any enabled capture event on channel 0. 1 = Timer resets on enabled capture event. 0 = Timer not affected by capture event.	0
17	CAP2MCI2_FE	Enable capture event on channel 2 on the falling edge of the MCI2 input. 1 = Enabled 0 = Disabled	0
16	CAP2MCI2_RE	Enable capture event on channel 2 on the rising edge of the MCI2 input. 1 = Enabled 0 = Disabled	0
15	CAP2MCI1_FE	Enable capture event on channel 2 on the falling edge of the MCI1 input. 1 = Enabled 0 = Disabled	0
14	CAP2MCI1_RE	Enable capture event on channel 2 on the rising edge of the MCI1 input. 1 = Enabled 0 = Disabled	0
13	CAP2MCI0_FE	Enable capture event on channel 2 on the falling edge of the MCI0 input. 1 = Enabled 0 = Disabled	0
12	CAP2MCI0_RE	Enable capture event on channel 2 on the rising edge of the MCI0 input. 1 = Enabled 0 = Disabled	0
11	CAP1MCI2_FE	Enable capture event on channel 1 on the falling edge of the MCI2 input. 1 = Enabled 0 = Disabled	0
10	CAP1MCI2_RE	Enable capture event on channel 1 on the rising edge of the MCI2 input. 1 = Enabled 0 = Disabled	0
9	CAP1MCI1_FE	Enable capture event on channel 1 on the falling edge of the MCI1 input. 1 = Enabled 0 = Disabled	0

Table 591. MCPWM Capture control register read address (MCCAPCON - read address 0x400E 800C) bit description

Bit	Symbol	Description	Reset Value
8	CAP1MCI1_RE	Enable capture event on channel 1 on the rising edge of the MCI1 input. 1 = Enabled 0 = Disabled	0
7	CAP1MCI0_FE	Enable capture event on channel 1 on the falling edge of the MCI0 input. 1 = Enabled 0 = Disabled	0
6	CAP1MCI0_RE	Enable capture event on channel 1 on the rising edge of the MCI0 input. 1 = Enabled 0 = Disabled	0
5	CAP0MCI2_FE	Enable capture event on channel 0 on the falling edge of the MCI2 input. 1 = Enabled 0 = Disabled	0
4	CAP0MCI2_RE	Enable capture event on channel 0 on the rising edge of the MCI2 input. 1 = Enabled 0 = Disabled	0
3	CAP0MCI1_FE	Enable capture event on channel 0 on the falling edge of the MCI1 input. 1 = Enabled 0 = Disabled	0
2	CAP0MCI1_RE	Enable capture event on channel 0 on the rising edge of the MCI1 input. 1 = Enabled 0 = Disabled	0
1	CAP0MCI0_FE	Enable capture event on channel 0 on the falling edge of the MCI0 input. 1 = Enabled 0 = Disabled	0
0	CAP0MCI0_RE	Enable capture event on channel 0 on the rising edge of the MCI0 input. 1 = Enabled 0 = Disabled	0

30.7.4.2 MCPWM Capture control register set address (MCFBCON_SET - 0x400E 8010)

Writing a one to the bits at the MCFBCON_SET register address sets the corresponding bits in the MCFBCON register. The bit allocation is identical to the MCFBCON register. This register is WO.

Table 592. MCPWM Capture control register set address (MCFBCON_SET - 0x400E 8010) bit description

Bit	Description
31:0	Writing a one to any of the bits sets the corresponding bit in the MCFBCON register. See Table 591 for bit allocation.

30.7.4.3 MCPWM Capture control register clear address (MCFBCON_CLR - 0x400E 8014)

Writing a one to the bits at the MCFBCON_SET register address clears the corresponding bits in the MCFBCON register. The bit allocation is identical to the MCFBCON register. This register is write-only.

Table 593. MCPWM Capture control register clear address (MCFBCON_CLR - 0x400E 8014) bit description

Bit	Description
31:0	Writing a one to any of the bits clears the corresponding bit in the MCFBCON register. See Table 591 for bit allocation.

30.7.5 MCPWM Interrupt registers

The Motor Control PWM module includes the following interrupt sources:

Table 594. Motor Control PWM interrupts

Symbol	Description
ILIM[2:0]	Limit interrupt for channels 0 to 2.
IMAT[2:0]	Match interrupt for channels 0 to 2.
ICAP[2:0]	Capture interrupt for channels 0 to 2.
ABORT	Fast abort interrupt

All MCPWM interrupt related registers contain one bit for each source as shown in Table 595.

Table 595. Interrupt sources bit allocation table

Bit	Name	Description
0	ILIM0	Limit interrupt for channel 0.
1	IMAT0	Match interrupt for channel 0.
2	ICAP0	Capture interrupt for channel 0.
3	Reserved	Reserved, do not modify.
4	ILIM1	Limit interrupt for channels 1.
5	IMAT1	Match interrupt for channels 1.
6	ICAP1	Capture interrupt for channels 1.
7	Reserved	Reserved, do not modify.
8	ILIM2	Limit interrupt for channels 2.
9	IMAT2	Match interrupt for channels 2.
10	ICAP2	Capture interrupt for channels 2.
14:11	Reserved	Reserved, do not modify.
15	ABORT	Fast abort interrupt
31:16	Reserved	Reserved, do not modify.

30.7.5.1 MCPWM interrupt enable register read address (MCINTEN - 0x400E 8050)

The MCINTEN register read address provides the enabled/disabled status of each PWM interrupt. This register address is a read-only. To enable or disable the interrupts, use the MCINTEN_SET and the MCINTEN_CLR register addresses.

Table 596. MCPWM interrupt enable register read address (MCINTEN - 0x400E 8050) bit description

Bit	Description	Reset value
31:0	See Table 595 for bit allocation. 1 = Interrupt is enabled. 0 = Interrupt is disabled.	0

30.7.5.2 MCPWM interrupt enable register set address (MCINTEN_SET - 0x400E 8054)

The MCINTEN_SET register address sets the bits in the MCINTEN register. Writing a one to any of the bits at this register address enables the corresponding interrupt. This register address is write-only.

Table 597. MCPWM interrupt enable register set address (MCINTEN_SET - 0x400E 8054) bit description

Bit	Description
31:0	Writing a one to any bit enables the corresponding interrupt. See Table 595 for bit allocation.

30.7.5.3 MCPWM interrupt enable register clear address (MCINTEN_CLR - 0x400E 8058)

The MCINTEN_CLR register address clears the bits in the MCINTEN register. Writing a one to any of the bits at this register address disables the corresponding interrupt. This register address is write-only.

Table 598. MCPWM interrupt enable register clear address (MCINTEN_CLR - 0x400E 8058) bit description

Bit	Description
31:0	Writing a one to any bit disables (clears) the corresponding interrupt. See Table 595 for bit allocation.

30.7.5.4 MCPWM interrupt flags register read address (MCINTFLAG - 0x400E 8068)

The MCINTFLAG register includes all MCPWM interrupt flags, which are set when the

corresponding hardware event occurs, or when a one is written to the corresponding bit at the MCINTFLAG_SET register address. When corresponding bits in this register and MCINTEN are both 1, the MCPWM asserts its interrupt request to the Interrupt Controller module.This register address is read-only. To set or clear the register bits in the MCINTFLAG register use the MCINTFLAG_SET and the MCINTFLAG_CLR register addresses.

Table 599. MCPWM interrupt flags register read address (MCINTFLAG - 0x400E 8068) bit description

Bit	Description	Reset value
31:0	Reading this address provides the status of each interrupt flag. See Table 595 for bit allocation. 1 = Interrupt flag set. 0 = Interrupt flag cleared.	0

30.7.5.5 MCPWM interrupt flags register set address (MCINTFLAG_SET - 0x400E 806C)

The MCINTFLAG_SET register address sets the bits in the MCINTFLAG register. Writing a one to any of the bits at this register address sets the corresponding interrupt flag.This register address is write-only.

Table 600. MCPWM interrupt flags register set address (MCINTFLAG_SET - 0x400E 806C) bit description

Bit	Description
31:0	See Table 595 for bit allocation. Writing a one to any bit sets the corresponding interrupt flag.

30.7.5.6 MCPWM interrupt flags register clear address (MCINTFLAG_CLR - 0x400E 8070)

The MCINTFLAG_CLR register address clears the bits in the MCINTFLAG register. Writing a one to any of the bits at this register address clears the corresponding interrupt flag. This register address is write-only.

Table 601. MCPWM interrupt flags register clear address (MCINTFLAG_CLR - 0x400E 8070) bit description

Bit	Description
31:0	Writing a one to any bit clears the corresponding interrupt flag. See Table 595 for bit allocation.

30.7.6 MCPWM Timer value registers 0-2 (MCTC0 - 0x400E 8018, MCTC1 - 0x400E 801C, MCTC2 - 0x400E 8020)

These registers hold the current values of the 32-bit timers for channels 0-2. Each

value is incremented on every PCLK. The timer counts up from 0 until it reaches the value in its corresponding MCLIM register (or is stopped by writing to MCCON_CLR). A TC register can be read at any time, but can only be written to when its channel is stopped.

Table 602. MCPWM Timer value registers 0-2 (MCTC0 - 0x400E 8018, MCTC1 - 0x400E 801C, MCTC2 - 0x400E 8020) bit description

Bit	Symbol	Description	Reset value
31:0	MCTC0-2	Timer value for channel 0, 1, 2	0x0000 0000

30.7.7 MCPWM Limit value registers 0-2 (MCLIM0 - 0x400E 8024, MCLIM1 - 0x400E 8028, MCLIM2 - 0x400E 802C)

These registers hold the limiting values for timers 0-2. When a timer 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 MCCON 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 MCCON 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.

Table 603. MCPWM Limit value registers 0-2 (MCLIM0 - 0x400E 8024, MCLIM1 - 0x400E 8028, MCLIM2 - 0x400E 802C) bit description

Bit	Symbol	Description	Reset value
31:0	MCLIM0-2	Limit value for channel 0, 1, 2	0xFFFF FFFF

30.7.7.1 Match and Limit shadow write and operating registers

Writing to either a Limit or a Match register loads a shadow "write" register, and if the channel is stopped it also loads a shadow "operating" register that is compared to the TC. If the channel is running and its "disable update" bit in MCCON is 0, the operating registers are loaded from the write registers as follows:

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 MCLIM register address always returns the operating value.

Simultaneous update of the Limit and Match registers can be achieved by writing the MCLIMx and MCMATx registers and then clearing the DISUPx bits in the MCCON register (see Table 590). The simultaneous update will occur at the beginning of the next PWM cycle (also see Section 30.8.2).

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. You can consider the Limit register to be a "Period register" and the Match register to be a "Pulse Width register".

30.7.8 MCPWM Match value registers 0-2 (MCMAT0 - 0x400E 8030, MCMAT1 - 0x400E 8034, MCMAT2 - 0x400E 8038)

These registers also have "write" and "operating" shadow registers as described above for the Limit registers. The operating registers are also compared to the channels' TCs. See Section 30.7.7.1 for details about 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 604. MCPWM match value registers 0-2 (MCMAT0 - 0x400E 8030, MCMAT1 - 0x400E 8034, MCMAT2 - 0x400E 8038) bit description

Bit	Symbol	Description	Reset value
31:0	MCMAT0-2	Pulse width value for channel 0, 1, 2	0xFFFF FFFF

30.7.8.1 Match register in Edge-Aligned mode

If the channel's CENTER bit in MCCON 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 MCCON 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.

30.7.8.2 Match register in Center-Aligned mode

If the channel's CENTER bit in MCCON 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 MCCON 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.

30.7.8.3 0 and 100% duty cycle

To lock a channel's MCO outputs at the state "B active, A passive", simply write its Match register with a higher value than you write to its Limit register. The match never occurs.

There are lock the channel's outputs in the opposite state:

1. Write 0 to the Match register (this procedure has to be verified as of Oct 27 2008)
2. Write MAT higher than LIM as described above, and complement the POLA bit.

30.7.9 MCPWM Dead-time register (MCDT - 0x400E 803C)

This register holds the dead-time value for channels 0 to 2. If a channel's DTE bit in MCCON 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's other output changes 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, program the dead-time register 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 605. MCPWM Dead-time register (MCDT - 0x400E 803C) bit description

Bit	Symbol	Description	Reset value
31:30	-	reserved	
29:20	DT2	Dead time for channel 2.[2]	0x3FF
19:10	DT1	Dead time for channel 1.[2]	0x3FF
9:0	DT0	Dead time for channel 0.[1]	0x3FF

[1] In ACMODE = 1 (enable AC-mode), this field controls the dead time for all three channels.

[2] If ACMODE = 0.

30.7.10 MCPWM Communication pattern register (MCCP - 0x400E 8040)

This register is used in DC mode only. he 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 shadow “write” and “operational” versions. see Section 30.7.7.1 and Section 30.8.2 for additional details.

Table 606. MCPWM Communication pattern register (MCCP - address 0x400E 8040) bit description

Bit	Symbol	Description	Reset value
31:6	-	Reserved.	
5	CCPB2	0 = MCO2B off. 1 = MCO2B tracks internal MCOA0.	0
4	CCPA2	0 = MCO2A off. 1 = MCO2A tracks internal MCOA0.	0
3	CCPB1	0 = MCOB1 off. 1 = MCO1B tracks internal MCOA0.	0
2	CCPA1	0 = MCOA1 off. 1 = MCO1A tracks internal MCOA0.	0
1	CCPB0	0 = MCOB0 off. 1 = MCO0B tracks internal MCOA0.	0
0	CCPA0	0 = MCO0A off. 1 = MCO0A active.	0

30.7.11 MCPWM Capture registers

30.7.11.1 MCPWM Capture register read addresses (MCCAP0 - 0x400E 8044, MCCAP1 - 0x400E 8048, MCCAP2 - 0x400E 804C)

The MCCAPCON register (Table 591) allows software to select any edge(s) on any of the MCPWM inputs 0-2 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 register addresses are read-only, but the underlying registers can be cleared by writing to the CAP_CLR address

Table 607. MCPWM Capture register read addresses (MCCAP0 - 0x400E 8044, MCCAP1 - 0x400E 8048, MCCAP2 - 0x400E 804C) bit description

Bit	Symbol	Description	Reset value
31:0	MCCAP0-2	TC value at a capture event for channel 0, 1, 2.	0x0000 0000

30.7.11.2 MCPWM Capture register clear address (MCCAP_CLR - 0x400E 8074)

Writing a one to the bits in this register clears the corresponding MCCAP register. This register is write-only.

Table 608. MCPWM Capture register clear address (MCCAP_CLR - 0x400E 8074) bit description

Bit	Symbol	Description
31:3	-	Reserved
2	CAP_CLR2	Writing a 1 to this bit clears the MCCAP2 register.
1	CAP_CLR1	Writing a 1 to this bit clears the MCCAP1 register.
0	CAP_CLR0	Writing a 1 to this bit clears the MCCAP0 register.

30.8 MCPWM operation

30.8.1 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 0V and 3.3V is programmable.

The MCPWM can perform edge-aligned and center-aligned pulse-width modulation.

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. You can consider the Limit register to be a "Period register" and the Match register to be a "Pulse Width register".

30.8.1.1 Edge-aligned PWM without dead-time

As shown in Figure 117, in this mode the timer TC counts up from 0 to the value in the LIM register. The MCO state is passive until the TC matches the Match register, at which point the MCO state changes to active. When the TC matches the Limit register, the MCO state changes back to passive, and the TC is reset and starts counting up again.

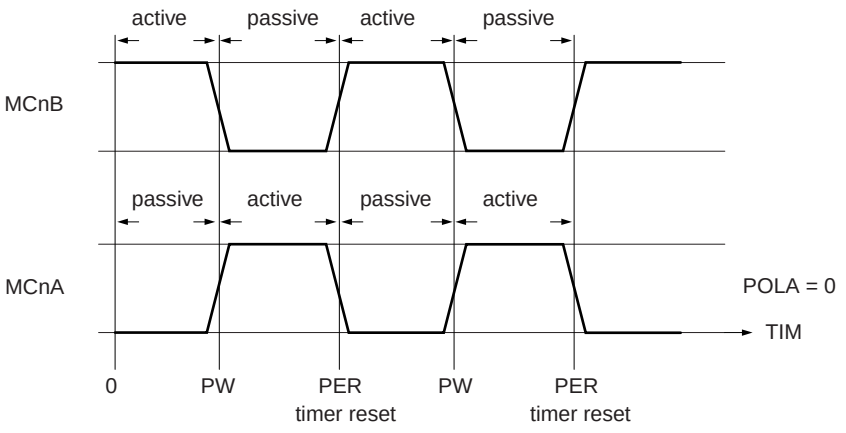


Fig 117. Center-aligned PWM waveform without dead time, POLA = 0

30.8.1.2 Center-aligned PWM without dead-time

As shown in Figure 118, 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. While the timer counts up, the MCO state is passive until the TC matches the Match register, at which point the MCO state changes to active. When the TC matches the Limit register it starts counting down. When the TC matches the Match register on the way down, the MCO state changes back to passive.

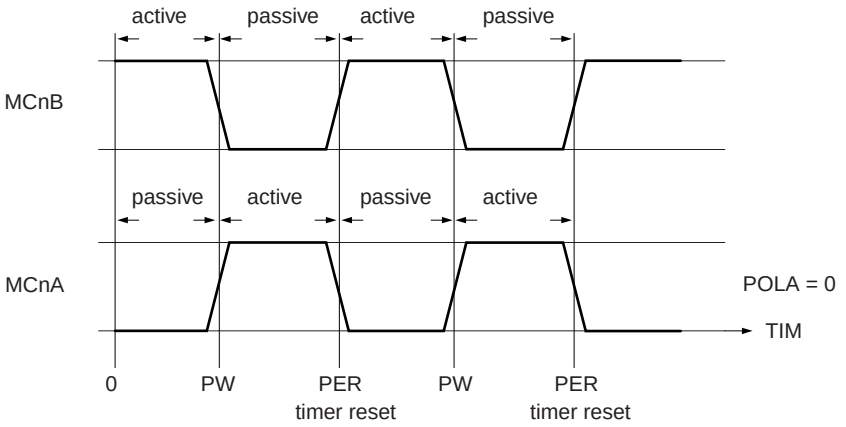


Fig 118. Center-aligned PWM waveform without dead time, POLA = 0

30.8.1.3 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 MCO outputs. 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 MCOA and MCOB output levels are both passive. Figure 119 shows operation in edge aligned mode with dead time, and Figure 120 shows center-aligned operation with dead time.

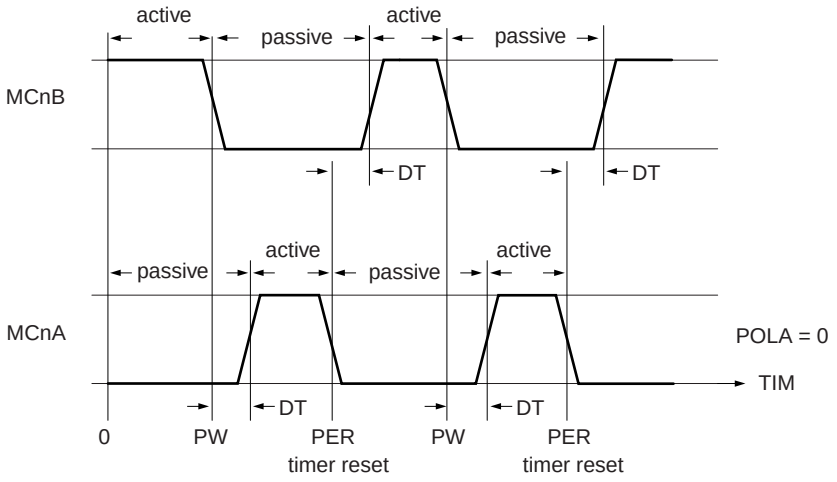


Fig 119. Edge-aligned PWM waveform with dead time, POLA = 0

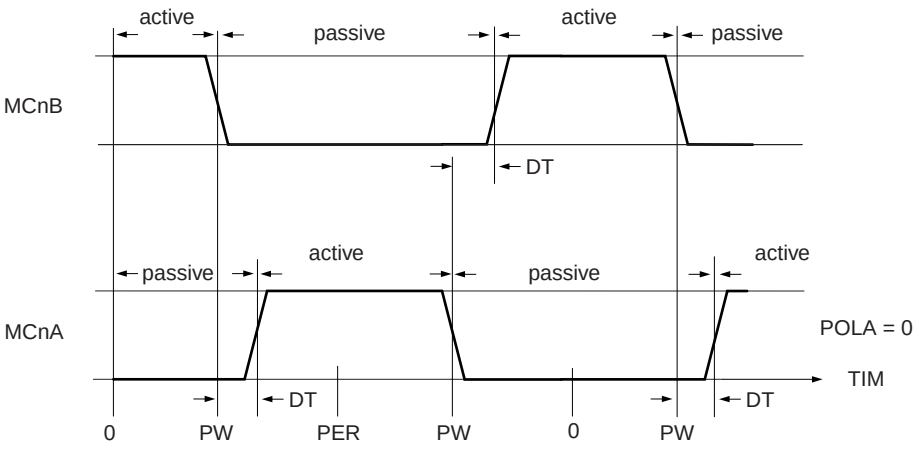


Fig 120. Center-aligned waveform with dead time, POLA = 0

30.8.2 Shadow registers and simultaneous updates

The Limit and Match registers (LIM and MAT) 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.

30.8.3 Fast Abort (ABORT)

The MCPWM has an external input $\overline{\text{MCABORT}}$. When this input goes low, all six MCO outputs assume their passive states, and the Abort interrupt is generated if enabled. The outputs remain locked in passive state until the ABORT interrupt flag is cleared or the Abort interrupt is disabled. The ABORT flag may not be cleared before the $\overline{\text{MCABORT}}$ input goes high.

30.8.4 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 MCPWM inputs 0-2 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 MCI 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 596), 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 MCO pin(s) in edge-aligned mode as described in Section 30.7.7 and Section 30.8.1.

30.8.5 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 MCOA0 signal can be routed to any or all of the MCO outputs. Each MCO output 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 MCOA0. The polarity of the off state is determined by the POLA0 bit.

All MCO outputs having 1 bits in the MCCP register are controlled by the internal MCOA0 signal.

The three MCOB 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 communication pattern occur at the beginning of a new PWM cycle. See Section 30.7.7.1 and Section 30.8.2 for more about writing and reading these registers.

Figure 121 shows sample waveforms of the MCO outputs in three-phase DC mode. Bits 1 and 3 in the MCCP register (corresponding to outputs MCOB1 and MCOB0) 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 MCOB outputs are **not** the opposite of the MCOA outputs.

As shown in Figure 121, bits 0, 2, 4, and 5 in the MCCP register are set to 1, meaning that MCOA1 and both MCO outputs for channel 2, follow the MCOA0 signal.

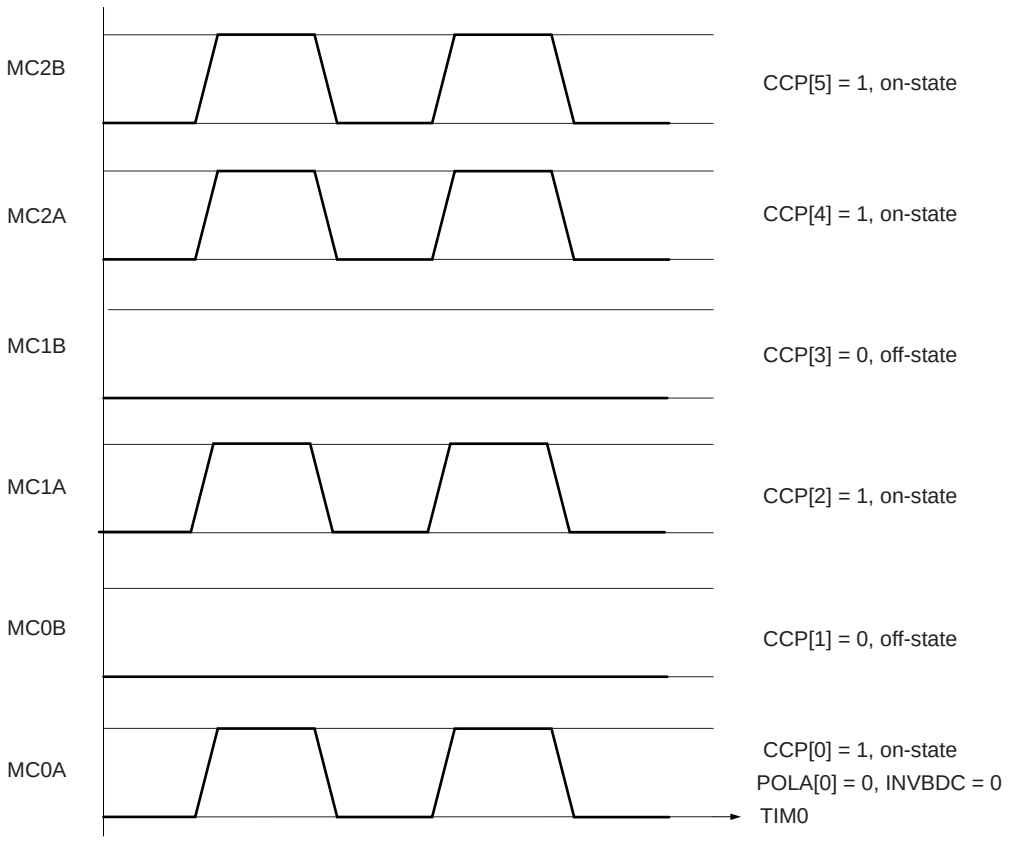


Fig 121. Three-phase DC mode sample waveforms

30.8.6 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 MCO output by comparing its MAT value to TC0.

Figure 122 shows sample waveforms for the six MCO outputs in three-phase AC mode. The POLA bits are set to 0 for all three channels, so that for all MCO outputs the active levels are high and the passive levels are low. Each channels has a different LIM 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.

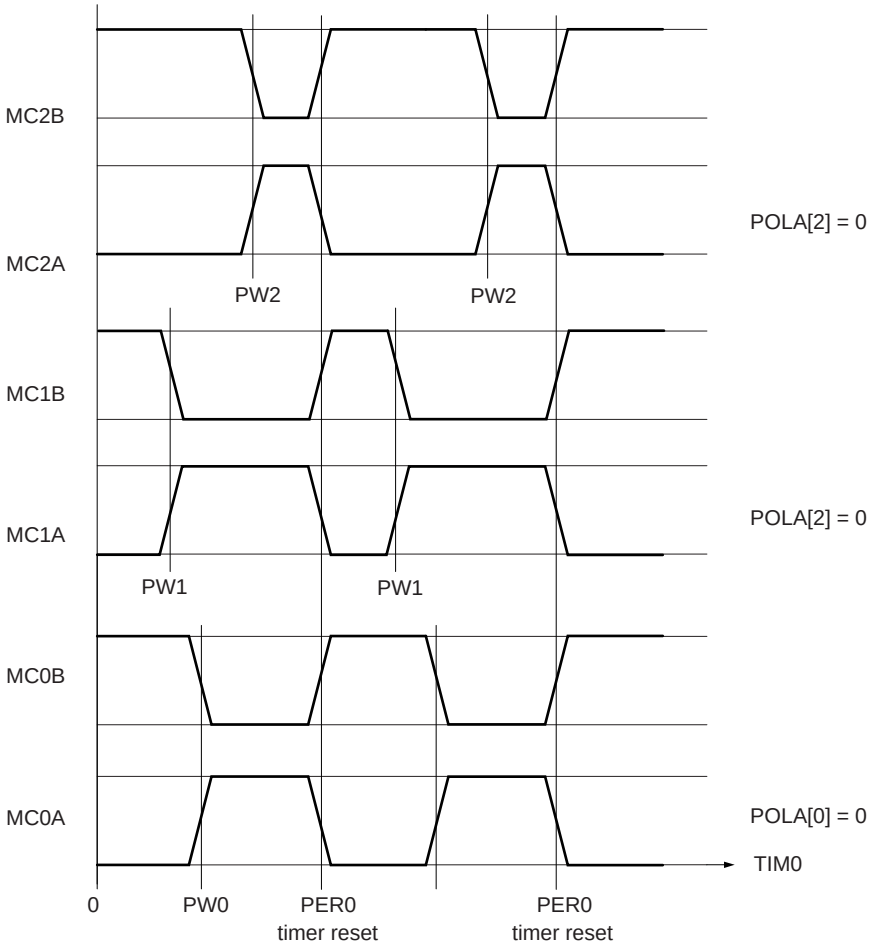


Fig 122. Three-phase AC mode sample waveforms, edge aligned PWM mode

30.8.7 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 MCI0-2.
- When all three channels' outputs are forced to "passive" state because the MCABORT pin goes low.

31.1 Features

- PERIPH_CLK or 32KHz clock source option.
- Programmable 4-bit Prescaler.
- Programmable duty cycle in 255 steps.
- PWM_OUT2 pin can be programmed to output the internal interrupt status (nIRQ or nFIQ).

31.2 Description

There are two free running Pulse Width Modulators (PWM1 and PWM2) located on the FAB bus. They are clocked separately with either PERIPH_CLK or the 32 kHz RTC clock. Note that PERIPH_CLK is stopped and the 32 kHz clock is not stopped in the microcontroller's STOP mode. If the PWMs are not used, the PWM1_CLK and PWM2_CLK can be switched off by programming the PWMCLK_CTRL register. See Table 609 for examples of PWM frequencies:

Table 609. PWM frequencies

Clock Source	Freq. div.	PWM_CLK frequency	PWM_RELOADV	PWM_OUT pin frequency
13 MHz	1 (min)	13 MHz	1 (min)	50 kHz (max)
	1 (min)	13 MHz	256 (max)	198 Hz
	15 (max)	866.7 kHz	256 (max)	13.2 Hz (min)
32 kHz	1 (min)	32 kHz	1 (min)	128 Hz (max)
	1 (min)	32 kHz	256 (max)	0.5 Hz
	15 (max)	2.18 kHz	256 (max)	0.033 Hz (min)

Both PWMs have a programmable duty cycle in 255 steps.

The outputs from PWM1 and PWM2 are connected to the external output pins PWM_OUT1 and PWM_OUT2. When a PWM is disabled, the corresponding output pin will resume the logic state set by the PWM_PIN_LEVEL bit in the control register.

The PWM_OUT2 pin can be programmed to output the internal interrupt status (nIRQ or nFIQ).

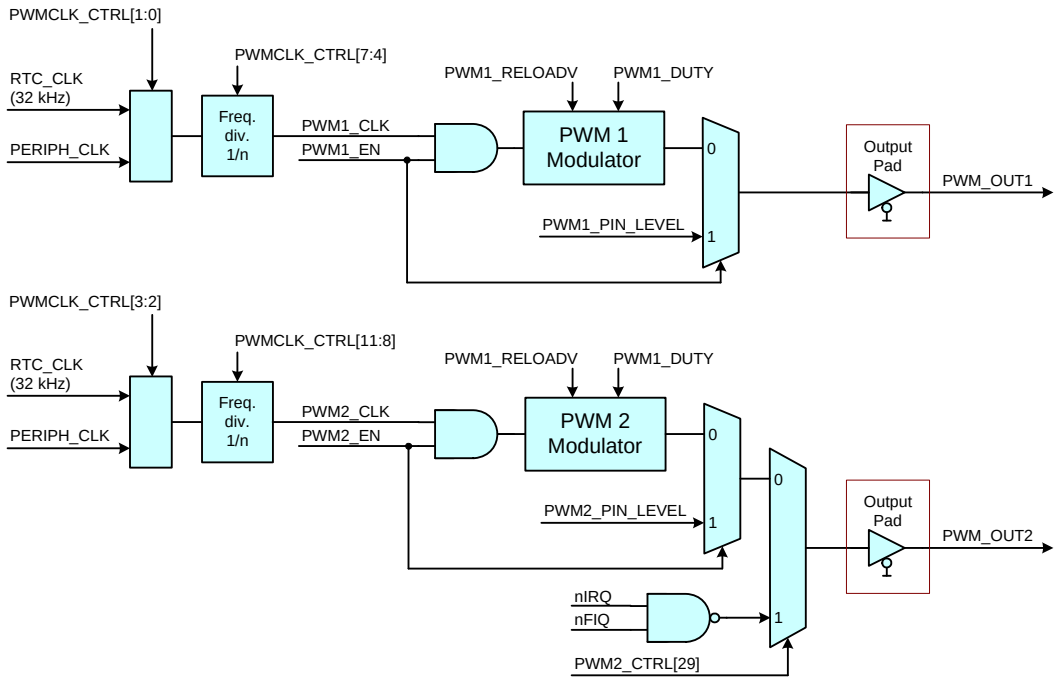


Fig 123. PWM block diagram

31.3 Register description

Table 610 shows the registers of PWM0 and PWM1.

