UM10237

LPC2468 User manual

Rev. 01 — 18 December 2006 **User manual**

Document information

Info	Content
Keywords	LPC2400, LPC2468, ARM, ARM7, 32-bit, Single-chip, External memory interface, USB 2.0, Device, Host, OTG, Ethernet, CAN, I2S, I2C, SPI, UART, PWM, IRC, Microcontroller
Abstract	Initial LPC2468 user manual release

Revision history

Rev	Date	Description
01	20061218	Initial LPC2468 user manual release

Contact information

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

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

UM10237

Chapter 1: Introductory information

Rev. 01 — 18 December 2006 **User manual**

1. Introduction

NXP Semiconductor designed the LPC2468 microcontroller around a 16-bit/32-bit ARM7TDMI-S CPU core with real-time debug interfaces that include both JTAG and embedded Trace. The LPC2468 microcontroller has 512 kB of on-chip high-speed Flash memory. This Flash memory includes a special 128-bit wide memory interface and accelerator architecture that enables the CPU to execute sequential instructions from Flash memory at the maximum 72 MHz system clock rate. This feature is available only on the LPC2000 ARM Microcontroller family of products. The LPC2468 can execute both 32-bit ARM and 16-bit Thumb instructions. Support for the two Instruction Sets means Engineers can choose to optimize their application for either performance or code size at the sub-routine level. When the core executes instructions in Thumb state it can reduce code size by more than 30 % with only a small loss in performance while executing instructions in ARM state maximizes core performance.

The LPC2468 microcontroller is ideal for multi-purpose communication applications. It incorporates a 10/100 Ethernet Media Access Controller (MAC), a USB full speed device/host/OTG controller with 4 kB of endpoint RAM, four UARTs, two Controller Area Network (CAN) channels, an SPI interface, two Synchronous Serial Ports (SSP), three I^2C interfaces, and an I^2S interface. Supporting this collection of serial communications interfaces are the following feature components; an on-chip 4 MHz internal precision oscillator, 98 kB of total RAM consisting of 64 kB of local SRAM, 16 kB SRAM for Ethernet, 16 kB SRAM for general purpose DMA, 2 kB of battery powered SRAM, and an External Memory Controller (EMC). These features make this device optimally suited for communication gateways and protocol converters. Complementing the many serial communication controllers, versatile clocking capabilities, and memory features are various 32-bit timers, an improved 10-bit ADC, 10-bit DAC, two PWM units, four external interrupt pins, and up to 160 fast GPIO lines. The LPC2468 connects 64 of the GPIO pins to the hardware based Vector Interrupt Controller (VIC) that means these external inputs can generate edge-triggered, interrupts. All of these features make the LPC2468 particularly suitable for industrial control and medical systems.

Important: The term “LPC2400“ in the following text will be used as a generic name for LPC2468.

2. Features

- ARM7TDMI-S processor, running at up to 72 MHz.
- 512 kB on-chip Flash program memory with In-System Programming (ISP) and In-Application Programming (IAP) capabilities. Flash program memory is on the ARM local bus for high performance CPU access.
- 98 kB on-chip SRAM includes:
  - 64 kB of SRAM on the ARM local bus for high performance CPU access.
  - 16 kB SRAM for Ethernet interface. Can also be used as general purpose SRAM.
  - 16 kB SRAM for general purpose DMA use also accessible by the USB.

  - 2 kB SRAM data storage powered from the RTC power domain
- Dual Advanced High-performance Bus (AHB) system allows simultaneous Ethernet DMA, USB DMA, and program execution from on-chip Flash with no contention.
- EMC provides support for asynchronous static memory devices such as RAM, ROM and Flash, as well as dynamic memories such as Single Data Rate SDRAM.
- Advanced Vectored Interrupt Controller (VIC), supporting up to 32 vectored interrupts.
- General Purpose AHB DMA controller (GPDMA) that can be used with the SSP, I^2S, and SD/MM interface as well as for memory-to-memory transfers.
- Serial Interfaces:
  - Ethernet MAC with MII/RMII interface and associated DMA controller. These functions reside on an independent AHB bus.
  - USB 2.0 full-speed dual port device/host/OTG controller with on-chip PHY and associated DMA controller.
  - Four UARTs with fractional baud rate generation, one with modem control I/O, one with IrDA support, all with FIFO.
  - CAN controller with two channels.
  - SPI controller.
  - Two SSP controllers, with FIFO and multi-protocol capabilities. One is an alternate for the SPI port, sharing its interrupt. SSPs can be used with the GPDMA controller.
  - Three I^2C-bus interfaces (one with open-drain and two with standard port pins).
  - I^2S (Inter-IC Sound) interface for digital audio input or output. It can be used with the GPDMA.
- Other peripherals:
  - SD/MMC memory card interface.
  - 160 General purpose I/O pins with configurable pull-up/down resistors.
  - 10-bit ADC with input multiplexing among 8 pins.
  - 10-bit DAC.
  - Four general purpose timers/counters with 8 capture inputs and 10 compare outputs. Each timer block has an external count input.
  - Two PWM/timer blocks with support for three-phase motor control. Each PWM has an external count inputs.
  - Real-Time Clock (RTC) with separate power domain, clock source can be the RTC oscillator or the APB clock.
  - 2 kB SRAM powered from the RTC power pin, allowing data to be stored when the rest of the chip is powered off.
  - WatchDog Timer (WDT). The WDT can be clocked from the internal RC oscillator, the RTC oscillator, or the APB clock.
- Standard ARM test/debug interface for compatibility with existing tools.
- Emulation trace module supports real-time trace.
- Single 3.3 V power supply (3.0 V to 3.6 V).
- Four reduced power modes: idle, sleep, power-down, and deep power-down.

- Four external interrupt inputs configurable as edge/level sensitive. All pins on PORT0 and PORT2 can be used as edge sensitive interrupt sources.
- Processor wake-up from Power-down mode via any interrupt able to operate during Power-down mode (includes external interrupts, RTC interrupt, USB activity, Ethernet wake-up interrupt, CAN bus activity, PORT0/2 pin interrupt).
- Two independent power domains allow fine tuning of power consumption based on needed features.
- Each peripheral has its own clock divider for further power saving. These dividers help reducing active power by 20 - 30 %.
- Brownout detect with separate thresholds for interrupt and forced reset.
- On-chip power-on reset.
- On-chip crystal oscillator with an operating range of 1 MHz to 24 MHz.
- 4 MHz internal RC oscillator trimmed to 1 % accuracy that can optionally be used as the system clock. When used as the CPU clock, does not allow CAN and USB to run.
- On-chip PLL allows CPU operation up to the maximum CPU rate without the need for a high frequency crystal. May be run from the main oscillator, the internal RC oscillator, or the RTC oscillator.
- Boundary scan for simplified board testing.
- Versatile pin function selections allow more possibilities for using on-chip peripheral functions.

3. Applications

- Industrial control
- Medical systems
- Protocol converter
- Communications

4. Ordering options

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC2468FBD208	LQFP208	plastic low profile quad flat package; 208 leads; body 28 × 28 × 1.4 mm	SOT459-1
LPC2468FET208	TFBGA208	plastic thin fine-pitch ball grid array package; 208 balls; body 15 x 15 x 0.7 mm	SOT950-1

Table 2. Ordering options

Type number	Flash (kB)	SRAM (kB)					External bus	Ethernet	USB OTG/ OHC/ DEV + 4 kB FIFO	CAN channels	SD/ MMC	GP DMA	ADC channels	DAC channels	Temp range
		Local bus	Ethernet buffer	GP/USB	RTC	Total									
LPC2468FBD208	512	64	16	16	2	98	Full 32-bit	MII/ RMII	yes	2	yes	yes	8	1	–40 °C to +85 °C
LPC2468FET208	512	64	16	16	2	98	Full 32-bit	MII/ RMII	yes	2	yes	yes	8	1	–40 °C to +85 °C

5. Architectural overview

The LPC2468 microcontroller consists of an ARM7TDMI-S CPU with emulation support, the ARM7 local bus for closely coupled, high speed access to the majority of on-chip memory, the AMBA AHB interfacing to high speed on-chip peripherals and external memory, and the AMBA APB for connection to other on-chip peripheral functions. The microcontroller permanently configures the ARM7TDMI-S processor for little-endian byte order.

The LPC2468 implements two AHB buses in order to allow the Ethernet block to operate without interference caused by other system activity. The primary AHB, referred to as AHB1, includes the VIC, GPDMA controller, and EMC.

The second AHB, referred to as AHB2, includes only the Ethernet block and an associated 16 kB SRAM. In addition, a bus bridge is provided that allows the secondary AHB to be a bus master on AHB1, allowing expansion of Ethernet buffer space into off-chip memory or unused space in memory residing on AHB1.

In summary, bus masters with access to AHB1 are the ARM7 itself, the GPDMA function, and the Ethernet block (via the bus bridge from AHB2). Bus masters with access to AHB2 are the ARM7 and the Ethernet block.

AHB peripherals are allocated a 2 MB range of addresses at the very top of the 4 GB ARM memory space. Each AHB peripheral is allocated a 16 kB address space within the AHB address space. Lower speed peripheral functions are connected to the APB bus. The AHB to APB bridge interfaces the APB bus to the AHB bus. APB peripherals are also allocated a 2 MB range of addresses, beginning at the 3.5 GB address point. Each APB peripheral is allocated a 16 kB address space within the APB address space.

The ARM7TDMI-S processor is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler than those of microprogrammed complex instruction set computers. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor core.

Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The ARM7TDMI-S processor also employs a unique architectural strategy known as Thumb, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The key idea behind Thumb is that of a super-reduced instruction set. Essentially, the ARM7TDMI-S processor has two instruction sets:

- the standard 32-bit ARM set
- a 16-bit Thumb set

The Thumb set's 16-bit instruction length allows it to approach higher density compared to standard ARM code while retaining most of the ARM's performance.

6. On-chip Flash programming memory

The LPC2468 incorporates 512 kB Flash memory system. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways. It may be programmed In System via the serial port (UART0). The application program may also erase and/or program the Flash while the application is running, allowing a great degree of flexibility for data storage field and firmware upgrades.

The Flash memory is 128 bits wide and includes pre-fetching and buffering techniques to allow it to operate at speeds of 72 MHz.

The LPC2468 provides a minimum of 100000 write/erase cycles and 20 years of data retention.

7. On-chip SRAM

The LPC2468 includes a SRAM memory of 64 kB reserved for the ARM processor exclusive use. This RAM may be used for code and/or data storage and may be accessed as 8 bits, 16 bits, and 32 bits.

A 16 kB SRAM block serving as a buffer for the Ethernet controller and a 16 kB SRAM associated with the second AHB bus can be used both for data and code storage, too. Remaining SRAM such as a 4 kB USB FIFO and a 2 kB RTC SRAM can be used for data storage only. The RTC SRAM is battery powered and retains the content in the absence of the main power supply.

8. Block diagram

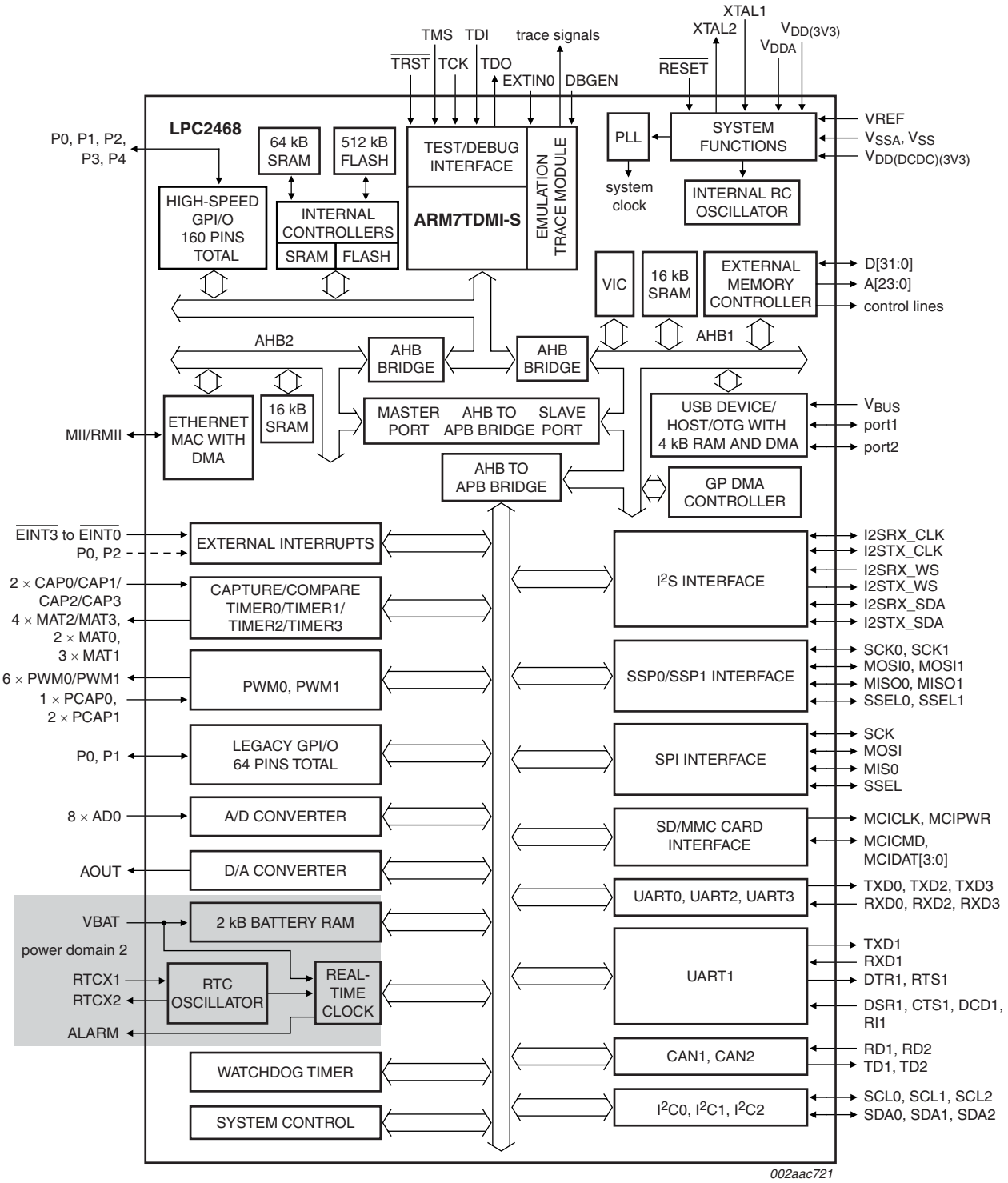


Fig 1. LPC2468 block diagram

1. Memory map and peripheral addressing

ARM processors have a single 4 GB address space. The following table shows how this space is used on NXP embedded ARM devices.

Table 3. LPC2468 memory usage and details

Address range	General use	Address range details and description	
0x0000 0000 to 0x3FFF FFFF	On-chip non-volatile memory and Fast I/O	0x0000 0000 - 0x0007 FFFF	Flash Memory (512 kB)
		0x3FFF C000 - 0x3FFF FFFF	Fast GPIO registers
0x4000 0000 to 0x7FFF FFFF	On-chip RAM	0x4000 0000 - 0x4000 FFFF	RAM (64 kB)
		0x7FE0 0000 - 0x7FE0 3FFF	Ethernet RAM (16 kB)
		0x7FD0 0000 - 0x7FD0 3FFF	USB RAM (16 kB)
0x8000 0000 to 0xDFFF FFFF	Off-Chip Memory	Four static memory banks, 16 MB each	
		0x8000 0000 - 0x80FF FFFF	Static memory bank 0
		0x8100 0000 - 0x81FF FFFF	Static memory bank 1
		0x8200 0000 - 0x82FF FFFF	Static memory bank 2
		0x8300 0000 - 0x83FF FFFF	Static memory bank 3
		Four dynamic memory banks, 256 MB each	
		0xA000 0000 - 0xAFFF FFFF	Dynamic memory bank 0
		0xB000 0000 - 0xBFFF FFFF	Dynamic memory bank 1
		0xC000 0000 - 0xCFFF FFFF	Dynamic memory bank 2
		0xD000 0000 - 0xDFFF FFFF	Dynamic memory bank 3
0xE000 0000 to 0xEFFF FFFF	APB Peripherals	36 peripheral blocks, 16 kB each	
0xF000 0000 to 0xFFFF FFFF	AHB peripherals		

2. Memory maps

The LPC2400 incorporates several distinct memory regions, shown in the following figures. Figure 2–2 shows the overall map of the entire address space from the user program viewpoint following reset. The interrupt vector area supports address remapping, which is described later in this section.

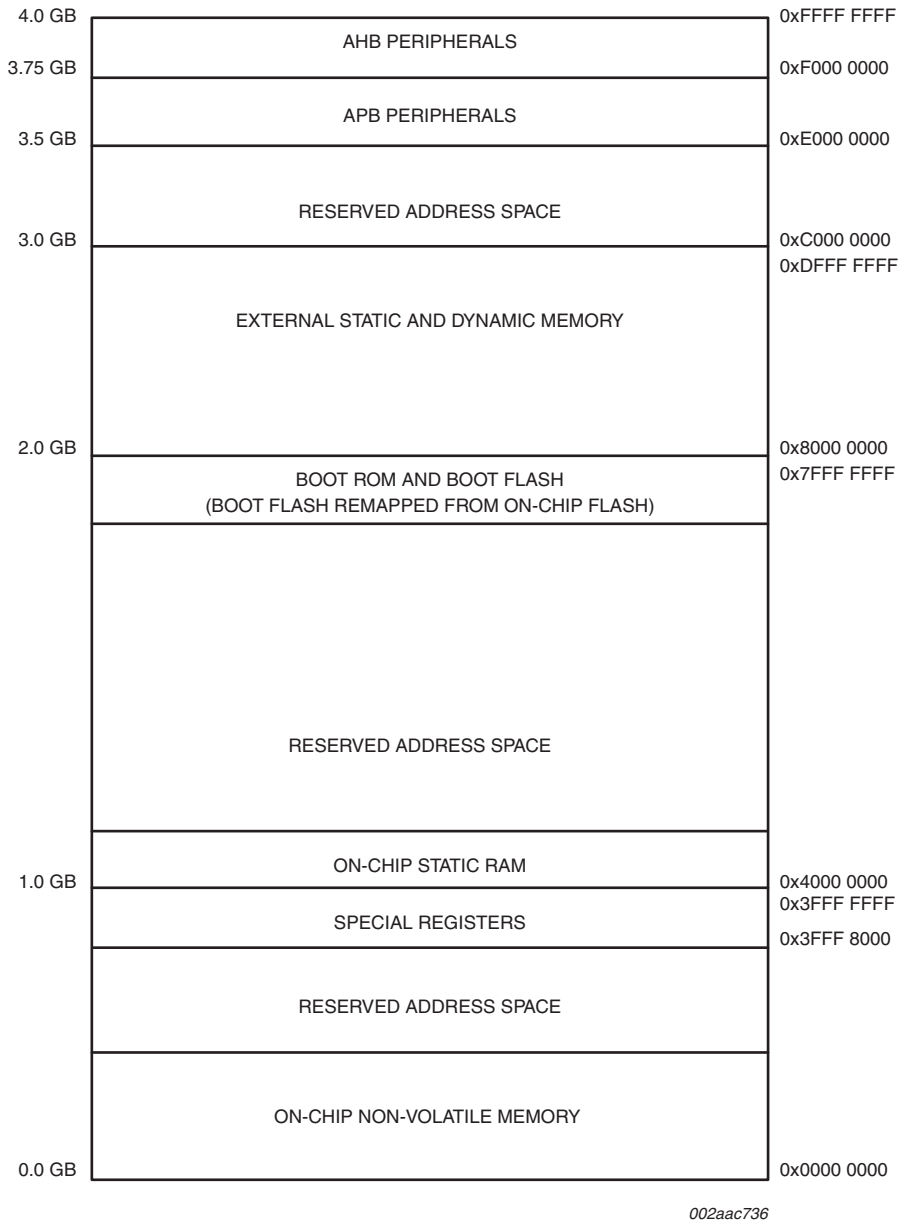


Fig 2. LPC2468 system memory map

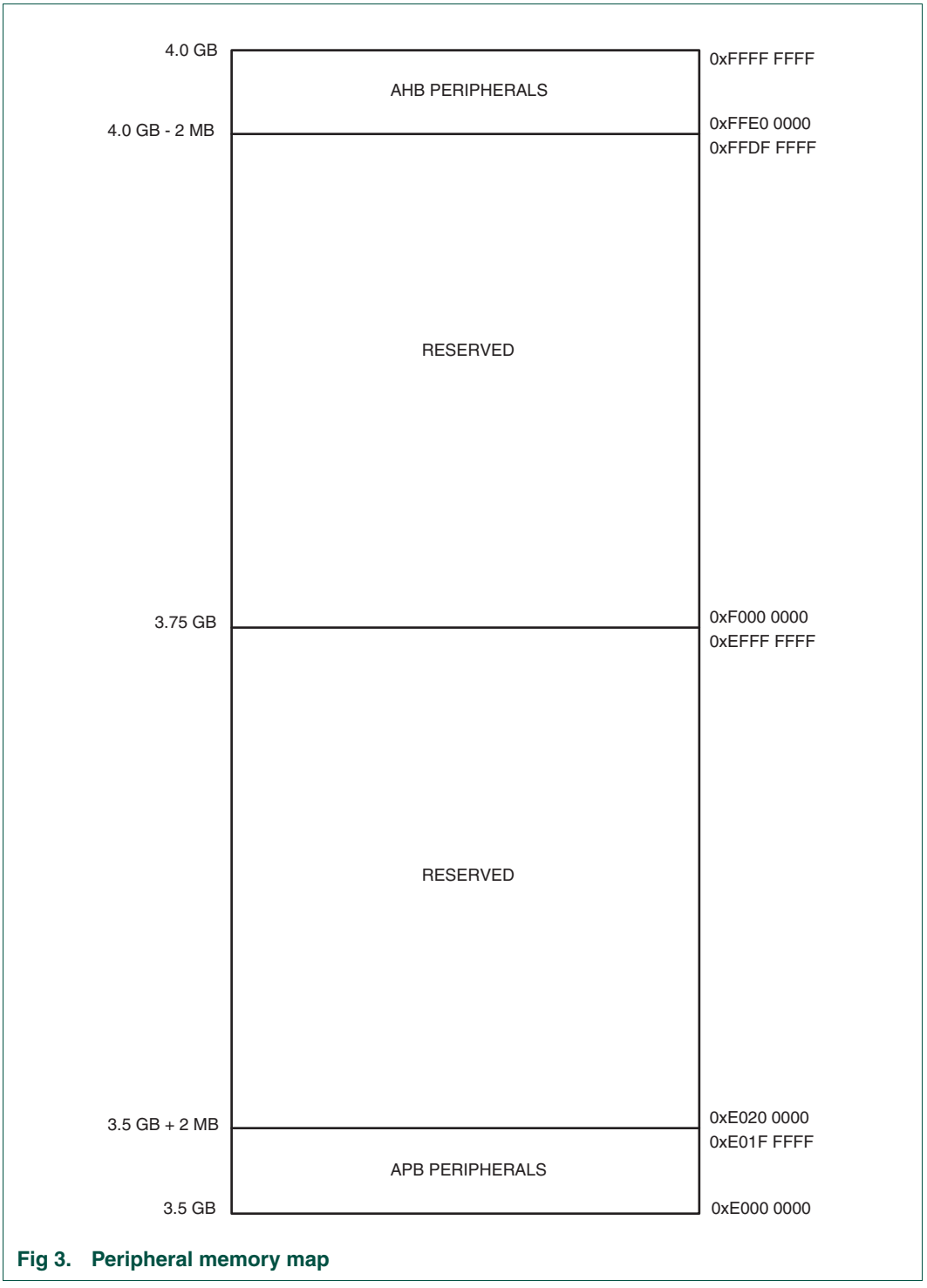


Fig 3. Peripheral memory map

Figure 4 and Table 2–4 show different views of the peripheral address space. Both the AHB and APB peripheral areas are 2 megabyte spaces which are divided up into 128 peripherals. Each peripheral space is 16 kilobytes in size. This allows simplifying the address decoding for each peripheral.

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

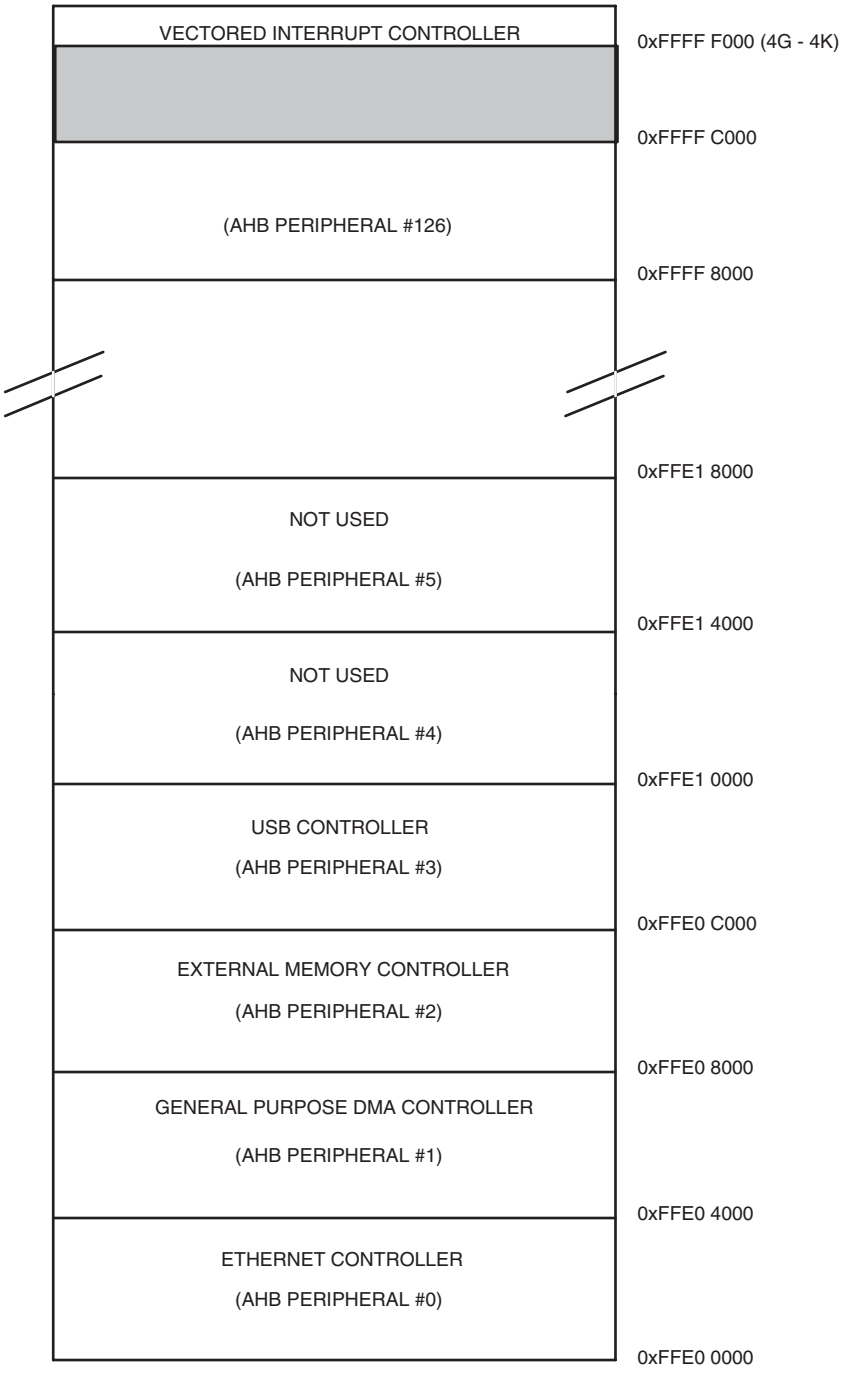


Fig 4. AHB peripheral map

3. APB peripheral addresses

The following table shows the APB address map. No APB peripheral uses all of the 16 kB space allocated to it. Typically each device's registers are "aliased" or repeated at multiple locations within each 16 kB range.

Table 4. APB peripherals and base addresses

APB Peripheral	Base Address	Peripheral Name
0	0xE000 0000	Watchdog Timer
1	0xE000 4000	Timer 0
2	0xE000 8000	Timer 1
3	0xE000 C000	UART0
4	0xE001 0000	UART1
5	0xE001 4000	PWM0
6	0xE001 8000	PWM1
7	0xE001 C000	I^2C0
8	0xE002 0000	SPI
9	0xE002 4000	RTC
10	0xE002 8000	GPIO
11	0xE002 C000	Pin Connect Block
12	0xE003 0000	SSP1
13	0xE003 4000	ADC
14	0xE003 8000	CAN Acceptance Filter RAM
15	0xE003 C000	CAN Acceptance Filter Registers
16	0xE004 0000	CAN Common Registers
17	0xE004 4000	CAN Controller 1
18	0xE004 8000	CAN Controller 2
19 to 22	0xE004 C000 to 0xE005 8000	Not used
23	0xE005 C000	I^2C1
24	0xE006 0000	Not used
25	0xE006 4000	Not used
26	0xE006 8000	SSP0
27	0xE006 C000	DAC
28	0xE007 0000	Timer 2
29	0xE007 4000	Timer 3
30	0xE007 8000	UART2
31	0xE007 C000	UART3
32	0xE008 0000	I^2C2
33	0xE008 4000	Battery RAM
34	0xE008 8000	I^2S
35	0xE008 C000	SD/MMC Card Interface
36 to 126	0xE009 0000 to 0xE01F BFFF	Not used
127	0xE01F C000	System Control Block

4. LPC2400 memory re-mapping and boot ROM

4.1 Memory map concepts and operating modes

The basic concept on the LPC2400 is that each memory area has a "natural" location in the memory map. This is the address range for which code residing in that area is written. The bulk of each memory space remains permanently fixed in the same location, eliminating the need to have portions of the code designed to run in different address ranges.

Because of the location of the interrupt vectors on the ARM7 processor (at addresses 0x0000 0000 through 0x0000 001C, as shown in Table 2–5 below), a small portion of the Boot ROM and SRAM spaces need to be re-mapped in order to allow alternative uses of interrupts in the different operating modes described in Table 2–6. Re-mapping of the interrupts is accomplished via the Memory Mapping Control feature (Section 2–5 “Memory mapping control” on page 15).

Table 5. ARM exception vector locations

Address	Exception
0x0000 0000	Reset
0x0000 0004	Undefined Instruction
0x0000 0008	Software Interrupt
0x0000 000C	Prefetch Abort (instruction fetch memory fault)
0x0000 0010	Data Abort (data access memory fault)
0x0000 0014	Reserved **Note:** Identified as reserved in ARM documentation, this location is used by the Boot Loader as the Valid User Program key. This is described in detail in "Flash Memory System and Programming" chapter on page 561.
0x0000 0018	IRQ
0x0000 001C	FIQ

Table 6. LPC2400 Memory mapping modes

Mode	Activation	Usage
Boot Loader mode	Hardware activation by any Reset	The Boot Loader **always** executes after any reset. The Boot ROM interrupt vectors are mapped to the bottom of memory to allow handling exceptions and using interrupts during the Boot Loading process. A sector of the flash memory (the Boot flash) is available to hold part of the Boot Code.
User Flash mode	Software activation by Boot code	Activated by the Boot Loader when a valid User Program Signature is recognized in memory and Boot Loader operation is not forced. Interrupt vectors are not re-mapped and are found in the bottom of the flash memory.
User RAM mode	Software activation by User program	Activated by a User Program as desired. Interrupt vectors are re-mapped to the bottom of the Static RAM.

4.2 Memory re-mapping

In order to allow for compatibility with future derivatives, the entire Boot ROM is mapped to the top of the on-chip memory space. In this manner, the use of larger or smaller flash modules will not require changing the location of the Boot ROM (which would require changing the Boot Loader code itself) or changing the mapping of the Boot ROM interrupt vectors. Memory spaces other than the interrupt vectors remain in fixed locations. Figure 2–5 shows the on-chip memory mapping in the modes defined above.

The portion of memory that is re-mapped to allow interrupt processing in different modes includes the interrupt vector area (32 bytes) and an additional 32 bytes for a total of 64 bytes, that facilitates branching to interrupt handlers at distant physical addresses. The remapped code locations overlay addresses 0x0000 0000 through 0x0000 003F. A typical user program in the flash memory can place the entire FIQ handler at address 0x0000 001C without any need to consider memory boundaries. The vector contained in the SRAM, external memory, and Boot ROM must contain branches to the actual interrupt handlers, or to other instructions that accomplish the branch to the interrupt handlers.

There are three reasons this configuration was chosen:

1. To give the FIQ handler in the flash memory the advantage of not having to take a memory boundary caused by the remapping into account.
2. Minimize the need to for the SRAM and Boot ROM vectors to deal with arbitrary boundaries in the middle of code space.
3. To provide space to store constants for jumping beyond the range of single word branch instructions.

Re-mapped memory areas, including the Boot ROM and interrupt vectors, continue to appear in their original location in addition to the re-mapped address.

Details on re-mapping and examples can be found in Section 2–5 “Memory mapping control” on page 15.

5. Memory mapping control

The Memory Mapping Control alters the mapping of the interrupt vectors that appear beginning at address 0x0000 0000. This allows code running in different memory spaces to have control of the interrupts.

5.1 Memory Mapping Control Register (MEMMAP - 0xE01F C040)

Whenever an exception handling is necessary, microcontroller will fetch an instruction residing on exception corresponding address as described in Table 2–5 “ARM exception vector locations” on page 14. The MEMMAP register determines the source of data that will fill this table.

Table 7. Memory mapping control registers

Name	Description	Access	Reset value	Address
MEMMAP	Memory mapping control. Selects whether the ARM interrupt vectors are read from the Boot ROM, User Flash, or RAM.	R/W	0x00	0xE01F C040

Table 8. Memory Mapping control register (MEMMAP - address 0xE01F C040) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MAP	00	Boot Loader Mode. Interrupt vectors are re-mapped to Boot ROM.	00
		01	User Flash Mode. Interrupt vectors are not re-mapped and reside in Flash.	
		10	User RAM Mode. Interrupt vectors are re-mapped to Static RAM.	
		11	User External Memory Mode.	
			Warning: Improper setting of this value may result in incorrect operation of the device.	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.2 Memory mapping control usage notes

Memory Mapping Control simply selects one out of three available sources of data (sets of 64 bytes each) necessary for handling ARM exceptions (interrupts).

For example, whenever a Software Interrupt request is generated, ARM core will always fetch 32 bit data "residing" on 0x0000 0008 see Table 2–5 “ARM exception vector locations” on page 14. This means that when MEMMAP[1:0] = 10 (User RAM Mode), read/fetch from 0x0000 0008 will provide data stored in 0x4000 0008. In case of MEMMAP[1:0] = 00 (Boot Loader Mode), read/fetch from 0x0000 0008 will provide data available also at 0x7FFF E008 (Boot ROM remapped from on-chip Bootloader).

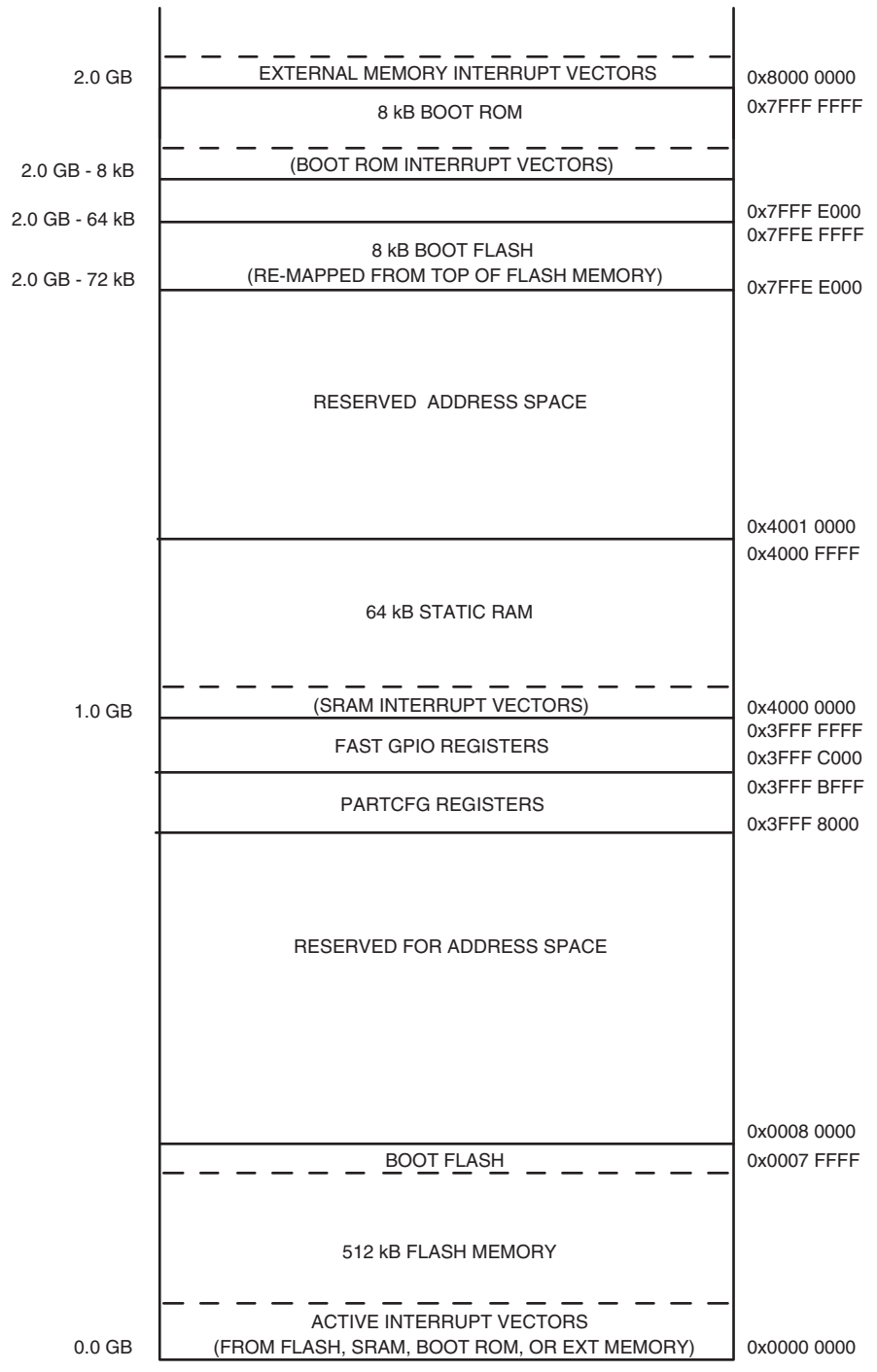


Fig 5. Map of lower memory is showing re-mapped and re-mappable areas

6. Prefetch abort and data abort exceptions

The LPC2400 generates the appropriate bus cycle abort exception if an access is attempted for an address that is in a reserved or unassigned address region. The regions are:

- Areas of the memory map that are not implemented for a specific ARM derivative. For the LPC2400, these are:
  - Address space between On-Chip Non-Volatile Memory and the Special Register space. Labelled "Reserved for On-Chip Memory" in Figure 2–2.
  - Address space between On-Chip Static RAM and the Boot ROM. Labelled "Reserved Address Space" in Figure 2–2.
  - External Memory
  - Reserved regions of the AHB and APB spaces. See Figure 2–3.
- Unassigned AHB peripheral spaces. See Figure 2–4.
- Unassigned APB peripheral spaces. See Table 2–4.