Table 610. Pulse Width Modulator register map

Address offset	Name	Description	Reset value	Type
0x4005 C000	PWM1_CTRL	PWM1 Control register.	0x0	R/W
0x4005 C004	PWM2_CTRL	PWM2 Control register.	0x0	R/W

31.3.1 PWM1 Control Register (PWM1_CTRL, RW - 0x4005 C000)

Table 611. PWM1 Control Register (PWM1_CTRL, RW - 0x4005 C000)

Bits	Description	Reset value
31	PWM1_EN This bit gates the PWM_CLK signal and enables the external output pin to the PWM_PIN_STATE logical level. 0 = PWM disabled. (Default) 1 = PWM enabled.	0
30	PWM1_PIN_LEVEL If the PWM1_EN bit is set to 0, the PWM_OUT1 pin is set to the PWM1_PIN_LEVEL logical state. (Default = 0)	0

Table 611. PWM1 Control Register (PWM1_CTRL, RW - 0x4005 C000) *…continued*

Bits	Description	Reset value
29:16	Not used Write is don't care, Read returns random value.	0
15:8	PWM1_RELOADV Reload value for the PWM output frequency. Fout = (PWM_CLK / PWM_RELOADV) / 256. A value of 0 is treated as 256. (Default = 0)	0x0
7:0	PWM1_DUTY Adjusts the output duty cycle. [Low]/[High] = [PWM_DUTY] / [256-PWM_DUTY], where 0 < PWM_DUTY <= 255. (Default = 0)	0x0

31.3.2 PWM2 Control Register (PWM2_CTRL, RW - 0x4005 C004)

Table 612. PWM2 Control Register (PWM2_CTRL, RW - 0x4005 C004)

Bits	Description	Reset value
31	PWM2_EN This bit gates the PWM_CLK signal and enables the external output pin to the PWM_PIN_STATE logical level. 0 = PWM disabled. (Default) 1 = PWM enabled.	0
30	PWM2_PIN_LEVEL If the PWM1_EN bit is set to 0, the PWM_OUT1 pin is set to the PWM1_PIN_LEVEL logical state. (Default = 0)	0
29	PWM2_INT 0 = Normal PWM_OUT2 functionality 1 = PWM_OUT2 outputs the internal interrupt status. A low level means that neither nIRQ or nFIQ is active. PWM_OUT2 = nIRQ 'NAND' nFIQ.	0
28:16	Not used Write is don't care, Read returns random value.	0
15:8	PWM2_RELOADV Reload value for the PWM output frequency. Fout = (PWM_CLK / PWM_RELOADV) / 256 (A value of 0 is treated as 256) Default = 0	0x0
7:0	PWM2_DUTY Adjusts the output duty cycle. [Low]/[High] = [PWM_DUTY] / [256-PWM_DUTY], where 0 < PWM_DUTY <= 255. (Default = 0)	0x0

32.1 Features

- There are four ports which describe the general purpose input, output and input/output pins available on the LPC32x0
- Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.
- Direction control of individual bits that support both input and output modes.
- For input/output pins, both the programmed output state and the actual pin state can be read.
- There are a total of 22 general purpose inputs, 24 general purpose outputs, and 6 general purpose input/outputs (Port 3).
- Additionally, 13 External memory controller (EMC) data lines (Port 2) may be used as GPIOs if a 16-bit SDRAM interface is used (rather than a 32-bit interface)
- Additionally, up to 24 EMC address lines (Port 1) may be used as GPIOs they are not used in for an EMC memory interface.
- Additionally, there are 8 additional GPIO lines available for use if the peripheral they share signals with are not used (Port 0).

32.2 Applications

- General purpose I/O
- Driving LEDs or other indicators
- Controlling or communicating with off-chip devices
- Sensing static inputs

32.3 Pin description

Table 613. Port 0 GPIO pin description

Pin name	Type	Description
p0[7:0]	I/O	General purpose input/outputs P0.0 through P0.7.

Table 614. Port 1 GPIO pin description

Pin name	Type	Description
P1[23:0]	I/O	General purpose input/outputs P1.0 through P1.23. (multiplexed with EMC_A[23:0])

Table 615. Port 2 GPIO pin description

Pin name	Type	Description
P2[12:0]	I/O	General purpose input/outputs P2.0 through P2.12. (multiplexed with EMC_D[31:19]

Table 616. Port 3 GPI, GPO and GPIO pin description

Pin name	Type	Description
GPI_[9:0]	I	General purpose input, GPI_00 through GPI_9.
GPI_[23:15]	I	General purpose input, GPI_15 through GPI_23.
GPI_25	I	General purpose input, GPI_25.
GPI_[28:27]	I	General purpose input, GPI_27 through GPI_28.
GPO_[23:0]	O	General purpose output, GPO_0 through GPO_23.
GPIO_[5:0]	I/O	General purpose input/output, GPI0_00 through GPI0_05.

32.4 GPIO functional description

The General Purpose Input/Output pin block controls the state of the output ports and allows access to input ports. Some pins are defined as General Purpose Outputs (GPOs), some are General Purpose Inputs (GPIs), and some can perform both functions as General Purpose Input/Outputs (GPIOs). At reset all output signals have a defined value. These values can be found in the Packaging, Pinout, and Pin Multiplexing chapter.

The GPIO block is accessed via the FAB bus and is clocked with PERIPH_CLK. The following three figures show the connections for the four I/O ports, (P0, P1, P2, AND P3). Ports 0, 1 and 2 are GPIO ports, while Port 3 contains different types of pins GPO, GPI and GPIO.

32.4.1 Port 0, 1, 2 and 3 Bidirectional pins

- As for Inputs, registers P[3:0]_INP_STATE reflect the current level of the GPIO input pins.
- As for Outputs, the P[3:0]_OUTP_SET and P[3:0]_OUTP_CLR registers control the level on the corresponding GPIO pins.
- The P[2:0]_DIR_SET and P[2:0]_DIR_CLR registers control the direction of the GPIO pins. The chosen direction of the GPIO pins can be read in the P[2:0]_DIR_STATE register.
- The programmed level of the output signal (not necessarily the actual pin level) can be read in the P[3:0]_OUTP_STATE register.

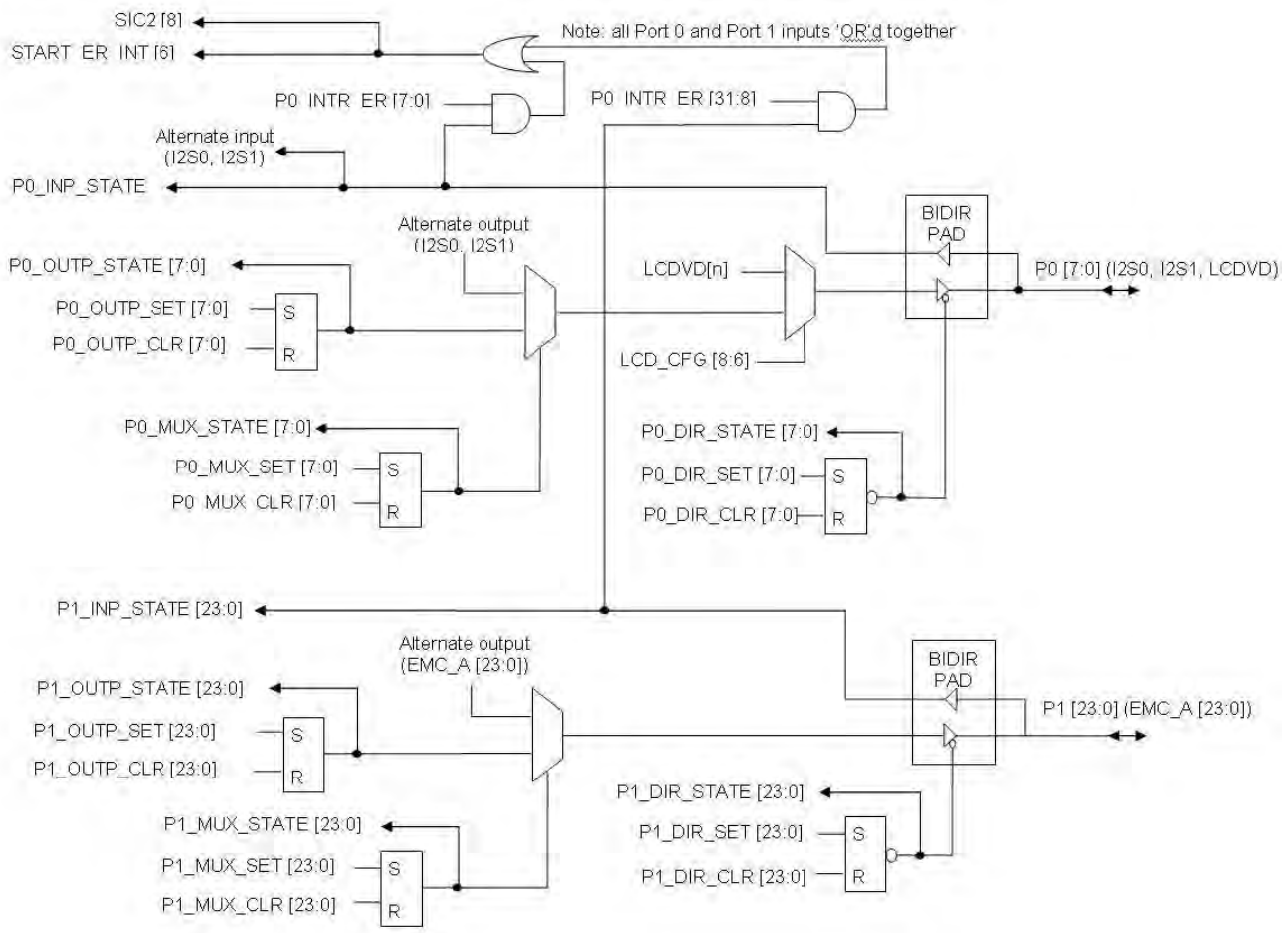


Fig 124. Port 0 GPIO/Alternate output and Port 1 and GPIO/EMC signals

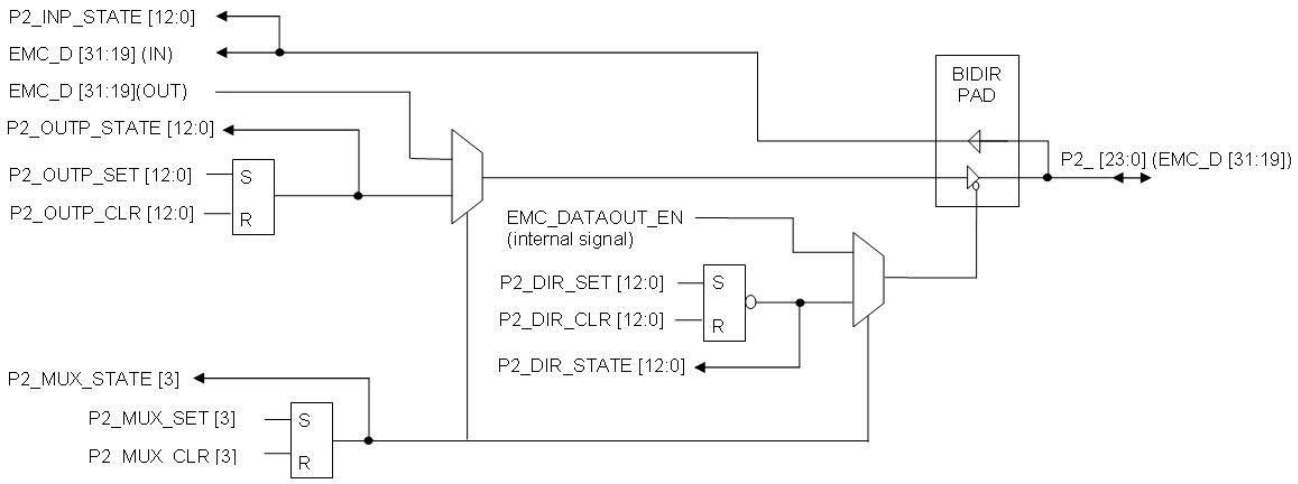


Fig 125. Port 2 EMC GPIO signals

32.4.1.1 Port 1 and Port 2 EMC bus GPIOs

When using a 16 bit SDRAM bus, or if no SRAMs are connected, some or all the EMC_D[31:19] and EMC_A[23:0] pins may be used as GPIOs. Following reset, the EMC_D[31:19] and EMC_A[23:0] pins are connected to the EMC block

Port 1

- The P1_INP_STATE register reflects the actual level of the P1[23:0] (EMC_A[23:0]) pins.
- The P1_OUTP_SET and P1_OUTP_CLR registers control the level on the corresponding P1[23:0] (EMC_A[23:0]) pins.
- The direction of the P1[23:0] (EMC_A[23:0]) pins can be selected using the P1_DIR_SET and P1_DIR_CLR registers. The selected direction of the P1[23:0] (EMC_A[23:0]) pins can be read in the P1_DIR_STATE register.

Port 2

- The P2_INP_STATE register reflects the actual level of the P2[12:0] (EMC_D[31:19]) pins.
- The P2_OUTP_SET and P2_OUTP_CLR registers control the level on the corresponding EMC_D[31:19] pins.
- The direction of the EMC_D[31:19] pins can be selected using the P2_DIR_SET and P2_DIR_CLR registers. The selected direction of the EMC_D[31:19] pins can be read in the P2_DIR_STATE register.

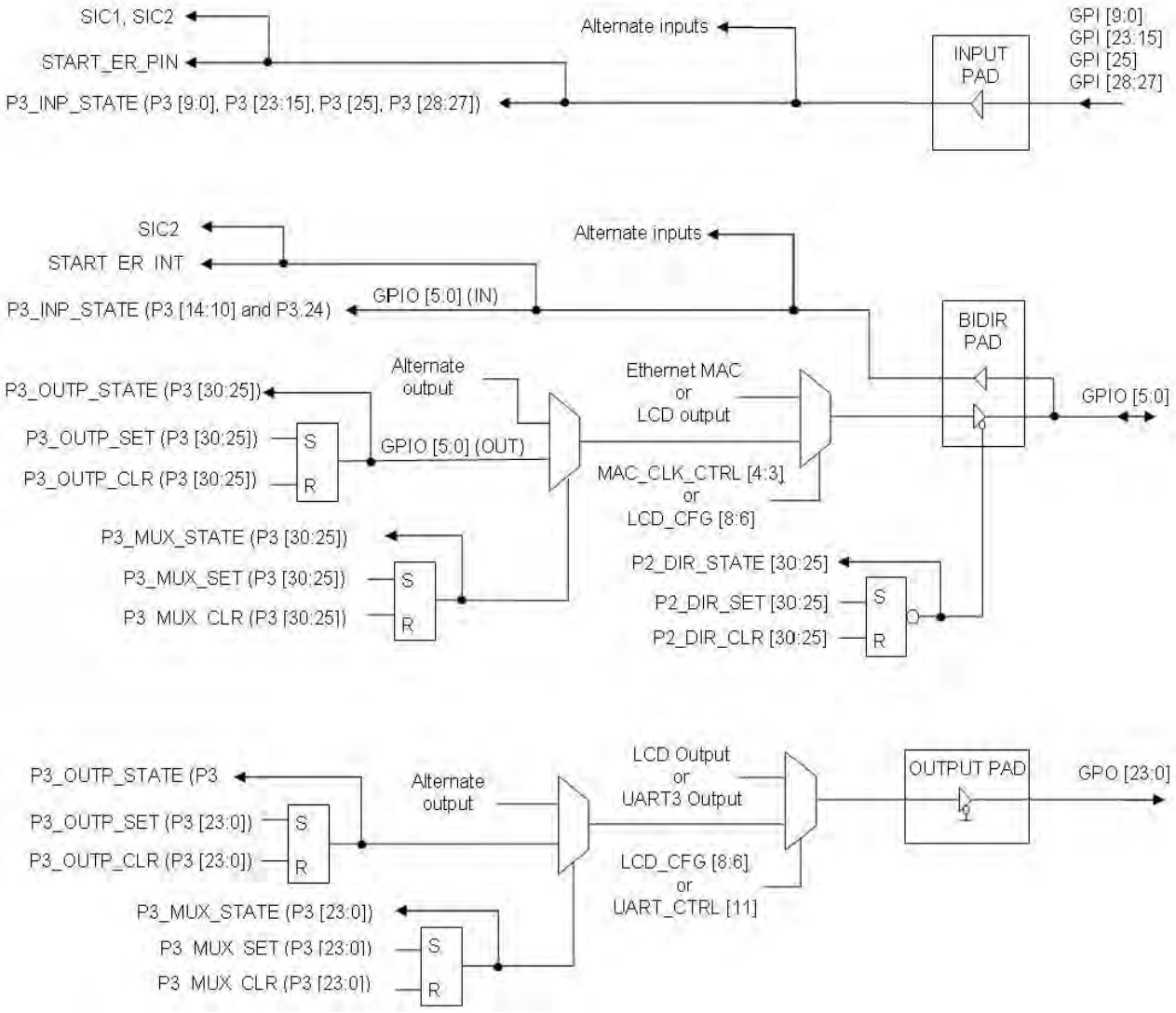


Fig 126. Port 3 GPI, GPO, GPIO signals

32.4.2 Port 3 Inputs

- The actual pin level can be read via the P3_INP_STATE register. A 'high' on the external pin will result in level '1' in the corresponding bit in the register.
- Many input signals are connected as start signals. Refer to the Clocking and Power Control chapter for details.
- Many input signals are connected as IRQ signals. Refer to the Interrupt Controller chapter for details.

32.4.3 Port 3 Outputs

- The level for the output can be controlled via the P3_OUTP_SET and P3_OUTP_CLR registers.

- The actual level for the output can be read via the P3_OUTP_STATE register.

32.4.4 Alternate functions

Some GPIO pins have alternate functions that are selected by using the P[3:0]_MUX_SET and P[3:0]_MUX_CLR registers. The selected state of these pins can be read in the P[3:0]_MUX_STATE registers.

GPO_0 has additional connections, allowing it to output one of 3 clock signals. This feature is intended primarily for system testing. The connections to GPO_0 are shown in Figure 127.

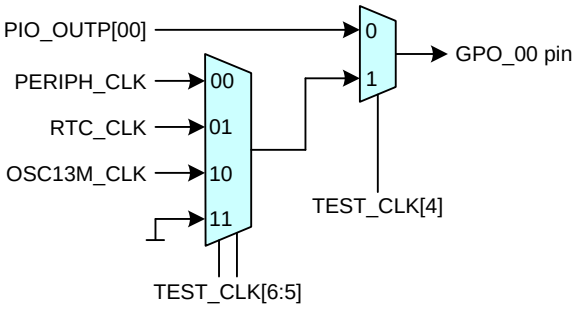


Fig 127. GPO_0 alternate functions

32.5 Register description

The registers in Table 617 give control over GPIO features available on the LPC32x0.

Table 617. Summary of GPIO Data and Configuration registers

Address	Name	Description	Reset state	Access
Port 0				
0x4002 8040	P0_INP_STATE	Port 0 Input pin state register. Reads the state of input pins.	-	RO
0x4002 8044	P0_OUTP_SET	Port 0 Output pin set register. Allows setting output pin(s).	-	WO
0x4002 8048	P0_OUTP_CLR	Port 0 Output pin clear register. Allows clearing output pin(s).	-	WO
0x4002 804C	P0_OUTP_STATE	Port 0 Output pin state register. Reads the state of output pins.	-	RO
0x4002 8050	P0_DIR_SET	Port 0 direction set register. Configures P0 I/O pins as outputs.	-	WO
0x4002 8054	P0_DIR_CLR	Port 0 direction clear register. Configures P0 I/O pins as inputs.	-	WO
0x4002 8058	P0_DIR_STATE	Port 0 direction state register. Reads direction of P0 I/O pins.	0	RO
Port 1				
0x4002 8060	P1_INP_STATE	Port 1 Input pin state register. Reads the state of P1 pins.	-	RO
0x4002 8064	P1_OUTP_SET	Port 1 Output pin set register. Sets output value of P1 pins.	-	WO
0x4002 8068	P1_OUTP_CLR	Port 1 Output pin clear register. Clears output value of P1 pins.	-	WO
0x4002 806C	P1_OUTP_STATE	Port 1 Output pin state register. Reads state of P1 output pins.	-	RO
0x4002 8070	P1_DIR_SET	Port 1 direction set register. Configures Port 1 I/O pins as outputs.	-	WO
0x4002 8074	P1_DIR_CLR	Port 1 direction clear register. Configures Port 1 I/O pins as inputs.	-	WO
0x4002 8078	P1_DIR_STATE	Port 1 direction state register. Reads direction value Port 1 I/O pins.	0	RO

Table 617. Summary of GPIO Data and Configuration registers

Address	Name	Description	Reset state	Access
Port 2				
0x4002 801C	P2_INP_STATE	Port 2 Input pin state register. Reads the state of Port 2 GPIO pins.	-	RO
0x4002 8020	P2_OUTP_SET	Port 2 Output pin set register. Sets Port 2 GPIO output value.	-	WO
0x4002 8024	P2_OUTP_CLR	Port 2 Output pin clear register. Clears Port 2 GPIO output value.	-	WO
0x4002 8010	P2_DIR_SET	Port 2 and Port 3 GPIO direction set register. configures direction of I/O pins P2.[23:0] and GPIO_[5:0].	-	WO
0x4002 8014	P2_DIR_CLR	Port 2 and Port 3 GPIO direction clear register. configures direction of I/O pins P2.[23:0] and GPIO_[5:0].	-	WO
0x4002 8018	P2_DIR_STATE	Port 2 and Port 3 GPIO direction state register. Reads pin direction status for I/O pins P2.[23:0] and GPIO_[5:0].	0	RO
Port 3				
0x4002 8000	P3_INP_STATE	Port 3 Input pin state register. Reads the state of GPIO[5:0] and GPI input pins.	-	RO
0x4002 8004	P3_OUTP_SET	Port 3 output pin set register. Sets GPIO[5:0] output and GPO_[23:0] pin(s).	-	WO
0x4002 8008	P3_OUTP_CLR	Port 3 output pin clear register. Clears GPIO_[5:0] output and GPO_[23:0] pin(s).	-	WO
0x4002 800C	P3_OUTP_STATE	Port 3 output pin state register. Reads the state of GPIO_[5:0] output and GPO_[23:0] pin(s).	-	RO

32.5.1 P0 Input Pin State register (P0_INP_STATE - 0x4002 8040)

The P0_INP_STATE register is a read-only register that provides the state of GPIO pins P0.7 through p0.0. This allows reading the pin values when the pins are used as GPIOs.

Table 618. P0 Input Pin State register for EMC address pins (P0_INP_STATE - 0x4002 8040)

P0_INP_STATE	Function	Description
0	P0.0 / I2S1RX_CLK	Reflects the state of the P0.0 pin.
1	P0.1 / I2S1RX_WS	Reflects the state of the P0.1 pin.
2	P0.2 / I2S0RX_SDA | (LCDVD[4])	Reflects the state of the P0.2 pin.
3	P0.3 / I2S0RX_CLK | (LCDVD[5])	Reflects the state of the P0.3 pin.
4	P0.4 / I2S0RX_WS | (LCDVD[6])	Reflects the state of the P0.4 pin.
5	P0.5 / I2S0TX_SDA | (LCDVD[7])	Reflects the state of the P0.5 pin.
6	P0.6 / I2S0TX_CLK | (LCDVD[12])	Reflects the state of the P0.6 pin.
7	P0.7 / I2S0TX_WS | (LCDVD[13])	Reflects the state of the P0.7 pin.

32.5.2 P0 Output Pin Set register (P0_OUTP_SET - 0x4002 8044)

The P0_OUTP_SET register is a write-only register that allows setting one or more of the P0 GPIO pins. This applies only if the P0 pins are selected as GPIOs (using the P0_MUX_CLR register, which is described in Section 33.4.2), and if the specified pins are configured as outputs via the P0_DIR_SET register.

Writing a one to a bit in P0_OUTP_SET results in the corresponding pin being driven high (if configured as an output). Writing a zero to a bit in P0_OUTP_SET has no effect.

Table 619. P0 Output Pin Set register (P1_OUTP_SET - 0x4002 8044)

P0_OUTP_SET	Function	Description
31:0		Same functions as P0_INP_STATE.

32.5.3 P0 Output Pin Clear register (P0_OUTP_CLR - 0x4002 8048)

The P0_OUTP_CLR register is a write-only register that allows clearing one or more of the P0 GPIO pins. This applies only if the P0 GPIO pins are selected as GPIOs using the P0_MUX_CLR register, which is described in Section 33.4.2, and if the specified pins are configured as outputs via the P0_DIR_SET register.

Writing a one to a bit in P0_OUTP_CLR results in the corresponding pin (if configured as an output) being driven low. Writing a zero to a bit in P0_OUTP_CLR has no effect.

Table 620. P0 Output Pin Clear register (P0_OUTP_CLR - 0x4002 8048)

P0_OUTP_CLR	Function	Description
31:0		Same functions as P0_INP_SET.

32.5.4 P0 Output Pin State Register (P0_OUTP_STATE - 0x4002 804C)

The P0_OUTP_STATE register is a read-only register that provides the state of all P0 outputs (if configured as output) pins.

Table 621. P0 Output Pin State Register (P0_OUTP_STATE - 0x4002 804C)

P0_OUTP_STATE	Function	Description
31:0		Same functions as P0_OUTP_SET.

32.5.5 P0 Direction Set Register (P0_DIR_SET - 0x4002 8050)

The P0_DIR_SET register is a write-only register that configures the data direction of the P0.0 through P0.7 pins when they are used as output pins. P0_DIR_SET is used in conjunction with P0_DIR_CLR.

Writing a one to a bit in P0_DIR_SET results in the corresponding I/O pin being configured as an output. Writing a zero to a bit in P0_DIR_SET has no effect.

In the case of P0.0 to P0.7, any value written to P0_DIR_SET applies only the pins are selected as GPIOs by the GPIO_MUX_SEL bit in the P0_MUX_CLR register, which is described in Section 33.4.1.

Table 622. P0 Direction Set Register (P0_DIR_SET - 0x4002 8050)

P0_DIR_SET	Function	Description
0	P0.0	Configure the P0.0 pin[1].
1	P0.1	Configure the P0.1 pin[1].
2	P0.2	Configure the P0.2 pin[1].
3	P0.3	Configure the P0.3 pin[1].
4	P0.4	Configure the P0.4 pin[1].
5	P0.5	Configure the P0.5 pin[1].
6	P0.6	Configure the P0.6 pin[1].
7	P0.7	Configure the P0.7 pin[1].

[1] Bit level definitions: 1 = set pin as output, 0 = set pin as input.

32.5.6 P0 Direction Clear Register (P0_DIR_CLR - 0x4002 8054)

The P0_DIR_CLR register is a write-only register that configures the data direction of P0.0 through p0.7 when they are used as I/0 lines. P0_DIR_CLR is used in conjunction with P0_DIR_SET.

Writing a one to a bit in P0_DIR_CLR results in the corresponding I/O pin being configured as an input. Writing a zero to a bit in P0_DIR_CLR has no effect.

In the case of p0.0 through p0.7, any value written to P0_DIR_CLR applies only if the pins are selected as GPIOs using the P0_MUX_CLR register, which is described in Section 33.4.1.

Table 623. P0 Direction Clear Register (P1_DIR_CLR - 0x4002 8054)

P0_DIR_CLR	Function	Description
31:0		Same functions as P0_DIR_SET.

32.5.7 P0 Direction State Register (P0_DIR_STATE - 0x4002 80058)

The P0_DIR_STATE register is a read-only register that reports the direction selection for P0.0 through P1.7 when they are used as I/O lines. The value read reflects the result of writes to P0_DIR_SET and P0_DIR_CLR.

A value of zero indicates that the pin is configured as an input. A value of one indicates that the pin is configured as an output. In the case of P0.0 through P0.7, the value read from P0_DIR_STATE only applies if these pins have been configured as GPIOs using the P0_MUX_CLR register, which is described in Section 33.4.1.

Table 624. P1 Direction State Register (P1_DIR_STATE - 0x4002 80078)

P3_DIR_STATE	Function	Description
31:0		Same functions as P1_DIR_SET.

32.5.8 P1 Input Pin State register for unused EMC Address pins (P1_INP_STATE - 0x4002 8060)

The P1_INP_STATE register is a read-only register that provides the state of EMC_A pins EMC_A[23:0]. This allows reading the pin values when the EMC address pins are used as GPIOs.

Table 625. P1 Input Pin State register for EMC address pins (P1_INP_STATE - 0x4002 8060)

P1_INP_STATE	Function	Description
0	EMC_A[0] / P1.0	Reflects the state of the P1.0 pin.
1	EMC_A[1] / P1.1	Reflects the state of the P1.1 pin.
2	EMC_A[2] / P1.2	Reflects the state of the P1.2 pin.
3	EMC_A[3] / P1.3	Reflects the state of the P1.3 pin.
4	EMC_A[4] / P1.4	Reflects the state of the P1.4 pin.
5	EMC_A[5] / P1.5	Reflects the state of the P1.5 pin.
6	EMC_A[6] / P1.6	Reflects the state of the P1.6 pin.
7	EMC_A[7] / P1.7	Reflects the state of the P1.7 pin.

Table 625. P1 Input Pin State register for EMC address pins (P1_INP_STATE - 0x4002 8060)

P1_INP_STATE	Function	Description
8	EMC_A[8] / P1.8	Reflects the state of the P1.8 pin.
9	EMC_A[9] / P1.9	Reflects the state of the P1.9 pin.
10	EMC_A[10] / P1.10	Reflects the state of the P1.10 pin.
11	EMC_A[11] / P1.11	Reflects the state of the P1.11 pin.
12	EMC_A[12] / P1.12	Reflects the state of the P1.12 pin.
13	EMC_A[13] / P1.13	Reflects the state of the P1.13 pin.
14	EMC_A[14] / P1.14	Reflects the state of the P1.14 pin.
15	EMC_A[15] / P1.15	Reflects the state of the P1.15 pin.
16	EMC_A[16] / P1.16	Reflects the state of the P1.16 pin.
17	EMC_A[17] / P1.17	Reflects the state of the P1.17 pin.
18	EMC_A[18] / P1.18	Reflects the state of the P1.18 pin.
19	EMC_A[19] / P1.19	Reflects the state of the P1.19 pin.
20	EMC_A[20] / P1.20	Reflects the state of the P1.20 pin.
21	EMC_A[21] / P1.21	Reflects the state of the P1.21 pin.
22	EMC_A[22] / P1.22	Reflects the state of the P1.22 pin.
23	EMC_A[23] / P1.23	Reflects the state of the P1.23 pin

32.5.9 P1 Output Pin Set register for EMC address pins (P1_OUTP_SET - 0x4002 8064)

The P1_OUTP_SET register is a write-only register that allows setting one or more of the EMC_A[23:0] pins. This applies only if the EMC pins EMC_A[23:0] are selected as GPIOs P1[23:0] in the P1_MUX_SET register, which is described in Section 33.4.4, and if the specified pins are configured as outputs via the P1_DIR_SET register.

Writing a one to a bit in P1_OUTP_SET results in the corresponding pin (if configured as an output) being driven high. Writing a zero to a bit in P1_OUTP_SET has no effect.

Table 626. P1 Output Pin Set register for EMC address pins (P1_OUTP_SET - 0x4002 8064)

P1_OUTP_SET	Function	Description
31:0		Same functions as P1_INP_STATE.