For these areas, both attempted data access and instruction fetch generate an exception. In addition, a Prefetch Abort exception is generated for any instruction fetch that maps to an AHB or APB peripheral address, or to the Special Register space located just below the SRAM at addresses 0x3FFF8000 through 0x3FFFFFFF.

Within the address space of an existing APB peripheral, a data abort exception is not generated in response to an access to an undefined address. Address decoding within each peripheral is limited to that needed to distinguish defined registers within the peripheral itself. For example, an access to address 0xE000 D000 (an undefined address within the UART0 space) may result in an access to the register defined at address 0xE000 C000. Details of such address aliasing within a peripheral space are not defined in the LPC2400 documentation and are not a supported feature.

If software executes a write directly to the flash memory, the MAM generates a data abort exception. Flash programming must be accomplished by using the specified flash programming interface provided by the Boot Code.

Note that the ARM core stores the Prefetch Abort flag along with the associated instruction (which will be meaningless) in the pipeline and processes the abort only if an attempt is made to execute the instruction fetched from the illegal address. This prevents accidental aborts that could be caused by prefetches that occur when code is executed very near a memory boundary.

1. Summary of system control block functions

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

- Reset
- Brown-Out Detection
- External Interrupt Inputs
- Miscellaneous System Controls and Status
- Code Security vs. Debugging

Each type of function has its own register(s) if any are required and unneeded bits are defined as reserved in order to allow future expansion. Unrelated functions never share the same register addresses

2. Pin description

Table 3–9 shows pins that are associated with System Control block functions.

Table 9. Pin summary

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

3. Register description

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

Table 10. Summary of system control registers

Name	Description	Access	Reset value	Address
External Interrupts				
EXTINT	External Interrupt Flag Register	R/W	0x00	0xE01F C140
EXTMODE	External Interrupt Mode register	R/W	0x00	0xE01F C148
EXTPOLAR	External Interrupt Polarity Register	R/W	0x00	0xE01F C14C

Table 10. Summary of system control registers

Name	Description	Access	Reset value	Address
Reset				
RSID	Reset Source Identification Register	R/W	see text	0xE01F C180
Chip Security				
CSPR	Code Security Protection Register	WO	0x00	0xE01F C184
Syscon Miscellaneous Registers				
SCS	System Control and Status	R/W	0x00	0xE01F C1A0

4. Reset

Reset has four sources on the LPC2400: the $\overline{\text{RESET}}$ pin, the Watchdog Reset, Power On Reset (POR) and the Brown Out Detection circuit (BOD). The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin. Assertion of chip Reset by any source, once the operating voltage attains a usable level, starts the Wakeup Timer (see description in Section 4–8 “Wakeup timer” in this chapter), causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, a fixed number of clocks have passed, and the Flash controller has completed its initialization. The relationship between Reset, the oscillator, and the Wakeup Timer are shown in Figure 3–6.

On the assertion of any of reset sources (POR, BOD reset, External reset and Watchdog reset), the following two sequences start simultaneously:

1. After IRC-start-up time (maximum of 60 μs on power-up), IRC provides stable clock output, the reset signal is latched and synchronized on the IRC clock. The 2-bit IRC wakeup timer starts counting when the synchronized reset is de-asserted. The boot code in the ROM starts when the 2-bit IRC wakeup timer times out. The boot code performs the boot tasks and may jump to the Flash. If the Flash is not ready to access, the MAM will insert wait cycles until the Flash is ready.
2. After IRC-start-up time (maximum of 60 μs on power-up), IRC provides stable clock output, the reset signal is synchronized on the IRC clock. The Flash wakeup-timer (9-bit) starts counting when the synchronized reset is de-asserted. The Flash wakeup-timer generates the 100 μs Flash start-up time. Once it times out, the Flash initialization sequence is started, which takes about 250 cycles. When it’s done, the MAM will be granted access to the Flash.

When the internal Reset is removed, the processor begins executing at address 0, which is initially the Reset vector mapped from the Boot Block. At that point, all of the processor and peripheral registers have been initialized to predetermined values.

The various Resets have some small differences. For example, a Power On Reset causes the value of certain pins to be latched to configure the part.

For more details on Reset, PLL and startup/boot code interaction see Section 4–5.2 “PLL and startup/boot code interaction”.

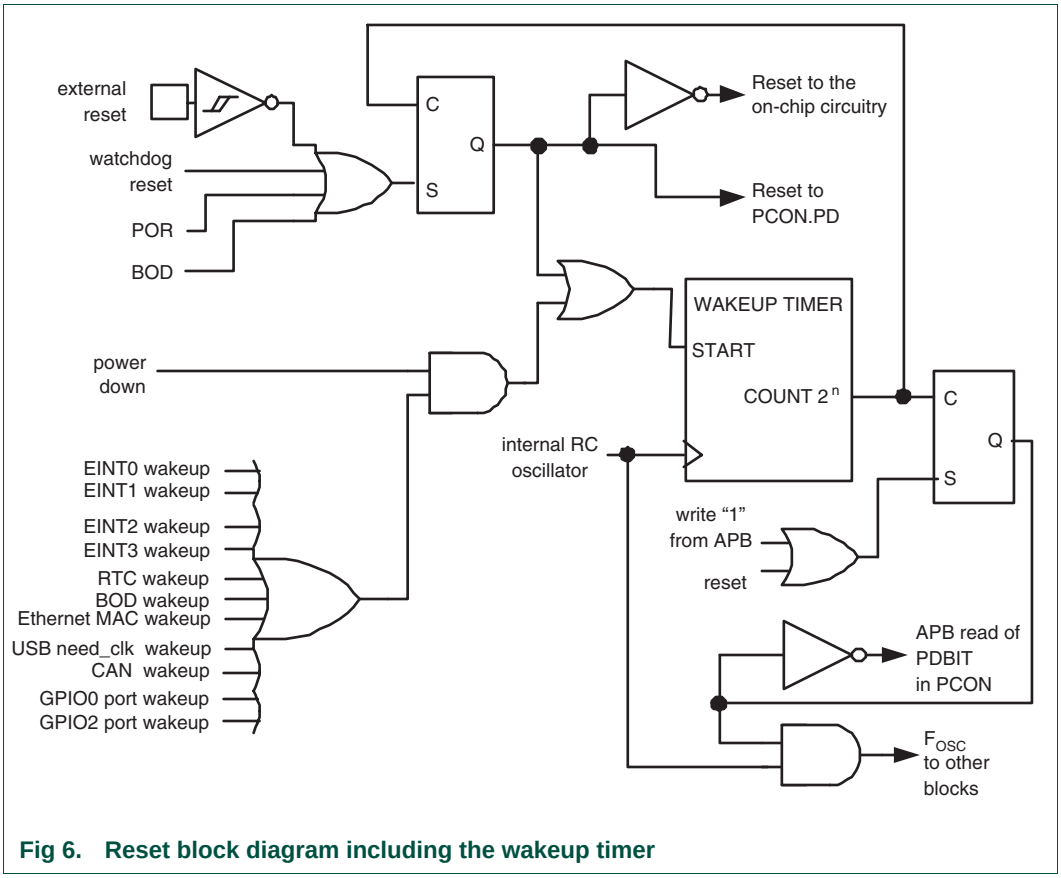


Fig 6. Reset block diagram including the wakeup timer

4.1 Reset Source Identification Register (RSIR - 0xE01F C180)

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

Table 11. Reset Source Identification register (RSID - address 0xE01F C180) bit description

Bit	Symbol	Description	Reset value
0	POR	Assertion of the POR signal sets this bit, and clears all of the other bits in this register. But if another Reset signal (e.g., External Reset) remains asserted after the POR signal is negated, then its bit is set. This bit is not affected by any of the other sources of Reset.	See text
1	EXTR	Assertion of the $\overline{\text{RESET}}$ signal sets this bit. This bit is cleared by POR, but is not affected by WDT or BOD reset.	See text

Table 11. Reset Source Identification register (RSID - address 0xE01F C180) bit description

Bit	Symbol	Description	Reset value
2	WDTR	This bit is set when the Watchdog Timer times out and the WDTRESET bit in the Watchdog Mode Register is 1. It is cleared by any of the other sources of Reset.	See text
3	BODR	This bit is set when the 3.3 V power reaches a level below 2.6 V. If the V_{DD} voltage dips from 3.3 V to 2.5 V and backs up, the BODR bit will be set to 1. If the $V_{DD(3V3)}$ voltage dips from 3.3 V to 2.5 V and continues to decline to the level at which POR is asserted (nominally 1 V), the BODR bit is cleared. if the $V_{DD(3V3)}$ voltage rises continuously from below 1 V to a level above 2.6 V, the BODR will be set to 1. This bit is not affected by External Reset nor Watchdog Reset. **Note:** Only in case when a reset occurs and the POR = 0, the BODR bit indicates if the $V_{DD(3V3)}$ voltage was below 2.6 V or not.	See text
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5. Brown-out detection

The LPC2400 includes 2-stage monitoring of the voltage on the $V_{DD(3V3)}$ pins. If this voltage falls below 2.95 V, the Brown-Out Detector (BOD) asserts an interrupt signal to the Vectored Interrupt Controller. This signal can be enabled for interrupt in the Interrupt Enable Register in the VIC (see Section 7–4.4 “Interrupt Enable Register (VICIntEnable - 0xFFFF F010)”) in order to cause a CPU interrupt; if not, software can monitor the signal by reading the Raw Interrupt Status Register (see Section 7–4.3 “Raw Interrupt Status Register (VICRawIntr - 0xFFFF F008)”).

The second stage of low-voltage detection asserts Reset to inactivate the LPC2400 when the voltage on the $V_{DD(3V3)}$ pins falls below 2.65 V. This Reset prevents alteration of the Flash as operation of the various elements of the chip would otherwise become unreliable due to low voltage. The BOD circuit maintains this reset down below 1 V, at which point the Power-On Reset circuitry maintains the overall Reset.

Both the 2.95 V and 2.65 V thresholds include some hysteresis. In normal operation, this hysteresis allows the 2.95 V detection to reliably interrupt, or a regularly-executed event loop to sense the condition.

But when Brown-Out Detection is enabled to bring the LPC2400 out of Power-Down mode (which is itself not a guaranteed operation -- see Section 4–7.7 “Power Mode Control register (PCON - 0xE01F C0C0)”), the supply voltage may recover from a transient before the Wakeup Timer has completed its delay. In this case, the net result of the transient BOD is that the part wakes up and continues operation after the instructions that set Power-Down Mode, without any interrupt occurring and with the BOD bit in the RSID being 0. Since all other wakeup conditions have latching flags (see Section 3–6.2 “External Interrupt flag register (EXTINT - 0xE01F C140)” and Section 28–4.2), a wakeup of this type, without any apparent cause, can be assumed to be a Brown-Out that has gone away.

6. External interrupt inputs

The LPC2400 includes four External Interrupt Inputs as selectable pin functions. In addition, external interrupts have the ability to wake up the CPU from Power down mode. This is controlled by the register INTWAKE, which is described in the Clocking and Power Control chapter under the Power Control heading

6.1 Register description

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

Table 12. External Interrupt registers

Name	Description	Access	Reset value[1]	Address
EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, EINT2 and EINT3. See Table 3–13.	R/W	0x00	0xE01F C140
EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level-sensitive. See Table 3–14.	R/W	0x00	0xE01F C148
EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt. See Table 3–15.	R/W	0x00	0xE01F C14C

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

6.2 External Interrupt flag register (EXTINT - 0xE01F C140)

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

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

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

Important: whenever a change of external interrupt operating mode (i.e. active level/edge) is performed (including the initialization of an external interrupt), corresponding bit in the EXTINT register must be cleared! For details see Section 3–6.3 “External Interrupt Mode register (EXTMODE - 0xE01F C148)” **and** Section 3–6.4 “External Interrupt Polarity register (EXTPOLAR - 0xE01F C14C)”.

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

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

Table 13. External Interrupt Flag register (EXTINT - address 0xE01F C140) bit description

Bit	Symbol	Description	Reset value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

6.3 External Interrupt Mode register (EXTMODE - 0xE01F C148)

The bits in this register select whether each EINT pin is level- or edge-sensitive. Only pins that are selected for the EINT function (see chapter Pin Connect Block on page 119) and enabled in the VICIntEnable register (Section 7–4.4 “Interrupt Enable Register (VICIntEnable - 0xFFFF F010)”) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

Table 14. External Interrupt Mode register (EXTMODE - address 0xE01F C148) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0	0	Level-sensitivity is selected for $\overline{\text{EINT0}}$.	0
		1	$\overline{\text{EINT0}}$ is edge sensitive.	
1	EXTMODE1	0	Level-sensitivity is selected for $\overline{\text{EINT1}}$.	0
		1	$\overline{\text{EINT1}}$ is edge sensitive.	
2	EXTMODE2	0	Level-sensitivity is selected for $\overline{\text{EINT2}}$.	0
		1	$\overline{\text{EINT2}}$ is edge sensitive.	
3	EXTMODE3	0	Level-sensitivity is selected for $\overline{\text{EINT3}}$.	0
		1	$\overline{\text{EINT3}}$ is edge sensitive.	
7:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.4 External Interrupt Polarity register (EXTPOLAR - 0xE01F C14C)

In level-sensitive mode, the bits in this register select whether the corresponding pin is high- or low-active. In edge-sensitive mode, they select whether the pin is rising- or falling-edge sensitive. Only pins that are selected for the EINT function (see "Pin Connect Block" chapter on page 119) and enabled in the VICIntEnable register (Section 7–4.4 “Interrupt Enable Register (VICIntEnable - 0xFFFF F010)”) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

Table 15. External Interrupt Polarity register (EXTPOLAR - address 0xE01F C14C) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTPOLAR0	0	$\overline{\text{EINT0}}$ is low-active or falling-edge sensitive (depending on EXTMODE0).	0
		1	$\overline{\text{EINT0}}$ is high-active or rising-edge sensitive (depending on EXTMODE0).	
1	EXTPOLAR1	0	$\overline{\text{EINT1}}$ is low-active or falling-edge sensitive (depending on EXTMODE1).	0
		1	$\overline{\text{EINT1}}$ is high-active or rising-edge sensitive (depending on EXTMODE1).	
2	EXTPOLAR2	0	$\overline{\text{EINT2}}$ is low-active or falling-edge sensitive (depending on EXTMODE2).	0
		1	$\overline{\text{EINT2}}$ is high-active or rising-edge sensitive (depending on EXTMODE2).	

Table 15. External Interrupt Polarity register (EXTPOLAR - address 0xE01F C14C) bit description

Bit	Symbol	Value	Description	Reset value
3	EXTPOLAR3	0	$\overline{\text{EINT3}}$ is low-active or falling-edge sensitive (depending on EXTMODE3).	0
		1	$\overline{\text{EINT3}}$ is high-active or rising-edge sensitive (depending on EXTMODE3).	
7:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7. Other system controls and status flags

Some aspects of controlling LPC2400 operation that do not fit into peripheral or other registers are grouped here.

7.1 System Controls and Status register (SCS - 0xE01F C1A0)

Table 16. System Controls and Status register (SCS - address 0xE01F C1A0) bit description

Bit	Symbol	Value	Description	Access	Reset value
0	GPIOM		GPIO access mode selection.	R/W	0
		0	GPIO ports 0 and 1 are accessed via APB addresses in a fashion compatible with previous LPC2000 devices.		
		1	High speed GPIO is enabled on ports 0 and 1, accessed via addresses in the on-chip memory range. This mode includes the port masking feature described in the GPIO chapter.		
1	EMC Reset Disable[1]		External Memory Controller Reset Disable.	R/W	0
		0	Both EMC resets are asserted when any type of reset event occurs. In this mode, all registers and functions of the EMC are initialized upon any reset condition.		
		1	Many portions of the EMC are only reset by a power-on or brown-out event, in order to allow the EMC to retain its state through a warm reset (external reset or watchdog reset). If the EMC is configured correctly, auto-refresh can be maintained through a warm reset.		
2	-	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	NA
3	MCIPWR Active Level[1]		MCIPWR pin control.	R/W	0
		0	The MCIPWR pin is low.		
		1	The MCIPWR pin is high.		
4	OSCRANGE		Main oscillator range select.	R/W	0
		0	The frequency range of the main oscillator is 1 MHz to 20 MHz.		
		1	The frequency range of the main oscillator is 15 MHz to 24 MHz.		
5	OSCEN		Main oscillator enable.	R/W	0
		0	The main oscillator is disabled.		
		1	The main oscillator is enabled, and will start up if the correct external circuitry is connected to the XTAL1 and XTAL2 pins.		

Table 16. System Controls and Status register (SCS - address 0xE01F C1A0) bit description

Bit	Symbol	Value	Description	Access	Reset value
6	OSCSTAT		Main oscillator status.	RO	0
		0	The main oscillator is not ready to be used as a clock source.		
		1	The main oscillator is ready to be used as a clock source. The main oscillator must be enabled via the OSCEN bit.		
31:7	-	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-	NA

[1] The state of this bit is preserved through a software reset, and only a POR or a BOD event will reset it to its default value.

8. Code security vs. debugging

Applications in development typically need the debugging and tracing facilities in the LPC2400. Later in the life cycle of an application, it may be more important to protect the application code from observation by hostile or competitive eyes. The following feature of the LPC2400 allows an application to control whether it can be debugged or protected from observation.

Details on the way Code Read Protection works can be found in "Flash Memory Programming Hardware" chapter on page 561.

Table 17. Code security register map

Name	Description	Access	Reset value	Address
CPSR	Controls whether debugging features are enabled.	R/W	0	0xE01F C184

8.1 Code Security Protection Register (CSPR - 0xE01F C184)

Table 18. Code Security Protection Register (CSPR - address 0xE01F C184) bit description

Bit	Symbol	Description	Reset value
31:0	CPSR	If the value 0x8765 4321 is written to this register within the first 256 clocks of execution of the Boot Code, the debugging features of the LPC2400 will be disabled, making the application code in Flash memory secure from observation. If that value is not written to this register within that time period, debugging features will be enabled. If the Boot Code detects a valid checksum in Flash (see the Boot Code description in the Flash Memory Programming Firmware chapter), it will access the contents of the word at Flash address 0x0000 01FC and write that value to this register. Thus, if the application contains the value 0x8765 4321 at address 0x0000 01FC, debugging will be disabled and thus the code in Flash will be protected from observation.	0

1. Summary of clocking and power control functions

This section describes the generation of the various clocks needed by the LPC2400 and options of clock source selection, as well as power control and wakeup from reduced power modes. Functions described in the following subsections include:

- Oscillators
- Clock Source Selection
- PLL
- Clock Dividers
- APB Divider
- Power Control
- Wakeup Timer

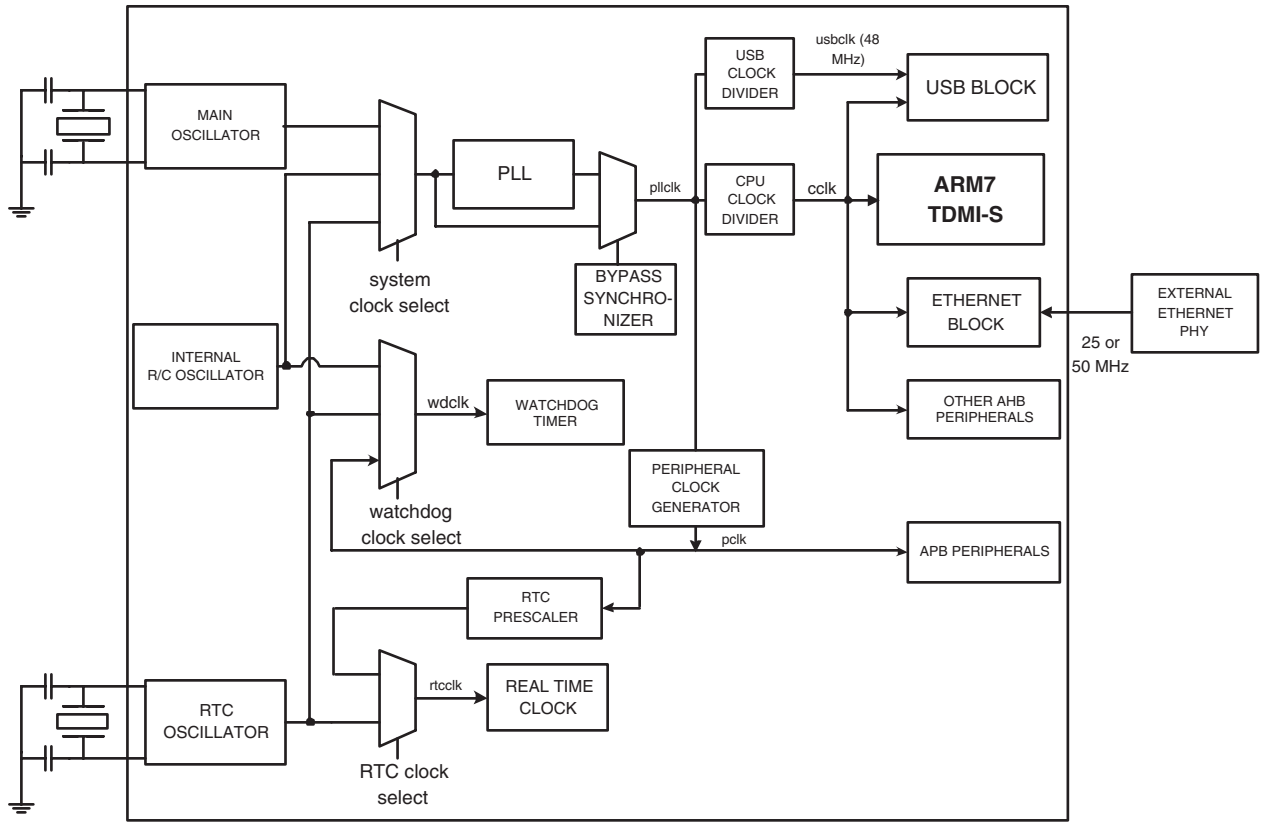


Fig 7. Clock generation for the LPC2400

2. Register description

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

Table 19. Summary of system control registers

Name	Description	Access	Reset value	Address
Clock source selection				
CLKSRCSEL	Clock Source Select Register	R/W	0	0xE01F C10C
Phase Locked Loop				
PLLCON	PLL Control Register	R/W	0	0xE01F C080
PLLCFG	PLL Configuration Register	R/W	0	0xE01F C084
PLLSTAT	PLL Status Register	RO	0	0xE01F C088
PLLFEED	PLL Feed Register	WO	NA	0xE01F C08C
Clock dividers				
CCLKCFG	CPU Clock Configuration Register	R/W	0	0xE01F C104
USBCLKCFG	USB Clock Configuration Register	R/W	0	0xE01F C108
IRCTRIM	IRC Trim Register	R/W	0xA0	0xE01FC1A4
PCLKSEL0	Peripheral Clock Selection register 0.	R/W	0	0xE01F C1A8
PCLKSEL1	Peripheral Clock Selection register 1.	R/W	0	0xE01F C1AC
Power control				
PCON	Power Control Register	R/W	0	0xE01F C0C0
INTWAKE	Interrupt Wakeup Register	R/W	0	0xE01F C144
PCONP	Power Control for Peripherals Register	R/W	0x03BE	0xE01F C0C4

3. Oscillators

The LPC2400 includes three independent oscillators. These are the Main Oscillator, the Internal RC Oscillator, and the RTC oscillator. Each oscillator can be used for more than one purpose as required in a particular application.

Following Reset, the LPC2400 will operate from the Internal RC Oscillator until switched by software. This allows systems to operate without any external crystal, and allows the Boot Loader code to operate at a known frequency. When Boot Block will branch to a user program, there could be an option to activate the main oscillator prior to entering user code.

3.1 Internal RC oscillator

The Internal RC Oscillator (IRC) may be used as the clock source for the watchdog timer, and/or as the clock that drives the PLL and subsequently the CPU. The precision of the IRC does not allow for use of the USB interface, which requires a much more precise time base. The nominal IRC frequency is 4 MHz.

Upon power up or any chip reset, the LPC2400 uses the IRC as the clock source. Software may later switch to one of the other available clock sources.

3.2 Main oscillator

The main oscillator can be used as the clock source for the CPU, with or without using the PLL. The main oscillator operates at frequencies of 1 MHz to 24 MHz. This frequency can be boosted to a higher frequency, up to the maximum CPU operating frequency, by the PLL. The oscillator output is called OSCCLK. The clock selected as the PLL input is PLLCLKIN and the ARM processor clock frequency is referred to as CCLK for purposes of rate equations, etc. elsewhere in this document. The frequencies of PLLCLKIN and CCLK are the same value unless the PLL is active and connected. Refer to the PLL description in this chapter for details.

Since chip operation always begins using the Internal RC Oscillator, and the main oscillator may never be used in some applications, it will only be started by software request. This is accomplished by setting the OSCEN bit in the SCS register, as described in the System Control Block chapter. The main oscillator provides a status flag (the OSCSTAT bit in the SCS register) so that software can determine when the oscillator is running and stable. At that point, software can control switching to the main oscillator as a clock source. Prior to starting the main oscillator, a frequency range must be selected by configuring the OSCRANGE bit in the SCS register.

3.3 RTC oscillator

The RTC oscillator can be used as the clock source for the RTC, and/or the watchdog timer. Also, the RTC oscillator can be used to drive the PLL and the CPU.

4. Clock source selection multiplexer

Several clock sources may be chosen to drive the PLL and ultimately the CPU and on-chip peripheral devices. The clock sources available are the main oscillator, the RTC oscillator, and the Internal RC oscillator.

The clock source selection can only be changed safely when the PLL is not connected. For a detailed description of how to change the clock source in a system using the PLL see Section 4–5.14 “PLL setup sequence”.

4.1 Clock Source Select register (CLKSRCSEL - 0xE01F C10C)

The PCLKSRCSEL register contains the bits that select the clock source for the PLL.

Table 20. Clock Source Select register (CLKSRCSEL - address 0xE01F C10C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CLKSRC		Selects the clock source for the PLL as follows:	0
		00	Selects the Internal RC oscillator as the PLL clock source (default).	
		01	Selects the main oscillator as the PLL clock source.	
		10	Selects the RTC oscillator as the PLL clock source.	
		11	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	
			Warning: Improper setting of this value, or an incorrect sequence of changing this value may result in incorrect operation of the device.	
7:2	-	0	Unused, always 0.	0

5. PLL (Phase Locked Loop)

The PLL accepts an input clock frequency in the range of 32 kHz to 50 MHz . The input frequency is multiplied up to a high frequency, then divided down to provide the actual clock used by the CPU and the USB block.

5.1 PLL operation

The PLL input, in the range of 32 kHZ to 50 MHz, may initially be divided down by a value "N", which may be in the range of 1 to 256. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. This can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 1 through 32768. The resulting frequency must be in the range of 275 MHz to 550 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is used to adjust the CCO frequency.

There are additional dividers at the PLL output to bring the frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description. A block diagram of the PLL is shown in Figure 4–8

PLL activation is controlled via the PLLCON register. The PLL multiplier and divider values are controlled by the PLLCFG register. These two registers are protected in order to prevent accidental alteration of PLL parameters or deactivation of the PLL. Since all chip operations, including the Watchdog Timer, could be dependent on the PLL if so configured (for example when it is providing the chip clock), accidental changes to the PLL setup could result in unexpected or fatal behavior of the microcontroller. The protection is accomplished by a feed sequence similar to that of the Watchdog Timer. Details are provided in the description of the PLLFEED register.

The PLL is turned off and bypassed following a chip Reset and by entering Power-down mode. PLL is enabled by software only.

It is important that the setup procedure described in Section 4–5.14 “PLL setup sequence” is followed as is or the PLL might not operate at all!.

5.2 PLL and startup/boot code interaction

The boot code for the LPC2400 is a little different from those for the previous NXP ARM7 LPC2000 chips. When there's no valid code (determined by the checksum word) in the user flash or the ISP enable pin (P2.10) is pulled low on startup, the ISP mode will be entered and the boot code will setup the PLL with the IRC. Therefore it can not be assumed that the PLL is disabled when the user opens a debug session to debug the application code. The user startup code must follow the steps described in this chapter to disconnect the PLL.

The boot code may also change the values for some registers when the chip enters ISP mode. For example, the GPIOM bit in the SCS register is set in the ISP mode. If the user doesn't notice it and clears the GPIOM bit in the application code, the application code will not be able to operate with the traditional GPIO function on PORT0 and PORT1.

5.3 Register description

The PLL is controlled by the registers shown in Table 4–21. More detailed descriptions follow. Writes to any unused bits are ignored. A read of any unused bits will return a logic zero.

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

Table 21. PLL registers

Name	Description	Access	Reset value[1]	Address
PLLCON	PLL Control Register. Holding register for updating PLL control bits. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W	0	0xE01F C080
PLLCFG	PLL Configuration Register. Holding register for updating PLL configuration values. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W	0	0xE01F C084
PLLSTAT	PLL Status Register. Read-back register for PLL control and configuration information. If PLLCON or PLLCFG have been written to, but a PLL feed sequence has not yet occurred, they will not reflect the current PLL state. Reading this register provides the actual values controlling the PLL, as well as the PLL status.	RO	0	0xE01F C088
PLLFEED	PLL Feed Register. This register enables loading of the PLL control and configuration information from the PLLCON and PLLCFG registers into the shadow registers that actually affect PLL operation.	WO	NA	0xE01F C08C

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

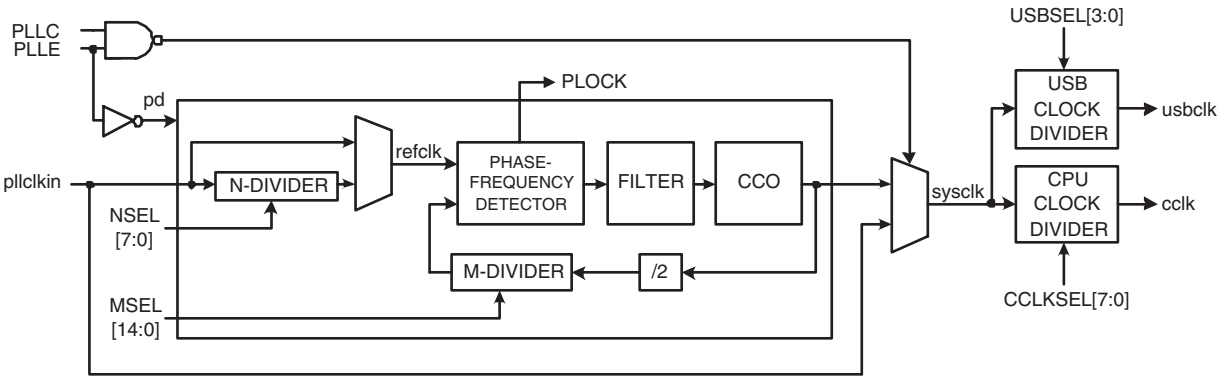


Fig 8. PLL block diagram

5.4 PLL Control register (PLLCON - 0xE01F C080)

The PLLCON register contains the bits that enable and connect the PLL. Enabling the PLL allows it to attempt to lock to the current settings of the multiplier and divider values. Connecting the PLL causes the processor and all chip functions to run from the PLL output clock. Changes to the PLLCON register do not take effect until a correct PLL feed sequence has been given (see Section 4–5.9 “PLL Feed register (PLLFEED - 0xE01F C08C)”).

Table 22. PLL Control register (PLLCON - address 0xE01F C080) bit description

Bit	Symbol	Description	Reset value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 4–24.	0
1	PLLC	PLL Connect. Having both PLLC and PLLE set to one followed by a valid PLL feed sequence, the PLL becomes the clock source for the CPU, as well as the USB subsystem and. Otherwise, the clock selected by the Clock Source Selection Multiplexer is used directly by the LPC2400. See PLLSTAT register, Table 4–24.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

The PLL must be set up, enabled, and Lock established before it may be used as a clock source. When switching from the oscillator clock to the PLL output or vice versa, internal circuitry synchronizes the operation in order to ensure that glitches are not generated. Hardware does not insure that the PLL is locked before it is connected or automatically disconnect the PLL if lock is lost during operation. In the event of loss of PLL lock, it is likely that the oscillator clock has become unstable and disconnecting the PLL will not remedy the situation.

5.5 PLL Configuration register (PLLCFG - 0xE01F C084)

The PLLCFG register contains the PLL multiplier and divider values. Changes to the PLLCFG register do not take effect until a correct PLL feed sequence has been given (see Section 4–5.9 “PLL Feed register (PLLFEED - 0xE01F C08C)”). Calculations for the PLL frequency, and multiplier and divider values are found in the Section 4–5.11 “PLL frequency calculation”.

Table 23. PLL Configuration register (PLLCFG - address 0xE01F C084) bit description

Bit	Symbol	Description	Reset value
14:0	MSEL	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations. The value stored here is M - 1. **Note:** Not all values of M are needed, and therefore some are not supported by hardware. For details on selecting values for MSEL see Section 4–5.11 “PLL frequency calculation”.	0
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
23:16	NSEL	PLL Pre-Divider value. Supplies the value "N" in the PLL frequency calculations. The value stored here is N - 1, giving a range for N of 1 through 256. **Note:** For details on selecting the right value for NSEL see Section 4–5.11 “PLL frequency calculation”.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.6 PLL Status register (PLLSTAT - 0xE01F C088)

The read-only PLLSTAT register provides the actual PLL parameters that are in effect at the time it is read, as well as the PLL status. PLLSTAT may disagree with values found in PLLCON and PLLCFG because changes to those registers do not take effect until a proper PLL feed has occurred (see Section 4–5.9 “PLL Feed register (PLLFEED - 0xE01F C08C)”).

Table 24. PLL Status register (PLLSTAT - address 0xE01F C088) bit description

Bit	Symbol	Description	Reset value
14:0	MSEL	Read-back for the PLL Multiplier value. This is the value currently used by the PLL, and is one less than the actual multiplier.	0
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
23:16	NSEL	Read-back for the PLL Pre-Divider value. This is the value currently used by the PLL, and is one less than the actual divider.	0
24	PLLE	Read-back for the PLL Enable bit. When one, the PLL is currently activated. When zero, the PLL is turned off. This bit is automatically cleared when Power-down mode is activated.	0

Table 24. PLL Status register (PLLSTAT - address 0xE01F C088) bit description

Bit	Symbol	Description	Reset value
25	PLLC	Read-back for the PLL Connect bit. When PLLC and PLLE are both one, the PLL is connected as the clock source for the LPC2400. When either PLLC or PLLE is zero, the PLL is bypassed. This bit is automatically cleared when Power-down mode is activated.	0
26	PLOCK	Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency. See text for details.	0
31:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.7 PLL Interrupt: PLOCK

The PLOCK bit in the PLLSTAT register reflects the lock status of the PLL. When the PLL is enabled, or parameters are changed, the PLL requires some time to establish lock under the new conditions. PLOCK can be monitored to determine when the PLL may be connected for use. The value of PLOCK may not be stable when the PLL reference frequency (F_{REF}, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 μs when FREF is greater than 400 kHz and 200 / FREF seconds when FREF is less than 400 kHz

PLOCK is connected to the interrupt controller. This allows for software to turn on the PLL and continue with other functions without having to wait for the PLL to achieve lock. When the interrupt occurs, the PLL may be connected, and the interrupt disabled.

5.8 PLL Modes

The combinations of PLLE and PLLC are shown in Table 4–25.

Table 25. PLL control bit combinations

PLLC	PLLE	PLL Function
0	0	PLL is turned off and disconnected. The PLL outputs the unmodified clock input.
0	1	The PLL is active, but not yet connected. The PLL can be connected after PLOCK is asserted.
1	0	Same as 00 combination. This prevents the possibility of the PLL being connected without also being enabled.
1	1	The PLL is active and has been connected as the system clock source.

5.9 PLL Feed register (PLLFEED - 0xE01F C08C)

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

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

The two writes must be in the correct sequence, and must be consecutive APB bus cycles. The latter requirement implies that interrupts must be disabled for the duration of the PLL feed operation. If either of the feed values is incorrect, or one of the previously mentioned conditions is not met, any changes to the PLLCON or PLLCFG register will not become effective.

Table 26. PLL Feed register (PLLFEED - address 0xE01F C08C) bit description

Bit	Symbol	Description	Reset value
7:0	PLLFEED	The PLL feed sequence must be written to this register in order for PLL configuration and control register changes to take effect.	0x00

5.10 PLL and Power-down mode

Power-down mode automatically turns off and disconnects the PLL. Wakeup from Power-down mode does not automatically restore the PLL settings, this must be done in software. Typically, a routine to activate the PLL, wait for lock, and then connect the PLL can be called at the beginning of any interrupt service routine that might be called due to the wakeup. It is important not to attempt to restart the PLL by simply feeding it when execution resumes after a wakeup from Power-down mode. This would enable and connect the PLL at the same time, before PLL lock is established.

5.11 PLL frequency calculation

The PLL equations use the following parameters:

Table 27. PLL frequency parameter

Parameter	Description
F_{IN}	the frequency of PLLCLKIN from the Clock Source Selection Multiplexer.
F_{CCO}	the frequency of the SYSCLK (output of the PLL Current Controlled Oscillator)
N	PLL Pre-divider value from the NSEL bits in the PLLCFG register (PLLCFG NSEL field + 1). N is an integer from 1 through 32.
M	PLL Multiplier value from the MSEL bits in the PLLCFG register (PLLCFG MSEL field + 1). Not all potential values are supported. See below.
F_{REF}	PLL internal reference frequency, FIN divided by N.