32.5.10 P1 Output Pin Clear register for EMC address pins (P1_OUTP_CLR - 0x4002 8068)

The P1_OUTP_CLR register is a write-only register that allows clearing one or more of the EMC pins EMC_A[23:0]. This applies only if the EMC pins EMC_A[23:0] are selected as GPIOs P1.[23:0] using the P1_MUX_SET register, which is described later in this chapter, and if the specified pins are configured as outputs via the P1_DIR_SET register.

Writing a one to a bit in P1_OUTP_CLR results in the corresponding pin (if configured as an output) being driven low. Writing a zero to a bit in P1_OUTP_CLR has no effect.

Table 627. P1 Output Pin Clear register for unused EMC address pins (P1_OUTP_CLR - 0x4002 8068)

P1_OUTP_CLR	Function	Description
31:0		Same functions as P1_INP_STATE.

32.5.11 P1 Output Pin State Register (P1_OUTP_STATE - 0x4002 806C)

The P1_OUTP_STATE register is a read-only register that provides the state of all P1 outputs (if configured as output) pins.

Table 628. P1 Output Pin State Register (P1_OUTP_STATE - 0x4002 806C)

P1_OUTP_STATE	Function	Description
31:0		Same functions as P1_OUTP_SET.

32.5.12 P1 Direction Set Register (P1_DIR_SET - 0x4002 8070)

The P1_DIR_SET register is a write-only register that configures the data direction of the EMC_A[23:0] when they are not used for EMC address lines. P1_DIR_SET is used in conjunction with P1_DIR_CLR.

Writing a one to a bit in the P1_DIR_SET register results in the corresponding I/O pin being configured as an output. Writing a zero to a bit in P1_DIR_SET has no effect.

In the case of EMC_A[23:0], any value written to P1_DIR_SET applies only if EMC_A[23:0] are selected as GPIOs P1.[23:0] using the P1_MUX_SET register, which is described later in this chapter.

Table 629. P1 Direction Set Register (P1_DIR_SET - 0x4002 8070)

P1_DIR_SET	Function	Description
0	P1.0	Configure the P1.0 pin[1].
1	P1.1	Configure the P1.1 pin[1].
2	P1.2	Configure the P1.2 pin[1].
3	P1.3	Configure the P1.3 pin[1].
4	P1.4	Configure the P1.4 pin[1].
5	P1.5	Configure the P1.5 pin[1].
6	P1.6	Configure the P1.6 pin[1].
7	P1.7	Configure the P1.7 pin[1].
8	P1.8	Configure the P1.8 pin[1].
9	P1.9	Configure the P1.9 pin[1].
10	P1.10	Configure the P1.10 pin[1].
11	P1.11	Configure the P1.11 pin[1].
12	P1.12	Configure the P1.12 pin[1].
13	P1.13	Configure the P1.13 pin[1].
14	P1.14	Configure the P1.14 pin[1].
15	P1.15	Configure the P1.15 pin[1].
17	P1.17	Configure the P1.17 pin[1].
18	P1.18	Configure the P1.18 pin[1].
19	P1.19	Configure the P1.19 pin[1].
20	P1.20	Configure the P1.20 pin[1].
16	P1.16	Configure the P1.16 pin[1].
21	P1.21	Configure the P1.21 pin[1].
22	P1.22	Configure the P1.22 pin[1].
23	P1.23	Configure the P1.23 pin[1].

[1] Bit level definitions: 1 = set pin as output, 0 = set pin as input.

32.5.13 P1 Direction Clear Register (P1_DIR_CLR - 0x4002 8074)

The P1_DIR_CLR register is a write-only register that configures the data direction of EMC_A[23:0] when they are used as GPIOs P1.[23:0] lines. P1_DIR_CLR is used in conjunction with P1_DIR_SET.

Writing a one to a bit in P1_DIR_CLR results in the corresponding I/O pin being configured as an input. Writing a zero to a bit in P1_DIR_CLR has no effect.

In the case of EMC_A[23:0], any value written to P1_DIR_CLR applies only if EMC_A[23:0] are selected as GPIOs P1.[23:0] using the P1_MUX_SET register, which is described later in this chapter.

Table 630. P1 Direction Clear Register (P1_DIR_CLR - 0x4002 8014)

P1_DIR_CLR	Function	Description
31:0		Same functions as P1_DIR_SET.

32.5.14 P1 Direction State Register (P1_DIR_STATE - 0x4002 80078)

The P1_DIR_STATE register is a read-only register that reports the direction selection for EMC_A[23:0] when they are not used for EMC address lines. The value read reflects the result of writes to P1_DIR_SET and P1_DIR_CLR.

A value of zero indicates that the pin is configured as an input. A value of one indicates that the pin is configured as an output. In the case of EMC_A[23:0], the value read from P1_DIR_STATE only applies if these pins have been configured as GPIOs P1.[23:0] using the P1_MUX_SET register, which is described in Section 33.4.4.

Table 631. P1 Direction State Register (P1_DIR_STATE - 0x4002 80078)

P1_DIR_STATE	Function	Description
31:0		Same functions as P1_DIR_SET.

32.5.15 P2 Input Pin State register for EMC pins (P2_INP_STATE - 0x4002 801C)

The P2_INP_STATE register is a read-only register that provides the state of EMC pins EMC_D[31:19]. This allows reading the pin values when the EMC data pins are used as GPIOs P2[12:0].

Table 632. Input Pin State register for EMC pins (P2_INP_STATE - 0x4002 801C)

P2_INP_STATE	Function	Description
0	EMC_D[19] / P2.0	Reflects the state of the P2.0 input pin.
1	EMC_D[20] / P2.1	Reflects the state of the P2.1 input pin.
2	EMC_D[21] / P2.2	Reflects the state of the P2.2 input pin.
3	EMC_D[22] / P2.3	Reflects the state of the P2.3 input pin.
4	EMC_D[23] / P2.4	Reflects the state of the P2.4 input pin.
5	EMC_D[24] / P2.5	Reflects the state of the P2.5 input pin.
6	EMC_D[25] / P2.6	Reflects the state of the P2.6 input pin.
7	EMC_D[26] / P2.7	Reflects the state of the P2.7 input pin.

Table 632. Input Pin State register for EMC pins (P2_INP_STATE - 0x4002 801C)

P2_INP_STATE	Function	Description
8	EMC_D[27] / P2.8	Reflects the state of the P2.8 input pin.
9	EMC_D[28] / P2.9	Reflects the state of the P2.9 input pin.
10	EMC_D[29] / P2.10	Reflects the state of the P2.10 input pin.
11	EMC_D[30] / P2.11	Reflects the state of the P2.11 input pin.
12	EMC_D[31] / P2.12	Reflects the state of the P2.12 input pin.
31:13	Reserved	The value read from a reserved bit is not defined.

32.5.16 P2 Output Pin Set register for EMC pins (P2_OUTP_SET - 0x4002 8020)

The P2_OUTP_SET register is a write-only register that allows setting one or more of the EMC_D[31:19] pins. This applies only if the EMC pins EMC_D[31:19] are selected as GPIOs P2 [12:0] using P2_MUX_SET register, which is described in Section 33.4.7, and if the specified pins are configured as outputs via the P3_DIR_SET register.

Writing a one to a bit in P2_OUTP_SET results in the corresponding pin (if configured as an output) being driven high. Writing a zero to a bit in P2_OUTP_SET has no effect.

Table 633. P2 Output Pin Set register for EMC data pins (P2_OUTP_SET - 0x4002 8020)

P2_OUTP_SET	Function	Description
31:0		Same functions as P2_INP_STATE.

32.5.17 P2 Output Pin Clear register for EMD data pins (P2_OUTP_CLR - 0x4002 8024)

The P2_OUTP_CLR register is a write-only register that allows clearing one or more of the EMC pins EMC_D[31:19]. This applies only if the EMC pins EMC_D[31:19] are selected as GPIOs P2 [12:0] using the P2_MUX_SET register, which is described later in this chapter, and if the specified pins are configured as outputs via the P2_DIR_SET register.

Writing a one to a bit in P2_OUTP_CLR results in the corresponding pin (if configured as an output) being driven low. Writing a zero to a bit in P2_OUTP_CLR has no effect.

Table 634. P2 Output Pin Clear register for EMC data pins (P2_OUTP_CLR - 0x4002 8024)

P2_OUTP_CLR	Function	Description
31:0		Same functions as P2_INP_STATE.

32.5.18 Port2 and Port3 Direction Set Register (P2_DIR_SET - 0x4002 8010)

The P2_DIR_SET register is a write-only register that configures the data direction of GPIO[5:0] pins and EMC_D[31:19] (P2.0 through P2.12) when they are not used as EMC data lines. P3_DIR_SET is used in conjunction with P3_DIR_CLR.

Writing a one to a bit in P2_DIR_SET results in the corresponding I/O pin being configured as an output. Writing a zero to a bit in P2_DIR_SET has no effect.

In the case of EMC_D[31:19], any value written to P2_DIR_SET applies only if EMC_D[31:19] are selected as GPIOs (P2.0 through P2.12) using the P2_MUX_SET register, which is described in Section 33.4.7.

Table 635. P2 Direction Set Register (P2_DIR_SET - 0x4002 8010)

P2_DIR_SET	Function	Description
0	EMC_D[19] / P2.0	Configure the P2.0 pin[1].
1	EMC_D[20] / P2.1	Configure the P2.1 pin[1].
2	EMC_D[21] / P2.2	Configure the P2.2 pin[1].
3	EMC_D[22] / P2.3	Configure the P2.3 pin[1].
4	EMC_D[23] / P2.4	Configure the P2.4 pin[1].
5	EMC_D[24] / P2.5	Configure the P2.5 pin[1].
6	EMC_D[25] / P2.6	Configure the P2.6 pin[1].
7	EMC_D[26] / P2.7	Configure the P2.7 pin[1].
8	EMC_D[27] / P2.8	Configure the P2.8 pin[1].
9	EMC_D[28] / P2.9	Configure the P2.9 pin[1].
10	EMC_D[29] / P2.10	Configure the P2.10 pin[1].
11	EMC_D[30] / P2.11	Configure the P2.11 pin[1].
12	EMC_D[31] / P2.12	Configure the P2.12 pin[1].
24:13	Reserved	Reserved, user software should not write ones to reserved bits.
25	GPIO_0	Configure the general purpose I/O pin GPIO_0[1].
26	GPIO_1	Configure the general purpose I/O pin GPIO_1[1].
27	GPIO_2	Configure the general purpose I/O pin GPIO_2[1].
28	GPIO_3	Configure the general purpose I/O pin GPIO_3[1].
29	GPIO_4	Configure the general purpose I/O pin GPIO_4[1].
30	GPIO_5	Configure the general purpose I/O pin GPIO_5[1].
31	Reserved	Reserved, user software should not write ones to reserved bits.

[1] Bit level definitions: 1 = set pin as output, 0 = set pin as input.

32.5.19 P2 Direction Clear Register (P2_DIR_CLR - 0x4002 8014)

The P2_DIR_CLR register is a write-only register that configures the data direction of PORT 3 GPIO[5:0] pins and PORT 2 EMC_D[31:19]when they are not used for EMC data. P2_DIR_CLR is used in conjunction with P2_DIR_SET.

Writing a one to a bit in P2_DIR_CLR results in the corresponding I/O pin being configured as an input. Writing a zero to a bit in P2_DIR_CLR has no effect.

In the case of EMC_D[31:19], any value written to P2_DIR_CLR applies only if EMC_D[31:19] are selected as GPIOs (P2.0 through P2.12) using the P2_MUX_SET register, which is described in Section 33.4.7.

Table 636. P2 Direction Clear Register (P2_DIR_CLR - 0x4002 8014)

P2_DIR_CLR	Function	Description
31:0		Same functions as P2_DIR_SET.

32.5.20 P2 Direction State Register (P2_DIR_STATE - 0x4002 80018)

The P2_DIR_STATE register is a read-only register that reports the direction selection for GPIO[5:0] pins and EMC_D[31:19] when they are not used for EMC data (P2.0 through P2.12). The value read reflects the result of writes to P2_DIR_SET and P2_DIR_CLR.

A value of zero indicates that the pin is configured as an input. A value of one indicates that the pin is configured as an output. In the case of EMC_D[31:19], the value read from P2_DIR_STATE only applies if these pins have been configured as GPIOs (P2.0 through P2.12) using the P2_MUX_SET register, which is described in Section 33.4.7.

Table 637. P2 Direction State Register (P2_DIR_STATE - 0x4002 80018)

P2_DIR_STATE	Function	Description
31:0		Same functions as P2_DIR_SET.

32.5.21 P3 Input Pin State Register (P3_INP_STATE - 0x4002 8000)

The P3_INP_STATE register is a read-only register that provides the state of a set of general purpose inputs, the inputs of a set of GPIO pins, and alternately the state of some selected peripheral pins.

Table 638. Input Pin State Register (P3_INP_STATE - 0x4002 8000)

P3_INP_STATE	Function	Description
0	GPI_0 / I2S1RX_SDA	Reflects the general purpose input pin GPI_0.
1	GPI_1 / SERVICE_N	Reflects the general purpose input pin GPI_1 / SERVICE_N.
2	GPI_2 / CAP2.0 | (ENET_RXD3)	Reflects the general purpose input pin GPI_2.
3	GPI_3	Reflects the general purpose input pin GPI_3.
4	GPI_4 / SPI1_BUSY	Reflects the general purpose input pin GPI_4 / SPI1_BUSY.
5	GPI_5 / U3_DCD	Reflects the general purpose input pin GPI_5.
6	GPI_6 / HSTIM_CAP | (ENET_RXD2)	Reflects the general purpose input pin GPI_6 / HSTIM_CAP.
7	GPI_7	Reflects the general purpose input pin GPI_7.
8	GPI_8 / KEY_COL6 / SPI2_BUSY | (ENET_RX_DV)	Reflects the general purpose input pin GPI_8 / KEY_COL6 / SPI2_BUSY.
9	GPI_9 / KEY_COL7 | (ENET_COL)	Reflects the general purpose input pin GPI_9 / KEY_COL7
10	GPIO_0	Reflects the general purpose I/O pin GPIO_0.
11	GPIO_1	Reflects the general purpose I/O pin GPIO_1.
12	GPIO_2 / KEY_ROW6 | (ENET_MDC)	Reflects the general purpose I/O pin GPIO_2 / KEY_ROW6
13	GPIO_3 / KEY_ROW7 | (ENET_MDIO)	Reflects the general purpose I/O pin GPIO_3 / KEY_ROW7
14	GPIO_4 / SSEL1 | (LCDVD[22])	Reflects the general purpose I/O pin GPIO_4.
15	GPI_15 / U1_RX | (CAP1.0)	Reflects the state of the input pin GPI_15 / U1_RX.
16	GPI_16 / U2_HCTS | (U3_CTS)	Reflects the state of the input pin GPI_16 / U2_HCTS.
17	GPI_17 / U2_RX | (U3_DSR)	Reflects the state of the input pin GPI_17 / U2_RX.
18	GPI_18 / U3_RX	Reflects the state of the input pin GPI_18 / U3_RX.
19	GPI_19 / U4_RX	Reflects the general purpose input pin GPI_19 / U4_RX.
20	GPI_20 / U5_RX	Reflects the state of the input pin GPI_20 / U5_RX.
21	GPI_21 / U6_IRRX	Reflects the state of the input pin GPI_21 / U6_IRRX.
22	GPI_22 / U7_HCTS / CAP0.1 | (LCDCLKIN)	Reflects the state of the input pin GPI_22 / U7_HCTS.

Table 638. Input Pin State Register (P3_INP_STATE - 0x4002 8000)

P3_INP_STATE	Function	Description
23	GPI_23 / U7_RX / CAP0.0 | (LCDVD[10])	Reflects the state of the input pin GPI_23 / U7_RX.
24	GPIO_5 / SSEL0 / MCFB0	Reflects the general purpose I/O pin GPIO_5.
25	GPI_25 / SPI1_DATIN / MISO0 / MCFB2	Reflects the state of the input pin GPI_25 / SPI1_DATIN.
26	Reserved	The value read from a reserved bit is not defined.
27	GPI_27 / SPI2_DATIN / MISO1 | (LCDVD[21])	Reflects the state of the input pin GPI_27 / SPI2_DATIN.
28	GPI_28 / U3_RI	Reflects the general purpose input pin GPI_28.
31:29	Reserved	The value read from a reserved bit is not defined.

32.5.22 P3 Output Pin Set Register (P3_OUTP_SET - 0x4002 8004)

The P3_OUTP_SET register is a write-only register that allows setting one or more general purpose output GPO[23:0] and GPIO[5:0] pins.

Writing a one to a bit in P3_OUTP_SET results in the corresponding GPO[23:0] output or GPIO[5:0] (if configured as an output) pin being driven high. Writing a zero to a bit in P3_OUTP_SET has no effect.

Table 639. P3 Output Pin Set Register (P3_OUTP_SET - 0x4002 8004)

P3_OUTP_SET	Function	Description
31	Reserved	Reserved, user software should not write ones to reserved bits.
30	GPIO_5 / SSEL0 / MCFB0	Reflects the general purpose I/O pin GPIO_5.
29	GPIO_4 / SSEL1 | (LCDVD[22])	Reflects the general purpose I/O pin GPIO_4.
28	GPIO_3 / KEY_ROW7 | (ENET_MDIO)	Reflects the general purpose I/O pin GPIO_3.
27	GPIO_2 / KEY_ROW6 | (ENET_MDC)	Reflects the general purpose I/O pin GPIO_2.
26	GPIO_1	Reflects the general purpose I/O pin GPIO_1.
25	GPIO_0	Reflects the general purpose I/O pin GPIO_0.
24	Reserved	Reserved, user software should not write ones to reserved bits.
23	GPO_23 / U2_HRTS | (U3_RTS)	Reflects the general purpose output pin GPO_23.
22	GPO_22 / U7_HRTS / LCDVD[14]	Reflects the general purpose output pin GPO_22
21	GPO_21 / U4_TX | (LCDVD[3])	Reflects the general purpose output pin GPO_21.
20	GPO_20	Reflects the general purpose output pin GPO_20.
19	GPO_19	Reflects the general purpose output pin GPO_19.
18	GPO_18 / MC0A | LCDLP	Reflects the general purpose output pin GPO_18.
17	GPO_17	Reflects the general purpose output pin GPO_17.
16	GPO_16 / MC0B | (LCDENAB/LCDM)	Reflects the general purpose output pin GPO_16.
15	GPO_15 / MC1A | (LCDFP)	Reflects the general purpose output pin GPO_15.
14	GPO_14	Reflects the general purpose output pin GPO_14.
13	GPO_13 / MC1B | (LCDDCLK)	Reflects the general purpose output pin GPO_13.
12	GPO_12 / MC2A | (LCDLE)	Reflects the general purpose output pin GPO_12.
11	GPO_11	Reflects the general purpose output pin GPO_11.
10	GPO_10 / MC2B | (LCDPWR)	Reflects the general purpose output pin GPO_10.

Table 639. P3 Output Pin Set Register (P3_OUTP_SET - 0x4002 8004)

P3_OUTP_SET	Function	Description
9	GPO_9 | (LCDVD[9])	Reflects the general purpose output pin GPO_9.
8	GPO_8 | (LCDVD[8])	Reflects the general purpose output pin GPO_8.
7	GPO_7 | (LCDVD[2])	Reflects the general purpose output pin GPO_7.
6	GPO_6 | (LCDVD[18])	Reflects the general purpose output pin GPO_6.
5	GPO_5	Reflects the general purpose output pin GPO_5.
4	GPO_4	Reflects the general purpose output pin GPO_4.
3	GPO_3 | (LCDVD[1])	Reflects the general purpose output pin GPO_3.
2	GPO_2 / T1_MAT.0 | (LCDVD[0])	Reflects the general purpose output pin GPO_2.
1	GPO_1	Reflects the general purpose output pin GPO_1.
0	GPO_0 | (TST_CLK1)	Reflects the general purpose output pin GPO_0.

32.5.23 P3 Output Pin Clear Register (P3_OUTP_CLR - 0x4002 8008)

The P3_OUTP_CLR register is a write-only register that allows clearing one or more general purpose output GPO[32:0] and GPIO[5:0] configured as output pins.

Writing a one to a bit in P3_OUTP_CLR results in the corresponding output GPO[32:0] or GPIO[5:0] (if configured as an output) pin being driven low. Writing a zero to a bit in P3_OUTP_CLR has no effect.

Table 640. P3 Output Pin Clear Register (P3_OUTP_CLR - 0x4002 8008)

P3_OUTP_CLR	Function	Description
31:0		Same functions as P3_OUTP_SET.

32.5.24 P3 Output Pin State Register (P3_OUTP_STATE - 0x4002 800C)

The P3_OUTP_STATE register is a read-only register that provides the state of all general purpose output and GPIO[5:0] (if configured as an output) pins.

Table 641. P3 Output Pin State Register (P3_OUTP_STATE - 0x4002 800C)

P3_OUTP_STATE	Function	Description
31:0		Same functions as P3_OUTP_SET.

33.1 Introduction

There are not enough pins to support the full set of peripheral devices available on the LPC32x0, so the pins are multiplexed and configuration is controlled by registers, which direct pin access to the peripherals in the LPC32x0.

Some peripherals in the LPC32x0 use a single configuration register to connect or disconnect a block of pins to a peripheral. These peripherals are listed here, along with the name of the registers that control the configuration.

- Ethernet MAC, (MAC_CLK_CTRL[4:3]) Section 33.2.1
- LCD controller, (LCD_CFG[8:6]) Section 33.2.2
- UART 3 hardware support for modem control, (UART_CTRL[11]) Section 33.2.3
- SD_Card interface, (MS_CTRL[10]) Section 33.2.4
- EMC controller data lines (P2_MUX_SET[3], P2_MUX_CLR[3]) Section 33.4.7

In addition, there are many configuration registers that determine or control the multiplexing of individual pins to GPI, GPO and GPIO, and peripheral devices. These registers are summarized here.

- Peripheral muxing
  - P_MUX_SET (MAT3, SSP0, SSP2, T3, T1, MCPWM) Section 33.3.1
  - P_MUX_CLR (I2S1, SPI2, SPI1, U7, T2, T0) Section 33.3.2
- Port 0 muxing
  - P0_MUX_SET (I2S0, 12S1) Section 33.4.1
  - P0_MUX_CLR (P0.[7:0]) Section 33.4.2
- Port 1 muxing
  - P1_MUX_SET (P0.[23:0]) Section 33.4.4
  - P1_MUX_CLR (EMC_D[23:0]) Section 33.4.5
- Port 2 muxing
  - P2_MUX_SET (Misc. Alternate functions) Section 33.4.7
  - P2_MUX_CLR (GPIO) Section 33.4.8
- Port 3 muxing
  - P3_MUX_SET (GPO) Section 33.4.10
  - P3_MUX_CLR (MCPWM) Section 33.4.10

The remaining sections in this chapter describe in greater detail the registers that control pin multiplexing in the LPC32x0.

33.2 Peripheral block control registers

The peripheral registers described in this section use a single configuration register to connect or disconnect a block of pins to a given peripheral.

33.2.1 Ethernet MAC Clock Control Register (MAC_CLK_CTRL - 0x4000 4090)

The MAC_CLK_CTRL register controls configuration features of the Ethernet MAC. In the context of multiplexing pins, bits [4:3] in this register enable or disable the pins connected to the Ethernet MAC hardware interface. These two bits can set three possible Ethernet MAC port configurations.

- If bits [4:3] are set to 00 or 10 then the Ethernet MAC is not connected to port pins.
- If bits [4:3] are set to 01 then the port signals used for MII and MIIM are connected to the port pins.
- If bits [4:3] are set to 11 then the port signals used for RMII and MIIM are connected to the port pins.

The MAC_CLK_CTRL register is described in Table 642. The multiplexing bits in the MAC_CLK_CTRL[4:3] register takes overrides the multiplexing of all other functions shared on the pins as shown in Table 643.

Table 642. Ethernet MAC Clock Control Register (MAC_CLK_CTRL 0x4000 4090)

MAC_CLK_CTRL	Name	Description	Reset value
4:3	HDW_INF_CTRL	Ethernet MAC Hardware interface control 00 = Do not connect Ethernet MAC to Port pins. 01 = Connect Ethernet MAC to Port pins in MII Mode. 10 = Do not connect Ethernet MAC to Port pins. 11 = Connect Ethernet MAC to Port pins in RMII Mode.	00
2	MASTER_CLK	Master Interface Clock 0 = Disabled. 1 = Enabled.	0
0	REG_CLK	Control Registers Clock 0 = disabled. 1 = enabled	0
1	SLAVE_CLK	Slave Interface Clock 0 = disabled 1 = enabled	0

Table 643 shows the active pins which are used when the ethernet MAC is enabled and configured to use either an MII interface or an RMII interface.

Table 643. Ethernet MAC pin multiplexing description

Pin Name	Ethernet MAC Interface Function	Type	Pin Description	MII Pin Connected ([4:3] = 01)	RMII Pin Connected ([4:3] = 11)
KEY_ROW3 | ENET_TX_EN	ENET_TX_EN	Output	Transmit data enable.	Y	Y
KEY_ROW2 | ENET_TXD3	ENET_TXD3	Output	Transmit data 3	Y	N
KEY_ROW1 | ENET_TXD2	ENET_TXD2	Output	Transmit data 2	Y	N
KEY_ROW5 | ENET_TXD1	ENET_TXD1	Output	Transmit data 1.	Y	Y
KEY_ROW4 | ENET_TXD0	ENET_TXD0	Output	Transmit data 0	Y	Y
KEY_ROW0 | ENET_TX_ER	ENET_TX_ER	Output	Transmit error	Y	N
KEY_COL0 | ENET_TX_CLK	ENET_TX_CLK	Input	Transmit clock	Y	N
GPI_8 / KEY_COL6 / SPI2_BUSY | ENET_RX_DV	ENET_RX_DV	Input	Receive data valid	Y	N
GPI_2 / CAP2.0 | ENET_RXD3	ENET_RXD3	Input	Receive data 3	Y	N
GPI_6 / HSTIM_CAP | ENET_RXD2	ENET_RXD2	Input	Receive data 2	Y	N
KEY_COL5 | ENET_RXD1	ENET_RXD1	Input	Receive data 1	Y	Y
KEY_COL4 | ENET_RXD0	ENET_RXD0	Input	Receive data 0	Y	Y
KEY_COL2 | ENET_RX_ER	ENET_RX_ER	Input	Receive error	Y	Y
GPI_9 / KEY_COL7 | ENET_COL	ENET_COL	Input	Collision detect	Y	N
KEY_COL3 | ENET_CRS	ENET_CRS	Input	Carrier sense	Y	Y
KEY_COL1 | ENET_RX_CLK / ENET_REF_CLK	ENET_REF_CLK/ ENET_RX_CLK	Input	Reference clock/ Receive clock	Y	Y
GPIO_2 / KEY_ROW6 | ENET_MDC	ENET_MDC	Output	MIIM clock.	Y	Y
GPIO_3 / KEY_ROW7 | ENET_MDIO	ENET_MDIO	Input/ Output	MI data input and output	Y	Y

33.2.2 LCD Configuration register (LCD_CFG, RW - 0x4000 4054)

The LCD_CFG register controls the selection of the output pins needed for different LCD panel configurations, as well as prescaling of the clock used for LCD data generation.The bitfields in the LCD_CFG register are described in Table 644.

The LCD_CFG register takes precedence over all other functions shared on the pins shown in Table 645 and Table 646 and controlled by bits [8:6] shown in Table 644.

Table 644. LCD Configuration register (LCD_CFG, RW - 0x4000 4054)

Bits	Function	Description	Reset value
31:9	reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
8	DISPLAY_TYPE	Sets the Display type: 0 = TFT Display 1 = STN Display	0x0
7:6	MODE_SELECT	Selects output pin group. See Table 645 and Table 646	0x0
5	HCLK_ENABLE	Enables HCLK signal to LCD controller	0x0
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.	0x0

Table 645. Mode Select Bits for TFT Display Type (bit 8 =0)

Screens	Description	Value of Bits 7:6	Number of external pins
Single Panel	12-bit (4:4:4)	0x0	18[1]
Single Panel	16-bit (5:6:5)	0x1	22[1]
Single Panel	16-bit (1:5:5:5)	0x2	24[1]
Single Panel	24-bit	0x3	30[1]

[1] See Table 183 for complete pin listing

Table 646. Mode Select Bits for STN Display Type (bit 8 =1)

Screens	Description	Value of Bits 7:6	Number of external pins
Single Panel	4-bit Monochrome	0x0	10[1]
Single Panel	8-bit Monochrome	0x1	14[1]
Single Panel	Color	0x1	14[1]
Dual Panel	4-bit Monochrome	0x2	14[2]
Dual Panel	8-bit Monochrome	0x3	22[2]
Dual Panel	Color	0x3	22[2]

[1] See Table 181 for complete pin listing

[2] See Table 182 for complete pin listing

33.2.3 UART Control Register (UART_CTRL - 0x4005 4000)

The UART_CTRL register controls some configuration features of the standard UARTs. In the context of multiplexing pins, The UART_CTRL[11] bit enables or disables the pin configuration for the hardware modem control functions of UART 3. If UART_CTRL[11] is set, then UART 3 does not use the modem control pins. If UART_CTRL[11] is cleared, then UART 3 uses modem control pins.

Table 647 shows the control functions in the UART_CTRL register. Table 648 summarizes the pins enabled on UART 3. The UART_CTRL[11] bit takes precedence over all other functions shared on the pins shown in Table 648.

Table 647. UART Control Register (UART_CTRL - 0x4005 4000)

UART_CTRL	Function	Description	Reset value
11	UART3_MD_CTRL	0 = does not use modem control pins. 1 = UART3 uses modem control pins	0
10	HDPX_INV	0 = IRRX6 is not inverted. 1 = IRRX6 is inverted. This inversion comes in addition to the IRRX6_INV controlled inversion.	0
9	HDPX_EN	0 = IRRX6 is not disabled by TXD. 1 = IRRX6 is masked while TXD is low. This is used for stopping IRRXD6 data received from the IrDA transceiver while transmitting (optical reflection suppression).	0
8:6	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
5	UART6_IRDA	0 = UART6 uses the IrDA modulator/demodulator. 1 = UART6 bypasses the IrDA modulator/demodulator.	0
4	IRTX6_INV	0 = The IRTX6 pin is not inverted. 1 = The IRTX6 pin is inverted.	0
3	IRRX6_INV	0 = The IRRX6 pin is not inverted. 1 = The IRRX6 pin is inverted.	0
2	IR_RxLength	0 = The IRDA expects Rx pulses 3/16 of the selected bit period. 1 = The IRDA expects Rx pulses 3/16 of a 115.2 kbps bit period.	0
1	IR_TxLength	0 = The IRDA Tx uses 3/16 of the selected bit period. 1 = The IRDA Tx uses 3/16 of a 115.2 kbps bit period.	0
0	UART5_MODE	0 = The UART5 TX/RX function is only routed to the U5_TX and U5_RX pins. 1 = The UART5 TX/RX function is also routed to the USB D+ and D- pins.	0

Table 648. UART 3 Pin Usage Summary

UART3	Pin Name	Direction	Description	UART_CTRL (bit 11 = 1) connected	UART_CTRL (bit 11 = 0) connected
TxD	U3_TX	Out	Transmitted Data	Y	Y
RxD	GPI_18 / U3_RX	In	Received Data.	Y	Y
DTR	U2_TX | U3_DTR	Out	Data Terminal Ready	Y	N
DSR	GPI_17 /U2_RX | U3_DSR	In	Data Set Ready	Y	N
RTS	GPO_23 / U2_HRTS | U3_RTS	Out	Request to Send	Y	N
CTS	GPI_16 /U2_HCTS | U3_CTS	In	Clear to Send	Y	N
DCD	GPI_5 | U3_DCD	In	Carrier Detect	Y	N
RI	GPI_28 | U3_RI	In	Ring Indicator	Y	N

33.2.4 Memory Card Control register (MS_CTRL - 0x4000 4080)

The MS_CTRL register selects whether the SD card interface is enabled.