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

$$F_{CCO} = (2 \times M \times F_{IN}) / N$$

The PLL inputs and settings must meet the following:

- F_{IN} is in the range of 32 kHz to 50 MHz.
- F_{CCO} is in the range of 275 MHz to 550 MHz.

The PLL equation can be solved for other PLL parameters:

$$M = (F_{CCO} \times N) / (2 \times F_{IN})$$

$$N = (2 \times M \times F_{IN}) / F_{CCO}$$

$$F_{IN} = (F_{CCO} \times N) / (2 \times M)$$

Allowed values for M:

At higher oscillator frequencies, in the MHz range, values of M from 6 through 512 are allowed. This supports the entire useful range of both the main oscillator and the IRC.

For lower frequencies, specifically when the RTC is used to clock the PLL, a set of 65 additional M values have been selected for supporting baud rate generation, CAN/USB operation, and attaining even MHz frequencies. These values are shown in Table 4–28

Table 28. Additional Multiplier Values for use with a Low Frequency Clock Input

Low Frequency PLL Multipliers				
4272	4395	4578	4725	4807
5127	5188	5400	5493	5859
6042	6075	6104	6409	6592
6750	6836	6866	6958	7050
7324	7425	7690	7813	7935
8057	8100	8545	8789	9155
9613	10254	10376	10986	11719
12085	12207	12817	13184	13672
13733	13916	14099	14420	14648
15381	15564	15625	15869	16113
16479	17578	18127	18311	19226
19775	20508	20599	20874	21149
21973	23071	23438	23804	24170

5.12 Procedure for determining PLL settings

PLL parameter determination can be simplified by using a spreadsheet available from NXP. To determine PLL parameters by hand, the following general procedure may be used:

1. Determine if the application requires use of the USB interface. The USB requires a 50% duty cycle clock of 48 MHz within a very small tolerance, which means that F_{CCO} must be an even integer multiple of 48 MHz (i.e. an integer multiple of 96 MHz), within a very small tolerance.
2. Choose the desired processor operating frequency (CCLK). This may be based on processor throughput requirements, need to support a specific set of UART baud rates, etc. Bear in mind that peripheral devices may be running from a lower clock frequency than that of the processor (see Section 4–6 “Clock dividers” on page 40 and Section 4–7 “Power control” on page 43). Find a value for F_{CCO} that is close to a multiple of the desired CCLK frequency, bearing in mind the requirement for USB support in [1] above, and that lower values of F_{CCO} result in lower power dissipation.
3. Choose a value for the PLL input frequency (F_{IN}). This can be a clock obtained from the main oscillator, the RTC oscillator, or the on-chip RC oscillator. For USB support, the main oscillator should be used.
4. Calculate values for M and N to produce a sufficiently accurate F_{CCO} frequency. The desired M value -1 will be written to the MSEL field in PLLCFG. The desired N value -1 will be written to the NSEL field in PLLCFG.

In general, it is better to use a smaller value for N, to reduce the level of multiplication that must be accomplished by the CCO. Due to the difficulty in finding the best values in some cases, it is recommended to use a spreadsheet or similar method to show many possibilities at once, from which an overall best choice may be selected. A spreadsheet is available from NXP for this purpose.

5.13 Examples of PLL settings

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

Example 1)

Assumptions:

- The USB interface will be used in the application. The lowest integer multiple of 96 MHz that falls within the PLL operating range (288 MHz) will be targeted.
- The desired CPU rate = 60 MHz.
- An external 4 MHz crystal or clock source will be used as the system clock source.

Calculations:

$M = (F_{CCO} \times N) / (2 \times F_{IN})$

Start by assuming N = 1, since this produces the smallest multiplier needed for the PLL. So, $M = 288 \times 10^6 / (2 \times 4 \times 10^6) = 36$. Since the result is an integer, there is no need to look further for a good set of PLL configuration values. The value written to PLLCFG would be 0x23 (N - 1 = 0; M - 1 = 35 = 0x23).

The potential CPU clock rate can be determined by dividing F_{CCO} by the desired CPU frequency: $288 \times 10^6 / 60 \times 10^6 = 4.8$. The nearest integer value for the CPU Clock Divider is then 5, giving us 57.6 MHz as the nearest value to the desired CPU clock rate.

If it is important to obtain exactly 60 MHz, an F_{CCO} rate must be found that can be divided down to both 48 MHz and 60 MHz. The only possibility is 480 MHz. Divided by 10, this gives the 48 MHz with a 50% duty cycle needed by the USB block. Divided by 8, it gives 60 MHz for the CPU clock. PLL settings for 480 MHz are N = 1 and M = 60.

Example 2)

Assumptions:

- The USB interface will not be used in the application.
- The desired CPU rate = 72 MHz
- The 32.768 kHz RTC clock source will be used as the system clock source

Calculations:

$M = (F_{CCO} \times N) / (2 \times F_{IN})$

The smallest frequency for F_{CCO} that can produce our desired CPU clock rate and is within the PLL operating range is 288 MHz (4×72 MHz). Start by assuming N = 1, since this produces the smallest multiplier needed for the PLL.

So, M = 288 × 10^6 / (2 × 32,768) = 4,394.53125. This is not an integer, so the CPU frequency will not be exactly 288 MHz with this setting. Since this case is less obvious, it may be useful to make a table of possibilities for different values of N (see Table 4–29).

Table 29. Potential values for PLL example

N	M	M Rounded	F_{REF} (Hz)	F_{CCO} (Hz)	Actual CCLK (Hz)	% Error
1	4394.53125	4395	32768	288.0307	72.0077	0.0107
2	8789.0625	8789	16384	287.9980	71.9995	-0.0007
3	13183.59375	13184	10922.67	288.0089	72.0022	0.0031
4	17578.125	17578	8192	287.9980	71.9995	-0.0007
5	21972.65625	21973	6553.6	288.0045	72.0011	0.0016

Beyond N = 7, the value of M is out of range or not supported, so the table stops there. In the table, the calculated M value is rounded to the nearest integer. If this results in CCLK being above the maximum operating frequency (72 MHz), it is allowed if it is not more than ½% above the maximum frequency.

In general, larger vlaues of F_{REF} result in a more stable PLL when the input clock is a low frequency. Even the first table entry shows a very small error of just over 1 hundredth of a percent, or 107 parts per million (ppm). If that is not accurate enough in the application, the second case gives a much smaller error of 7 ppm.

Remember that when a frequency below about 1 MHz is used as the PLL clock source, not all multiplier values are available. As it turns out, all of the rounded M values found in Table 4–29 of this exmaple are supported, as may be confirmed in Table 4–28.

If PLL calculations suggest use of unsupported multiplier values, those values must be disregarded and other values examined to find the best fit. Multiplier values one count off from calculated values may also be good possibilities..

The value written to PLLCFG for the second table entry would be 0x12254 (N - 1 = 1 = 0x1; M - 1 = 8788 = 0x2254).

5.14 PLL setup sequence

The following sequence must be followed step by step in order to have the PLL initialized an running:

1. Disconnect the PLL with one feed sequence if PLL is already connected.
2. Disable the PLL with one feed sequence.
3. Change the CPU Clock Divider setting to speed up operation without the PLL, if desired.
4. Write to the Clock Source Selection Control register to change the clock source.
5. Write to the PLLCFG and make it effective with one feed sequence. The PLLCFG can only be updated when the PLL is disabled.
6. Enable the PLL with one feed sequence.
7. Change the CPU Clock Divider setting for the operation with the PLL. It's critical to do this before connecting the PLL.

8. Wait for the PLL to achieve lock by monitoring the PLOCK bit in the PLLSTAT register, or using the PLOCK interrupt, or wait for a fixed time when the input clock to PLL is slow (i.e. 32 kHz). The value of PLOCK may not be stable when the PLL reference frequency (FREF, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 µs when FREF is greater than 400 kHz and 200 / FREF seconds when FREF is less than 400 kHz.
9. Connect the PLL with one feed sequence.

It's very important not to merge any steps above. For example, don't update the PLLCFG and enable the PLL simultaneously with the same feed sequence.

6. Clock dividers

The output of the PLL must be divided down for use by the CPU and the USB block. Separate dividers are provided such that the CPU frequency can be determined independently from the USB block, which always requires 48 MHz with a 50% duty cycle for proper operation.

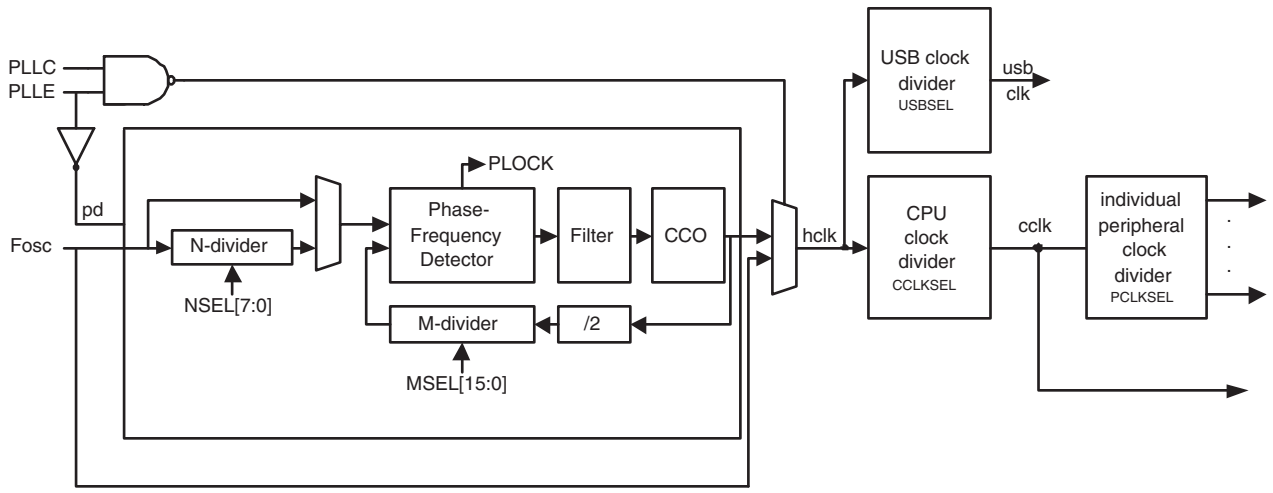


Fig 9. PLL and clock dividers

6.1 CPU Clock Configuration register (CCLKCFG - 0xE01F C104)

The CCLKCFG register controls the division of the PLL output before it is used by the CPU. When the PLL is bypassed, the division may be by 1. When the PLL is running, the output must be divided in order to bring the CPU clock frequency (CCLK) within operating limits. An 8 bit divider allows a range of options, including slowing CPU operation to a low rate for temporary power savings without turning off the PLL.

Note: when the USB interface is used in an application, CCLK must be at least 18 MHz in order to support internal operations of the USB block.

Table 30. CPU Clock Configuration register (CCLKCFG - address 0xE01F C104) bit description

Bit	Symbol	Description	Reset value
7:0	CCLKSEL	Selects the divide value for creating the CPU clock (CCLK) from the PLL output. **Warning:** Improper setting of this value may result in incorrect operation of the device.	0x00

The CCLK is derived from the PLL output signal, divided by CCLKSEL + 1. Having CCLKSEL = 1 results in CCLK being one half the PLL output.

6.2 USB Clock Configuration register (USBCLKCFG - 0xE01F C108)

The USBCLKCFG register controls the division of the PLL output before it is used by the USB block. If the PLL is bypassed, the division may be by 1. In that case, the PLL input frequency must be 48 MHz, with a 500 ppm tolerance. When the PLL is running, the output must be divided in order to bring the USB clock frequency to 48 MHz with a 50% duty cycle. A 4-bit divider allows obtaining the correct USB clock from any even multiple of 48 MHz (i.e. any mutliple of 96 MHz) within the PLL operating range.

Table 31. USB Clock Configuration register (USBCLKCFG - address 0xE01F C108) bit description

Bit	Symbol	Description	Reset value
3:0	USBSEL	Selects the divide value for creating the USB clock from the PLL output. **Warning:** Improper setting of this value will result in incorrect operation of the USB interface.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

The USB clock is derived from the PLL output signal, divided by USBSEL + 1. Having USBSEL = 1 results in USB's clock being one half the PLL output.

6.3 IRC Trim Register (IRCTRIM - 0xE01F C1A4)

This register is used to trim the on - chip 4 MHz oscillator.

Table 32. IRC Trim register (IRCTRIM - address 0xE01F C1A4) bit description

Bit	Symbol	Description	Reset value
7:0	IRCtrim	IRC trim value. It controls the on-chip 4 MHz IRC frequency.	0xA0
15:8	-	Reserved. Software must write 0 into these bits.	NA

6.4 Peripheral Clock Selection registers 0 and 1 (PCLKSEL0 - 0xE01F C1A8 and PCLKSEL1 - 0xE01F C1AC)

A pair of bits in a Peripheral Clock Selection register controls the rate of the clock signal that will be supplied to the corresponding peripheral as specified in Table 4–33, Table 4–34 and Table 4–35. For details on the HCLK clock see Figure 4–9.

Table 33. Peripheral Clock Selection register 0 (PCLKSEL0 - address 0xE01F C1A8) bit description

Bit	Symbol	Description	Reset value
1:0	PCLK_WDT	Peripheral clock selection for WDT.	00
3:2	PCLK_TIMER0	Peripheral clock selection for TIMER0.	00
5:4	PCLK_TIMER1	Peripheral clock selection for TIMER1.	00
7:6	PCLK_UART0	Peripheral clock selection for UART0.	00
9:8	PCLK_UART1	Peripheral clock selection for UART1.	00
11:10	PCLK_PWM0	Peripheral clock selection for PWM0.	00
13:12	PCLK_PWM1	Peripheral clock selection for PWM1.	00
15:14	PCLK_I2C0	Peripheral clock selection for I2C0.	00
17:16	PCLK_SPI	Peripheral clock selection for SPI.	00
19:18	PCLK_RTC[1]	Peripheral clock selection for RTC.	00
21:20	PCLK_SSP1	Peripheral clock selection for SSP1.	00
23:22	PCLK_DAC	Peripheral clock selection for DAC.	00
25:24	PCLK_ADC	Peripheral clock selection for ADC.	00
27:26	PCLK_CAN1	Peripheral clock selection for CAN1.	00
29:28	PCLK_CAN2	Peripheral clock selection for CAN2.	00
31:30	PCLK_ACF	Peripheral clock selection for CAN filtering.	00

[1] For PCLK_RTC only, the value '01' is illegal. Do not write '01' to the PCLK_RTC. Attempting to write '01' results in the previous value being unchanged.

Table 34. Peripheral Clock Selection register 1 (PCLKSEL1 - address 0xE01F C1AC) bit description

Bit	Symbol	Description	Reset value
1:0	PCLK_BAT_RAM	Peripheral clock selection for the battery supported RAM.	00
3:2	PCLK_GPIO	Peripheral clock selection for GPIOs.	00
5:4	PCLK_PCB	Peripheral clock selection for the Pin Connect block.	00
7:6	PCLK_I2C1	Peripheral clock selection for I2C1.	00
9:8	-	Unused, always read as 0.	00
11:10	PCLK_SSP0	Peripheral clock selection for SSP0.	00
13:12	PCLK_TIMER2	Peripheral clock selection for TIMER2.	00
15:14	PCLK_TIMER3	Peripheral clock selection for TIMER3.	00
17:16	PCLK_UART2	Peripheral clock selection for UART2.	00
19:18	PCLK_UART3	Peripheral clock selection for UART3.	00
21:20	PCLK_I2C2	Peripheral clock selection for I2C2.	00
23:22	PCLK_I2S	Peripheral clock selection for I2S.	00
25:24	PCLK_MCI	Peripheral clock selection for MCI.	00
27:26	-	Unused, always read as 0.	00
29:28	PCLK_SYSCON	Peripheral clock selection for the System Control block.	00
31:30	-	Unused, always read as 0.	00

Table 35. Peripheral Clock Selection register bit values

PCLKSEL0 and PCLKSEL1 individual peripheral's clock select options	Function	Reset value
00	PCLK_xyz = CCLK/4	00
01	PCLK_xyz = CCLK[1]	
10	PCLK_xyz = CCLK/2	
11	Peripheral's clock is selected to PCLK_xyz = HCLK/8 except for CAN1, CAN2, and CAN filtering when '11' selects PCLK_xyz = HCLK/6.	

[1] For PCLK_RTC only, the value '01' is illegal. Do not write '01' to the PCLK_RTC. Attempting to write '01' results in the previous value being unchanged.

7. Power control

The LPC2400 supports a variety of power control features. There are four special modes of processor power reduction: Idle mode, Sleep mode, Power-down mode, and Deep Power-down mode. The CPU clock rate may also be controlled as needed by changing clock sources, re-configuring PLL values, and/or altering the CPU clock divider value. This allows a trade-off of power versus processing speed based on application requirements. In addition, Peripheral Power Control allows shutting down the clocks to individual on-chip peripherals, allowing fine tuning of power consumption by eliminating all dynamic power use in any peripherals that are not required for the application.

The LPC2400 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, referred to as the Battery RAM. This feature is described in more detail later in this chapter under the heading Power Domains, and in the Real Time Clock and Battery RAM chapter.

7.1 Idle mode

When Idle mode is entered, the clock to the core is stopped. Resumption from the Idle mode does not need any special sequence but re-enabling the clock to the ARM core.

In Idle mode, execution of instructions is suspended until either a Reset or interrupt occurs. Peripheral functions continue operation during Idle mode and may generate interrupts to cause the processor to resume execution. Idle mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses.

7.2 Sleep mode

When the chip enters the Sleep mode, the main oscillator is powered down and all clocks are stopped. The output of the IRC is disabled but the IRC is not powered down for a fast wakeup later. The 32 kHz RTC oscillator is not stopped because the RTC interrupts may be used as the wakeup source. The Flash is left in the standby mode allowing a very quick wakeup. The PLL is automatically turned off and disconnected. The CCLK and USBCLK clock dividers automatically get reset to zero.

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

On the wakeup of sleep mode, if the IRC was used before entering sleep mode, the 2-bit IRC timer starts counting and the code execution and peripherals activities will resume after the timer expires (4 cycles). If the main external oscillator was used, the 12-bit main oscillator timer starts counting and the code execution will resume when the timer expires (4096 cycles). Customer must not forget to re-configure the PLL and clock dividers after the wakeup.

7.3 Power-down mode

Power-down mode does everything that Sleep mode does, but also turns off the Flash memory. This saves more power, but requires waiting for resumption of Flash operation before execution of code or data access in the Flash memory can be accomplished.

When the chip enters power-down mode, the IRC, the main oscillator and all clocks are stopped. The 32Khz RTC oscillator is not stopped because the RTC interrupts may be used as the wakeup source. The flash is forced into power-down mode. The PLL is automatically turned off and disconnected. The CCLK and USBCLK clock dividers automatically get reset to zero.

On the wakeup of power-down mode, if the IRC was used before entering power-down mode, after IRC-start-up time (60 μs), the 2-bit IRC timer starts counting and expires in 4 cycles. The code execution can then be resumed immediately upon the expiration of the IRC timer if the code was running from SRAM. In the meantime, the Flash wakeup-timer generates Flash start-up time 100 μs. When it times out, access to the Flash is enabled. Customer must not forget to re-configure the PLL and clock dividers after the wakeup.