It also controls pad pull-up and pull-down, and clocks to the related peripheral blocks. The non-multiplexing features of this register are described in Section 4.11.20

Table 649. Memory Card Control register (MS_CTRL - 0x4000 4080)

Bit	Function	Reset value
31:11	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
10	Disable SD_card pins. If the SD Card interface is not used, this bit (10) should be programmed to 1, bit 9 should be programmed to 0, and bits 6 through 8 should be programmed to 1. 0 = enable SD_card interface pins. 1 = disable SD_card interface pins and enable Peripheral MUX registers.	0
9	Enables clock and pull-ups to MSSDIO pins. If the SD Card interface is not used, this bit should be programmed to 0, and bits 6 through 8 should be programmed to 1. To enable clocking, the SD_card controller requires both bits 5 and 9 be set to 1 0 = MSSDIO pull-up and clock disabled. 1 = MSSDIO pull-up and clock enable.	0
8	MSSDIO2 and MSSDIO3 pad control. 0 = MSSDIO2 and 3 pad has pull-up enabled. 1 = MSSDIO2 and 3 pad has no pull-up.	0
7	MSSDIO1 pad control. 0 = MSSDIO1 pad has pull-up enabled. 1 = MSSDIO1 pad has no pull-up.	0

Table 649. Memory Card Control register (MS_CTRL - 0x4000 4080) *...continued*

Bit	Function	Reset value
6	MSSDIO0/MSBS pad control. 0 = MSSDIO0 pad has pull-up enable. 1 = MSSDIO0 pad has no pull-up.	0
5	SD Card clock control. This bit controls MSSDCLK to the SD Card block. The registers in the peripheral block cannot be accessed if the clock is stopped. To enable clocking, the SD_card controller requires both bits 5 and 9 be set to 1. 0 = Clocks disabled. 1 = Clocks enabled.	0
4	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-
3:0	These register bits control the divider ratio when generating the clock from the ARM PLL output clock. Software must insure that the maximum clock frequency of the targeted device is not exceeded. 0000 = MSSDCLK stopped. Divider in low power mode. 0001 = MSSDCLK equals ARM PLL output clock divided by 1. …… 1110 = MSSDCLK equals ARM PLL output clock divided by 14. 1111 = MSSDCLK equals ARM PLL output clock divided by 15.	0

Table 650. SD card interface pin description

Pin name	SC_Card Function	Type	Description	MS_CTRL (bit 10 = 0)	MS_CTRL (bit 10 = 1)
MS_SCLK | MAT2.0	MS_SCLK	Output	SD card clock output.	SD_Card MS_SCLK	P_MUX Pin
MS_BS | MAT2.1	MS_BS	Input	SD card command input/output.	SD_Card MS_BS	P_MUX Pin
MS_DIO0 | MAT0.0	MS_DIO0	Output	SD card data line.	SD_Card MS_DIO0	P_MUX Pin
MS_DIO1 | MAT0.1	MS_DIO1	Output	SD card data line.	SD_Card MS_DIO1	P_MUX Pin
MS_DIO2 | MAT0.2	MS_DIO2	Output	SD card data line.	SD_Card MS_DIO2	P_MUX Pin
MS_DIO3 | MAT0.3	MS_DIO3	Output	SD card data line.	SD_Card MS_DIO3	P_MUX Pin

33.3 Peripheral MUX Registers description

The registers in Table 655 give control over some individually multiplexed pin functions of some peripheral features available on the LPC32x0. However, several peripheral connections may override the functions controlled by the peripheral control register.

- Some pins controlled by the peripheral mux registers may be overridden by the LCD_CFG[8:6] register bits, which control the LCD hardware interface.
- If the MS_CTRL[10] bit is cleared (SD_card is enabled) the SD card interface overrides the pin functions controlled by the peripheral mux registers.

- If the TIMCLK1_CTRL1[6] bit is set (MCCON is enabled) some of the Motor Control PWM signals overide the signals selected by the P_MUX_SET Register. Specifically the P_MUX_SET[10:9] are overridden by the MCFB1 and MCFB2 signals.

Table 651. Summary of Peripheral Multiplexing registers

Address	Name	Description	Reset state	Access
0x4002 8100	P_MUX_SET	Peripheral multiplexer control set register. Controls the selection of alternate functions on certain pins.	-	WO
0x4002 8104	P_MUX_CLR	Peripheral multiplexer control clear register. Controls the selection of default functions on certain pins.	-	WO
0x4002 8108	P_MUX_STATE	Peripheral multiplexer state register. Reads back the selection of functions on certain pins.	0x0000 0000	RO

33.3.1 Peripheral multiplexer Set register (P_MUX_SET - 0x4002 8100)

The P_MUX_SET register is a write-only register that allows configuring selected pins for one of two peripheral functions. This multiplexer selects between two alternate peripheral functions.

Writing a one to a bit in P_MUX_SET results in the corresponding pin being configured for the alternate function. Writing a zero to a bit in P_MUX_SET has no effect.

Table 652. Peripheral multiplexer Set register (P_MUX_SET - 0x4002 8100)

P_MUX_SET	Pin Name	Description
31:16	Reserved	Reserved, user software should not write ones to reserved bits.
15	U7_TX / MAT1.1 | (LCDVD[11])	1 = Configure as MAT1.1. note: if LCD is enabled this mux bit may be overridden by LCDVD[11]
14:13	Reserved	Reserved, user software should not write ones to reserved bits.
12	SPI1_CLK / SCK0	1 = Configure as SCK0.
11	Reserved	Reserved, user software should not write ones to reserved bits
10	SPI1_DATIN / MISO0 / MCFB2	1 = Configure as MISO0. note: If TIMCLK_CTRL1[6] = 1 Override MISO0 and configure as MCFB2
9	SPI1_DATIO / MOSI0 / MCFB1	1 = Configure as MOSI0. note: If TIMCLK_CTRL1[6] = 1 Override MOSI0 and configure as MCFB1
8	SPI2_CLK / SCK1 | (LCDVD[23])	1 = Configure as SCK1. note: if LCD is enabled this mux bit may be overridden by LCDVD[23]
7	Reserved	Reserved, user software should not write ones to reserved bits
6	SPI2_DATIN / MISO1 | (LCDVD[21])	1 = Configure as MISO1. note: if LCD is enabled this mux bit may be overridden by LCDVD[21]

Table 652. Peripheral multiplexer Set register (P_MUX_SET - 0x4002 8100)

P_MUX_SET	Pin Name	Description
5	SPI2_DATIO / MOSI1 | (LCDVD[20])	1 = Configure as MOSI1. note: if LCD is enabled this mux bit may be overridden by LCDVD[20]
4	I2S1TX_WS / CAP3.0	1 = Configure as CAP3.0.
3	I2S1TX_CLK / MAT3.0	1 = Configure as MAT3.0.
2	I2S1TX_SDA / MAT3.1	1 = Configure as MAT3.1.
1:0	Reserved	Reserved, user software should not write ones to reserved bits.

33.3.2 Peripheral multiplexer Clear register (P_MUX_CLR - 0x4002 8104)

The P_MUX_CLR register is a write-only register that allows configuring selected pins for one of two peripheral functions. This multiplexer selects between two alternate peripheral functions.

Writing a one to a bit in P_MUX_CLR register configures the corresponding pin to the default function. Writing a zero to a bit in P_MUX_CLR has no effect.

Table 653. Peripheral multiplexer Clear register (P_MUX_CLR - 0x4002 8104)

P_MUX_CLR	Function	Description
31:21	Reserved	Reserved, user software should not write ones to reserved bits.
20	(MS_DIO0) | MAT0.0	1 = Configure as MAT0.0. note: if SD_card is enabled MS_DIO0 function overrides this bit
19	(MS_DIO1) | MAT0.1	1 = Configure as MAT0.1. note: if SD_card is enabled MS_DIO1 function overrides this bit
18	(MS_DIO2) | MAT0.2	1 = Configure as MAT0.2. note: if SD_card is enabled MS_DIO2 function overrides this bit
17	(MS_DIO3) | MAT0.3	1 = Configure as MAT0.3. note: if SD_card is enabled MS_DIO3 function overrides this bit
16	Reserved	Reserved, user software should not write ones to reserved bits
15	U7_TX / MAT1.1 | (LCDVD[11])	1 = Configure as U7_TX. note: if LCD is enabled this mux bit may be overridden by LCDVD[11]
14:13	Reserved	Reserved, user software should not write ones to reserved bits.
12	SPI1_CLK / SCK0	1 = Configure as SPI1_CLK.
11	Reserved	Reserved, user software should not write ones to reserved bits
10	SPI1_DATIN / MISO0 / MCFB2	1 = Configure as SPI1_DATIN.
9	SPI1_DATIO / MOSI0 / MCFB1	1 = Configure as SPI1_DATI.
8	SPI2_CLK / SCK1 | (LCDVD[23])	1 = Configure as SPI2_CLK. note: if LCD is enabled this mux bit may be overridden by LCDVD[23]
7	Reserved	Reserved, user software should not write ones to reserved bits

Table 653. Peripheral multiplexer Clear register (P_MUX_CLR - 0x4002 8104)

P_MUX_CLR	Function	Description
6	SPI2_DATIN / MISO1 | (LCDVD[21])	1 = Configure as SPI2_DATIN. note: if LCD is enabled this mux bit may be overridden by LCDVD[21]
5	SPI2_DATIO / MOSI1 | (LCDVD[20])	1 = Configure as SPI2_DATIO. note: if LCD is enabled this mux bit may be overridden by LCDVD[20]
4	I2S1TX_WS / CAP3.0	1 = Configure as I2S1TX_WS.
3	I2S1TX_CLK / MAT3.0	1 = Configure as I2S1TX_CLK.
2	I2S1TX_SDA / MAT3.1	1 = Configure as I2S1TX_SDA.
1:0	Reserved	Reserved, user software should not write ones to reserved bits.

33.3.3 Peripheral multiplexer State register (P_MUX_STATE - 0x4002 8108)

The P_MUX_STATE register is a read-only register that reports the function selected for certain pins. The value read reflects the result of writes to P_MUX_SET and P_MUX_CLR.

A value of zero indicates that the pin is configured to the default function. A value of one indicates that the pin is configured to the alternate function.

Table 654. Peripheral multiplexer State register (P_MUX_STATE - 0x4002 8108)

P_MUX_STATE	Function	Description	Reset value
31:0	-	Same functions as P_MUX_SET.	0

33.4 Port 0, 1, 2, and 3 MUX Register descriptions

The registers in Table 655 control the GPI, GPO, and GPIO pin multiplexing features available on the LPC32x0. However, several peripheral connections may override the functions controlled by the peripheral control register.

- Some pins controlled by the P0, P2, and P3 mux registers may be overridden by the LCD_CFG[8:6] register bits, which control the LCD hardware interface.
- Some pins controlled by the P2, and P3 mux registers may be overridden by the MAC_CLK_CTRL[4:3] register bits, which control the Ethernet MAC hardware interface.

Table 655. Summary of GPIO Multiplexing registers

Address	Name	Description	Reset state	Access
Port 0				
0x4002 8120	P0_MUX_SET	Port 0 multiplexer set register. selects alternate functions on certain Port 0 pins.	-	WO
0x4002 8124	P0_MUX_CLR	Port 0 multiplexer clear register. selects default GPIO function on certain pins.	-	WO
0x4002 8128	P0_MUX_STATE	Port 0 multiplexer state register. Reads selection of alternate or I/O functions on Port 0 pins.	0x0000 0000	RO
Port 1				
0x4002 8130	P1_MUX_SET	Port 1 multiplexer set register. selects alternate functions on certain Port 1 pins.	-	WO
0x4002 8134	P1_MUX_CLR	Port 1 multiplexer clear register. selects default GPIO function on certain Port 1 pins.	-	WO
0x4002 8138	P1_MUX_STATE	Port 1 multiplexer state register. Reads selection of alternate or I/O functions on Port 1 pins.	0x0000 0000	RO
Port 2				
0x4002 8028	P2_MUX_SET	Port 2 multiplexer set register. selects alternate functions on certain Port 2 pins.	-	WO
0x4002 802C	P2_MUX_CLR	Port 2 multiplexer clear register. selects default GPIO functions on certain Port 2 pins.	-	WO
0x4002 8030	P2_MUX_STATE	Port 2 multiplexer state register. Reads selection of alternate or I/O functions on Port 2 pins.	0x0000 0000	RO
Port 3				
0x4002 8110	P3_MUX_SET	Port 3 multiplexer set register. Controls the selection of alternate functions on Port 3 pins.	-	WO
0x4002 8114	P3_MUX_CLR	Port 3 multiplexer clear register. selects default GPIO functions on certain Port 3 pins.	-	WO
0x4002 8118	P3_MUX_STATE	Port 3 multiplexer state register. Reads selection of alternate or I/O functions on Port 3 pins.	0x0000 0000	RO

33.4.1 Port 0 multiplexer Set register (P0_MUX_SET - 0x4002 8120)

The P0_MUX_SET register is a write-only register that allows configuring selected pins for one of two functions. In each case, the default function is a GPIO-type function and the other is a peripheral function.

Writing a one to a bit in P0_MUX_SET results in the corresponding pin being configured for the alternate peripheral function. Writing a zero to a bit in P0_MUX_SET has no effect.

Table 656. Port 0 Multiplexer Set register (P0_MUX_SET - 0x4002 8120)

P0_MUX_SEL	Function	Description
31:8	Reserved	Reserved, user software should not write ones to reserved bits.
7	P0.7 / I2S0TX_WS | (LCDVD[13])	1 = Configure as I2S0TX_WS note: if LCD is enabled this mux bit may be overridden by LCDVD[13]
6	P0.6 / I2S0TX_CLK | (LCDVD[12])	1 = Configure as I2S0TX_CLK note: if LCD is enabled this mux bit may be overridden by LCDVD[12]
5	P0.5 / I2S0TX_SDA | (LCDVD[7])	1 = Configure as I2S0TX_SDA note: if LCD is enabled this mux bit may be overridden by LCDVD[7]
4	P0.4 / I2S0RX_WS | (LCDVD[6])	1 = Configure as I2S0RX_WS note: if LCD is enabled this mux bit may be overridden by LCDVD[6]
3	P0.3 / I2S0RX_CLK | (LCDVD[5])	1 = Configure as I2S0RX_CLK note: if LCD is enabled this mux bit may be overridden by LCDVD[5]
2	P0.2 / I2S0RX_SDA | (LCDVD[4])	1 = Configure as I2S0RX_SDA note: if LCD is enabled this mux bit may be overridden by LCDVD[4]
1	P0.1 / I2S1RX_WS	1 = Configure as I2S1RX_WS
0	P0.0 / I2S1RX_CLK	1 = Configure as I2S1RX_CLK

33.4.2 Port 0 multiplexer Clear register (P0_MUX_CLR - 0x4002 8124)

The P0_MUX_CLR register is a write-only register that allows configuring selected pins for one of two functions. In each case, one function is a GPIO-type function and the other is a peripheral function.

Writing a one to a bit in P0_MUX_CLR results in the corresponding pin being configured for the default function P0[7:0]. Writing a zero to a bit in P0_MUX_CLR has no effect.

Table 657. Port 0 multiplexer Clear register (P0_MUX_CLR - 0x4002 8124)

P0_MUX_SEL	Function	Description
31:8	Reserved	Reserved, user software should not write ones to reserved bits.
7	P0.7 / I2S0TX_WS | (LCDVD[13])	1 = Configure as P0.7 note: if LCD is enabled this mux bit may be overridden by LCDVD[13]
6	P0.6 / I2S0TX_CLK | (LCDVD[12])	1 = Configure as P0.6 note: if LCD is enabled this mux bit may be overridden by LCDVD[12]
5	P0.5 / I2S0TX_SDA | (LCDVD[7])	1 = Configure as P0.5 note: if LCD is enabled this mux bit may be overridden by LCDVD[7]
4	P0.4 / I2S0RX_WS | (LCDVD[6])	1 = Configure as P0.4 note: if LCD is enabled this mux bit may be overridden by LCDVD[6]

Table 657. Port 0 multiplexer Clear register (P0_MUX_CLR - 0x4002 8124)

P0_MUX_SEL	Function	Description
3	P0.3 / I2S0RX_CLK | (LCDVD[5])	1 = Configure as P0.3 note: if LCD is enabled this mux bit may be overridden by LCDVD[5]
2	P0.2 / I2S0RX_SDA | (LCDVD[4])	1 = Configure as P0.2 note: if LCD is enabled this mux bit may be overridden by LCDVD[4]
1	P0.1 / I2S1RX_WS	1 = Configure as P0.1
0	P0.0 / I2S1RX_CLK	1 = Configure as P0.0

33.4.3 Port 0 multiplexer State register (P0_MUX_STATE - 0x4002 8128)

The P0_MUX_STATE register is a read-only register that reports the function selected for certain pins. The value read reflects the result of writes to P0_MUX_SET and P0_MUX_CLR.

A value of zero indicates that the pin is configured to the default GPIO P0.[7:0] function. A value of one indicates that the pin is configured to the alternate function.

Table 658. Port 0 Multiplexer State register (P0_MUX_STATE - 0x4002 8128)

P0_MUX_STATE	Function	Description	Reset value
31:0	-	Same functions as P0_MUX_SET.	0

33.4.4 Port 1 multiplexer Set register (P1_MUX_SET - 0x4002 8130)

The P1_MUX_SET register is a write-only register that allows configuring selected pins for one of two functions. In each case, one function is an EMC address and the other is a GPIO-type function.

Writing a one to a bit in P1_MUX_SET results in the corresponding pin being configured as a GPIO function P1[23:0]. Writing a zero to a bit in P1_MUX_SET has no effect.

Table 659. Port 1 Multiplexer Set register (P1_MUX_SET - 0x4002 8130)

P1_MUX_SET	Function	Description
31:24	Reserved	Reserved, user software should not write ones to reserved bits.
23	EMC_A[23] / P1.23	1: Configure as P1.23
22	EMC_A[22] / P1.22	1: Configure as P1.22
21	EMC_A[21] / P1.21	1: Configure as P1.21
20	EMC_A[20] / P1.20	1: Configure as P1.20
19	EMC_A[19] / P1.19	1: Configure as P1.19
18	EMC_A[18] / P1.18	1: Configure as P1.18
17	EMC_A[17] / P1.17	1: Configure as P1.17
16	EMC_A[16] / P1.16	1: Configure as P1.16
15	EMC_A[15] / P1.15	1: Configure as P1.15
14	EMC_A[14] / P1.14	1: Configure as P1.14
13	EMC_A[13] / P1.13	1: Configure as P1.13

Table 659. Port 1 Multiplexer Set register (P1_MUX_SET - 0x4002 8130)

P1_MUX_SET	Function	Description
12	EMC_A[12] / P1.12	1: Configure as P1.12
11	EMC_A[11] / P1.11	1: Configure as P1.11
10	EMC_A[10] / P1.10	1: Configure as P1.10
9	EMC_A[09] / P1.9	1: Configure as P1.9
8	EMC_A[08] / P1.8	1: Configure as P1.8
7	EMC_A[07] / P1.7	1: Configure as P1.7
6	EMC_A[06] / P1.6	1: Configure as P1.6
5	EMC_A[05] / P1.5	1: Configure as P1.5
4	EMC_A[04] / P1.4	1: Configure as P1.4
3	EMC_A[03] / P1.3	1: Configure as P1.3
2	EMC_A[02] / P1.2	1: Configure as P1.2
1	EMC_A[01] / P1.1	1: Configure as P1.1
0	EMC_A[00] / P1.0	1: Configure as P1.0

33.4.5 Port 1 multiplexer Clear register (P1_MUX_CLR - 0x4002 8134)

The P1_MUX_CLR register is a write-only register that allows configuring selected pins for one of two functions. In each case, one function is an EMC address and the other is a GPIO-type function.

Writing a one to a bit in P1_MUX_CLR results in the corresponding pin being configured for the default function as an EMC address. Writing a zero to a bit in P1_MUX_CLR has no effect.

Table 660. P1 multiplexer Clear register (P1_MUX_CLR - 0x4002 8134)

P1_MUX_CLR	Function	Description
31:24	Reserved	Reserved, user software should not write ones to reserved bits.
23	EMC_A[23] / P1.23	1 = Configure as EMC_A[23]
22	EMC_A[22] / P1.22	1 = Configure as EMC_A[22]
21	EMC_A[21] / P1.21	1 = Configure as EMC_A[21]
20	EMC_A[20] / P1.20	1 = Configure as EMC_A[20]
19	EMC_A[19] / P1.19	1 = Configure as EMC_A[19]
18	EMC_A[18] / P1.18	1 = Configure as EMC_A[18]
17	EMC_A[17] / P1.17	1 = Configure as EMC_A[17]
16	EMC_A[16] / P1.16	1 = Configure as EMC_A[16]
15	EMC_A[15] / P1.15	1 = Configure as EMC_A[15]
14	EMC_A[14] / P1.14	1 = Configure as EMC_A[14]
13	EMC_A[13] / P1.13	1 = Configure as EMC_A[13]
12	EMC_A[12] / P1.12	1 = Configure as EMC_A[12]
11	EMC_A[11] / P1.11	1 = Configure as EMC_A[11]
10	EMC_A[10] / P1.10	1 = Configure as EMC_A[10]
9	EMC_A[09] / P1.9	1 = Configure as EMC_A[09]
8	EMC_A[08] / P1.8	1 = Configure as EMC_A[08]

Table 660. P1 multiplexer Clear register (P1_MUX_CLR - 0x4002 8134)

P1_MUX_CLR	Function	Description
7	EMC_A[07] / P1.7	1 = Configure as EMC_A[07]
6	EMC_A[06] / P1.6	1 = Configure as EMC_A[06]
5	EMC_A[05] / P1.5	1 = Configure as EMC_A[05]
4	EMC_A[04] / P1.4	1 = Configure as EMC_A[04]
3	EMC_A[03] / P1.3	1 = Configure as EMC_A[03]
2	EMC_A[02] / P1.2	1 = Configure as EMC_A[02]
1	EMC_A[01] / P1.1	1 = Configure as EMC_A[01]
0	EMC_A[00] / P1.0	1 = Configure as EMC_A[00]

33.4.6 Port 1 multiplexer State register (P1_MUX_STATE - 0x4002 8138)

The P1_MUX_STATE register is a read-only register that reports the function selected for certain pins. The value read reflects the result of writes to P1_MUX_SET and P1_MUX_CLR.

A value of zero indicates that the pin is configured to the default function as an EMC address pin. A value of one indicates that the pin is configured to the alternate P1[23:0] GPIO function.

Table 661. P1 Multiplexer State register (P1_MUX_STATE - 0x4002 8138)

P1_MUX_STATE	Function	Description	Reset value
23:0	-	Same functions as P1_MUX_SET.	0
31:24	Reserved	Reserved bits	-

33.4.7 Port 2 multiplexer Set register (P2_MUX_SET - 0x4002 8028)

The P2_MUX_SET register is a write-only register that allows configuring selected pins for one of two functions. One bit in this register, EMC_D_SEL enables all EMC data lines, the other bits in this register select either an alternate GPIO-type function or an alternate peripheral function. The EMC data pins can be used as general purpose GPIO when 16 bit SDRAM or DDRAM is used.

Note: this register also controls three Port 3 pins; one GPO and four GPIO pins.

Writing a one to P2_MUX_SET bits 0, 1, 2, 4 or 5 configures the GPO or GPIO pins as the alternate function.

Writing a one to bit 3 in the P2_MUX_SET register results in all of the corresponding EMC_D[31:19] pins being configured as GPIO pins P2[12:0].

Writing a zero to a bit in P2_MUX_SET has no effect.

Table 662. Port 2 Multiplexer Set register (P2_MUX_SET - 0x4002 8028)

P2_MUX_SET	Function	Description
31:6	Reserved	Reserved, user software should not write ones to reserved bits.
5	GPIO_5 / SSEL0 / MCFB0	1 = pin connected to SSEL0. note: If TIMCLK_CTRL1[6] = 1 Override SSEL0 and configure as MCFB0
4	GPIO_4 / SSEL1 | (LCDVD[22])	1 = pin connected to SSEL1. note: if LCD is enabled this mux bit is overridden by LCDVD[22]
3	EMC_D_SEL	1 = EMC_D[31:19] pins are connected to the GPIO block. These pins can be used as general purpose GPIO when a 16 bit SDRAM or DDRAM is used.
2	GPO_21 / U4_TX | (LCDVD[3])	1 = Configure pin as U4_TX. note: if LCD is enabled this mux bit is overridden by LCDVD[3]
1	GPIO_3 / KEY_ROW7 | (ENET_MDIO)	1 = Configure pin as KeyScan Row[7]. note: If Ethernet MAC is enabled this mux bit is overridden by ENET_MDIO function.
0	GPIO_2 / KEY_ROW6 | (ENET_MDC)	1 = Configure pin as KeyScan Row[6]. note: If Ethernet MAC is enabled this mux bit is overridden by ENET_MDC function.

33.4.8 Port 2 multiplexer Clear register (P2_MUX_CLR - 0x4002 802C)

The P2_MUX_CLR register is a write-only register that allows configuring selected pins for one of two functions. One function is either a EMC data line select or a GPIO-type function and the other is a peripheral function. The EMC data pins can be used as general purpose GPIO when 16 bit SDRAM or DDRAM is used.

Note: this register also controls three Port 3 pins; one GPO and four GPIO pins.

Writing a one to bit 3 in the P2_MUX_CLR register results in all of the corresponding EMC_D[31:19] pins being configured as EMC_D[31:19] pins.

Writing a one to P2_MUX_CLR bits 0,1, 2, 4, or 5 configures the GPO or GPIO pins as the GPO or GPIO depending on the bit.

Writing a zero to a bit in the P2_MUX_CLR register has no effect.

Table 663. P2 multiplexer Clear register (P2_MUX_CLR - 0x4002 802C)

P2_MUX_CLR	Function	Description
31:6	Reserved	Reserved, user software should not write ones to reserved bits.
5	GPIO_5 / SSEL0 / MCFB0	1 = pin connected to GPIO_5.
4	GPIO_4 / SSEL1 | (LCDVD[22])	1 = pin connected to GPIO_4. note: if LCD is enabled this mux bit is overridden by LCDVD[22]
3	EMC_D_SEL	1 = EMC_D[31:19] are connected to the SDRAM controller.

Table 663. P2 multiplexer Clear register (P2_MUX_CLR - 0x4002 802C)

P2_MUX_CLR	Function	Description
2	GPO_21 / U4_TX | (LCDVD[3])	1 = Configure as GPO_21 note: if LCD is enabled this mux bit is overridden by LCDVD[3]
1	GPIO_3 / KEY_ROW7 | (ENET_MDIO)	1 = Configure as GPIO_3 note: If Ethernet MAC is enabled this mux bit is overridden by ENET_MDIO function.
0	GPIO_2 / KEY_ROW6 | (ENET_MDC)	1 = Configure as GPIO_2 note: If Ethernet MAC is enabled this mux bit is overridden by ENET_MDC function.

33.4.9 Port 2 multiplexer State register (P2_MUX_STATE - 0x4002 8030)

The P2_MUX_STATE register is a read-only register that reports the function selected for certain pins. The value read reflects the result of writes to P2_MUX_SET and P2_MUX_CLR.

A value of zero indicates that the pin is configured to the default function. A value of one indicates that the pin is configured to the alternate peripheral function.

Table 664. P2 multiplexer State register (P2_MUX_STATE - 0x4002 8030)

P2_MUX_STATE	Function	Description
31:0		Same functions as P2_MUX_SET.

33.4.10 Port 3 multiplexer Set register (P3_MUX_SET - 0x4002 8110)

The P3_MUX_SET register is a write-only register that allows configuring selected pins for one of two functions. In each case, one function is a GPO function and the other is a peripheral function.

Writing a one to a bit in P3_MUX_SET results in the corresponding pin being configured for the alternate function. Writing a zero to a bit in P3_MUX_SET has no effect.

Table 665. P3 multiplexer Set register (P3_MUX_SET - 0x4002 8028)

P3_MUX_SET	Function	Description
31:19	Reserved	Reserved, user software should not write ones to reserved bits.
18	GPO_18 / MC0A / LCDLP	1 = Configure as MC0A, if TIMCLK_CTRL1[6] = 1. note: if LCD is enabled this mux bit is overridden by LCDVD[0]
17	Reserved	Reserved, user software should not write ones to reserved bits.
16	GPO_16 / MC0B / LCDENAB / LCDM	1 = Configure as MC0B, if TIMCLK_CTRL1[6] = 1. note: if LCD is enabled this mux bit is overridden by LCDENAB / LCDM
15	GPO_15 / MC1A / LCDFP	1 = Configure as MC1A, if TIMCLK_CTRL1[6] = 1. note: if LCD is enabled this mux bit is overridden by LCDFP
14	Reserved	Reserved, user software should not write ones to reserved bits.
13	GPO_13 / MC1B / LCDDCLK	1 = Configure as MC1B, if TIMCLK_CTRL1[6] = 1. note: if LCD is enabled this mux bit is overridden by LCDDCLK

Table 665. P3 multiplexer Set register (P3_MUX_SET - 0x4002 8028)

P3_MUX_SET	Function	Description
12	GPO_12 / MC2A / LCDLE	1 = Configure as MC2A, if TIMCLK_CTRL1[6] = 1. note: if LCD is enabled this mux bit is overridden by LCDLE
11	Reserved	Reserved, user software should not write ones to reserved bits.
10	GPO_10 / MC2B / LCDPWR	1 = Configure as MC2B, if TIMCLK_CTRL1[6] = 1. note: if LCD is enabled this mux bit is overridden by LCDPWR
9:3	Reserved	Reserved, user software should not write ones to reserved bits.
2	GPO_2 / MAT1.0 | (LCDVD[0])	1 = Configure as MAT1.0. note: if LCD is enabled this mux bit is overridden by LCDVD[0]
1	Reserved	Reserved, user software should not write ones to reserved bits.

33.4.11 P3 Multiplexer Clear register (P3_MUX_CLR - 0x4002 8114)

The P3_MUX_CLR register is a write-only register that allows configuring selected pins for one of two functions. In each case, one function is a GPIO-type function and the other is a peripheral function.

Writing a one to a bit in P3_MUX_CLR results in the corresponding pin being configured for the default GPO, GPI or GPIO function. Writing a zero to a bit in P3_MUX_CLR has no effect.

Table 666. P3 multiplexer Clear register (P3_MUX_CLR - 0x4002 8114)

P3_MUX_CLR	Function	Description
31:19	Reserved	Reserved, user software should not write ones to reserved bits.
18	GPO_18 / MC0A / LCDLP	1 = Configure as GPO_18. note: if LCD is enabled this mux bit is overridden by LCDVD[0]
17	Reserved	Reserved, user software should not write ones to reserved bits.
16	GPO_16 / MC0B / LCDENAB / LCDM	1 = Configure as GPO_16. note: if LCD is enabled this mux bit is overridden by LCDENAB / LCDM
15	GPO_15 / MC1A / LCDFP	1 = Configure as GPO_15. note: if LCD is enabled this mux bit is overridden by LCDFP
14	Reserved	Reserved, user software should not write ones to reserved bits.
13	GPO_13 / MC1B / LCDDCLK	1 = Configure as GPO_13. note: if LCD is enabled this mux bit is overridden by LCDDCLK
12	GPO_12 / MC2A / LCDLE	1 = Configure as GPO_12. note: if LCD is enabled this mux bit is overridden by LCDLE
11	Reserved	Reserved, user software should not write ones to reserved bits.
10	GPO_10 / MC2B / LCDPWR	1 = Configure as GPO_10 note: if LCD is enabled this mux bit is overridden by LCDPWR

Table 666. P3 multiplexer Clear register (P3_MUX_CLR - 0x4002 8114)

P3_MUX_CLR	Function	Description
9:3	Reserved	Reserved, user software should not write ones to reserved bits.
2	GPO_2 / MAT1.0 | (LCDVD[0])	1 = Configure as GPO_2. note: if LCD is enabled this mux bit is overridden by LCDVD[0]
1:0	Reserved	Reserved, user software should not write ones to reserved bits.

33.4.12 Port 3 Multiplexer State register (P3_MUX_STATE - 0x4002 8118)

The P3_MUX_STATE register is a read-only register that reports the function selected for certain pins. The value read reflects the result of writes to P3_MUX_SET and P3_MUX_CLR.

A value of zero indicates that the pin is configured to the default function. A value of one indicates that the pin is configured to the alternate function.

Table 667. Port 3 Multiplexer State register (P3_MUX_STATE - 0x4002 8118)

P3_MUX_STATE	Function	Description
31:0		Same functions as P3_MUX_SET.

34.1 LPC32x0 pinout for the TFBGA296 package

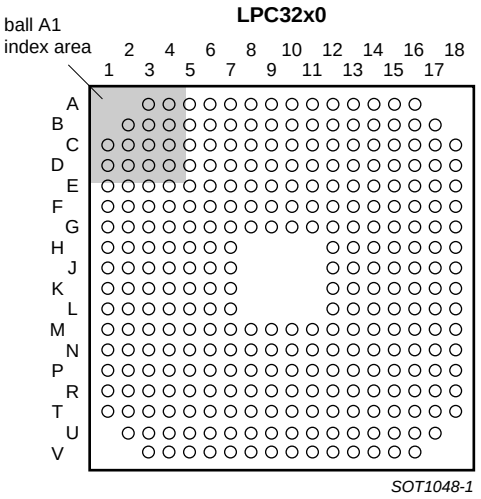


Fig 128.Pin configuration for SOT1048-1 (TFBGA296)

Table 668. Pin allocation table (TFBGA296)

Pin	Symbol	Pin	Symbol	Pin	Symbol
Row A					
				A3	I2C2_SCL
A4	I2S1TX_CLK/MAT3[0]	A5	I2C1_SCL	A6	MS_BS/MAT2[1]
A7	MS_DIO1/MAT0[1]	A8	MS_DIO0/MAT0[0]	A9	SPI2_DATIO/MOSI1/LCDVD[20]
A10	SPI2_DATIN/MISO1/LCDVD[21]/ GPI_27	A11	GPIO_1	A12	GPIO_
A13	GPO_21/U4_TX/LCDVD[3]	A14	GPO_15/MCOA1/LCDFP	A15	GPO_7/LCDVD[2]
A16	GPO_6/LCDVD[18]				
Row B					
		B2	GPO_20	B3	GPO_5
B4	I2S1TX_WS/CAP3[0]	B5	P0[0]/I2S1RX_CLK	B6	I2C1_SDA
B7	MS_SCLK/MAT2[0]	B8	MS_DIO2/MAT0[2]	B9	SPI1_DATIO/MOSI0/MCI2
B10	SPI2_CLK/SCK1/LCDVD[23]	B11	GPIO_4/SSEL1/LCDVD[22]	B12	GPO_12/MCOA2/LCDLE
B13	GPO_13/MCOB1/LCDDCLK	B14	GPO_2/MAT1[0]/LCDVD[0]	B15	GPI_19/U4_RX
B16	GPI_8/KEY_COL6/ SPI2_BUSY/ENET_RX_DV	B17	n.c.		
Row C					
C1	$\overline{\text{FLASH\_RD}}$	C2	GPO_19	C3	GPO_0 /TST_CLK1
C4	$\overline{\text{USB\_ATX\_INT}}$	C5	USB_SE0_ VM/U5_TX	C6	TST_CLK2

Table 668. Pin allocation table (TFBGA296)

Pin	Symbol	Pin	Symbol	Pin	Symbol
C7	GPI_6/HSTIM_CAP/ ENET_RXD2	C8	MS_DIO3/MAT0[3]	C9	SPI1_CLK/SCK0
C10	SPI1_DATIN/MISO0/GPI_25/ MCI1	C11	GPIO_3/KEY_ROW7/ ENET_MDIO	C12	GPO_9/LCDVD[9]
C13	GPO_8/LCDVD[8]	C14	GPI_2/CAP2[0]/ ENET_RXD3	C15	GPI_1/$\overline{\text{SERVICE}}$
C16	GPI_0/I2S1RX_SDA	C17	KEY_ROW4/ENET_TXD0	C18	KEY_ROW5/ENET_TXD1
Row D					
D1	FLASH_RDY	D2	FLASH_ALE	D3	GPO_14
D4	GPO_1	D5	USB_DAT_VP/U5_RX	D6	$\overline{\text{USB\_OE\_TP}}$
D7	P0[1]/I2S1RX_WS	D8	GPO_4	D9	GPIO_2/KEY_ROW6/ENET_MDC
D10	GPO_16/MCOB0/LCDENAB/ LCDM	D11	GPO_18/MCOA0/LCDLP	D12	GPO_3/LCDVD[1]
D13	GPI_7/CAP4[0]/$\overline{\text{MCABORT}}$	D14	PWM_OUT1/LCDVD[16]	D15	PWM_OUT2/INTSTAT/LCDVD[19]
D16	KEY_ROW3/ENET_TX_EN	D17	KEY_COL2/ENET_RX_ER	D18	KEY_COL3/ENET_CRS
Row E					
E1	FLASH_IO[3]	E2	FLASH_IO[7]	E3	$\overline{\text{FLASH\_CE}}$
E4	I2C2_SDA	E5	USB_I2C_SCL	E6	USB_I2C_SDA
E7	I2S1TX_SDA/MAT3[1]	E8	GPO_11	E9	GPIO_5/SSEL0/MCI0
E10	GPO_22/U7_HRTS/LCDVD[14]	E11	GPO_10/MCOB2/LCDPWR	E12	GPI_9/KEY_COL7/ENET_COL
E13	GPI_4/SPI1_BUSY	E14	KEY_ROW1/ENET_TXD2	E15	KEY_ROW0/ENET_TX_ER
E16	KEY_COL1/ENET_RX_CLK/ ENET_REF_CLK	E17	U7_RX/CAP0[0]/LCDVD[10]/ GPI_23	E18	U7_TX/MAT1[1]/LCDVD[11]
Row F					
F1	FLASH_IO[2]	F2	$\overline{\text{FLASH\_WR}}$	F3	FLASH_CLE
F4	GPI_3	F5	VSS_IOC	F6	VSS_IOB
F7	VDD_IOC	F8	VDD_IOB	F9	VDD_IOD
F10	VSS_IOD	F11	VSS_IOD	F12	VSS_IOD
F13	VDD_IOD	F14	KEY_ROW2/ENET_TXD3	F15	KEY_COL0/ENET_TX_CLK
F16	KEY_COL5/ENET_RXD1	F17	U6_IRRX/GPI_21	F18	U5_RX/GPI_20
Row G					
G1	$\overline{\text{EMC\_DYCS1}}$	G2	FLASH_IO[5]	G3	FLASH_IO[6]
G4	$\overline{\text{RESOUT}}$	G5	VSS_IOC	G6	VDD_IOC
G7	VDD_CORE	G8	VSS_CORE	G9	VDD_CORE
G10	VSS_CORE	G11	VDD_CORE	G12	VSS_CORE
G13	U7_HCTS/CAP0[1]/ LCDCLKIN/GPI_22	G14	DBGEN	G15	KEY_COL4/ENET_RXD0
G16	U6_IRTX	G17	SYSCLKEN/LCDVD[15]	G18	JTAG_TMS
Row H					
H1	$\overline{\text{EMC\_OE}}$	H2	FLASH_IO[0]	H3	FLASH_IO[1]
H4	FLASH_IO[4]	H5	VSS_IOC	H6	VDD_IOC
H7	VSS_CORE				
				H12	VSS_IOD

Table 668. Pin allocation table (TFBGA296)

Pin	Symbol	Pin	Symbol	Pin	Symbol
H13	VDD_IOA	H14	JTAG_TCK	H15	U5_TX
H16	HIGHCORE/LCDVD[17]	H17	JTAG_NTRST	H18	JTAG_RTCK
Row J					
J1	EMC_A[20]/P1[20]	J2	EMC_A[21]/P1[21]	J3	EMC_A[22]/P1[22]
J4	EMC_A[23]/P1[23]	J5	VDD_IOC	J6	VDD_EMC
J7	VDD_CORE				
				J12	VDD_CORE
J13	VDD_IOA	J14	U3_RX/GPI_18	J15	JTAG_TDO
J16	JTAG_TDI	J17	U3_TX	J18	U2_HCTS/U3_CTS/GPI_16
Row K					
K1	EMC_A[19]/P1[19]	K2	EMC_A[18]/P1[18]	K3	EMC_A[16]/P1[16]
K4	EMC_A[17]/P1[17]	K5	VSS_EMC	K6	VDD_EMC
K7	VDD_EMC				
				K12	VSS_CORE
K13	VSS_IOA	K14	VDD_RTC	K15	U1_RX/CAP1[0]/GPI_15
K16	U1_TX	K17	U2_TX/U3_DTR	K18	U2_RX/U3_DSR/GPI_17
Row L					
L1	EMC_A[15]/P1[15]	L2	EMC_CKE1	L3	EMC_A[0]/P1[0]
L4	EMC_A[1]/P1[1]	L5	VSS_EMC	L6	VDD_EMC
L7	VSS_CORE				
				L12	VDD_COREFXD
L13	VDD_RTCCORE	L14	VSS_RTCCORE	L15	P0[4]/I2S0RX_WS/LCDVD[6]
L16	P0[5]/I2S0TX_SDA/LCDVD[7]	L17	P0[6]/I2S0TX_CLK/ LCDVD[12]	L18	P0[7]/I2S0TX_WS/LCDVD[13]
Row M					
M1	EMC_A[2]/P1[2]	M2	EMC_A[3]/P1[3]	M3	EMC_A[4]/P1[4]
M4	EMC_A[8]/P1[8]	M5	VSS_EMC	M6	VDD_EMC
M7	VDD_CORE	M8	VDD_EMC	M9	VSS_CORE
M10	VSS_CORE	M11	VDD_CORE	M12	VSS_CORE
M13	VDD_COREFXD	M14	$\overline{\text{RESET}}$	M15	ONSW
M16	GPO_23/U2_HRTS/U3_RTS	M17	P0[2]/I2S0RX_SDA/ LCDVD[4]	M18	P0[3]/I2S0RX_CLK/LCDVD[5]
Row N					
N1	EMC_A[5]/P1[5]	N2	EMC_A[6]/P1[6]	N3	EMC_A[7/P1[7]
N4	EMC_A[12]/P1[12]	N5	VSS_EMC	N6	VSS_EMC
N7	VDD_EMC	N8	VDD_EMC	N9	VDD_EMC
N10	VDD_EMC	N11	VDD_EMC	N12	VDD_AD
N13	VDD_AD	N14	VDD_FUSE	N15	VDD_RTCOSC
N16	GPI_5/U3_DCD	N17	GPI_28/U3_RI	N18	GPO_17
Row P					
P1	EMC_A[9]/P1[9]	P2	EMC_A[10]/P1[10]	P3	EMC_A[11]/P1[11]

Table 668. Pin allocation table (TFBGA296)

Pin	Symbol	Pin	Symbol	Pin	Symbol
P4	EMC_DQM[1]	P5	EMC_DQM[3]	P6	VSS_EMC
P7	VSS_EMC	P8	VSS_EMC	P9	VSS_EMC
P10	VSS_EMC	P11	VSS_EMC	P12	EMC_BLS[3]
P13	VSS_AD	P14	VSS_OSC	P15	VDD_PLLUSB
P16	RTCX_IN	P17	RTCX_OUT	P18	VSS_RTCOSC
Row R					
R1	EMC_A[13]/P1[13]	R2	EMC_A[14]/P1[14]	R3	EMC_DQM[0]
R4	$\overline{\text{EMC\_WR}}$	R5	$\overline{\text{EMC\_CAS}}$	R6	$\overline{\text{EMC\_DYCS0}}$
R7	EMC_D[1]	R8	EMC_D[7]	R9	EMC_D[17]/EMC_DQS1
R10	EMC_D[24]/P2[5]	R11	$\overline{\text{EMC\_CS1}}$	R12	EMC_BLS[2]
R13	TS_XP	R14	PLL397_LOOP	R15	SYSX_OUT
R16	VSS_PLLUSB	R17	VDD_PLLHCLK	R18	VSS_PLLHCLK
Row T					
T1	EMC_DQM[2]	T2	$\overline{\text{EMC\_RAS}}$	T3	EMC_CLK
T4	EMC_CLKIN	T5	EMC_D[2]	T6	EMC_D[6]
T7	EMC_D[11]	T8	EMC_D[14]	T9	EMC_D[20]/P2[1]
T10	EMC_D[23]/P2[4]	T11	EMC_D[27]/P2[8]	T12	$\overline{\text{EMC\_CS2}}$
T13	EMC_BLS[1]	T14	ADIN1/TS_XM	T15	VSS_PLL397
T16	VDD_PLL397	T17	SYSX_IN	T18	VDD_OSC
Row U					
		U2	n.c.	U3	EMC_CKE0
U4	EMC_D[0]	U5	EMC_D[3]	U6	EMC_D[9]
U7	EMC_D[12]	U8	EMC_D[15]	U9	EMC_D[19]/P2[0]
U10	EMC_D[22]/P2[3]	U11	EMC_D[26]/P2[7]	U12	EMC_D[30]/P2[11]
U13	$\overline{\text{EMC\_CS0}}$	U14	EMC_BLS[0]	U15	ADIN0/TS_YM
U16	TS_YP	U17	n.c.		
Row V					
				V3	EMC_D[4]
V4	EMC_D[5]	V5	EMC_D[8]	V6	EMC_D[10]
V7	EMC_D[13]	V8	EMC_D[16]/EMC_DQS0	V9	EMC_D[18]/$\overline{\text{EMC\_CLK}}$
V10	EMC_D[21]/P2[2]	V11	EMC_D[25]/P2[6]	V12	EMC_D[28]/P2[9]
V13	EMC_D[29]/P2[10]	V14	EMC_D[31]/P2[12]	V15	$\overline{\text{EMC\_CS3}}$
V16	ADIN2/TS_AUX_IN				

Table 669. LPC32x0 pin allocation for TFBGA296 package by pin name

Symbol	Ball #	Symbol	Ball #
ADIN0/TS_YM	U15	JTAG_TCK	H14
ADIN1/TS_XM	T14	JTAG_TDI	J16
ADIN2/TS_AUX_IN	V16	JTAG_TDO	J15
DBGEN	G14	JTAG_TMS	G18
EMC_A[00] / P1.0	L3	KEY_COL0 / ENET_TX_CLK	F15

Table 669. LPC32x0 pin allocation for TFBGA296 package by pin name

Symbol	Ball #	Symbol	Ball #
EMC_A[01] / P1.1	L4	KEY_COL1 / ENET_RX_CLK / ENET_REF_CLK	E16
EMC_A[02] / P1.2	M1	KEY_COL2 / ENET_RX_ER	D17
EMC_A[03] / P1.3	M2	KEY_COL3 / ENET_CRS	D18
EMC_A[04] / P1.4	M3	KEY_COL4 / ENET_RXD0	G15
EMC_A[05] / P1.5	N1	KEY_COL5 / ENET_RXD1	F16
EMC_A[06] / P1.6	N2	KEY_ROW0 / ENET_TX_ER	E15
EMC_A[07] / P1.7	N3	KEY_ROW1 / ENET_TXD2	E14
EMC_A[08] / P1.8	M4	KEY_ROW2 / ENET_TXD3	F14
EMC_A[09] / P1.9	P1	KEY_ROW3 / ENET_TX_EN	D16
EMC_A[10] / P1.10	P2	KEY_ROW4 / ENET_TXD0	C17
EMC_A[11] / P1.11	P3	KEY_ROW5 / ENET_TXD1	C18
EMC_A[12] / P1.12	N4	MS_BS / MAT2.1	A6
EMC_A[13] / P1.13	R1	MS_DIO0 / MAT0.0	A8
EMC_A[14] / P1.14	R2	MS_DIO1 / MAT0.1	A7
EMC_A[15] / P1.15	L1	MS_DIO2 / MAT0.2	B8
EMC_A[16] / P1.16	K3	MS_DIO3 / MAT0.3	C8
EMC_A[17] / P1.17	K4	MS_SCLK / MAT2.0	B7
EMC_A[18] / P1.18	K2	N.C.	B17
EMC_A[19] / P1.19	K1	N.C.	U17
EMC_A[20] / P1.20	J1	N.C.	U2
EMC_A[21] / P1.21	J2	ONSW	M15
EMC_A[22] / P1.22	J3	P0.0 / I2S1RX_CLK	B5
EMC_A[23] / P1.23	J4	P0.1 / I2S1RX_WS	D7
EMC_BLS[0]	U14	P0.2 / I2S0RX_SDA / LCDVD[4]	M17
EMC_BLS[1]	T13	P0.3 / I2S0RX_CLK / LCDVD[5]	M18
EMC_BLS[2]	R12	P0.4 / I2S0RX_WS / LCDVD[6]	L15
EMC_BLS[3]	P12	P0.5 / I2S0TX_SDA / LCDVD[7]	L16
EMC_CAS_N	R5	P0.6 / I2S0TX_CLK / LCDVD[12]	L17
EMC_CKE0	U3	P0.7 / I2S0TX_WS / LCDVD[13]	L18
EMC_CKE1	L2	PLL397_LOOP	R14
EMC_CLK	T3	PWM_OUT1 / LCDVD[16]	D14
EMC_CLKIN	T4	PWM_OUT2 / LCDVD[19]	D15
EMC_CS0_N	U13	RESET_N	M14
EMC_CS1_N	R11	RESOUT_N	G4
EMC_CS2_N	T12	RTCX_IN	P16
EMC_CS3_N	V15	RTCX_OUT	P17
EMC_D[00]	U4	SPI1_CLK / SCK0	C9
EMC_D[01]	R7	SPI1_DATIN / MISO0 / GPI_25 / MCFB1	C10
EMC_D[02]	T5	SPI1_DATIO / MOSI0 / MCFB2	B9
EMC_D[03]	U5	SPI2_CLK / SCK1 / LCDVD[23]	B10
EMC_D[04]	V3	SPI2_DATIN / MISO1 / LCDVD[21] / GPI_27	A10

Table 669. LPC32x0 pin allocation for TFBGA296 package by pin name

Symbol	Ball #	Symbol	Ball #
EMC_D[05]	V4	SPI2_DATIO / MOSI1 / LCDVD[20]	A9
EMC_D[06]	T6	SYSCLKEN / LCDVD[15]	G17
EMC_D[07]	R8	SYSX_IN	T17
EMC_D[08]	V5	SYSX_OUT	R15
EMC_D[09]	U6	TS_XOUT	R13
EMC_D[10]	V6	TS_YOUT	U16
EMC_D[11]	T7	TST_CLK2	C6
EMC_D[12]	U7	U1_RX / CAP1.0 / GPI_15	K15
EMC_D[13]	V7	U1_TX	K16
EMC_D[14]	T8	U2_HCTS / U3_CTS / GPI_16	J18
EMC_D[15]	U8	U2_RX / U3_DSR / GPI_17	K18
EMC_D[16] / EMC_DQS0	V8	U2_TX / U3_DTR	K17
EMC_D[17] / EMC_DQS1	R9	U3_RX / GPI_18	J14
EMC_D[18] / EMC_CLK_N	V9	U3_TX	J17
EMC_D[19] / P2.0	U9	U5_RX / GPI_20	F18
EMC_D[20] / P2.1	T9	U5_TX	H15
EMC_D[21] / P2.2	V10	U6_IRRX / GPI_21	F17
EMC_D[22] / P2.3	U10	U6_IRTX	G16
EMC_D[23] / P2.4	T10	U7_HCTS / CAP0.1 / LCDCLKIN / GPI_22	G13
EMC_D[24] / P2.5	R10	U7_RX / CAP0.0 / LCDVD[10] / GPI_23	E17
EMC_D[25] / P2.6	V11	U7_TX / MAT1.1 / LCDVD[11]	E18
EMC_D[26] / P2.7	U11	USB_ATX_INT_N	C4
EMC_D[27] / P2.8	T11	USB_DAT_VP / U5_RX	D5
EMC_D[28] / P2.9	V12	USB_I2C_SCL	E5
EMC_D[29] / P2.10	V13	USB_I2C_SDA	E6
EMC_D[30] / P2.11	U12	USB_OE_TP_N	D6
EMC_D[31] / P2.12	V14	USB_SE0_ VM / U5_TX	C5
EMC_DQM[0]	R3	VDD_AD	N12
EMC_DQM[1]	P4	VDD_AD	N13
EMC_DQM[2]	T1	VDD_CORE	G11
EMC_DQM[3]	P5	VDD_CORE	G7
EMC_DYCS0_N	R6	VDD_CORE	G9
EMC_DYCS1_N	G1	VDD_CORE	J12
EMC_OE_N	H1	VDD_CORE	J7
EMC_RAS_N	T2	VDD_CORE	M11
EMC_WR_N	R4	VDD_CORE	M7
FLASH_ALE	D2	VDD_COREFXD	L12
FLASH_CE_N	E3	VDD_COREFXD	M13
FLASH_CLE	F3	VDD_EMC	J6
FLASH_IO[00]	H2	VDD_EMC	K6
FLASH_IO[01]	H3	VDD_EMC	K7

Table 669. LPC32x0 pin allocation for TFBGA296 package by pin name

Symbol	Ball #	Symbol	Ball #
FLASH_IO[02]	F1	VDD_EMC	L6
FLASH_IO[03]	E1	VDD_EMC	M6
FLASH_IO[04]	H4	VDD_EMC	M8
FLASH_IO[05]	G2	VDD_EMC	N10
FLASH_IO[06]	G3	VDD_EMC	N11
FLASH_IO[07]	E2	VDD_EMC	N7
FLASH_RD_N	C1	VDD_EMC	N8
FLASH_RDY	D1	VDD_EMC	N9
FLASH_WR_N	F2	VDD_FUSE	N14
GPI_0 / I2S1RX_SDA	C16	VDD_IOA	H13
GPI_1 / SERVICE_N	C15	VDD_IOA	J13
GPI_2 / CAP2.0 / ENET_RXD3	C14	VDD_IOB	F8
GPI_3	F4	VDD_IOC	F7
GPI_4 / SPI1_BUSY	E13	VDD_IOC	G6
GPI_5 / U3_DCD	N16	VDD_IOC	H6
GPI_6 / HSTIM_CAP / ENET_RXD2	C7	VDD_IOC	J5
GPI_7 / CAP4.0 / MCABORT	D13	VDD_IOD	F13
GPI_8 / KEY_COL6 / SPI2_BUSY / ENET_RX_DV	B16	VDD_IOD	F9
GPI_9 / KEY_COL7 / ENET_COL	E12	VDD_OSC	T18
GPI_19 / U4_RX	B15	VDD_PLL397	T16
GPI_28 / U3_RI	N17	VDD_PLLHCLK	R17
GPIO_0	A12	VDD_PLLUSB	P15
GPIO_1	A11	VDD_RTC	K14
GPIO_2 / KEY_ROW6 / ENET_MDC	D9	VDD_RTCCORE	L13
GPIO_3 / KEY_ROW7 / ENET_MDIO	C11	VDD_RTCOSC	N15
GPIO_4 / SSEL1 / LCDVD[22]	B11	VSS_AD	P13
GPIO_5 / SSEL0 / MCFB0	E9	VSS_CORE	G10
GPO_0 / TST_CLK1	C3	VSS_CORE	G12
GPO_1	D4	VSS_CORE	G8
GPO_2 / MAT1.0 / LCDVD[0]	B14	VSS_CORE	H7
GPO_3 / LCDVD[1]	D12	VSS_CORE	K12
GPO_4	D8	VSS_CORE	L7
GPO_5	B3	VSS_CORE	M10
GPO_6 / LCDVD[18]	A16	VSS_CORE	M12
GPO_7 / LCDVD[2]	A15	VSS_CORE	M9
GPO_8 / LCDVD[8]	C13	VSS_EMC	K5
GPO_9 / LCDVD[9]	C12	VSS_EMC	L5
GPO_10 / MC2B / LCDPWR	E11	VSS_EMC	M5
GPO_11	E8	VSS_EMC	N5
GPO_12 / MC2A / LCDLE	B12	VSS_EMC	N6
GPO_13 / MC1B / LCDDCLK	B13	VSS_EMC	P10

Table 669. LPC32x0 pin allocation for TFBGA296 package by pin name

Symbol	Ball #	Symbol	Ball #
GPO_14	D3	VSS_EMC	P11
GPO_15 / MC1A / LCDFP	A14	VSS_EMC	P6
GPO_16 / MC0B / LCDENAB / LCDM	D10	VSS_EMC	P7
GPO_17	N18	VSS_EMC	P8
GPO_18 / MC0A / LCDLP	D11	VSS_EMC	P9
GPO_19	C2	VSS_IOA	K13
GPO_20	B2	VSS_IOB	F6
GPO_21 / U4_TX / LCDVD[3]	A13	VSS_IOC	F5
GPO_22 / U7_HRTS / LCDVD[14]	E10	VSS_IOC	G5
GPO_23 / U2_HRTS / U3_RTS	M16	VSS_IOC	H5
HIGHCORE / LCDVD[17]	H16	VSS_IOD	F10
I2C1_SCL	A5	VSS_IOD	F11
I2C1_SDA	B6	VSS_IOD	F12
I2C2_SCL	A3	VSS_IOD	H12
I2C2_SDA	E4	VSS_OSC	P14
I2S1TX_CLK / MAT3.0	A4	VSS_PLL397	T15
I2S1TX_SDA / MAT3.1	E7	VSS_PLLHCLK	R18
I2S1TX_WS / CAP3.0	B4	VSS_PLLUSB	R16
JTAG_NTRST	H17	VSS_RTCCORE	L14
JTAG_RTCK	H18	VSS_RTCOSC	P18

34.2 Pin descriptions

This section provides a cross reference of peripheral signal name to pin. Each peripheral is listed by signal name and the associated pin. These tables also list the alternate peripheral functions that share the pin, which can be helpful when selecting peripherals.

Table 670. Definition of parameter abbreviations

Parameter	Abbreviation
I/O type	I = input O = output I/O = bi-directional OT = output/high impedance I/OT = bi-directional, or high impedance
Pin detail[1]	BK: pin has a bus keeper function that weakly retains the last logic level driven on an I/O pin. I/O pin voltage vs Keeper current: Vdd = 1 μA (max) 0.67 x Vdd = 55 μA (max) 0.33 x Vdd = 60 μA (max) 0 V = 1 μA (max) pullup: pin has a nominal 50 μA internal pullup connected (see specific device data sheet for detailed information) pulldown: pin has a nominal 50 μA internal pulldown connected (see specific device data sheet for detailed information) P: pin has programmable input characteristics, see relevant peripheral chapters for details.

[1] Additional Function tables appear below some of the tables showing pins associated with specific functions. These indicate when the pin is not named for the related function but is an alternate function on a pin with a different purpose.

34.2.1 System Pins

Table 671. Summary of System pin pinout for TFBGA296 package

BGA 296 Pin	Function	Description	Alternate Function	I/O Type	Reset State	Pin Power Supply	Usage Notes
T17	SYSX_IN	System clock oscillator input	-	analog in	input	VDD_OSC	[1]
R15	SYSX_OUT	System clock oscillator output	-	analog out	out	VDD_OSC	[1]
M14	RESET_N	Reset input, active low	-	I	input	VDD_RTC	
G17	SYSCLKEN	Clock request out for external clock source	LCDVD[15]	I/O T	H	VDD_IOD	[2]
C15	SERVICE_N	boot select input, active low	GPI_1	I	input	VDD_IOD	
P16	RTCX_IN	RTC oscillator input	-	analog in	input	VDD_RTC	[3]
P17	RTCX_OUT	RTC oscillator output	-	analog out	out	VDD_RTC	
G4	RESOUT_N	Reset out. Reflects external & WDT reset	-	O	L/H[4]	VDD_IOC	[4]
M15	ONSW	RTC match out for external power control	-	O	L	VDD_RTC	
H16	HIGHCORE	Core voltage control out	LCDVD[17]	O	L	VDD_IOD	[5]
C3	TST_CLK1	Test clock 1 out	GPO_0	O	L	VDD_IOC	[6][7]
C6	TST_CLK2	Test clock 2 out	-	O	L	VDD_IOB	[8]
R14	PLL397_LOOP	PLL397 loop filter (for external components)	-		-	VDD_PLL397	[9]

[1] System oscillator runs only when pin DBGEN is LOW.

[2] SYSCLKEN either drives high or is in an input when not configured as a GPO (See the PWR_CTRL register in the Clocking and Power control chapter)

[3] The RTCX_IN pin supports a square wave clock signal with full VDD_RTC swing.

[4] RESOUT_N is low when RESET_N is low and goes high when RESET_N goes high. RESOUT_N may also be configured to reflect an internal watchdog reset.

[5] When HIGHCORE drives low it means a normal core voltage (1.2 V) is needed. When HIGHCORE drives high, a low core voltage (0.9 V) is allowed.

[6] TST_CLK1 can output selected internal clocks for test purposes, refer to the TEST_CLK register description in the Clocking and Power Control section for details.

[7] TST_CLK1 can output a maximum of 26 MHz when VDD_IOC is supplied with 1.8 V, and a maximum of 52 MHz when VDD_IOC is supplied with 2.8 V.

[8] TST_CLK2 can output selected internal clocks for test purposes, refer to the TEST_CLK register description in the Clocking and Power Control section for details.

[9] PLL397_LOOP requires two external capacitors and one resistor to be connected. See the Clocking and Power control chapter for details.

Table 672. DEBUG pinout for TFBGA296 package

BGA 296 Pin	Name	Description	Voltage Domain	Type	Default
H14	JTAG_TCK	JTAG1 clock input (user adds pullup/down on PCB)	VDD_IOD	I	input
H18	JTAG_RTCK	JTAG1 return clock out	VDD_IOD	O	L
H17	JTAG_NTRST	JTAG1 reset input	VDD_IOD	I: PU	input
G18	JTAG_TMS	JTAG1 test mode select input	VDD_IOD	I: PU	input
J16	JTAG_TDI	JTAG1 data input	VDD_IOD	I: PU	input
J15	JTAG_TDO	JTAG1 data out	VDD_IOD	O	L
G14	DBGEN	Device test input 0 = JTAG in-circuit debug; normal mode 1 = I/O cell boundary scan; test mode	VDD_IOD	I: PD	input

[1] General note: Inputs from JTAG emulators may have strong drivers. It is recommended to add termination resistors on the PCB.