7.4 Deep Power-down mode

Deep Power-down mode is like Power-down mode, but the on-chip regulator that supplies power to internal logic is also shut off. This produces the lowest possible power consumption without actually removing power from the entire chip. Since Deep Power-down mode shuts down the on-chip logic power supply, there is no register or memory retention, and resumption of operation involves the same activities as a full-chip reset.

If power is supplied to the LPC2400 during Deep Power-down mode, wakeup can be caused by the RTC Alarm or external Reset.

While in Deep Power-down mode, external device power may be removed. In this case, the LPC2400 will start up when external power is restored.

Essential data may be retained through Deep Power-down mode (or through complete powering off of the chip) by storing data in the Battery RAM, as long as the external power to the VBAT pin is maintained.

7.5 Peripheral power control

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

7.6 Register description

The Power Control function uses registers shown in Table 4–36. More detailed descriptions follow.

Table 36. Power Control registers

Name	Description	Access	Reset value[1]	Address
PCON	Power Control Register. This register contains control bits that enable the two reduced power operating modes of the LPC2400. See Table 4–37.	R/W	0x00	0xE01F C0C0
INTWAKE	Interrupt Wakeup Register. Controls which interrupts will wake the LPC2400 from power-down mode. See Table 4–39	R/W	0x00	0xE01F C144
PCONP	Power Control for Peripherals Register. This register contains control bits that enable and disable individual peripheral functions, allowing elimination of power consumption by peripherals that are not needed.	R/W		0xE01F C0C4

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

7.7 Power Mode Control register (PCON - 0xE01F C0C0)

Reduced power modes are controlled via the PCON register, as described in Table 4–37.

Table 37. Power Mode Control register (PCON - address 0xE01F COCO) bit description

Bit	Symbol	Description	Reset value
0	PM0 (IDL)	Power mode control bit 0. See text and table below for details.	0
1	PM1 (PD)	Power mode control bit 1. See text and table below for details.	0
2	BODPDM	Brown-Out Power-down Mode. When BODPDM is 1, the Brown-Out Detect circuitry will turn off when chip Power-down mode is entered, resulting in a further reduction in power usage. However, the possibility of using Brown-Out Detect as a wakeup source from Power-down mode will be lost. When 0, the Brown-Out Detect function remains active during Power-down mode. See the System Control Block chapter for details of Brown-Out detection.	0
3	BOGD	Brown-Out Global Disable. When BOGD is 1, the Brown-Out Detect circuitry is fully disabled at all times, and does not consume power. When 0, the Brown-Out Detect circuitry is enabled. See the System Control Block chapter for details of Brown-Out detection.	0

Table 37. Power Mode Control register (PCON - address 0xE01F COCO) bit description

Bit	Symbol	Description	Reset value
4	BORD	Brown-Out Reset Disable. When BORD is 1, the second stage of low voltage detection (2.6 V) will not cause a chip reset. When BORD is 0, the reset is enabled. The first stage of low voltage detection (2.9 V) Brown-Out interrupt is not affected. See the System Control Block chapter for details of Brown-Out detection.	0
6:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	PM2	Power mode control bit 2. See text and table below for details.	0

Encoding of Reduced Power Modes

The PM2, PM1, and PM0 bits in PCON allow entering reduced power modes as needed. The encoding of these bits allows backward compatibility with devices that previously only supported Idle and Power-down modes. Table 4–38 below shows the encoding for the four reduced power modes supported by the LPC2400.

Table 38. Encoding of reduced power modes

PM2, PM1, PM0	Description
000	Normal operation
001	Idle mode. Causes the processor clock to be stopped, while on-chip peripherals remain active. Any enabled interrupt from a peripheral or an external interrupt source will cause the processor to resume execution. See text for details.
101	Sleep mode. This mode is similar to Power-down mode (the oscillator and all on-chip clocks are stopped), but the Flash memory is left in Standby mode. This allows a more rapid wakeup than Power-down mode because the Flash reference voltage regulator start-up time is not needed. See text for details.
010	Power-down mode. Causes the oscillator and all on-chip clocks to be stopped. A wakeup condition from an external interrupt can cause the oscillator to re-start, the PD bit to be cleared, and the processor to resume execution. See text for details.
110	Deep Power-down mode. This is the most extreme power saving mode. As in Power-down mode, Deep Power-down mode causes the oscillator and all on-chip clocks to be stopped, but also turns off the on-chip DC-DC converter that supplies power to internal circuitry. See text for details.
Others	Reserved, not currently used.

7.8 Interrupt Wakeup Register (INTWAKE - 0xE01F C144)

Enable bits in the INTWAKE register allow the external interrupts to wake up the processor if it is in Power-down mode. The related EINTn function must be mapped to the pin in order for the wakeup process to take place. It is not necessary for the interrupt to be enabled in the Vectored Interrupt Controller for a wakeup to take place. This arrangement allows additional capabilities, such as having an external interrupt input wake up the processor from Power-down mode without causing an interrupt (simply resuming operation), or allowing an interrupt to be enabled during Power-down without waking the processor up if it is asserted (eliminating the need to disable the interrupt if the wakeup feature is not desirable in the application). Details of the wakeup operations are shown in Table 4–39.

For an external interrupt pin to be a source that would wake up the microcontroller from Power-down mode, it is also necessary to clear the corresponding interrupt flag (see Section 3–6.2 “External Interrupt flag register (EXTINT - 0xE01F C140)”).

Table 39. Interrupt Wakeup register (INTWAKE - address 0xE01F C144) bit description

Bit	Symbol	Description	Reset value
0	EXTWAKE0	When one, assertion of $\overline{\text{EINT0}}$ will wake up the processor from Power-down mode.	0
1	EXTWAKE1	When one, assertion of $\overline{\text{EINT1}}$ will wake up the processor from Power-down mode.	0
2	EXTWAKE2	When one, assertion of $\overline{\text{EINT2}}$ will wake up the processor from Power-down mode.	0
3	EXTWAKE3	When one, assertion of $\overline{\text{EINT3}}$ will wake up the processor from Power-down mode.	0
4	ETHWAKE	When one, assertion of the Wake-up on LAN interrupt (WakeupInt) of the Ethernet block will wake up the processor from Power-down mode.	0
5	USBWAKE	When one, activity on the USB bus will wake up the processor from Power-down mode. Any change of state on the USB data pins will cause a wakeup when this bit is set. For details on the relationship of USB to Power-down Mode and wakeup, see the relevant USB chapter(s).	0
6	CANWAKE	When one, activity of the CAN bus will wake up the processor from Power-down mode. Any change of state on the CAN receive pins will cause a wakeup when this bit is set.	0
7	GPIOWAKE	When one, specified activity on GPIO pins enabled for wakeup will wake up the processor from Power-down mode. See the GPIO chapter for details.	0
13:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
14	BODWAKE	When one, Brown-Out Detect interrupt will wake up the processor from Power-down mode. **Note:** since there is a delay before execution begins, there is no guarantee that execution will resume before $V_{DD(3V3)}$ has fallen below the lower BOD threshold, which prevents execution. If execution does resume, there is no guarantee of how long the processor will continue execution before the lower BOD threshold terminates execution. These issues depend on the slope of the decline of $V_{DD(3V3)}$. High decoupling capacitance (between $V_{DD(3V3)}$ and ground) in the vicinity of the LPC2400 will improve the likelihood that software will be able to do what needs to be done when power is in the process of being lost.	0
15	RTCWAKE	When one, assertion of an RTC interrupt will wake up the processor from Power-down mode.	0

7.9 Power Control for Peripherals register (PCONP - 0xE01F COC4)

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

Some peripherals, particularly those that include analog functions, may consume power that is not clock dependent. These peripherals may contain a separate disable control that turns off additional circuitry to reduce power. Information on peripheral specific power saving features may be found in the chapter describing that peripheral.

Each bit in PCONP controls one peripheral as shown in Table 4–40. The bit numbers correspond to the related peripheral number as shown in the APB peripheral map Table 2–4 “APB peripherals and base addresses” in the "LPC2400 Memory Addressing" chapter.

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

Important: valid read from a peripheral register and valid write to a peripheral register is possible only if that peripheral is enabled in the PCONP register!

Table 40. Power Control for Peripherals register (PCONP - address 0xE01F C0C4) bit description

Bit	Symbol	Description	Reset value
0	-	Unused, always 0	0
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	PCPWM0	PWM0 power/clock control bit.	1
6	PCPWM1	PWM1 power/clock control bit.	1
7	PCI2C0	The I^2C0 interface power/clock control bit.	1
8	PCSPI	The SPI interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSSP1	The SSP 1 interface power/clock control bit.	1
11	PCEMC	External Memory Controller	1
12	PCAD	A/D converter (ADC) power/clock control bit. **Note:** Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	0
13	PCAN1	CAN Controller 1 power/clock control bit.	0
14	PCAN2	CAN Controller 2 power/clock control bit.	0
18:15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19	PCI2C1	The I^2C1 interface power/clock control bit.	1
20	-	Unused, always 0	0
21	PCSSP0	The SSP0 interface power/clock control bit.	1
22	PCTIM2	Timer 2 power/clock control bit.	0
23	PCTIM3	Timer 3 power/clock control bit.	0
24	PCUART2	UART 2 power/clock control bit.	0
25	PCUART3	UART 3 power/clock control bit.	0
26	PCI2C2	I^2S interface 2 power/clock control bit.	1

Table 40. Power Control for Peripherals register (PCONP - address 0xE01F C0C4) bit description

Bit	Symbol	Description	Reset value
27	PCI2S	I^2S interface power/clock control bit.	0
28	PCSDC	SD card interface power/clock control bit.	0
29	PCGPDMA	GP DMA function power/clock control bit.	0
30	PCENET	Ethernet block power/clock control bit.	0
31	PCUSB	USB interface power/clock control bit.	0

7.10 Power control usage notes

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

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

7.11 Power domains

The LPC2400 provides two independent power domains that allow the bulk of the device to have power removed while maintaining operation of the Real Time Clock and the Battery RAM.

The VBAT pin supplies power only to the RTC and the Battery RAM. These two functions require a minimum of power to operate, which can be supplied by an external battery. When the CPU and the rest of chip functions are stopped and power removed, the RTC can supply an alarm output that may be used by external hardware to restore chip power and resume operation. Details may be found in the Real Time Clock and Battery RAM chapter on page 547.

Note: Both the RTC and the Battery RAM are enabled/disabled by a single bit in the PCONP register. Therefore, the Battery RAM cannot be accessed unless the RTC is enabled.

8. Wakeup timer

The LPC2400 begins operation at power-up and when awakened from Power-down mode or Deep Power-down mode by using the 4 MHz IRC oscillator as the clock source. This allows chip operation quickly in these cases. If the main oscillator or the PLL is needed by the application, software will need to enable these features and wait for them to stabilize before they are used as a clock source.

When the main oscillator is initially activated, the wakeup timer allows software to ensure that the main oscillator is fully functional before the processor uses it as a clock source and starts to execute instructions. This is important at power-on, all types of Reset, and

whenever any of the aforementioned functions are turned off for any reason. Since the oscillator and other functions are turned off during Power-down mode, any wakeup of the processor from Power-down mode makes use of the Wakeup Timer.

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

Once a clock is detected, the Wakeup Timer counts a fixed number of clocks (4096), then sets the flag (OSCSTAT bit in the SCS register) that indicates that the main oscillator is ready for use. Software can then switch to the main oscillator and, if needed, start the PLL. Refer to the Main Oscillator description in this chapter for details.

1. Introduction

The LPC2468 External Memory Controller (EMC) is an ARM PrimeCell™ MultiPort Memory Controller peripheral offering support for asynchronous static memory devices such as RAM, ROM and Flash, as well as dynamic memories such as Single Data Rate SDRAM. The EMC is an Advanced Microcontroller Bus Architecture (AMBA) compliant peripheral.

2. Features

- Dynamic memory interface support including Single Data Rate SDRAM.
- Asynchronous static memory device support including RAM, ROM, and Flash, with or without asynchronous page mode.
- Low transaction latency.
- Read and write buffers to reduce latency and to improve performance.
- 8 bit, 16 bit, and 32 bit wide static memory support.
- 16 bit and 32 bit wide chip select SDRAM memory support.
- Static memory features include:
  - Asynchronous page mode read.
  - Programmable wait states.
  - Bus turnaround delay.
  - Output enable and write enable delays.
  - Extended wait.
- Four chip selects for synchronous memory and four chip selects for static memory devices.
- Power-saving modes dynamically control CKE and CLKOUT to SDRAMs.
- Dynamic memory self-refresh mode controlled by software.
- Controller supports 2 k, 4 k, and 8 k row address synchronous memory parts. That is typical 512 MB, 256 MB, and 128 MB parts, with 4, 8, 16, or 32 data bits per device.
- Separate reset domains allow the for auto-refresh through a chip reset if desired.

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

3. Functional overview

This chapter describes the major functional blocks of the EMC.

4. EMC functional description

Figure 5–10 shows a block diagram of the EMC.

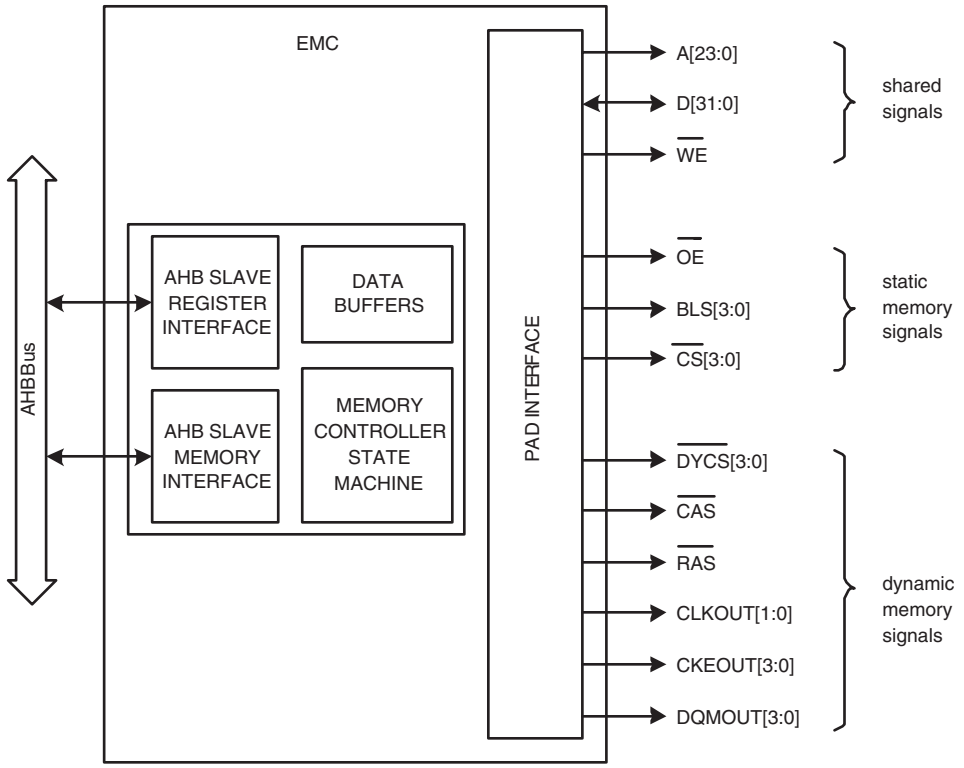


Fig 10. EMC block diagram

The functions of the EMC blocks are described in the following sections:

- AHB slave register interface.
- AHB slave memory interfaces.
- Data buffers.
- Memory controller state machine.
- Pad interface.

Note: For 32 bit wide chip selects data is transferred to and from dynamic memory in SDRAM bursts of four. For 16 bit wide chip selects SDRAM bursts of eight are used.

5. AHB slave register interface

The AHB slave register interface block enables the registers of the EMC to be programmed. This module also contains most of the registers and performs the majority of the register address decoding.

To eliminate the possibility of endianness problems, all data transfers to and from the registers of the EMC must be 32 bits wide.

Note: If an access is attempted with a size other than a word (32 bits), it causes an ERROR response to the AHB bus and the transfer is terminated.

5.1 AHB slave memory interface

The AHB slave memory interface allows access to external memories.

5.1.1 Memory transaction endianness

The endianness of the data transfers to and from the external memories is determined by the Endian mode (N) bit in the EMCConfig register.

Note: The memory controller must be idle (see the busy field of the EMCStatus Register) before endianness is changed, so that the data is transferred correctly.

5.1.2 Memory transaction size

Memory transactions can be 8, 16, or 32 bits wide. Any access attempted with a size greater than a word (32 bits) causes an ERROR response to the AHB bus and the transfer is terminated.

5.1.3 Write protected memory areas

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

5.2 Data buffers

The AHB interface reads and writes via buffers to improve memory bandwidth and reduce transaction latency. The EMC contains four 16-word buffers. The buffers can be used as read buffers, write buffers, or a combination of both. The buffers are allocated automatically.

The buffers are always enabled for dynamic memory. They can be enabled or disabled for static memory using the EMCStaticConfig Registers.

5.2.1 Write buffers

Write buffers are used to:

- Merge write transactions so that the number of external transactions are minimized.

  Buffer data until the EMC can complete the write transaction, improving AHB write latency.

  Convert all dynamic memory write transactions into quadword bursts on the external memory interface. This enhances transfer efficiency for dynamic memory.
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Write buffer operation:

- If the buffers are enabled, an AHB write operation writes into the Least Recently Used (LRU) buffer, if empty.

  If the LRU buffer is not empty, the contents of the buffer are flushed to memory to make space for the AHB write data.
- If a buffer contains write data it is marked as dirty, and its contents are written to memory before the buffer can be reallocated.

The write buffers are flushed whenever:

- The memory controller state machine is not busy performing accesses to external memory.

  The memory controller state machine is not busy performing accesses to external memory, and an AHB interface is writing to a different buffer.

Note: For dynamic memory, the smallest buffer flush is a quadword of data. For static memory, the smallest buffer flush is a byte of data.

5.2.2 Read buffers

Read buffers are used to:

- Buffer read requests from memory. Future read requests that hit the buffer read the data from the buffer rather than memory, reducing transaction latency.

  Convert all read transactions into quadword bursts on the external memory interface. This enhances transfer efficiency for dynamic memory.
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Read buffer operation:

- If the buffers are enabled and the read data is contained in one of the buffers, the read data is provided directly from the buffer.
- If the read data is not contained in a buffer, the LRU buffer is selected. If the buffer is dirty (contains write data), the write data is flushed to memory. When an empty buffer is available the read command is posted to the memory.

A buffer filled by performing a read from memory is marked as not-dirty (not containing write data) and its contents are not flushed back to the memory controller unless a subsequent AHB transfer performs a write that hits the buffer.

5.3 Memory controller state machine

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

5.4 Pad interface

The pad interface block provides the interface to the pads. The pad interface uses feedback clocks, FBCLKIN[3:0], to resynchronize SDRAM read data from the off-chip to on-chip domains.

6. Low-power operation

In many systems, the contents of the memory system have to be maintained during low-power sleep modes. The EMC provides a mechanism to place the dynamic memories into self-refresh mode.

Self-refresh mode can be entered by software by setting the SREFREQ bit in the EMCDynamicControl Register and polling the SREFACK bit in the EMCStatus Register.

Any transactions to memory that are generated while the memory controller is in self-refresh mode are rejected and an error response is generated to the AHB bus. Clearing the SREFREQ bit in the EMCDynamicControl Register returns the memory to normal operation. See the memory data sheet for refresh requirements.

Note: The static memory can be accessed as normal when the SDRAM memory is in self-refresh mode.

6.1 Low-power SDRAM Deep-sleep Mode

The EMC supports JEDEC low-power SDRAM deep-sleep mode. Deep-sleep mode can be entered by setting the deep-sleep mode (DP) bit in the EMCDynamicControl Register. The device is then put into a low-power mode where the device is powered down and no longer refreshed. All data in the memory is lost.

6.2 Low-power SDRAM partial array refresh

The EMC supports JEDEC low-power SDRAM partial array refresh. Partial array refresh can be programmed by initializing the SDRAM memory device appropriately. When the memory device is put into self-refresh mode only the memory banks specified are refreshed. The memory banks that are not refreshed lose their data contents.

7. Memory bank select

Eight independently-configurable memory chip selects are supported:

- Pins CSn3 to CSn0 are used to select static memory devices.
- Pins DYCSn3 to DYCSn0 are used to select dynamic memory devices.

Static memory chip select ranges are each 16 megabytes in size, while dynamic memory chip selects cover a range of 256 megabytes each. Table 5–41 shows the address ranges of the chip selects.

Table 41. Memory bank selection

Chip select pin	Address range	Memory type	Size of range
$\overline{CS0}$	0x8000 000 - 0x80FF FFFF	Static	16 MB
$\overline{CS1}$	0x8100 000 - 0x81FF FFFF	Static	16 MB
$\overline{CS2}$	0x8200 000 - 0x82FF FFFF	Static	16 MB
$\overline{CS3}$	0x8300 000 - 0x83FF FFFF	Static	16 MB
$\overline{DYCS0}$	0xA000 000 - 0xAFFF FFFF	Dynamic	256 MB
$\overline{DYCS1}$	0xB000 000 - 0xBFFF FFFF	Dynamic	256 MB
$\overline{DYCS2}$	0xC000 000 - 0xCFFF FFFF	Dynamic	256 MB
$\overline{DYCS3}$	0xD000 000 - 0xDFFF FFFF	Dynamic	256 MB

8. Reset

The EMC receives two reset signals. One is Power-On Reset (POR), asserted when chip power is applied, and when a brown-out condition is detected (see the System Control Block chapter for details of Brown-Out Detect). The other reset is from the external Reset pin and the Watchdog Timer.

A configuration bit in the SCS register, called EMC_Reset_Disable, allows control of how the EMC is reset. The default configuration (EMC_Reset_Disable = 0) is that both EMC resets are asserted when any type of reset event occurs. In this mode, all registers and functions of the EMC are initialized upon any reset condition.

If EMC_Reset_Disable is set to 1, many portions of the EMC are only reset by a power-on or brown-out event, in order to allow the EMC to retain its state through a warm reset (external reset or watchdog reset). If the EMC is configured correctly, auto-refresh can be maintained through a warm reset.

9. Pin description

Table 5–42 shows the interface and control signal pins for the EMC.

Table 42. Pad interface and control signal descriptions

Name	Type	Value on POR reset	Value during self-refresh	Description
A[23:0]	Output	0x0000 0000	Depends on static memory accesses	External memory address output. Used for both static and SDRAM devices. SDRAM memories use only bits [14:0].
D[31:0]	Input/ Output	Data outputs = 0x0000 0000	Depends on static memory accesses	External memory data lines. These are inputs when data is read from external memory and outputs when data is written to external memory.
$\overline{OE}$	Output	1	Depends on static memory accesses	Low active output enable for static memory devices.
BLS[3:0]	Output	0xF	Depends on static memory accesses	Low active byte lane selects. Used for static memory devices.
$\overline{WE}$	Output	1	Depends on static memory accesses	Low active write enable. Used for SDRAM and static memories.
$\overline{CS}$[3:0]	Output	0xF	Depends on static memory accesses	Static memory chip selects. Default active LOW. Used for static memory devices.
$\overline{DYCS}$[3:0]	Output	0xF	0xF	SDRAM chip selects. Used for SDRAM devices.
$\overline{CAS}$	Output	1	1	Column address strobe. Used for SDRAM devices.
$\overline{RAS}$	Output	1	1	Row address strobe. Used for SDRAM devices.
CLKOUT[1:0]	Output	Follows CCLK	Follows CCLK	SDRAM clocks. Used for SDRAM devices.
CKEOUT[3:0]	Output	0xF	0x0	SDRAM clock enables. Used for SDRAM devices. One is allocated for each Chip Select.
DQMOUT[3:0]	Output	0xF	0xF	Data mask output to SDRAMs. Used for SDRAM devices and static memories.

10. Register description

This chapter describes the EMC registers and provides details required when programming the microcontroller. The EMC registers are shown in Table 5–43.

Table 43. EMC register summary

Address	Register Name	Description	Warm Reset Value	POR Reset Value	Type
0xFFE0 8000	EMCControl	Controls operation of the memory controller.	0x1	0x3	R/W
0xFFE0 8004	EMCStatus	Provides EMC status information.	-	0x5	RO
0xFFE0 8008	EMCConfig	Configures operation of the memory controller	-	0x0	R/W
0xFFE0 8020	EMCDynamic Control	Controls dynamic memory operation.	-	0x006	R/W
0xFFE0 8024	EMCDynamic Refresh	Configures dynamic memory refresh operation.	-	0x0	R/W
0xFFE0 8028	EMCDynamic ReadConfig	Configures the dynamic memory read strategy.	-	0x0	R/W
0xFFE0 8030	EMCDynamicRP	Selects the precharge command period.	-	0x0F	R/W
0xFFE0 8034	EMCDynamic RAS	Selects the active to precharge command period.	-	0xF	R/W
0xFFE0 8038	EMCDynamic SREX	Selects the self-refresh exit time.	-	0xF	R/W
0xFFE0 803C	EMCDynamic APR	Selects the last-data-out to active command time.	-	0xF	R/W
0xFFE0 8040	EMCDynamic DAL	Selects the data-in to active command time.	-	0xF	R/W
0xFFE0 8044	EMCDynamicWR	Selects the write recovery time.	-	0xF	R/W
0xFFE0 8048	EMCDynamicRC	Selects the active to active command period.	-	0x1F	R/W
0xFFE0 804C	EMCDynamic RFC	Selects the auto-refresh period.	-	0x1F	R/W
0xFFE0 8050	EMCDynamic XSR	Selects the exit self-refresh to active command time.	-	0x1F	R/W
0xFFE0 8054	EMCDynamic RRD	Selects the active bank A to active bank B latency.	-	0xF	R/W
0xFFE0 8058	EMCDynamic MRD	Selects the load mode register to active command time.	-	0xF	R/W
0xFFE0 8100	EMCDynamic Config0	Selects the configuration information for dynamic memory chip select 0.	-	0x0	R/W
0xFFE0 8104	EMCDynamic RasCas0	Selects the RAS and CAS latencies for dynamic memory chip select 0.	-	0x303	R/W
0xFFE0 8120	EMCDynamic Config1	Selects the configuration information for dynamic memory chip select 1.	-	0x0	R/W
0xFFE0 8124	EMCDynamic RasCas1	Selects the RAS and CAS latencies for dynamic memory chip select 1.	-	0x303	R/W
0xFFE0 8140	EMCDynamic Config2	Selects the configuration information for dynamic memory chip select 2.	-	0x0	R/W
0xFFE0 8144	EMCDynamic RasCas2	Selects the RAS and CAS latencies for dynamic memory chip select 2.	-	0x303	R/W
0xFFE0 8160	EMCDynamic Config3	Selects the configuration information for dynamic memory chip select 3.	-	0x0	R/W
0xFFE0 8164	EMCDynamic RasCas3	Selects the RAS and CAS latencies for dynamic memory chip select 3.	-	0x303	R/W
0xFFE0 8200	EMCStatic Config0	Selects the memory configuration for static chip select 0.	-	0x0	R/W
0xFFE0 8204	EMCStatic WaitWen0	Selects the delay from chip select 0 to write enable.	-	0x0	R/W
0xFFE0 8208	EMCStatic WaitOen0	Selects the delay from chip select 0 or address change, whichever is later, to output enable.	-	0x0	R/W
0xFFE0 820C	EMCStatic WaitRd0	Selects the delay from chip select 0 to a read access.	-	0x1F	R/W

Table 43. EMC register summary

Address	Register Name	Description	Warm Reset Value	POR Reset Value	Type
0xFFE0 8210	EMCStatic WaitPage0	Selects the delay for asynchronous page mode sequential accesses for chip select 0.	-	0x1F	R/W
0xFFE0 8214	EMCStatic WaitWr0	Selects the delay from chip select 0 to a write access.	-	0x1F	R/W
0xFFE0 8218	EMCStatic WaitTurn0	Selects the number of bus turnaround cycles for chip select 0.	-	0xF	R/W
0xFFE0 8220	EMCStatic Config1	Selects the memory configuration for static chip select 1.	-	0x0	R/W
0xFFE0 8224	EMCStatic WaitWen1	Selects the delay from chip select 1 to write enable.	-	0x0	R/W
0xFFE0 8228	EMCStatic WaitOen1	Selects the delay from chip select 1 or address change, whichever is later, to output enable.	-	0x0	R/W
0xFFE0 822C	EMCStatic WaitRd1	Selects the delay from chip select 1 to a read access.	-	0x1F	R/W
0xFFE0 8230	EMCStatic WaitPage1	Selects the delay for asynchronous page mode sequential accesses for chip select 1.	-	0x1F	R/W
0xFFE0 8234	EMCStatic WaitWr1	Selects the delay from chip select 1 to a write access.	-	0x1F	R/W
0xFFE0 8238	EMCStatic WaitTurn1	Selects the number of bus turnaround cycles for chip select 1.	-	0xF	R/W
0xFFE0 8240	EMCStatic Config2	Selects the memory configuration for static chip select 2.	-	0x0	R/W
0xFFE0 8244	EMCStatic WaitWen2	Selects the delay from chip select 2 to write enable.	-	0x0	R/W
0xFFE0 8248	EMCStatic WaitOen2	Selects the delay from chip select 2 or address change, whichever is later, to output enable.	-	0x0	R/W
0xFFE0 824C	EMCStatic WaitRd2	Selects the delay from chip select 2 to a read access.	-	0x1F	R/W
0xFFE0 8250	EMCStatic WaitPage2	Selects the delay for asynchronous page mode sequential accesses for chip select 2.	-	0x1F	R/W
0xFFE0 8254	EMCStatic WaitWr2	Selects the delay from chip select 2 to a write access.	-	0x1F	R/W
0xFFE0 8258	EMCStatic WaitTurn2	Selects the number of bus turnaround cycles for chip select 2.	-	0xF	R/W
0xFFE0 8260	EMCStatic Config3	Selects the memory configuration for static chip select 3.	-	0x0	R/W
0xFFE0 8264	EMCStatic WaitWen3	Selects the delay from chip select 3 to write enable.	-	0x0	R/W
0xFFE0 8268	EMCStatic WaitOen3	Selects the delay from chip select 3 or address change, whichever is later, to output enable.	-	0x0	R/W
0xFFE0 826C	EMCStatic WaitRd3	Selects the delay from chip select 3 to a read access.	-	0x1F	R/W
0xFFE0 8270	EMCStatic WaitPage3	Selects the delay for asynchronous page mode sequential accesses for chip select 3.	-	0x1F	R/W
0xFFE0 8274	EMCStatic WaitWr3	Selects the delay from chip select 3 to a write access.	-	0x1F	R/W
0xFFE0 8278	EMCStatic WaitTurn3	Selects the number of bus turnaround cycles for chip select 3.	-	0xF	R/W
0xFFE0 8880	EMCStatic ExtendedWait	Time long static memory read and write transfers.	-	0x0	R/W

10.1 EMC Control register (EMCControl - 0xFFE0 8000)

The EMCControl register is a read/write register that controls operation of the memory controller. The control bits can be altered during normal operation. Table 5–44 shows the bit assignments for the EMCControl register.

Table 44. EMC Control register (EMCControl - address 0xFFE0 8000) bit description

Bit	Symbol	Value	Description	Reset Value
0	EMC Enable (E)		Indicates if the EMC is enabled or disabled:	1
		0	Disabled	
		1	Enabled (POR and warm reset value).	
			Disabling the EMC reduces power consumption. When the memory controller is disabled the memory is not refreshed. The memory controller is enabled by setting the enable bit, or by reset.	
			This bit must only be modified when the EMC is in idle state.[1]	
1	Address mirror (M)		Indicates normal or reset memory map:	1
		0	Normal memory map.	
		1	Reset memory map. Static memory chip select 1 is mirrored onto chip select 0 and chip select 4 (POR reset value).	
			On POR, chip select 1 is mirrored to both chip select 0 and chip select 1 and chip select 4 memory areas. Clearing the M bit enables chip select 0 and chip select 4 memory to be accessed.	
2	Low-power mode (L)		Indicates normal, or low-power mode:	0
		0	Normal mode (warm reset value).	
		1	Low-power mode.	
			Entering low-power mode reduces memory controller power consumption. Dynamic memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit (L), or by POR.	
			This bit must only be modified when the EMC is in idle state.[1]	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] The external memory cannot be accessed in low-power or disabled state. If a memory access is performed an AHB error response is generated. The EMC registers can be programmed in low-power and/or disabled state.

10.2 EMC Status register (EMCStatus - 0xFFE0 8004)

The read-only EMCStatus register provides EMC status information. Table 5–45 shows the bit assignments for the EMCStatus register.

Table 45. EMC Status register (EMCStatus - address 0xFFE0 8008) bit description

Bit	Symbol	Value	Description	Reset Value
0	Busy (B)		This bit is used to ensure that the memory controller enters the low-power or disabled mode cleanly by determining if the memory controller is busy or not:	1
		0	EMC is idle (warm reset value).	
		1	EMC is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode (POR reset value).	
1	Write buffer status (S)		This bit enables the EMC to enter low-power mode or disabled mode cleanly:	0
		0	Write buffers empty (POR reset value)	
		1	Write buffers contain data.	
2	Self-refresh acknowledge (SA)		This bit indicates the operating mode of the EMC:	1
		0	Normal mode	
		1	Self-refresh mode (POR reset value).	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.3 EMC Configuration register (EMCConfig - 0xFFE0 8008)

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 5–46 shows the bit assignments for the EMCConfig register.

Table 46. EMC Configuration register (EMCConfig - address 0xFFE0 8008) bit description

Bit	Symbol	Value	Description	Reset Value
0			Endian mode:	0
		0	Little-endian mode (POR reset value).	
		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.	
7:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8			CCLK : CLKOUT[3:0] ratio:	0
		0	1:1 (POR reset value)	
		1	1:2 (this option is not available on the LPC2468)	
			This bit must contain 0 for proper operation of the EMC.	
31:9	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.4 Dynamic Memory Control register (EMCDynamicControl - 0xFFE0 8020)

The EMCDynamicControl register controls dynamic memory operation. The control bits can be altered during normal operation. Table 5–47 shows the bit assignments for the EMCDynamicControl register.

Table 47. Dynamic Conrol register (EMCDynamicControl - address 0xFFE0 8020) bit description

Bit	Symbol	Value	Description	Reset Value
0	Dynamic memory clock enable (CE)	0	Clock enable of idle devices are deasserted to save power (POR reset value).	0
		1	All clock enables are driven HIGH continuously.[1]	
1	Dynamic memory clock control (CS)	0	CLKOUT stops when all SDRAMs are idle and during self-refresh mode.	1
		1	CLKOUT runs continuously (POR reset value).	
			When clock control is LOW the output clock CLKOUT is stopped when there are no SDRAM transactions. The clock is also stopped during self-refresh mode.	
2	Self-refresh request, EMCSREFREQ (SR)	0	Normal mode.	1
		1	Enter self-refresh mode (POR reset value).	
			By writing 1 to this bit self-refresh can be entered under software control. Writing 0 to this bit returns the EMC to normal mode. The self-refresh acknowledge bit in the EMCStatus register must be polled to discover the current operating mode of the EMC.[2]	
4:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	Memory clock control (MMC)	0	CLKOUT enabled (POR reset value).	0
		1	CLKOUT disabled.[3]	
6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8:7	SDRAM initialization (I)	00	Issue SDRAM NORMAL operation command (POR reset value).	00
		01	Issue SDRAM MODE command.	
		10	Issue SDRAM PALL (precharge all) command.	
		11	Issue SDRAM NOP (no operation) command)	
12:9	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 47. Dynamic Conrol register (EMCDynamicControl - address 0xFFE0 8020) bit description

Bit	Symbol	Value	Description	Reset Value
13	Low-power SDRAM deep-sleep mode (DP)	0	Normal operation (POR reset value).	0
		1	Enter deep power down mode.	
31:14	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

[2] The memory controller exits from power-on reset with the self-refresh bit HIGH. To enter normal functional mode set this bit LOW.

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

10.5 Dynamic Memory Refresh Timer register (EMCDynamicRefresh - 0xFFE0 8024)

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 5–48 shows the bit assignments for the EMCDynamicRefresh register.

Table 48. Dynamic Memory Refresh Timer register (EMCDynamicRefresh - address 0xFFE0 8024) bit description

Bit	Symbol	Value	Description	Reset Value
10:0	Refresh timer (REFRESH)		Indicates the multiple of 16 CCLKs between SDRAM refresh cycles.	0
		0x0	Refresh disabled (POR reset value).	
		0x1	0x7FF = n x16 = 16n CCLKs between SDRAM refresh cycles.	
			For example: 0x1 = 1 x 16 = 16 CCLKs between SDRAM refresh cycles. 0x8 = 8 x 16 = 128 CCLKs between SDRAM refresh cycles.	
31:11	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

(16 x 10-6 x 50 x 106) / 16 = 50 or 0x32

If auto-refresh through warm reset is requested (by setting the EMC_Reset_Disable bit), the timing of auto-refresh must be adjusted to allow a sufficient refresh rate when the clock rate is reduced during the wakeup period of a reset cycle. During this period, the EMC (and all other portions of the LPC2468 that are being clocked) run from the IRC oscillator at 4 MHz. So, 4 MHz must be considered the CCLK rate for refresh calculations if auto-refresh through warm reset is requested.

Note: The refresh cycles are evenly distributed. However, there might be slight variations when the auto-refresh command is issued depending on the status of the memory controller.

10.6 Dynamic Memory Read Configuration register (EMCDynamicReadConfig - 0xFFE0 8028)

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.

Important: Especially it should be highlighted that the default clock delay methodology requires the output clock to be delayed **externally** to the chip to avoid hold time issue for the SDRAM. In most application boards, there will be no such external delay circuit and the application should write correct value to the EMCDynamicReadConfig register to use Command Delay Strategy. The Clock Delay Strategy is the default setting on reset!

Table 5–49 shows the bit assignments for the EMCDynamicReadConfig register.

Table 49. Dynamic Memory Read Configuration register (EMCDynamicReadConfig - address 0xFFE0 8028) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Read data strategy (RD)	00	Clock out delayed strategy, using CLKOUT (command not delayed, clock out delayed). POR reset value.	0x0
		01	Command delayed strategy, using EMCCLKDELAY (command delayed, clock out not delayed).	
		10	Command delayed strategy plus one clock cycle, using EMCCLKDELAY (command delayed, clock out not delayed).	
		11	Command delayed strategy plus two clock cycles, using EMCCLKDELAY (command delayed, clock out not delayed).	
31:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.7 Dynamic Memory Percentage Command Period register (EMCDynamictRP - 0xFFE0 8030)

The EMCDynamicTRP register enables you to program the precharge command period, tRP. This register must only be modified during system initialization. This value is normally found in SDRAM data sheets as tRP. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–50 shows the bit assignments for the EMCDynamicTRP register.