Table 673. NO CONNECT pinout for TFBGA296 package

BGA 296 Pin	Name	Description	Voltage Domain	Type	Default
B17	N.C.	No connection			
U2	N.C.	No connection			
U17	N.C.	No connection			

34.2.2 EMC, NAND Flash, & SD card pinouts

Table 674. EMC pinout for TFBGA296 package

BGA 296 Pin	EMC Function	Description	Alternate Function	Power Supply Voltage	Type	Default
R3	EMC_DQM[0]	SDRAM data mask 0 out / EMC_LDM	-	VDD_EMC	O	L
P4	EMC_DQM[1]	SDRAM data mask 1 out / EMC_UDM	-	VDD_EMC	O	L
T1	EMC_DQM[2]	SDRAM data mask 2 out	-	VDD_EMC	O	L
P5	EMC_DQM[3]	SDRAM data mask 3 out	-	VDD_EMC	O	L
R6	EMC_DYCS0_N	SDRAM chip select 0, active low	-	VDD_EMC	O	H
G1	EMC_DYCS1_N	SDRAM chip select 1, active low	-	VDD_EMC	O	H
T2	EMC_RAS_N	SDRAM row address strobe, active low	-	VDD_EMC	O	H
R5	EMC_CAS_N	SDRAM column addr strobe out, active low	-	VDD_EMC	O	H
U3	EMC_CKE0	SDRAM bank 0 Clock enable out	-	VDD_EMC	O	L
L2	EMC_CKE1	SDRAM bank 1 Clock enable out	-	VDD_EMC	O	L
T3	EMC_CLK	SDRAM clock out	-	VDD_EMC	O	L/R
T4	EMC_CLKIN	SDRAM clock feedback	-	VDD_EMC	I	input
R4	EMC_WR_N	EMC write strobe, active low	-	VDD_EMC	O	H
H1	EMC_OE_N	EMC static memory output enable	-	VDD_EMC	O	H
U14	EMC_BLS[0]	EMC static memory byte lane 0 select out	-	VDD_EMC	O	H
T13	EMC_BLS[1]	EMC static memory byte lane 1 select out	-	VDD_EMC	O	H
R12	EMC_BLS[2]	EMC static memory byte lane 2 select out	-	VDD_EMC	O	H
P12	EMC_BLS[3]	EMC static memory byte lane 3 select out	-	VDD_EMC	O	H
U13	EMC_CS0_N	EMC static memory chip select 0	-	VDD_EMC	O	H
R11	EMC_CS1_N	EMC static memory chip select 1	-	VDD_EMC	O	H
T12	EMC_CS2_N	EMC static memory chip select 2	-	VDD_EMC	O	H
V15	EMC_CS3_N	EMC static memory chip select 3	-	VDD_EMC	O	H
L3	EMC_A[00]	EMC address bit 0	P1.0	VDD_EMC	I/O	L
L4	EMC_A[01]	EMC address bit 1	P1.1	VDD_EMC	I/O	L
M1	EMC_A[02]	EMC address bit 2	P1.2	VDD_EMC	I/O	L
M2	EMC_A[03]	EMC address bit 3	P1.3	VDD_EMC	I/O	L
M3	EMC_A[04]	EMC address bit 4	P1.4	VDD_EMC	I/O	L
N1	EMC_A[05]	EMC address bit 5	P1.5	VDD_EMC	I/O	L
N2	EMC_A[06]	EMC address bit 6	P1.6	VDD_EMC	I/O	L
N3	EMC_A[07]	EMC address bit 7	P1.7	VDD_EMC	I/O	L
M4	EMC_A[08]	EMC address bit 8	P1.8	VDD_EMC	I/O	L
P1	EMC_A[09]	EMC address bit 9	P1.9	VDD_EMC	I/O	L
P2	EMC_A[10]	EMC address bit 10	P1.10	VDD_EMC	I/O	L
P3	EMC_A[11]	EMC address bit 11	P1.11	VDD_EMC	I/O	L
N4	EMC_A[12]	EMC address bit 12	P1.12	VDD_EMC	I/O	L
R1	EMC_A[13]	EMC address bit 13 / BA 0	P1.13	VDD_EMC	I/O	L
R2	EMC_A[14]	EMC address bit 14 / BA 1	P1.14	VDD_EMC	I/O	L

Table 674. EMC pinout for TFBGA296 package

BGA 296 Pin	EMC Function	Description	Alternate Function	Power Supply Voltage	Type	Default
L1	EMC_A[15]	EMC address bit 15	P1.15	VDD_EMC	I/O	L
K3	EMC_A[16]	EMC address bit 16	P1.16	VDD_EMC	I/O	L
K4	EMC_A[17]	EMC address bit 17	P1.17	VDD_EMC	I/O	L
K2	EMC_A[18]	EMC address bit 18	P1.18	VDD_EMC	I/O	L
K1	EMC_A[19]	EMC address bit 19	P1.19	VDD_EMC	I/O	L
J1	EMC_A[20]	EMC address bit 20	P1.20	VDD_EMC	I/O	L
J2	EMC_A[21]	EMC address bit 21	P1.21	VDD_EMC	I/O	L
J3	EMC_A[22]	EMC address bit 22	P1.22	VDD_EMC	I/O	L
J4	EMC_A[23]	EMC address bit 23	P1.23	VDD_EMC	I/O	L
U4	EMC_D[00]	EMC data bit 0	-	VDD_EMC	I/O: BK	input
R7	EMC_D[01]	EMC data bit 1	-	VDD_EMC	I/O: BK	input
T5	EMC_D[02]	EMC data bit 2	-	VDD_EMC	I/O: BK	input
U5	EMC_D[03]	EMC data bit 3	-	VDD_EMC	I/O: BK	input
V3	EMC_D[04]	EMC data bit 4	-	VDD_EMC	I/O: BK	input
V4	EMC_D[05]	EMC data bit 5	-	VDD_EMC	I/O: BK	input
T6	EMC_D[06]	EMC data bit 6	-	VDD_EMC	I/O: BK	input
R8	EMC_D[07]	EMC data bit 7	-	VDD_EMC	I/O: BK	input
V5	EMC_D[08]	EMC data bit 8	-	VDD_EMC	I/O: BK	input
U6	EMC_D[09]	EMC data bit 9	-	VDD_EMC	I/O: BK	input
V6	EMC_D[10]	EMC data bit 10	-	VDD_EMC	I/O: BK	input
T7	EMC_D[11]	EMC data bit 11	-	VDD_EMC	I/O: BK	input
U7	EMC_D[12]	EMC data bit 12	-	VDD_EMC	I/O: BK	input
V7	EMC_D[13]	EMC data bit 13	-	VDD_EMC	I/O: BK	input
T8	EMC_D[14]	EMC data bit 14	-	VDD_EMC	I/O: BK	input
U8	EMC_D[15]	EMC data bit 15	-	VDD_EMC	I/O: BK	input
V8	EMC_D[16] / EMC_DQS0	EMC data bit 16, DDR data strobe 0 I/O	EMC_D[16] / EMC_DQS0	VDD_EMC	I/O: BK	input
R9	EMC_D[17] / EMC_DQS1	EMC data bit 17, DDR data strobe 1 I/O	EMC_D[17] / EMC_DQS1	VDD_EMC	I/O: BK	input
V9	EMC_D[18] / EMC_CLK_N	EMC data bit 18, inverted clock out	EMC_D[18] / EMC_CLK_N	VDD_EMC	I/O: P	input
U9	EMC_D[19]	EMC data bit 19	P2.0	VDD_EMC	I/O: P	input
T9	EMC_D[20]	EMC data bit 20	P2.1	VDD_EMC	I/O: P	input
V10	EMC_D[21]	EMC data bit 21	P2.2	VDD_EMC	I/O: P	input
U10	EMC_D[22]	EMC data bit 22	P2.3	VDD_EMC	I/O: P	input
T10	EMC_D[23]	EMC data bit 23	P2.4	VDD_EMC	I/O: P	input
R10	EMC_D[24]	EMC data bit 24	P2.5	VDD_EMC	I/O: P	input
V11	EMC_D[25]	EMC data bit 25	P2.6	VDD_EMC	I/O: P	input
U11	EMC_D[26]	EMC data bit 26	P2.7	VDD_EMC	I/O: P	input
T11	EMC_D[27]	EMC data bit 27	P2.8	VDD_EMC	I/O: P	input

Table 674. EMC pinout for TFBGA296 package

BGA 296 Pin	EMC Function	Description	Alternate Function	Power Supply Voltage	Type	Default
V12	EMC_D[28]	EMC data bit 28	P2.9	VDD_EMC	I/O: P	input
V13	EMC_D[29]	EMC data bit 29	P2.10	VDD_EMC	I/O: P	input
U12	EMC_D[30]	EMC data bit 30	P2.11	VDD_EMC	I/O: P	input
V14	EMC_D[31]	EMC data bit 31	P2.12	VDD_EMC	I/O: P	input

Table 675. Flash Memory Interface pinout for TFBGA296 package

BGA 296 Pin	Pin Name	Description	Alternate Function	Voltage Domain	Type	Default
D2	FLASH_ALE	Flash address latch enable	-	VDD_IOC	O	L
E3	FLASH_CE_N	Flash chip enable	-	VDD_IOC	O	H
F3	FLASH_CLE	Flash command latch enable	-	VDD_IOC	O	L
C1	FLASH_RD_N	Flash read enable	-	VDD_IOC	O	H
D1	FLASH_RDY	Flash ready (from Flash device)	-	VDD_IOC	I	input
F2	FLASH_WR_N	Flash write enable	-	VDD_IOC	O	H
H2	FLASH_IO[00]	Flash data bus, bit 0	-	VDD_IOC	I/O: BK	input
H3	FLASH_IO[01]	Flash data bus, bit 1	-	VDD_IOC	I/O: BK	input
F1	FLASH_IO[02]	Flash data bus, bit 2	-	VDD_IOC	I/O: BK	input
E1	FLASH_IO[03]	Flash data bus, bit 3	-	VDD_IOC	I/O: BK	input
H4	FLASH_IO[04]	Flash data bus, bit 4	-	VDD_IOC	I/O: BK	input
G2	FLASH_IO[05]	Flash data bus, bit 5	-	VDD_IOC	I/O: BK	input
G3	FLASH_IO[06]	Flash data bus, bit 6	-	VDD_IOC	I/O: BK	input
E2	FLASH_IO[07]	Flash data bus, bit 7	-	VDD_IOC	I/O: BK	input

Table 676. SD_Card pinout for TFBGA296 package

BGA 296 Pin	SD_Card Function	Description	Alternate Function	Voltage Domain	Type	Default
B7	MS_SCLK	SD card clock out	MAT2.0	VDD_IOD	I/O	L
A6	MS_BS	SD card command out	MAT2.1	VDD_IOD	I/O: P	L
A8	MS_DIO0	SD card data 0	MAT0.0	VDD_IOD	I/O: P	input
A7	MS_DIO1	SD card data 1	MAT0.1	VDD_IOD	I/O: P	input
B8	MS_DIO2	SD card data 2	MAT0.2	VDD_IOD	I/O: P	input
C8	MS_DIO3	SD card data 3	MAT0.3	VDD_IOD	I/O: P	input

34.2.3 LCD Pinout

Table 677. LCD pinout for TFBGA296 package

296 Pin	LCD Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
E11	LCDPWR	LCD panel power enable	GPO_10 / MC2B	VDD_IOD	O	L
B12	LCDLE	LCD line end signal	GPO_12 / MC2A	VDD_IOD	O	L
B13	LCDDCLK	LCD clock output	GPO_13 / MC1B	VDD_IOD	O	L
A14	LCDFP	LCD frame/sync pulse	GPO_15 / MC1A	VDD_IOD	O	L
D10	LCDENAB / LCDM	LCD STN AC bias / TFT data enable	GPO_16 / MC0B	VDD_IOD	O	L
D11	LCDLP	LCD line sync / horizontal sync	GPO_18 / MC0A	VDD_IOD	O	L
G13	LCDCLKIN	LCD panel clk in	GPI_22 / U7_HCTS / CAP0.1	VDD_IOD	I	input
B14	LCDVD[0]	LCD data bit 0	GPO_2 / MAT1.0	VDD_IOD	O	L
D12	LCDVD[1]	LCD data bit 1	GPO_3	VDD_IOD	O	H
A15	LCDVD[2]	LCD data bit 2	GPO_7	VDD_IOD	O	H
A13	LCDVD[3]	LCD data bit 3	GPO_21 / U4_TX	VDD_IOD	O	L
M17	LCDVD[4]	LCD data bit 4	P0.2 / I2S0RX_SDA	VDD_IOA	I/O	input
M18	LCDVD[5]	LCD data bit 5	P0.3 / I2S0RX_CLK	VDD_IOA	I/O	input
L15	LCDVD[6]	LCD data bit 6	P0.4 / I2S0RX_WS	VDD_IOA	I/O	input
L16	LCDVD[7]	LCD data bit 7	P0.5 / I2S0TX_SDA	VDD_IOA	I/O	input
C13	LCDVD[8]	LCD data bit 8	GPO_8	VDD_IOD	O	L
C12	LCDVD[9]	LCD data bit 9	GPO_9	VDD_IOD	O	L
E17	LCDVD[10]	LCD data bit 10	GPI_23 / U7_RX / CAP0.0	VDD_IOD	I/O	input
E18	LCDVD[11]	LCD data bit 11	U7_TX / MAT1.1	VDD_IOD	O	H
L17	LCDVD[12]	LCD data bit 12	P0.6 / I2S0TX_CLK	VDD_IOA	I/O	input
L18	LCDVD[13]	LCD data bit 13	P0.7 / I2S0TX_WS	VDD_IOA	I/O	input
E10	LCDVD[14]	LCD data bit 14	GPO_22 / U7_HRTS	VDD_IOD	O	L
G17	LCDVD[15]	LCD data bit 15	SYSCLKEN	VDD_IOD	I/O T	H
D14	LCDVD[16]	LCD data bit 16	PWM_OUT1	VDD_IOD	O	L
H16	LCDVD[17]	LCD data bit 17	HIGHCORE	VDD_IOD	O	L
A16	LCDVD[18]	LCD data bit 18	GPO_6	VDD_IOD	O	L
D15	LCDVD[19]	LCD data bit 19	PWM_OUT2	VDD_IOD	O	L
A9	LCDVD[20]	LCD data bit 20	SPI2_DATIO / MOSI1	VDD_IOD	I/O	input
A10	LCDVD[21]	LCD data bit 21	GPI_27 / SPI2_DATIN / MISO1	VDD_IOD	I/O	input
B11	LCDVD[22]	LCD data bit 22	GPIO_4 / SSEL1	VDD_IOD	I/O	input
B10	LCDVD[23]	LCD data bit 23	SPI2_CLK / SCK1	VDD_IOD	I/O	input

34.2.4 USB & Ethernet pinouts

Table 678. USB pinout for TFBGA296 package

BGA 296 Pin	USB Function	Description	Alternate Function	Voltage Domain	Type	Default
C4	USB_ATX_INT_N	Interrupt from USB ATX	-	VDD_IOC	I	input
D5	USB_DAT_VP	USB transmit data, D+ receive	U5_RX	VDD_IOC	I/O: P	input
E5	USB_I2C_SCL	I2C clock for USB ATX interface	-	VDD_IOC	I/O T	T
E6	USB_I2C_SDA	I2C data for USB ATX interface	-	VDD_IOC	I/O T	T
D6	USB_OE_TP_N	USB transmit enable for DAT/SE0	-	VDD_IOC	I/O: P	H
C5	USB_SE0_ VM	USB single ended zero transmit, D- receive	U5_TX	VDD_IOC	I/O: P	input

Table 679. Etherent MAC pinout for TFBGA296 package

BGA 296 Pin	Ethernet MAC Function	Description	Alternate Functions(s)	Voltage Domain	Type	Default
MII pins						
F15	ENET_TX_CLK	Ethernet transmit clock	KEY_COL0	VDD_IOD	I	input
B16	ENET_RX_DV	Ethernet receive data valid input	GPI_8 / KEY_COL6 / SPI2_BUSY	VDD_IOD	I	input
C7	ENET_RXD2	Ethernet receive data 2	GPI_6 / HSTIM_CAP	VDD_IOB	I:BK	input
C14	ENET_RXD3	Ethernet receive data 3	GPI_2 / CAP2.0	VDD_IOD	I	input
E15	ENET_TX_ER	Ethernet transmit error	KEY_ROW0	VDD_IOD	I/O T	H
E14	ENET_TXD2	Ethernet transmit data 2	KEY_ROW1	VDD_IOD	I/O T	H
F14	ENET_TXD3	Ethernet transmit data 3	KEY_ROW2	VDD_IOD	I/O T	H
RMII pins (and MII pins)						
D16	ENET_TX_EN	Ethernet transmit enable	KEY_ROW3	VDD_IOD	I/O T	H
C17	ENET_TXD0	Ethernet transmit data 0	KEY_ROW4	VDD_IOD	I/O T	H
C18	ENET_TXD1	Ethernet transmit data 1	KEY_ROW5	VDD_IOD	I/O T	H
G15	ENET_RXD0	Ethernet receive data 0	KEY_COL4	VDD_IOD	I	input
F16	ENET_RXD1	Ethernet receive data 1	KEY_COL5	VDD_IOD	I	input
D17	ENET_RX_ER	Ethernet receive error input	KEY_COL2	VDD_IOD	I	input
E12	ENET_COL	Ethernet collision input	GPI_9 / KEY_COL7	VDD_IOD	I	input
D18	ENET_CRS	Ethernet carrier sense input	KEY_COL3	VDD_IOD	I	input
E16	ENET_RX_CLK / ENET_REF_CLK	Ethernet receive clock (MII mode), Ethernet reference clock (RMII mode)	KEY_COL1	VDD_IOD	I	input
MIIM pins						
D9	ENET_MDC	Ethernet PHY interface clock	GPIO_2 / KEY_ROW6	VDD_IOD	I/O	input
C11	ENET_MDIO	Ethernet PHY interface data	GPIO_3 / KEY_ROW7	VDD_IOD	I/O	input

34.2.5 ADC, Touchscreen, Key scan pinouts

Table 680. ADC pinout for TFBGA296 package

BGA 296 Pin	ADC Function	Description	Alternate Function	Voltage Domain	Type	Default
U15	ADIN0	ADC input 0	TS_YM	VDD_AD	analog in	input
T14	ADIN1	ADC input 1	TS_XM	VDD_AD	analog in	input
V16	ADIN2	ADC input 2	TS_AUX_IN	VDD_AD	analog in	input

Table 681. Touch Screen pinout for TFBGA296 package

BGA 296 Pin	Touchscreen Function	Description	Alternate Function	Voltage Domain	Type	Default
R13	TS_XP	Touchscreen X Plus	-	VDD_AD	I/O	T
U16	TS_YP	Touchscreen Y Plus	-	VDD_AD	I/O	T
U15	TS_YM	Touchscreen Y Minus	ADIN0	VDD_AD	analog in	input
T14	TS_XM	Touchscreen X Minus	ADIN1	VDD_AD	analog in	input
V16	TS_AUX_IN	Auxiliary ADC input	ADIN2	VDD_AD	analog in	input

Table 682. Key Scanner pinout for TFBGA296 package

BGA 296 Pin	Keyscan Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
F15	KEY_COL0	Keyscan column 0 in	ENET_TX_CLK	VDD_IOD	I	input
E16	KEY_COL1	Keyscan column 1 in	ENET_RX_CLK / ENET_REF_CLK	VDD_IOD	I	input
D17	KEY_COL2	Keyscan column 2 in	ENET_RX_ER	VDD_IOD	I	input
D18	KEY_COL3	Keyscan column 3 in	ENET_CRS	VDD_IOD	I	input
G15	KEY_COL4	Keyscan column 4 in	ENET_RXD0	VDD_IOD	I	input
F16	KEY_COL5	Keyscan column 5 in	ENET_RXD1	VDD_IOD	I	input
B16	KEY_COL6	Keyscan column 6 in	GPI_8 / SPI2_BUSY / ENET_RX_DV	VDD_IOD	I	input
E12	KEY_COL7	Keyscan column 7 in	GPI_9 / ENET_COL	VDD_IOD	I	input
E15	KEY_ROW0	Keyscan row 0 out	ENET_TX_ER	VDD_IOD	I/O T	H
E14	KEY_ROW1	Keyscan row 1 out	ENET_TXD2	VDD_IOD	I/O T	H
F14	KEY_ROW2	Keyscan row 2 out	ENET_TXD3	VDD_IOD	I/O T	H
D16	KEY_ROW3	Keyscan row 3 out	ENET_TX_EN	VDD_IOD	I/O T	H
C17	KEY_ROW4	Keyscan row 4 out	ENET_TXD0	VDD_IOD	I/O T	H
C18	KEY_ROW5	Keyscan row 5 out	ENET_TXD1	VDD_IOD	I/O T	H
D9	KEY_ROW6	Keyscan row 6 out	GPIO_2 / ENET_MDC	VDD_IOD	I/O	input
C11	KEY_ROW7	Keyscan row 7 out	GPIO_3 / ENET_MDIO	VDD_IOD	I/O	input

34.2.6 I/O Ports 0, 1, 2, & 3 pinouts

Table 683. Port 0 pinout for TFBGA296 package

BGA 296 Pin	Port 0 Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
B5	P0.0	Port 0 bit 0	I2S1RX_CLK	VDD_IOB	I/O	input
D7	P0.1	Port 0 bit 1	I2S1RX_WS	VDD_IOB	I/O	input
M17	P0.2	Port 0 bit 2	I2S0RX_SDA / LCDVD[4]	VDD_IOA	I/O	input
M18	P0.3	Port 0 bit 3	I2S0RX_CLK / LCDVD[5]	VDD_IOA	I/O	input
L15	P0.4	Port 0 bit 4	I2S0RX_WS / LCDVD[6]	VDD_IOA	I/O	input
L16	P0.5	Port 0 bit 5	I2S0TX_SDA / LCDVD[7]	VDD_IOA	I/O	input
L17	P0.6	Port 0 bit 6	I2S0TX_CLK / LCDVD[12]	VDD_IOA	I/O	input
L18	P0.7	Port 0 bit 7	I2S0TX_WS / LCDVD[13]	VDD_IOA	I/O	input

Table 684. Port 1 pinout for TFBGA296 package

BGA 296 Pin	Port 1 Function	Description	Alternate Function	Voltage Domain	Type	Default
L3	P1.0	Port 1 bit 0	EMC_A[00]	VDD_EMC	I/O	L
L4	P1.1	Port 1 bit 1	EMC_A[01]	VDD_EMC	I/O	L
M1	P1.2	Port 1 bit 2	EMC_A[02]	VDD_EMC	I/O	L
M2	P1.3	Port 1 bit 3	EMC_A[03]	VDD_EMC	I/O	L
M3	P1.4	Port 1 bit 4	EMC_A[04]	VDD_EMC	I/O	L
N1	P1.5	Port 1 bit 5	EMC_A[05]	VDD_EMC	I/O	L
N2	P1.6	Port 1 bit 6	EMC_A[06]	VDD_EMC	I/O	L
N3	P1.7	Port 1 bit 7	EMC_A[07]	VDD_EMC	I/O	L
M4	P1.8	Port 1 bit 8	EMC_A[08]	VDD_EMC	I/O	L
P1	P1.9	Port 1 bit 9	EMC_A[09]	VDD_EMC	I/O	L
P2	P1.10	Port 1 bit 10	EMC_A[10]	VDD_EMC	I/O	L
P3	P1.11	Port 1 bit 11	EMC_A[11]	VDD_EMC	I/O	L
N4	P1.12	Port 1 bit 12	EMC_A[12]	VDD_EMC	I/O	L
R1	P1.13	Port 1 bit 13	EMC_A[13] / BA 0	VDD_EMC	I/O	L
R2	P1.14	Port 1 bit 14	EMC_A[14] / BA 1	VDD_EMC	I/O	L
L1	P1.15	Port 1 bit 15	EMC_A[15]	VDD_EMC	I/O	L
K3	P1.16	Port 1 bit 16	EMC_A[16]	VDD_EMC	I/O	L
K4	P1.17	Port 1 bit 17	EMC_A[17]	VDD_EMC	I/O	L
K2	P1.18	Port 1 bit 18	EMC_A[18]	VDD_EMC	I/O	L
K1	P1.19	Port 1 bit 19	EMC_A[19]	VDD_EMC	I/O	L
J1	P1.20	Port 1 bit 20	EMC_A[20]	VDD_EMC	I/O	L
J2	P1.21	Port 1 bit 21	EMC_A[21]	VDD_EMC	I/O	L
J3	P1.22	Port 1 bit 22	EMC_A[22]	VDD_EMC	I/O	L
J4	P1.23	Port 1 bit 23	EMC_A[23]	VDD_EMC	I/O	L

Table 685. Port 2 pinout for TFBGA296 package

BGA 296 Pin	Port 2 Function	Description	Alternate Function	Voltage Domain	Type	Default
U9	P2.0	Port 2 bit 0	EMC_D[19]	VDD_EMC	I/O: P	input
T9	P2.1	Port 2 bit 1	EMC_D[20]	VDD_EMC	I/O: P	input
V10	P2.2	Port 2 bit 2	EMC_D[21]	VDD_EMC	I/O: P	input
U10	P2.3	Port 2 bit 3	EMC_D[22]	VDD_EMC	I/O: P	input
T10	P2.4	Port 2 bit 4	EMC_D[23]	VDD_EMC	I/O: P	input
R10	P2.5	Port 2 bit 5	EMC_D[24]	VDD_EMC	I/O: P	input
V11	P2.6	Port 2 bit 6	EMC_D[25]	VDD_EMC	I/O: P	input
U11	P2.7	Port 2 bit 7	EMC_D[26]	VDD_EMC	I/O: P	input
T11	P2.8	Port 2 bit 8	EMC_D[27]	VDD_EMC	I/O: P	input
V12	P2.9	Port 2 bit 9	EMC_D[28]	VDD_EMC	I/O: P	input
V13	P2.10	Port 2 bit 10	EMC_D[29]	VDD_EMC	I/O: P	input
U12	P2.11	Port 2 bit 11	EMC_D[30]	VDD_EMC	I/O: P	input
V14	P2.12	Port 2 bit 12	EMC_D[31]	VDD_EMC	I/O: P	input

Table 686. Port 3 GPI pinout for TFBGA296 package

BGA 296 Pin	Port 3 Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
C16	GPI_0	GP input 0	I2S1RX_SDA	VDD_IOD	I	input
C15	GPI_1	GP input 1	SERVICE_N	VDD_IOD	I	input
C14	GPI_2	GP input 2	CAP2.0 / ENET_RXD3	VDD_IOD	I	input
F4	GPI_3	GP input 3	-	VDD_IOC	I	input
E13	GPI_4	GP input 4	SPI1_BUSY	VDD_IOD	I	input
N16	GPI_5	GP input 5	U3_DCD	VDD_IOA	I	input
C7	GPI_6	GP input 6	HSTIM_CAP / ENET_RXD2	VDD_IOB	I:BK	input
D13	GPI_7	GP input 7	CAP4.0 / MCABORT	VDD_IOD	I	input
B16	GPI_8	GP input 8	KEY_COL6 / SPI2_BUSY / ENET_RX_DV	VDD_IOD	I	input
E12	GPI_9	GP input 9	KEY_COL7 / ENET_COL	VDD_IOD	I	input
K15	GPI_15	GP input 15	U1_RX / CAP1.0	VDD_IOA	I/O	input
J18	GPI_16	GP input 16	U2_HCTS / U3_CTS	VDD_IOA	I/O	input
K18	GPI_17	GP input 17	U2_RX / U3_DSR	VDD_IOA	I/O	input
J14	GPI_18	GP input 18	U3_RX	VDD_IOD	I/O	input
B15	GPI_19	GP input 19	U4_RX	VDD_IOD	I	input
F18	GPI_20	GP input 20	U5_RX	VDD_IOD	I/O	input
F17	GPI_21	GP input 21	U6_IRRX	VDD_IOD	I/O	input
G13	GPI_22	GP input 22	U7_HCTS / CAP0.1 / LCDCLKIN	VDD_IOD	I	input

Table 686. Port 3 GPI pinout for TFBGA296 package

BGA 296 Pin	Port 3 Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
E17	GPI_23	GP input 23	U7_RX / CAP0.0 / LCDVD[10]	VDD_IOD	I/O	input
C10	GPI_25	GP input 25	SPI1_DATIN / MISO0 / MCFB1	VDD_IOD	I/O	input
A10	GPI_27	GP input 27	SPI2_DATIN / MISO1 / LCDVD[21]	VDD_IOD	I/O	input
N17	GPI_28	GP input 28	U3_RI	VDD_IOA	I	input

Table 687. Port 3 GPO pinout for TFBGA296 package

BGA 296 Pin	Port 3 Function	Description	Alternate Function	Voltage Domain	Type	Default
C3	GPO_0	GP out 00	TST_CLK1	VDD_IOC	O	L
D4	GPO_1	GP out 01	-	VDD_IOC	O	L
B14	GPO_2	GP out 02	MAT1.0 / LCDVD[0]	VDD_IOD	O	L
D12	GPO_3	GP out 03	LCDVD[1]	VDD_IOD	O	H
D8	GPO_4	GP out 04	-	VDD_IOB	O	L
B3	GPO_5	GP out 05	-	VDD_IOC	O	H
A16	GPO_6	GP out 06	LCDVD[18]	VDD_IOD	O	L
A15	GPO_7	GP out 07	LCDVD[2]	VDD_IOD	O	H
C13	GPO_8	GP out 08	LCDVD[8]	VDD_IOD	O	L
C12	GPO_9	GP out 09	LCDVD[9]	VDD_IOD	O	L
E11	GPO_10	GP out 10	MC2B / LCDPWR	VDD_IOD	O	L
E8	GPO_11	GP out 11	-	VDD_IOB	O	L
B12	GPO_12	GP out 12	MC2A / LCDLE	VDD_IOD	O	L
B13	GPO_13	GP out 13	MC1B / LCDDCLK	VDD_IOD	O	L
D3	GPO_14	GP out 14	-	VDD_IOC	O	L
A14	GPO_15	GP out 15	MC1A / LCDFP	VDD_IOD	O	L
D10	GPO_16	GP out 16	MC0B / LCDENAB / LCDM	VDD_IOD	O	L
N18	GPO_17	GP out 17	-	VDD_IOA	O	L
D11	GPO_18	GP out 18	MC0A / LCDLP	VDD_IOD	O	L
C2	GPO_19	GP out 19	-	VDD_IOC	O	L
B2	GPO_20	GP out 20	-	VDD_IOC	O	H
A13	GPO_21	GP out 21	U4_TX / LCDVD[3]	VDD_IOD	O	L
E10	GPO_22	GP out 22	U7_HRTS / LCDVD[14]	VDD_IOD	O	L
M16	GPO_23	GP out 23	U2_HRTS / U3_RTS	VDD_IOA	O	L

Table 688. Port 3 GPIO pinout for TFBGA296 package

BGA 296 Pin	Port 3 Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
A12	GPIO_	GP I/O 00	-	VDD_IOD	I/O	input
A11	GPIO_1	GP I/O 01	-	VDD_IOD	I/O	input
D9	GPIO_2	GP I/O 02	KEY_ROW6 / ENET_MDC	VDD_IOD	I/O	input
C11	GPIO_3	GP I/O 03	KEY_ROW7 / ENET_MDIO	VDD_IOD	I/O	input
B11	GPIO_4	GP I/O 04	SSEL1 / LCDVD[22]	VDD_IOD	I/O	input
E9	GPIO_5	GP I/O 05	SSEL0 / MCFB0	VDD_IOD	I/O	input

34.2.7 UART, SPI, SSP and I2S pinouts

Table 689. UART pinouts for TFBGA296 package

BGA 296 Pin	UART Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
UART 1						
K15	U1_RX	14-clock Uart 1 receive	GPI_15 / CAP1.0	VDD_IOA	I/O	input
K16	U1_TX	14-clock Uart 1 transmit	U1_TX	VDD_IOA	O	H
UART 2						
J18	U2_HCTS	14-clock Uart 2 CTS input	GPI_16 / U3_CTS	VDD_IOA	I/O	input
K18	U2_RX	14-clock Uart 2 receive	GPI_17 / U3_DSR	VDD_IOA	I/O	input
K17	U2_TX	14-clock Uart 2 transmit	U2_TX / U3_DTR	VDD_IOA	O	H
M16	U2_HRTS	14-clock Uart 2 RTS out	GPO_23 / U3_RTS	VDD_IOA	O	L
UART 3						
J14	U3_RX	Uart 3 receive	GPI_18	VDD_IOD	I/O	input
J17	U3_TX	Uart 3 transmit	-	VDD_IOD	O	H
M16	U3_RTS	Uart 3 request to send out	GPO_23 / U2_HRTS	VDD_IOA	O	L
J18	U3_CTS	Uart 3 clear to send	GPI_16 / U2_HCTS	VDD_IOA	I/O	input
K18	U3_DSR	Uart 3 data set ready	GPI_17 / U2_RX	VDD_IOA	I/O	input
K17	U3_DTR	Uart 3 data terminal ready out	U2_TX	VDD_IOA	O	H
N16	U3_DCD	Uart 3 data carrier detect input	GPI_5	VDD_IOA	I	input
N17	U3_RI	Uart 3 ring indicator input	GPI_28	VDD_IOA	I	input
UART 4						
A13	U4_TX	Uart 4 transmit	GPO_21 / LCDVD[3]	VDD_IOD	O	L
B15	U4_RX	Uart 4 receive	GPI_19	VDD_IOD	I	input
UART 5						
F18	U5_RX	Uart 5 receive,	GPI_20	VDD_IOD	I/O	input
H15	U5_TX	Uart 5 transmit	-	VDD_IOD	O	H
UART 5 (USB)						
D5	U5_RX	USB transmit data, D+ receive	USB_DAT_VP	VDD_IOC	I/O: P	input

Table 689. UART pinouts for TFBGA296 package

BGA 296 Pin	UART Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
C5	U5_TX	USB single ended zero transmit, D-receive	USB_SE0_ VM	VDD_IOC	I/O: P	input
UART 6						
F17	U6_IRRX	Uart 6 receive (with IrDA)	GPI_21	VDD_IOD	I/O	input
G16	U6_IRTX	Uart 6 transmit (with IrDA)	-	VDD_IOD	O	L
UART 7						
G13	U7_HCTS	14-clock Uart 7 CTS in	GPI_22 / CAP0.1 / LCDCLKIN	VDD_IOD	I	input
E17	U7_RX	14-clock Uart 7 receive	GPI_23 / CAP0.0 / LCDVD[10]	VDD_IOD	I/O	input
E18	U7_TX	14-clock Uart 7 transmit	MAT1.1 / LCDVD[11]	VDD_IOD	O	H
E10	U7_HRTS	14-clock Uart 7 RTS out	GPO_22 / LCDVD[14]	VDD_IOD	O	L

Table 690. SPI pinouts for TFBGA296 package

BGA 296 Pin	SPI Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
SPI 1						
C9	SPI1_CLK	SPI1 clock out	SCK0	VDD_IOD	I/O	input
C10	SPI1_DATIN	SPI1 data in	MISO0 / GPI_25 / MCFB1	VDD_IOD	I/O	input
B9	SPI1_DATIO	SPI1 data out (and opt. input)	MOSI0 / MCFB2	VDD_IOD	I/O	input

Table 690. SPI pinouts for TFBGA296 package

BGA 296 Pin	SPI Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
SPI 2						
B10	SPI2_CLK	SPI2 clock out	SCK1 / LCDVD[23]	VDD_IOD	I/O	input
A10	SPI2_DATIN	SPI2 data in	GPI_27 / MISO1 / LCDVD[21]	VDD_IOD	I/O	input
A9	SPI2_DATIO	SPI2 data out	MOSI1 / LCDVD[20]	VDD_IOD	I/O	input

Table 691. SSP pinout for TFBGA296 package

BGA 296 Pin	SSP Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
SSP 0						
E9	SSEL0	SSP0 Slave Select	GPIO_5 / MCFB0	VDD_IOD	I/O	input
C9	SCK0	SSP0 clock out	SPI1_CLK	VDD_IOD	I/O	input
C10	MISO0	SSP0 MISO	SPI1_DATIN / GPI_25 / MCFB1	VDD_IOD	I/O	input
B9	MOSI0	SSP0 MOSI	SPI1_DATIO / MCFB2	VDD_IOD	I/O	input
SSP 1						
B11	SSEL1	SSP1 Slave Select	GPIO_4 / LCDVD[22]	VDD_IOD	I/O	input
B10	SCK1	SSSP1 clock out	SPI2_CLK / LCDVD[23]	VDD_IOD	I/O	input
A10	MISO1	SSP1 MISO	GPI_27 / SPI2_DATIN / LCDVD[21]	VDD_IOD	I/O	input
A9	MOSI1	SSP1 MOSI	SPI2_DATIO / LCDVD[20]	VDD_IOD	I/O	input

Table 692. I2C pinouts for TFBGA296 package

BGA 296 Pin	I2C Function	Description	Alternate Function	Voltage Domain	Type	Default
I2C 1						
A5	I2C1_SCL	I2C1 serial clock input./output	-	VDD_IOB	I/O: T	T
B6	I2C1_SDA	I2C1 serial data input/output	-	VDD_IOB	I/O: T	T
I2C 2						
A3	I2C2_SCL	I2C2 serial clock input/output	-	VDD_IOC	I/O: T	T
E4	I2C2_SDA	I2C2 serial data input/output	-	VDD_IOC	I/O: T	T

Table 693. I2S pinouts for TFBGA296 package

BGA 296 Pin	I2S Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
I2S 0						
M17	I2S0RX_SDA	I2S0 receive data	P0.2 / LCDVD[4]	VDD_IOA	I/O	input
M18	I2S0RX_CLK	I2S0 receive clock	P0.3 / LCDVD[5]	VDD_IOA	I/O	input
L15	I2S0RX_WS	I2S0 receive word select	P0.4 / LCDVD[6]	VDD_IOA	I/O	input
L16	I2S0TX_SDA	I2S0 transmit data	P0.5 / LCDVD[7]	VDD_IOA	I/O	input

Table 693. I2S pinouts for TFBGA296 package

BGA 296 Pin	I2S Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
L17	I2S0TX_CLK	I2S0 transmit clock	P0.6 / LCDVD[12]	VDD_IOA	I/O	input
L18	I2S0TX_WS	I2S0 transmit word select	P0.7 / LCDVD[13]	VDD_IOA	I/O	input
I2S 1						
A4	I2S1TX_CLK	I2S1 transmit clock	MAT3.0	VDD_IOB	I/O	L
E7	I2S1TX_SDA	I2S1 transmit data	MAT3.1	VDD_IOB	I/O	input
B4	I2S1TX_WS	I2S1 transmit word select	CAP3.0	VDD_IOB	I/O	input
C16	I2S1RX_SDA	I2S1 receive data	GPI_0	VDD_IOD	I	input
B5	I2S1RX_CLK	I2S1 receive clock	P0.0	VDD_IOB	I/O	input
D7	I2S1RX_WS	I2S1 receive word select	P0.1	VDD_IOB	I/O	input

34.2.8 Timer & PWM pinouts

Table 694. Timer pinouts for TFBGA296 package

BGA 296 Pin	Timer Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
High Speed Timer						
C7	HSTIM_CAP	HS timer capture input	GPI_6 / ENET_RXD2	VDD_IOB	I:BK	input
Timer 0						
A8	MAT0.0	Timer 0 Mat 0	MS_DIO0	VDD_IOD	I/O: P	input
A7	MAT0.1	Timer 0 Mat 1	MS_DIO1	VDD_IOD	I/O: P	input
B8	MAT0.2	Timer 0 Mat 2	MS_DIO2	VDD_IOD	I/O: P	input
C8	MAT0.3	Timer 0 Mat 3	MS_DIO3	VDD_IOD	I/O: P	input
G13	CAP0.1	Timer 0 Cap 1	GPI_22 / U7_HCTS LCDCLKIN	VDD_IOD	I	input
E17	CAP0.0	Timer 0 Cap 0	GPI_23 / U7_RX / LCDVD[10]	VDD_IOD	I/O	input
Timer 1						
B14	MAT1.0	Timer 1 Mat 0	GPO_2 / LCDVD[0]	VDD_IOD	O	L
E18	MAT1.1	Timer 1 Mat 1	U7_TX / LCDVD[11]	VDD_IOD	O	H
K15	CAP1.0	Timer 1 Cap 0	GPI_15 / U1_RX	VDD_IOA	I/O	input
Timer 2						
B7	MAT2.0	Timer 2 Mat 0	MS_SCLK	VDD_IOD	I/O	L
A6	MAT2.1	Timer 2 Mat 1	MS_BS	VDD_IOD	I/O: P	L
C14	CAP2.0	Timer 2 Cap 0	GPI_2 / ENET_RXD3	VDD_IOD	I	input
Timer 3						
A4	MAT3.0	Timer 3 Mat 0	I2S1TX_CLK	VDD_IOB	I/O	L
E7	MAT3.1	Timer 3 Mat 1	I2S1TX_SDA	VDD_IOB	I/O	input
B4	CAP3.0	Timer 3 Cap 0	I2S1TX_WS	VDD_IOB	I/O	input
Timer 4						
D13	CAP4.0	Timer 4 Cap 0	GPI_7 / MCABORT	VDD_IOD	I	input

Table 695. Motor Control PWM pinouts for TFBGA296 package

BGA 296 Pin	PWM Function	Description	Alternate Function(s)	Voltage Domain	Type	Default
D11	MC0A	Motor Control PWM channel 0, output A.	GPO_18 / LCDLP	VDD_IOD	O	L
D10	MC0B	Motor Control PWM channel 0, output B.	GPO_16 / LCDENAB / LCDM	VDD_IOD	O	L
A14	MC1A	Motor Control PWM channel 1, output A.	GPO_15 / LCDFP	VDD_IOD	O	L
B13	MC1B	Motor Control PWM channel 1, output B.	GPO_13 / LCDDCLK	VDD_IOD	O	L
B12	MC2A	Motor Control PWM channel 2, output A.	GPO_12 / LCDLE	VDD_IOD	O	L
E11	MC2B	Motor Control PWM channel 2, output B.	GPO_10 / LCDPWR	VDD_IOD	O	L
D13	MCABORT	Motor Control PWM Fast Abort	GPI_7 / CAP4.0	VDD_IOD	I	input
E9	MCFB0	Motor Control PWM channel 0, feedback input.	GPIO_5 / SSEL0	VDD_IOD	I/O	input
C10	MCFB1	Motor Control PWM channel 1, feedback input.	SPI1_DATIN / MISO0 / GPI_25	VDD_IOD	I/O	input
B9	MCFB2	Motor Control PWM channel 2, feedback input.	SPI1_DATIO / MOSI0	VDD_IOD	I/O	input

34.2.9 Power pinout

Table 696. Power pinout for TFBGA296 package

BGA 296 Pin	Name	Description	Voltage Domain	Type	Default
N12	VDD_AD	3.3 V supply for ADC (TS)	VDD_AD	power	-
N13	VDD_AD	3.3 V supply for ADC (TS)	VDD_AD	power	-
G7	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
G9	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
G11	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
J7	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
J12	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
M7	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
M11	VDD_CORE	1.2 V (or 0.9 V) supply for core	VDD_CORE	power	-
L12	VDD_COREFXD	Fixed 1.2 V supply for core	VDD_COREFXD	power	-
M13	VDD_COREFXD	Fixed 1.2 V supply for core	VDD_COREFXD	power	-
J6	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
K6	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
K7	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
L6	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
M6	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
M8	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
N7	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
N8	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
N9	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-

Table 696. Power pinout for TFBGA296 package

BGA 296 Pin	Name	Description	Voltage Domain	Type	Default
N10	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
N11	VDD_EMC	1.8 V/2.5 V/3.3 V supply for EMC	VDD_EMC	power	-
N14	VDD_FUSE	Fuse VPP supply	(special)	power	-
H13	VDD_IOA	Supply for 1.8 V or 3.3 V I/O	VDD_IOA	power	-
J13	VDD_IOA	Supply for 1.8 V or 3.3 V I/O	VDD_IOA	power	-
F8	VDD_IOB	Supply for 1.8 V/3.3 V; for GPI_6, GPO_4/11, I2C1	VDD_IOB	power	-
F7	VDD_IOC	Supply for 1.8 V/3.3 V I/O	VDD_IOC	power	-
G6	VDD_IOC	Supply for 1.8 V/3.3 V I/O	VDD_IOC	power	-
H6	VDD_IOC	Supply for 1.8 V/3.3 V I/O	VDD_IOC	power	-
J5	VDD_IOC	Supply for 1.8 V/3.3 V I/O	VDD_IOC	power	-
F9	VDD_IOD	Supply for 1.8 V/3.3 V I/O	VDD_IOD	power	-
F13	VDD_IOD	Supply for 1.8 V/3.3 V I/O	VDD_IOD	power	-
T18	VDD_OSC	1.2 V supply for main oscillator	VDD_OSC	power	-
T16	VDD_PLL397	1.2 V supply for 397x PLL	VDD_PLL397	power	-
R17	VDD_PLLHCLK	1.2 V supply for HCLK PLL	VDD_PLLHCLK	power	-
P15	VDD_PLLUSB	1.2 V supply for USB PLL	VDD_PLLUSB	power	-
K14	VDD_RTC	1.2 V supply for RTC I/O	VDD_RTC	power	-
L13	VDD_RTCCORE	1.2 V supply for RTC	VDD_RTC	power	-
N15	VDD_RTCOSC	1.2 V supply for RTC oscillator	VDD_RTC	power	-

Table 697. Ground pinout for TFBGA296 package

BGA 296 Pin	Name	Description	Voltage Domain	Type	Default
P13	VSS_AD	Ground for ADC (Touchscreen)	VDD_AD	power	-
G8	VSS_CORE	Ground for core	VDD_CORE	power	-
G10	VSS_CORE	Ground for core	VDD_CORE	power	-
G12	VSS_CORE	Ground for core	VDD_CORE	power	-
H7	VSS_CORE	Ground for core	VDD_CORE	power	-
K12	VSS_CORE	Ground for core	VDD_CORE	power	-
L7	VSS_CORE	Ground for core	VDD_CORE	power	-
M9	VSS_CORE	Ground for core	VDD_CORE	power	-
M10	VSS_CORE	Ground for core	VDD_CORE	power	-
M12	VSS_CORE	Ground for core	VDD_CORE	power	-
K5	VSS_EMC	Ground for EMC	VDD_EMC	power	-
L5	VSS_EMC	Ground for EMC	VDD_EMC	power	-
M5	VSS_EMC	Ground for EMC	VDD_EMC	power	-
N5	VSS_EMC	Ground for EMC	VDD_EMC	power	-
N6	VSS_EMC	Ground for EMC	VDD_EMC	power	-
P6	VSS_EMC	Ground for EMC	VDD_EMC	power	-

Table 697. Ground pinout for TFBGA296 package

BGA 296 Pin	Name	Description	Voltage Domain	Type	Default
P7	VSS_EMC	Ground for EMC	VDD_EMC	power	-
P8	VSS_EMC	Ground for EMC	VDD_EMC	power	-
P9	VSS_EMC	Ground for EMC	VDD_EMC	power	-
P10	VSS_EMC	Ground for EMC	VDD_EMC	power	-
P11	VSS_EMC	Ground for EMC	VDD_EMC	power	-
K13	VSS_IOA	Ground for VDD_IOA voltage domain (1.8 V or 3.3 V I/O)	VDD_IOA	power	-
F6	VSS_IOB	Ground for VDD_IOB voltage domain (GPI_6, GPO_4/11, I2C1)	VDD_IOB	power	-
F5	VSS_IOC	Ground for VDD_IOC voltage domain	VDD_IOC	power	-
G5	VSS_IOC	Ground for VDD_IOC voltage domain	VDD_IOC	power	-
H5	VSS_IOC	Ground for VDD_IOC voltage domain	VDD_IOC	power	-
F10	VSS_IOD	Ground for VDD_IOD voltage domain	VDD_IOD	power	-
F11	VSS_IOD	Ground for VDD_IOD voltage domain	VDD_IOD	power	-
F12	VSS_IOD	Ground for VDD_IOD voltage domain	VDD_IOD	power	-
H12	VSS_IOD	Ground for VDD_IOD voltage domain	VDD_IOD	power	-
P14	VSS_OSC	Ground for main oscillator	VDD_OSC	power	-
T15	VSS_PLL397	Ground for 397x PLL	VDD_PLL397	power	-
R18	VSS_PLLHCLK	Ground for HCLK PLL	VDD_PLLHCLK	power	-
R16	VSS_PLLUSB	Ground for USB PLL	VDD_PLLUSB	power	-
L14	VSS_RTCCORE	Ground for RTC	VDD_RTC	power	-
P18	VSS_RTCOSC	Ground for RTC oscillator	VDD_RTC	power	-

35.1 Features

The LPC32x0 supports several boot options. The boot process can identify, copy to internal SRAM (IRAM) and execute an external program from the following external sources:

- UART boot using either the UART5 or USB transceiver.
- SLC/MLC NAND Flash boot.
- SPI boot using SSP0.

The boot process can also identify and execute in place an external program from the following external source:

- EMC boot from external static memory.

35.2 Description

The built-in 16 KB ROM contains a program which runs a boot procedure to load code from one of four external sources, UART5, SSP0 (in SPI mode), EMC Static CS0 memory, or NAND FLASH to internal RAM (IRAM).

After reset, execution always begins from the internal ROM. The program in ROM is called the bootstrap and is described here and a top level flowchart is shown in Figure 129.

Remark: While the bootstrap code executes, it reserves IRAM space from 0x0000 E000 to 0x0000 FFFF for stack, variables and other software functions. The bootstrap loads user boot code as a single contiguous block and if the specified code size is greater than 56 kB it will corrupt the IRAM space reserved by the bootstrap code and possibly fail. This limits the size of user boot code loaded by the bootstrap to 56 kB or less from UART boot (with start address < 0x0000 FFFF), NAND Flash Boot from large page device or SPI boot. This limit does not apply to EMC boot.

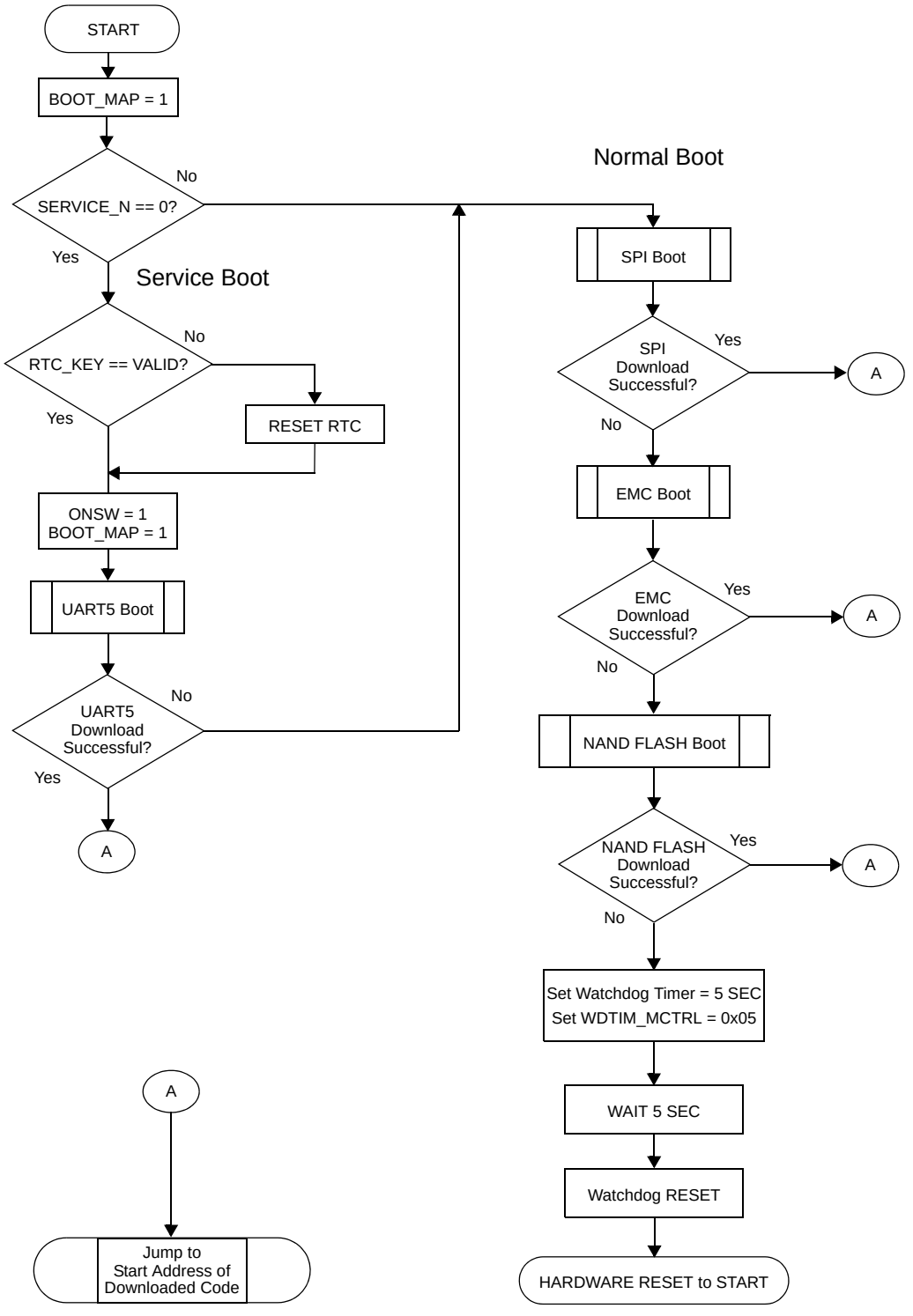


Fig 129. Bootloader top level flow chart

The bootstrap software starts by reading the SERVICE_N input (GPI_1). If SERVICE_N is low, the bootstrap does a service boot. The code transferred to IRAM will typically be secondary programming software. If the SERVICE_N pin is high, the bootstrap routine jumps to normal boot.

The normal boot procedure starts by testing for a bootable SPI memory connected to SSP0. If successful, the code is copied to executable memory (IRAM) and software control is passed to the transferred code by jumping to the start address of the code block. If the test for SPI memory fails, the bootloader tests for the presence of boot code in EMC Static CS0 memory. If boot code is available in static memory, the memory is configured and the bootloader jumps to 0xE000 0004 and begins execution. If boot code is not available in external static memory, the bootloader tests for the presence of boot code in NAND Flash. If boot code is present it transfers a block of code to IRAM and software control is transferred to the start address of the boot code.

The next two sections describe the service boot and normal boot procedures in greater detail.

35.2.1 Service Boot

The first step in a service boot is to check the RTC_KEY register value, if the RTC_KEY is valid, it will retain the information from the last initialization done by application software. The RTC_KEY register must be set to the value 0xB5C13F27. If the RTC_KEY is not valid, the RTC will be reset using the “software controlled RTC reset” bit in the RTC_CTRL register.

Next, the bootstrap sets the ONSW output pin high and remaps IRAM to 0x0000 0000 by.

35.2.1.1 UART 5 Boot

The Bootstrap begins a UART 5 boot. The UART 5 boot starts by checking if a transceiver is connected or if the transceiver is OTG compliant. Note: the bootloader only supports a single I^2C device address, which is 0x2c. It does not test for the other three available device addresses, 0x2d, 0x2e, and 0x2f. If using an ISP1301 transceiver this means setting the ADR/PWR pin (A0) low during reset.

If a transceiver is connected and compliant it will set the USB transceiver to UART mode using the USB I2C. See Section 17.2.3.2.6 in the USB OTG Controller chapter for more details. The bootstrap will then initialize UART5 and begin the UART data download protocol.

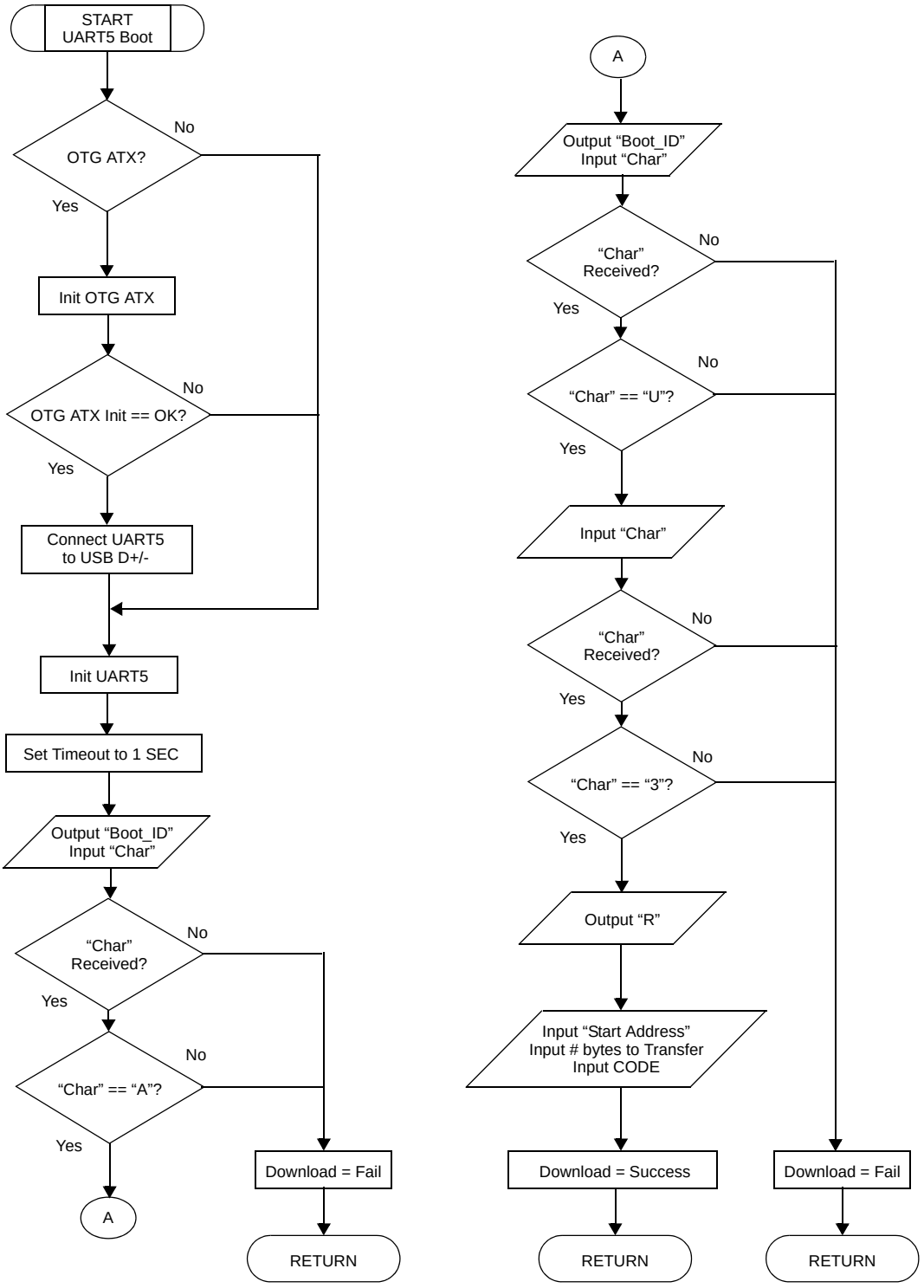


Fig 130. UART SERVICE boot procedure

The UART data download protocol expects an external device to be connected to either UART5 or the USB transceiver set to UART mode. The first action of the data download protocol is to send the boot_id from UART5, the peripheral is expected to respond with an 'A' (0x41). If an 'A' is received, UART5 retransmits the boot_id. The peripheral is expected to respond by sending a 'U' (0x55) and a '3' (0x33). If the 'U' and the '3' are received, then UART5 responds with an 'R' (0x52). The peripheral is then expected to send a start address (32 bit value; sent as four sequential UART bytes), followed by the number of bytes of code to transfer (32 bit value; sent as four sequential UART bytes). At this point, the code will be transferred. The code transfer is stored byte by byte beginning with the start address, and when the correct number of bytes is received, the bootstrap exits by jumping to the start address of the transferred code. The bootstrap has relinquished control and transferred execution to the downloaded program.

If the 'A', 'U', '3' sequence not is received within one second, it will time-out, and the bootstrap jumps to the Normal boot procedure.

Table 698. UART boot handshake

LPC32x0	Connected device
Boot ID ⇒	1 byte
⇐ 'A'	1 byte
Boot ID ⇒	1 byte
⇐ 'U' (0x55)	1 byte
⇐ '3' (0x33)	1 byte
'R' (0x52) ⇒	1 byte
⇐ start address	4 bytes
⇐ code size	4 bytes
⇐ code	

Table 699. Bootstrap download protocol communication parameters for UART5

Parameter	Description
Data bits	8
Parity	None
Stop bits	1
Speed	115200. The 13 MHz crystal must be connected to SYSX_IN/OUT for the bootloader to operate properly.
Start address	32 bits. Least significant byte first
Number of bytes	32 bits. Least significant byte first

35.2.1.1.1 Boot block register map

The Boot Block Registers are implemented as ROM locations in the IROM, not as real registers.

Table 700. Interface Configuration data (ICR)

Address	Section	Functional block	Register	Type	Width	Reset value
0x0C00 3FFC	AHB-1	Boot	BOOT_ID	R	31:0	0x35 = '5'

Register name: **BOOT_ID**

Function: Boot identification register

Table 701. BOOT_ID

Bit #	Bit name	Function
31:0	Boot_id	Boot_id = 0x35 = '5' The Boot_id is the version number of the boot code software. It will be changed only when a change is made in the boot code.

35.2.2 Normal Boot

The normal boot procedure starts by testing for a bootable SPI memory connected to SSP0. If successful, the code is copied to executable memory (IRAM) and software control is passed to the transferred code by jumping to the start address (0x0000 0000) of the code block. If the test for SPI memory fails, the bootloader tests for the presence of boot code in EMC Static CS0 memory. If boot code is available in static memory, the memory is configured and the bootloader jumps to 0xE000 0004 and begins execution. If boot code is not available in external static memory, the bootloader tests for the presence of boot code in NAND Flash. If boot code is present it transfers a block of code to IRAM and software control is transferred to the start address (0x0000 0000) of the boot code.

35.2.2.1 SPI Boot

Pin Description

Table 702. SPI boot SSP0 and related boot pins

Byte Address	Function
SPI1_CLK/SCK0	SCK0
SPI1_DATIN[1]	MISO0
SPI1_DATIO	MOSI0
GPIO_5	GPIO output to drive the SSEL pin of the external SPI flash.

[1] The pin SPI1_DATIN is also checked by the bootloader to differentiate between SPI flash and SPI EEPROM: 0 = Boot from SPI Flash, 1 = Boot from SPI EEPROM

Boot Procedure

The bootloader first reads the validation word and the download-data-length from the SPI memory device. If the validation word is not correct or the download-data-length is either 0xFFFFFFFF or 0x00000000 it means a bootable SPI memory device is not connected or the boot data is not valid in the SPI device.The SPI boot fails and continues to the next possible boot device.

If there's valid data, the bootloader reads and copies all required data from the SPI memory device to the IRAM starting at 0x00000000. The bootloader then sets the BOOT_MAP register to 1 and transfers execution to the loaded code by jumping to 0x00000000.

Important: The bootloader does not check if the download length exceeds the size of the IRAM for the specific part. An attempt to store more than the 56 Kbytes allowed into the IRAM may result in unpredictable behavior.

The validation word 0x13579BDF and download-data-length are stored as the first two 32-bit words (in Little Endian format) in the SPI memory device. The user data is stored after the download data length word.

Table 703. SPI flash/EEPROM data format (standard mode)

Byte Address	Memory content
1	0xDF
2	0x9B
3	0x57
4	0x13
5	Download_data_length byte 0
6	Download_data_length byte 1
7	Download_data_length byte 2
8	Download_data_length byte 3
9	Data byte 0
…	Data byte 1
…	…
Download_data_length + 7	Last Data byte (Download_data_length)

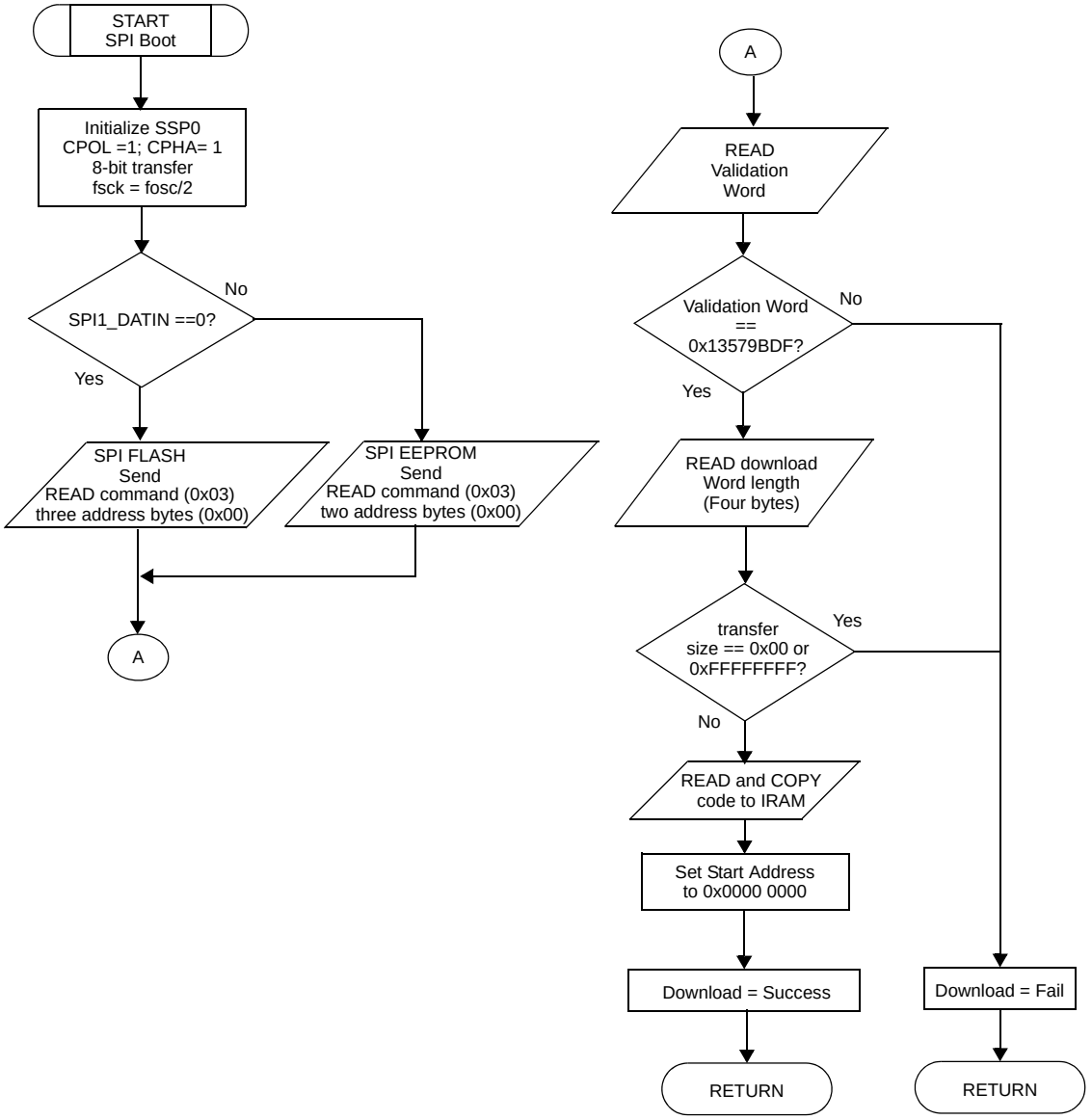


Fig 131. SPI boot procedure

35.2.2.2 EMC Static Memory Boot

Boot Procedure

The bootloader first reads the SPI1_DATIN pin value and writes it to the PB bit in the EMCStaticConfig0 register. Then the bootloader reads the memory width configuration nibble (MW) at address 0xE000 0000 to determine the memory width. If it's not valid memory width, the EMC boot fails and bootloader continues with the next boot attempt. Otherwise, the bootloader write the memory width value to the EMCStaticConfig0 register for the following accesses.The bootloader then checks the validation word at 0x0. If it's

not valid, the EMC boot fails and bootloader continues with the next boot attempt. Otherwise, the bootloader sets BOOT_MAP to 1 and jumps to 0xE000 0004 to execute the code.

Data format stored in the external static memory is shown in Table 704 and Table 705. The validation code and memory width configuration is stored as a 32-bit word data in the format of 0x13579BDm at address 0xE000 0000.

The only acceptable validation words at 0xE000 0000 are 0x13579BD2 for 32-bit memory, 0x13579BD1 for 16-bit memory, and 0x13579BD0 for 8-bit memory.

Table 704. EMC parallel static memory data format CS0

Address	Memory content
0xE000 0000	0x13579BDm[1]
0xE000 0004	Word 0
0xE000 0008	Word 1
...	...
...	Last Word in EMC memory

[1] The nibble **m** represents the width of static memory width (MW) see Table 705 for acceptable values.

Table 705. EMC static memory width format nibble MW

lower nibble at 0xE000 0000	Memory Width
nn00[1]	8-bit
nn01[1]	16-bit
nn10[1]	32-bit
nn11[1]	Reserved

[1] The two bits marked **nn** should be zero.

The static memory bank 0 (CS0) is only static memory which is bootable and can be 8-bit, 16-bit or 32-bit wide.

Remark: The value of the pin SPI1_DATIN is read by the bootloader and written to the PB bit in EMCStaticConfig0 register to ensure correct BLSn behavior for the specific memory structure.

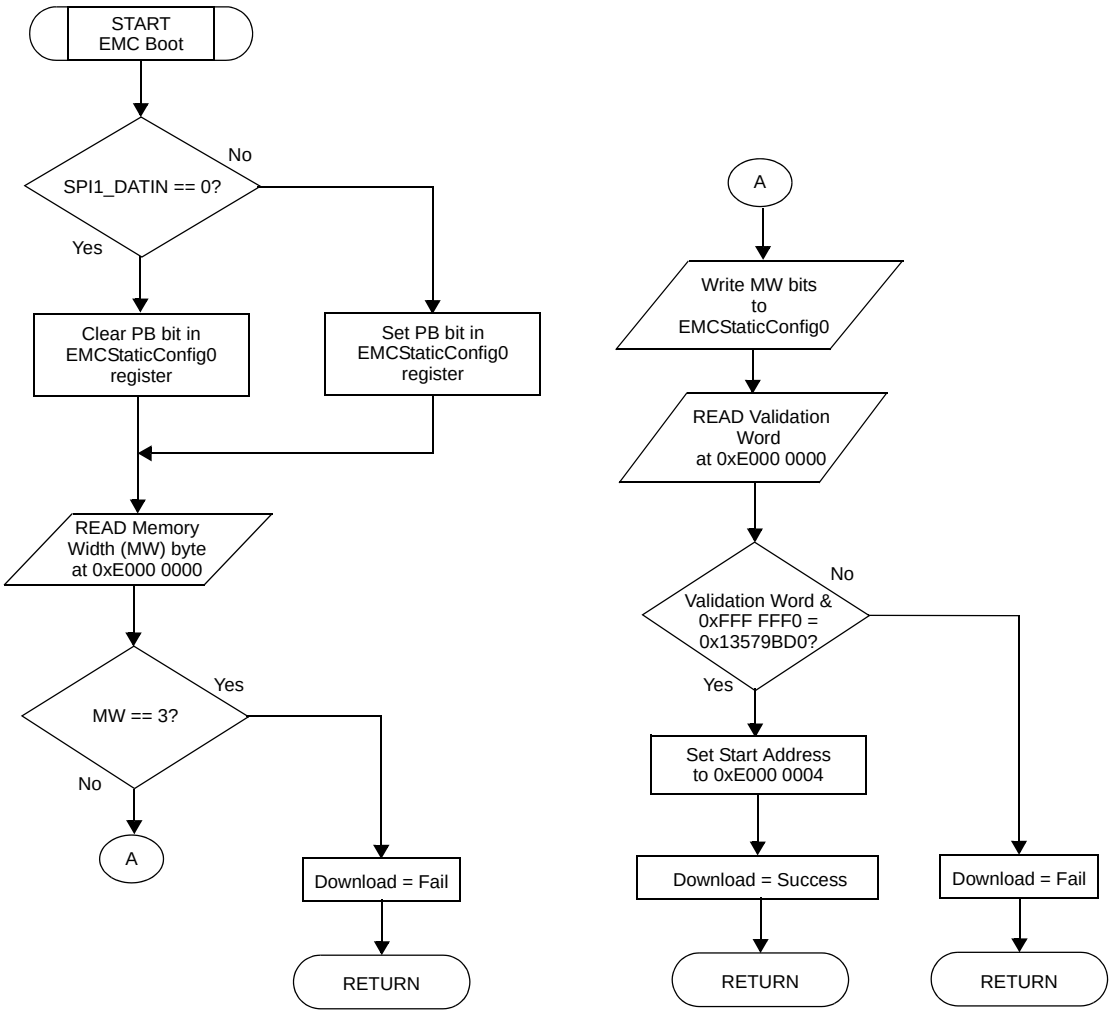


Fig 132. EMC boot procedure

35.2.2.3 NAND flash boot procedure

Table 706. NAND flash devices recognized by the bootloader[1]

Manufacturer	Flash number	Flash size	FLASH IO with	FLASH Maker Code	FLASH Device Code	Address cycle	use 0x30 command
SAMSUNG	K9F5608Q0B	256 Mbit	8 bit	ECh	35h	3	no
SAMSUNG	K9K1208Q0C	512 Mbit	8 bit	ECh	36h	4	no
SAMSUNG	K9F1208Q0C	512 Mbit	8 bit	ECh	36h	4	no
SAMSUNG	K9F1G08Q0C	1Gbit	8 bit	ECh	A1h	4	Yes
SAMSUNG	K9F2G08Q0C	2 Gbit	8 bit	ECh	AAh	5	Yes

[1] The NAND devices shown in this table are hardcoded in the boot ROM. The LPC32x0 boot ROM will also automatically detect other large and small page NAND flash devices using a generic flash detect process.

35.2.2.3.1 How the flash boot procedure reads data from flash and transfers it to IRAM

While booting from NAND Flash, the boot code needs to find out how many pages to copy and the type of NAND Flash. The first page in the first block or the second block of the NAND Flash shall contain the information needed for the boot code to work.

The boot loader tries to identify the NAND Flash device by reading the device ID. If any of the devices listed in Table 706 are connected, then the bootloader will proceed to transfer data from the NAND flash device to IRAM. If the read ID command fails to identify the NAND flash device the bootloader will continue to query the memory and try to determine if a valid ICR is present in the NAND flash and use it to work out the configuration information provided by a valid ICR.

The boot code reads out the first page with the MLC NAND Flash controller configured in 16-bit mode. This means that only the first byte in word 0, word 2, word 4 … etc. contains the data needed for the bootloader.

Table 707. 8-bit flash read as 16-bit flash

word 0		word 1		word 2		word 3		word 4		word 5		word 6		word 7		word 8		word 9	
XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
	d0				d1				d2				d3				d4		

Once the ICR is read, the boot code sets the MLC NAND Flash controller to the appropriate mode and copies the required amount of code to IRAM. The code must be stored using the Reed-Solomon Encoding implemented in the MLC NAND Flash controller. The boot code is not capable of skipping over bad blocks. Therefore, a secondary boot code which is capable of skipping over bad blocks should be implemented. The NAND Flash devices are shipped from the factory with Block 0 always valid. The block size of the Small Page NAND Flash is 16 KB and the block size of the Large Page NAND Flash is 128 KB. The secondary boot code should not exceed 15.5 KB (Small page) or 54 KB (Large page).

35.2.2.3.2 How to store Interface Configuration data (ICR) in the flash

Data d0 to d3 are the ICR and nICR. These data are used to decide which Flash is connected and what read algorithm to use to read out the Flash contents when the Flash ID is not known by the boot code.

The ICR and nICR are stored four times into the Flash. This reduces the chance for bit errors to be fatal in the boot up sequence.

Table 708. Interface Configuration data (ICR)

Bit 7 = nbit 4	Bit 6 = nbit 2	Bit 5 = nbit 1	Bit 4 = nbit 0	ICR bit 3	ICR bit 2 small/large page	ICR bit 1 3/4 address	ICR bit0 8/16 bit	
1	1	1	1	0	0	0	0	small page, 3 address cycles, 8 bit interface
1	1	0	1	0	0	1	0	small page, 4 address cycles, 8 bit interface
1	0	1	1	0	1	0	0	large page, 4 address cycles, 8 bit interface
1	0	0	1	0	1	1	0	large page, 5 address cycles, 8 bit interface

35.2.2.3.3 How to store size information in the flash

Data d4 to d11 is used to store the size of the code to be copied from Flash to IRAM by the boot code. The size is specified as the (number of pages +1).

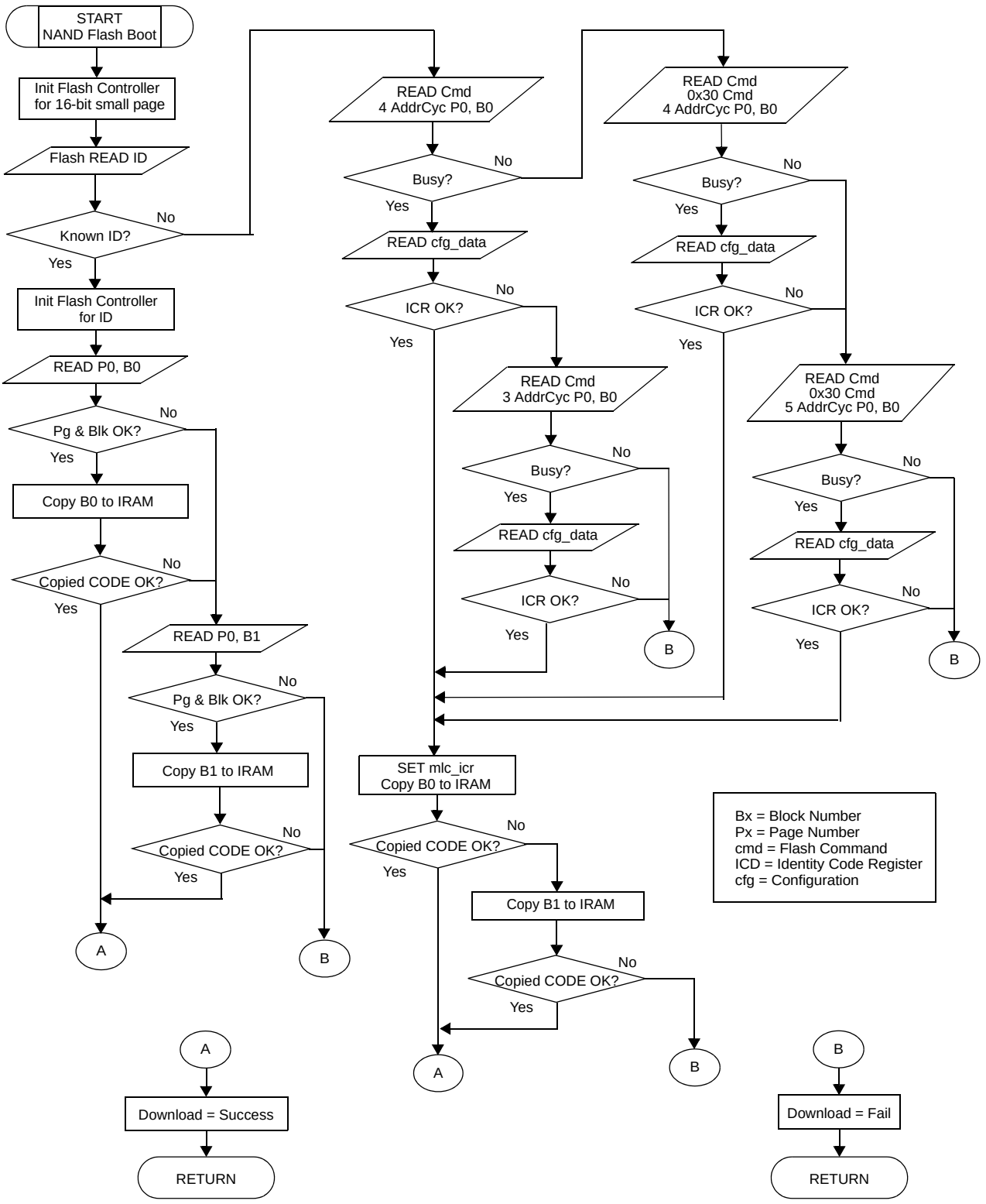
d4, d6, d8 and d10 shall be programmed to contain the size information.

d5, d7,d9 and d11 shall be programmed to contain the inverse of the size information.

The boot code XORs d4 and d5 and if the answer is 0xff then the right size information is in d4. If the XOR between d4 and d5 is not 0xff, then the boot code does an XOR between d6 and d7 and test against 0xff. If the results is 0xff then the right size information is in d6. If none of the d4 d5, d6 d7, d8, d9,d10 and d11 gives a XOR value equal to 0xff then the size of the transfer is unknown and the boot code fails.

35.2.2.3.4 How to store bad_block information

Data d12 holds the bad_block information for block 0. If block 0 is ok then d12 must be programmed to 0xaa.



The boot code sets the MLC NAND Flash controller to 16 bit mode, but only 8 bits are routed out of the package.

Fig 133. NAND FLASH boot procedure

35.2.3 Other advanced boot options

More advanced boot options such as boot from USB and Ethernet can be implemented as a secondary bootloader which is stored in the SPI flash, NAND flash, or Parallel static memory (EMC). The bootloader first loads the secondary bootloader code from the external SPI flash, NAND flash, or Parallel static memory (EMC) and then executes the secondary bootloader.

36.1 Features

- No target resources are required by the software debugger in order to start the debugging session.
- Allows the software debugger to talk via a JTAG (Joint Test Action Group) port directly to the core.
- Inserts instructions directly in to the ARM core.
- The ARM core or the System state can be examined, saved or changed depending on the type of instruction inserted.
- Allows instruction to execute at a slow debug speed or at a fast system speed.

36.2 Applications

The EmbeddedICE-RT logic provides on-chip debug support. The debugging of the target system requires a host computer running the debugger software and an EmbeddedICE-RT protocol converter. EmbeddedICE-RT protocol converter converts the Remote Debug Protocol commands to the JTAG data needed to access the ARM core present on the target system.

36.3 Description

The ARM Debug Architecture uses a JTAG port as a method of accessing the core. The scan chains that are around the core for production test are reused in the debug state to capture information from the databus and to insert new information into the core or the memory. There are two JTAG-style scan chains within the ARM core. A JTAG-style Test Access Port Controller controls the scan chains. In addition to the scan chains, the debug architecture uses EmbeddedICE-RT logic which resides on chip with the ARM core. The EmbeddedICE-RT has its own scan chain that is used to insert watchpoints and breakpoints for the ARM core.[21] The EmbeddedICE-RT logic consists of two real time watchpoint registers, together with a control and status register. One or both of the watchpoint registers can be programmed to halt the ARM core. Execution is halted when a match occurs between the values programmed into the EmbeddedICE-RT logic and the values currently appearing on the address bus, databus and some control signals. Any bit can be masked so that its value does not affect the comparison. Either watchpoint register can be configured as a watchpoint (i.e. on a data access) or a break point (i.e. on an instruction fetch). The watchpoints and breakpoints can be combined such that:

The conditions on both watchpoints must be satisfied before the ARM core is stopped. The CHAIN functionality requires two consecutive conditions to be satisfied before the core is halted. An example of this would be to set the first breakpoint to trigger on an access to a peripheral and the second to trigger on the code segment that performs the task switching. Therefore when the breakpoints trigger, the information regarding which task has switched out will be ready for examination.

21. For more details refer to IEEE Standard 1149.1 - 1990 Standard Test Access Port and Boundary Scan Architecture

The watchpoints can be configured such that a range of addresses are enabled for the watchpoints to be active. The RANGE function allows the breakpoints to be combined such that a breakpoint is to occur if an access occurs in the bottom 256 bytes of memory but not in the bottom 32 bytes.

The ARM core has a Debug Communication Channel function built in. The debug communication channel allows a program running on the target to communicate with the host debugger or another separate host without stopping the program flow or even entering the debug state. The debug communication channel is accessed as a co-processor 14 by the program running on the ARM core. The debug communication channel allows the JTAG port to be used for sending and receiving data without affecting the normal program flow. The debug communication channel data and control registers are mapped in to addresses in the EmbeddedICE-RT logic

36.4 Pin description

Table 709. EmbeddedICE-RT pin description

Pin name	Type	Description
TMS	Input	**Test Mode Select.** The TMS pin selects the next state in the TAP state machine.
TCK	Input	**Test Clock.** This allows shifting of the data in, on the TMS and TDI pins. It is a positive edge-triggered clock with the TMS and TCK signals that define the internal state of the device.
TDI	Input	**Test Data In.** This is the serial data input for the shift register.
TDO	Output	**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
TRST	Input	**Test Reset.** The $\overline{\text{TRST}}$ pin can be used to reset the test logic within the EmbeddedICE-RT logic.
RTCK	Output	**Returned Test Clock** Extra signal added to the JTAG port. Required for designs based on ARM processor core. Development systems can use this signal to maintain synchronization with targets. For details refer to "Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)".

36.5 Block diagram

The block diagram of the debug environment is shown below in Figure 134.

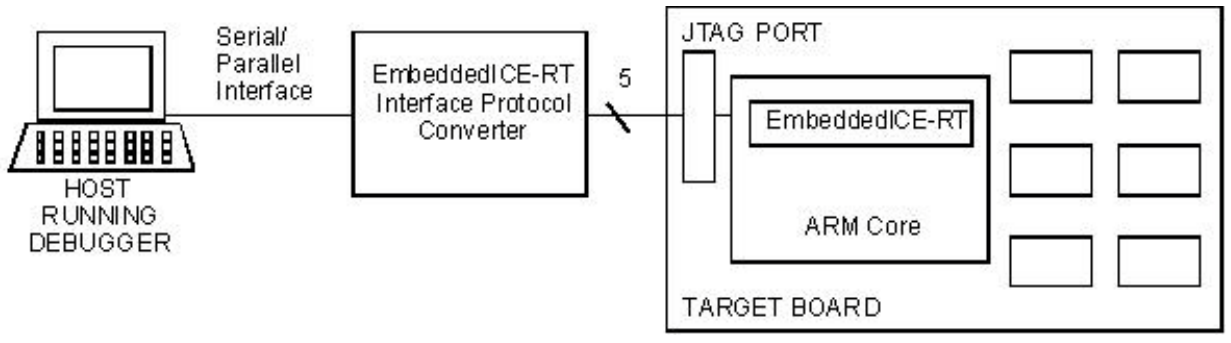


Fig 134. EmbeddedICE-RT debug environment block diagram

The TAPID register depends on the debug mode selected. The Embedded Trace Buffer (ETB) ID is 0x1B90 0F0F (see Table 710).

Table 710. TAPID/ETB register values

Chip revision	TAPID (DBGEN = LOW)	TAPID (DBGEN = HIGH)	ETB ID
Rev '—'	0x1790 0F0F	0x08D2 402B	0x1B90 0F0F
Rev A	0x1792 6F0F	0x88D2 402B	0x1B90 0F0F

37.1 Features

- Closely tracks the instructions that the ARM core is executing.
- On-chip trace data storage (ETB).
- All registers are programmed through JTAG interface.
- Does not consume power when trace is not being used.
- THUMB/Java instruction set support.

37.2 Applications

As the microcontroller has significant amounts of on-chip memory, it is not possible to determine how the processor core is operating simply by observing the external pins. The ETM provides real-time trace capability for deeply embedded processor cores. It outputs information about processor execution to a trace port. A software debugger allows configuration of the ETM using a JTAG interface and displays the trace information that has been captured, in a format that a user can easily understand. The ETB stores trace data produced by the ETM.

37.3 Description

The ETM is connected directly to the ARM core and not to the main AMBA system bus. It compresses the trace information and exports it through a narrow trace port. An internal Embedded Trace Buffer captures the trace information under software debugger control. ETM can broadcast the Instruction/data trace information. Bytecodes executed while in Java state can also be traced. The trace contains information about when the ARM core switches between states. Instruction trace (or PC trace) shows the flow of execution of the processor and provides a list of all the instructions that were executed. Instruction trace is significantly compressed by only broadcasting branch addresses as well as a set of status signals that indicate the pipeline status on a cycle by cycle basis. For data accesses either data or address or both can be traced. Trace information generation can be controlled by selecting the trigger resource. Trigger resources include address/data comparators, counters and sequencers. Since trace information is compressed the software debugger requires a static image of the code being executed. Self-modifying code can not be traced because of this restriction.

37.3.1 ETM9 configuration

The following standard configuration is selected for the ETM9 macrocell.

Table 711. ETM configuration

Resource description	Quantity[1]
Pairs of address comparators	8
Data Comparators	8
Memory Map Decoders	16
Counters	4
Sequencer Present	Yes
External Inputs	4 (Not brought out)
External Outputs	4 (Not brought out)
FIFOFULL Present	Yes
FIFO depth	45 bytes
Trace Packet Width	4/8/16 (Trace pins are not brought out)

[1] For details refer to ARM documentation “Embedded Trace Macrocell Specification (ARM IHI 0014E)”.

37.3.2 ETB configuration

The ETB has a 2048 × 24 bit RAM for instruction/data history storage.

37.4 Block diagram

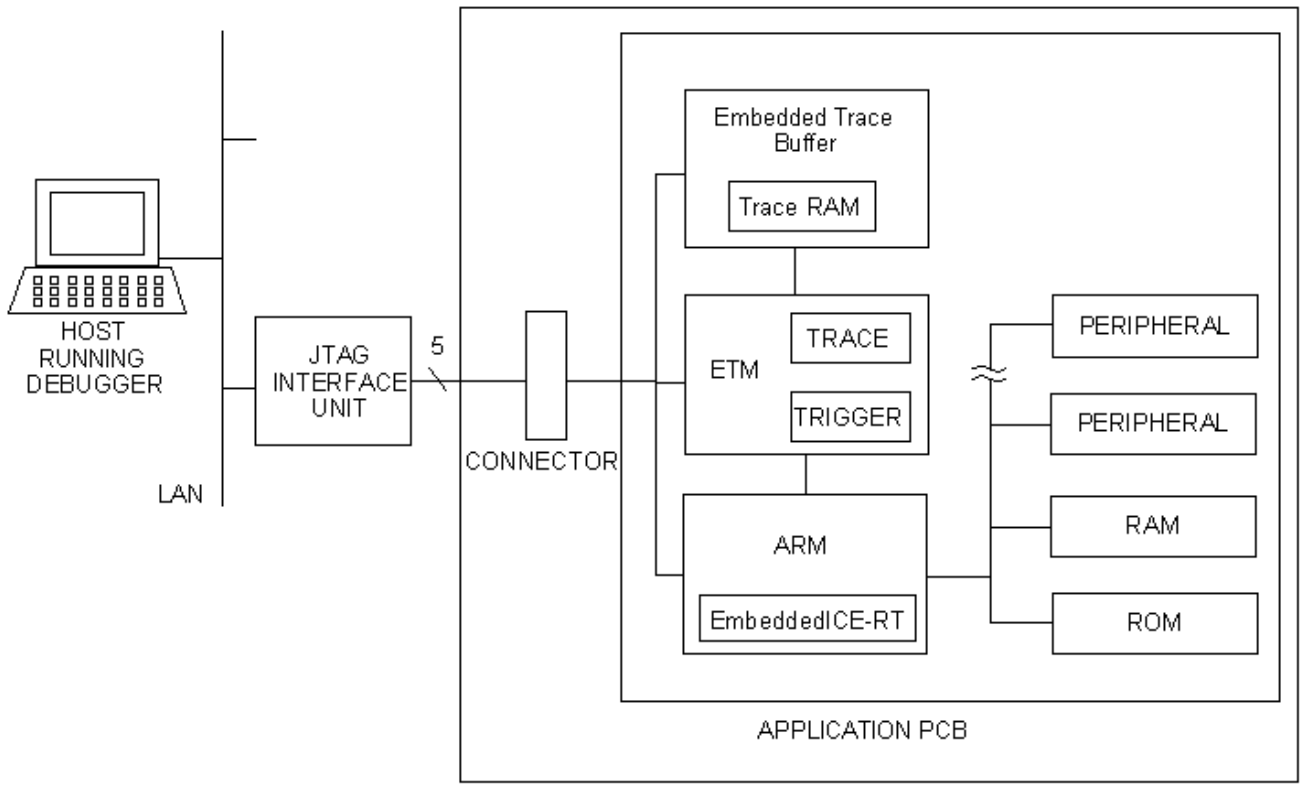


Fig 135. ETM/ETB debug environment block diagram

37.5 Register description

37.5.1 Debug Control register (DEBUG_CTRL, RW - 0x4004 0000)

Table 712. Debug Control register (DEBUG_CTRL, RW - 0x4004 0000)

Bits	Description	Reset value
31:5	Reserved	-
4	VFP9_CLKEN Controls VFP9 GCLK 0: CLK to VFP9 stopped 1: CLK enabled. (Default)	1
3	VFP_BIGEND Controls endianess of VFP 0: VFP use same endianess as the ARM926 (Default) 1: Force big endian. The combination: VFP_BIGEND = 1 and VFP9_CLKEN = 0 makes the ARM input CPEN low. This may save some power when VFP is not in use.	0
2	ARMDBG_DIS Controls if the debug logic is enabled or not. The core current is less with debug off. 0: ARM debug logic is on. (Default) 1: ARM debug logic is off.	0
1	Reserved	0
0	Reserved	0

37.5.2 Master Grant Debug Mode register (DEBUG_GRANT, RW - 0x4004 0004)

Table 713. Master Grant Debug Mode register (DEBUG_GRANT, RW - 0x4004 0004)

Bits	Description	Reset value
31:10	Not used. Write is don't care, Read returns random value	-
9:8	Reserved	-
7	USB Master. See description for bit 0	0
6:2	Reserved	-
1	DMA M1 Master	0
0	DMA M0 Master. If this bit is programmed to a one, the master will be allowed to finish it's current AHB Master access when ARM enters debug mode, but after the access the AHB matrix will withdraw the bus grant and prevent the master from doing more AHB transfers as long as ARM is in debug mode. When the ARM exits debug mode, the masters will be granted bus access again. The ARM debug output signal shall be synchronized to the AHB matrix HCLK before going into the matrix as the "debug_req" signal. 0 = Do not force GRANT inactive in ARM debug mode. (Default) 1 = Force GRANT inactive in ARM debug mode.	0

37.5.3 ETM registers

Please refer to ARM9 Embedded Trace Macrocell (ETM9) Technical Reference manual published by ARM

37.5.4 ETB registers

Please refer to Embedded Trace Buffer Technical Reference manual published by ARM.

38.1 Abbreviations

Table 714. Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AMBA	Advanced Microcontroller Bus Architecture
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
CISC	Complex Instruction Set Computer
CGU	Clock Generation Unit
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
FIQ	Fast Interrupt Request
GPIO	General Purpose Input/Output
IrDA	Infrared Data Association
IRQ	Interrupt Request
LCD	Liquid Crystal Display
PLL	Phase-Locked Loop
RISC	Reduced Instruction Set Computer
SD/MMC	Secure Digital/MultiMedia Card
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

38.2 Legal information

38.2.1 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.

38.2.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

38.2.3 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.

38.3 Tables

38.4 Figures

38.5 Contents

Chapter 1: LPC32x0 Introductory information

Chapter 2: LPC32x0 Bus architecture and memory map

Chapter 3: LPC32x0 System control block

Chapter 4: LPC32x0 Clocking and power control

Chapter 5: LPC32x0 Interrupt controller

Chapter 6: LPC32x0 General purpose DMA (GPDMA)

Chapter 7: LPC32x0 External Memory Controller (EMC)

Chapter 8: LPC32x0 Multi-level NAND flash controller

Chapter 9: LPC32x0 Single-level NAND flash controller

Chapter 10: LPC32x0 LCD controller

Chapter 11: LPC32x0 Touch screen controller

Chapter 12: LPC32x0 Analog-to-Digital Converter (ADC)

Chapter 13: LPC32x0 Keyboard scan interface

Chapter 14: LPC32x0 Ethernet Media Access Controller (MAC)

Chapter 15: LPC32x0 USB device controller

Chapter 16: LPC32x0 USB host (OHCI) controller

Chapter 17: LPC32x0 USB OTG controller

Chapter 18: LPC32x0 Secure Digital (SD) card interface

Chapter 19: LPC32x0 Standard Universal Asynchronous Receiver/Transmitter (UART)

Chapter 20: LPC32x0 14-clock Universal Asynchronous Receiver/Transmitter (UART)

Chapter 21: LPC32x0 SPI master interface

Chapter 22: LPC32x0 SSP interface

Chapter 23: LPC32x0 I2C interface

Chapter 24: LPC32x0 I2S interface

Chapter 25: LPC32x0 Standard Timers 0/1/2/3/4/5

Chapter 26: LPC32x0 High-speed Timer

Chapter 27: LPC32x0 Millisecond timer

Chapter 28: LPC32x0 Real-time clock (RTC) and battery RAM

Chapter 29: LPC32x0 Watchdog Timer (WDT)

Chapter 30: LPC32x0 Motor Control Pulse Width Modulator (MCPWM) interface

Chapter 31: LPC32x0 Simple Pulse Width Modulator (PWM) interface

Chapter 32: LPC32x0 General purpose inputs and outputs (GPI, GPO, GPIO)

Chapter 33: LPC32x0 Pin multiplexing

Chapter 34: LPC32x0 Pin configuration and packaging

Chapter 35: LPC32x0 Boot process

Chapter 36: LPC32x0 JTAG and Embedded ICE-RT controller

Chapter 37: LPC32x0 Embedded Trace Macrocell (ETM9) and Embedded Trace Buffer (ETB)

Chapter 38: Supplementary information

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 22 July 2011
Document identifier: UM10326