Table 50. Dynamic Memory Percentage Command Period register (EMCDynamictRP - address 0xFFE0 8030) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Precharge command period (tRP)	0x0 - 0xE	n + 1 clock cycles. The delay is in EMCCLK cycles.	0x0F
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.8 Dynamic Memory Active to Precharge Command Period register (EMCDynamictRAS - 0xFFE0 8034)

The EMCDynamicTRAS register enables you to program the active to precharge command period, tRAS. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRAS. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–51 shows the bit assignments for the EMCDynamicTRAS register.

Table 51. Dynamic Memory Active to Precharge Command Period register (EMCDynamictRAS - address 0xFFE0 8034) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Active to precharge command period (tRAS)	0x0 - 0xE	n + 1 clock cycles. The delay is in EMCCLK cycles.	0xF
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.9 Dynamic Memory Self-refresh Exit Time register (EMCDynamictSREX - 0xFFE0 8038)

The EMCDynamicTSREX register enables you to program the self-refresh exit time, tSREX. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tSREX, for devices without this parameter you use the same value as tXSR. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selectsmust be programmed.

Table 5–52 shows the bit assignments for the EMCDynamicTSREX register.

Table 52. Dynamic Memory Self-refresh Exit Time register (EMCDynamictSREX - address 0xFFE0 8038) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Self-refresh exit time (tSREX)	0x0 - 0xE	n + 1 clock cycles. The delay is in CCLK cycles.	0xF
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.10 Dynamic Memory Last Data Out to Active Time register (EMCDynamictAPR - 0xFFE0 803C)

The EMCDynamicTAPR register enables you to program the last-data-out to active command time, tAPR. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tAPR. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selectsmust be programmed.

Table 5–53 shows the bit assignments for the EMCDynamicTAPR register.

Table 53. Memory Last Data Out to Active Time register (EMCDynamictAPR - address 0xFFE0 803C) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Last-data-out to active command time (tAPR)	0x0 - 0xE	n + 1 clock cycles. The delay is in CCLK cycles.	0xF
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11 Dynamic Memory Data-in to Active Command Time register (EMCDynamictDAL - 0xFFE0 8040)

The EMCDynamicTDAL register enables you to program the data-in to active command time, tDAL. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tDAL, or tAPW. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–54 shows the bit assignments for the EMCDynamicTDAL register.

Table 54. Dynamic Memory Data-in to Active Command Time register (EMCDynamictDAL - address 0xFFE0 8040) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Data-in to active command (tDAL)	0x0 - 0xE	n clock cycles. The delay is in CCLK cycles.	0xF
		0xF	15 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.12 Dynamic Memory Write Recovery Time register (EMCDynamictWR - 0xFFE0 8044)

The EMCDynamicTWR register enables you to program the write recovery time, tWR. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tWR, tDPL, tRWL, or tRDL. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–55 shows the bit assignments for the EMCDynamicTWR register.

Table 55. Dynamic Memory Write recover Time register (EMCDynamictWR - address 0xFFE0 8044) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Write recovery time (tWR)	0x0 - 0xE	n + 1 clock cycles. The delay is in CCLK cycles.	0xF
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.13 Dynamic Memory Active to Active Command Period register (EMCDynamictRC - 0xFFE0 8048)

The EMCDynamicTRC register enables you to program the active to active command period, tRC. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRC. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–56 shows the bit assignments for the EMCDynamicTRC register.

Table 56. Dynamic Mempry Active to Active Command Period register (EMCDynamictRC - address 0xFFE0 8048) bit description

Bit	Symbol	Value	Description	Reset Value
4:0	Active to active command period (tRC)	0x0 - 0x1E	n + 1 clock cycles. The delay is in CCLK cycles.	0x1F
		0xF	32 clock cycles (POR reset value).	
31:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.14 Dynamic Memory Auto-refresh Period register (EMCDynamictRFC - 0xFFE0 804C)

The EMCDynamicTRFC register enables you to program the auto-refresh period, and auto-refresh to active command period, tRFC. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRFC, or sometimes as tRC. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–57 shows the bit assignments for the EMCDynamicTRFC register.

Table 57. Dynamic Memory Auto-refresh Period register (EMCDynamictRFC - address 0xFFE0 804C) bit description

Bit	Symbol	Value	Description	Reset Value
4:0	Auto-refresh period and auto-refresh to active command period (tRFC)	0x0 - 0x1E	n + 1 clock cycles. The delay is in CCLK cycles.	0x1F
		0xF	32 clock cycles (POR reset value).	
31:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.15 Dynamic Memory Exit Self-refresh register (EMCDynamictXSR - 0xFFE0 8050)

The EMCDynamicTXSR register enables you to program the exit self-refresh to active command time, tXSR. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tXSR. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–58 shows the bit assignments for the EMCDynamicTXSR register.

Table 58. Dynamic Memory Exit Self-refresh register (EMCDynamictXSR - address 0xFFE0 8050) bit description

Bit	Symbol	Value	Description	Reset Value
4:0	Exit self-refresh to active command time (tXSR)	0x0 - 0x1E	n + 1 clock cycles. The delay is in CCLK cycles.	0x1F
		0xF	32 clock cycles (POR reset value).	
31:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.16 Dynamic Memory Active Bank A to Active Bank B Time register (EMCDynamictRRD - 0xFFE0 8054)

The EMCDynamicTRRD register enables you to program the active bank A to active bank B latency, tRRD. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tRRD. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selects must be programmed.

Table 5–59 shows the bit assignments for the EMCDynamicTRRD register.

Table 59. Dynamic Memory Acitve Bank A to Active Bank B Time register (EMCDynamictRRD - address 0xFFE0 8054) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Active bank A to active bank B latency (tRRD)	0x0 - 0xE	n + 1 clock cycles. The delay is in CCLK cycles.	0xF
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.17 Dynamic Memory Load Mode register to Active Command Time (EMCDynamictMRD - 0xFFE0 8058)

The EMCDynamicTMRD register enables you to program the load mode register to active command time, tMRD. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tMRD, or tRSA. This register is accessed with one wait state.

Note: This register is used for all four dynamic memory chip selects. Therefore the worst case value for all of the chip selectsmust be programmed.

Table 5–60 shows the bit assignments for the EMCDynamicTMRD register.

Table 60. Dynamic Memory Load Mode register to Active Command Time (EMCDynamictMRD - address 0xFFE0 8058) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Load mode register to active command time (tMRD)	0x0 - 0xE	n + 1 clock cycles. The delay is in CCLK cycles.	0xF
		0xF	16 clock cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.18 Dynamic Memory Configuration registers (EMCDynamicConfig0-3 - 0xFFE0 8100, 120, 140, 160)

The EMCDynamicConfig0-3 registers enable you to program the configuration information for the relevant dynamic memory chip select. These registers are normally only modified during system initialization. These registers are accessed with one wait state.

Table 5–61 shows the bit assignments for the EMCDynamicConfig0-3 registers.

Table 61. Dynamic Memory Configuration registers (EMCDynamicConfig0-3 - address 0xFFE0 8100, 0xFFE0 8120, 0xFFE0 8140, 0xFFE0 8160) bit description

Bit	Symbol	Value	Description	Reset Value
2:0	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4:3	Memory device (MD)	00	SDRAM (POR reset value).	00
		01	Low-power SDRAM.	
		10	Micron SyncFlash.	
		11	Reserved.	
6:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
12:7	Address mapping (AM)		See Table 5–62	0
		0	000000 = reset value.[1]	

Table 61. Dynamic Memory Configuration registers (EMCDynamicConfig0-3 - address 0xFFE0 8100, 0xFFE0 8120, 0xFFE0 8140, 0xFFE0 8160) bit description

Bit	Symbol	Value	Description	Reset Value
13	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
14	Address mapping (AM)		See Table 5–62	0
		0	0 = reset value.	
18:15	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19	Buffer enable (B)	0	Buffer disabled for accesses to this chip select (POR reset value).	
		1	Buffer enabled for accesses to this chip select.[2]	
20	Write protect (P)	0	Writes not protected (POR reset value).	0
		1	Writes protected.	
31:21	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

[2] The buffers must be disabled during SDRAM and SyncFlash initialization. They must also be disabled when performing SyncFlash commands. The buffers must be enabled during normal operation.

Address mappings that are not shown in Table 5–62 are reserved.

Table 62. Address mapping

14	12	11:9	18:7	Description
16 bit external bus high-performance address mapping (Row, Bank, Column)				
0	0	000	00	16 MB (2Mx8), 2 banks, row length = 11, column length = 9
0	0	000	01	16 MB (1Mx16), 2 banks, row length = 11, column length = 8
0	0	001	00	64 MB (8Mx8), 4 banks, row length = 12, column length = 9
0	0	001	01	64 MB (4Mx16), 4 banks, row length = 12, column length = 8
0	0	010	00	128 MB (16Mx8), 4 banks, row length = 12, column length = 10
0	0	010	01	128 MB (8Mx16), 4 banks, row length = 12, column length = 9
0	0	011	00	256 MB (32Mx8), 4 banks, row length = 13, column length = 10
0	0	011	01	256 MB (16Mx16), 4 banks, row length = 13, column length = 9
0	0	100	00	512 MB (64Mx8), 4 banks, row length = 13, column length = 11
0	0	100	01	512 MB (32Mx16), 4 banks, row length = 13, column length = 10
16 bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
0	1	000	00	16 MB (2Mx8), 2 banks, row length = 11, column length = 9
0	1	000	01	16 MB (1Mx16), 2 banks, row length = 11, column length = 8
0	1	001	00	64 MB (8Mx8), 4 banks, row length = 12, column length = 9
0	1	001	01	64 MB (4Mx16), 4 banks, row length = 12, column length = 8
0	1	010	00	128 MB (16Mx8), 4 banks, row length = 12, column length = 10
0	1	010	01	128 MB (8Mx16), 4 banks, row length = 12, column length = 9

Table 62. Address mapping

14	12	11:9	18:7	Description
0	1	011	00	256 MB (32Mx8), 4 banks, row length = 13, column length = 10
0	1	011	01	256 MB (16Mx16), 4 banks, row length = 13, column length = 9
0	1	100	00	512 MB (64Mx8), 4 banks, row length = 13, column length = 11
0	1	100	01	512 MB (32Mx16), 4 banks, row length = 13, column length = 10
32 bit external bus high-performance address mapping (Row, Bank, Column)				
1	0	000	00	16 MB (2Mx8), 2 banks, row length = 11, column length = 9
1	0	000	01	16 MB (1Mx16), 2 banks, row length = 11, column length = 8
1	0	001	00	64 MB (8Mx8), 4 banks, row length = 12, column length = 9
1	0	001	01	64 MB (4Mx16), 4 banks, row length = 12, column length = 8
1	0	001	10	64 MB (2Mx32), 4 banks, row length = 11, column length = 8
1	0	010	00	128 MB (16Mx8), 4 banks, row length = 12, column length = 10
1	0	010	01	128 MB (8Mx16), 4 banks, row length = 12, column length = 9
1	0	010	10	128 MB (4Mx32), 4 banks, row length = 12, column length = 8
1	0	011	00	256 MB (32Mx8), 4 banks, row length = 13, column length = 10
1	0	011	01	256 MB (16Mx16), 4 banks, row length = 13, column length = 9
1	0	011	10	256 MB (8Mx32), 4 banks, row length = 13, column length = 8
1	0	100	00	512 MB (64Mx8), 4 banks, row length = 13, column length = 11
1	0	100	01	512 MB (32Mx16), 4 banks, row length = 13, column length = 10
32 bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
1	1	000	00	16 MB (2Mx8), 2 banks, row length = 11, column length = 9
1	1	000	01	16 MB (1Mx16), 2 banks, row length = 11, column length = 8
1	1	001	00	64 MB (8Mx8), 4 banks, row length = 12, column length = 9
1	1	001	01	64 MB (4Mx16), 4 banks, row length = 12, column length = 8
1	1	001	10	64 MB (2Mx32), 4 banks, row length = 11, column length = 8
1	1	010	00	128 MB (16Mx8), 4 banks, row length = 12, column length = 10
1	1	010	01	128 MB (8Mx16), 4 banks, row length = 12, column length = 9
1	1	010	10	128 MB (4Mx32), 4 banks, row length = 12, column length = 8
1	1	011	00	256 MB (32Mx8), 4 banks, row length = 13, column length = 10
1	1	011	01	256 MB (16Mx16), 4 banks, row length = 13, column length = 9
1	1	011	10	256 MB (8Mx32), 4 banks, row length = 13, column length = 8
1	1	100	00	512 MB (64Mx8), 4 banks, row length = 13, column length = 11
1	1	100	01	512 MB (32Mx16), 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.
- Four x 8 bit wide memory devices, choose a 32 bit wide address mapping.

- Two x 8 bit wide memory devices, choose a 16 bit wide address mapping.

10.19 Dynamic Memory RAS & CAS Delay registers (EMCDynamicRASCAS0-3 - 0xFFE0 8104, 124, 144, 164)

The EMCDynamicRasCas0-3 registers enable you to program the RAS and CAS latencies for the relevant dynamic memory. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

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

Table 5–63 shows the bit assignments for the EMCDynamicRasCas0-3 registers.

Table 63. Dynamic Memory RAS & CAS Delay registers (EMCDynamicRasCas0-3 - address 0xFFE0 8104, 0xFFE0 8124, 0xFFE0 8144, 0xFFE0 8164) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	RAS latency (active to read/write delay) (RAS)	00	Reserved.	11
		01	One CCLK cycle.	
		10	Two CCLK cycles.	
		11	Three CCLK cycles (POR reset value).	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9:8	CAS latency (CAS)	00	Reserved.	11
		01	One CCLK cycle.	
		10	Two CCLK cycles.	
		11	Three CCLK cycles (POR reset value).	
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.20 Static Memory Configuration registers (EMCStaticConfig0-3 - 0xFFE0 8200, 220, 240, 260)

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 5–64 shows the bit assignments for the EMCStaticConfig0-3 registers. Note that synchronous burst mode memory devices are not supported.

Table 64. Static Memory Configuration registers (EMCStaticConfig0-3 - address 0xFFE0 8200, 0xFFE0 8220, 0xFFE0 8240, 0xFFE0 8260) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Memory width (MW)	00	8 bit (POR reset value).	0
		01	16 bit.	
		10	32 bit.	
		11	Reserved.	
2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	Page mode (PM)		In page mode the EMC can burst up to four external accesses. Therefore devices with asynchronous page mode burst four or higher devices are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.	0
		0	Disabled (POR reset value).	
		1	Async page mode enabled (page length four).	
5:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
6	Chip select polarity (PC)		The value of the chip select polarity on power-on reset is 0.	0
		0	Active LOW chip select.	
		1	Active HIGH chip select.	
7	Byte lane state (PB)		The byte lane state bit, PB, enables different types of memory to be connected. For byte-wide static memories the BLSn[3:0] signal from the EMC is usually connected to WE (write enable). In this case for reads all the BLSn[3:0] bits must be HIGH. This means that the byte lane state (PB) bit must be LOW. 16 bit wide static memory devices usually have the BLSn[3:0] signals connected to the UBn and LBn (upper byte and lower byte) signals in the static memory. In this case a write to a particular byte must assert the appropriate UBn or LBn signal LOW. For reads, all the UB and LB signals must be asserted LOW so that the bus is driven. In this case the byte lane state (PB) bit must be HIGH.	0
		0	For reads all the bits in BLSn[3:0] are HIGH. For writes the respective active bits in BLSn[3:0] are LOW (POR reset value).	
		1	For reads the respective active bits in BLSn[3:0] are LOW. For writes the respective active bits in BLSn[3:0] are LOW.	
8	Extended wait (EW)		Extended wait (EW) uses the EMCStaticExtendedWait register to time both the read and write transfers rather than the EMCStaticWaitRd and EMCStaticWaitWr registers. This enables much longer transactions.[1]	0
		0	Extended wait disabled (POR reset value).	
		1	Extended wait enabled.	

Table 64. Static Memory Configuration registers (EMCStaticConfig0-3 - address 0xFFE0 8200, 0xFFE0 8220, 0xFFE0 8240, 0xFFE0 8260) bit description

Bit	Symbol	Value	Description	Reset Value
18:9	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19	Buffer enable (B)	00	Write buffer disabled (POR reset value).	0
		01	Write buffer enabled.	
20	Write protect (P)	00	Writes not protected (POR reset value).	0
		01	Write protected.	
31:21	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Extended wait and page mode cannot be selected simultaneously.

10.21 Static Memory Write Enable Delay registers (EMCStaticWaitWen0-3 - 0xFFE0 8204, 224, 244 ,264)

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 5–65 shows the bit assignments for the EMCStaticWaitWen0-3 registers.

Table 65. Static Memory Write Enable Delay registers (EMCStaticWaitWen0-3 - address 0xFFE0 8204,0xFFE0 8224, 0xFFE0 8244, 0xFFE0 8264) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Wait write enable (WAITWEN)		Delay from chip select assertion to write enable.	0x0
		0x0	One CCLK cycle delay between assertion of chip select and write enable (POR reset value).	
		0x1 - 0xF	(n + 1) CCLK cycle delay. The delay is (WAITWEN +1) x tCCLK.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.22 Static Memory Output Enable Delay registers (EMCStaticWaitOen0-3 - 0xFFE0 8208, 228, 248, 268)

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 5–66 shows the bit assignments for the EMCStaticWaitOen0-3 registers.

Table 66. Static Memory Output Enable delay registers (EMCStaticWaitOen03 - address 0xFFE0 8208, 0xFFE0 8228, 0xFFE0 8248, 0xFFE0 8268) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Wait output enable (WAITOEN)		Delay from chip select assertion to output enable.	0x0
		0x0	No delay (POR reset value).	
		0x1 - 0xF	n cycle delay. The delay is WAITOEN x tCCLK.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.23 Static Memory Read Delay registers (EMCStaticWaitRd0-3 - 0xFFE0 820C, 22C, 24C, 26C)

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 5–67 shows the bit assignments for the EMCStaticWaitRd0-3 registers.

Table 67. Static Memory Read Delay registers (EMCStaticWaitRd0-3 - address 0xFFE0 820C, 0xFFE0 822C, 0xFFE0 824C, 0xFFE0 826C) bit description

Bit	Symbol	Value	Description	Reset Value
4:0	Non-page mode read wait states or asynchronous page mode readfirst access wait state (WAITRD)		Non-page mode read or asynchronous page mode read, first read only:	0x1F
		0x0 - 0x1E	(n + 1) CCLK cycles for read accesses. For non-sequential reads, the wait state time is (WAITRD + 1) x tCCLK.	
		0x1F	32 CCLK cycles for read accesses (POR reset value).	
31:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.24 Static Memory Page Mode Read Delay registers (EMCStaticwaitPage0-3 - 0xFFE0 8210, 230, 250, 270)

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 5–68 shows the bit assignments for the EMCStaticWaitPage0-3 registers.

Table 68. Static Memory Page Mode Read Delay registers0-3 (EMCStaticWaitPage0-3 - address 0xFFE0 8210, 0xFFE0 8230, 0xFFE0 8250, 0xFFE0 8270) bit description

Bit	Symbol	Value	Description	Reset Value
4:0	Asynchronous page mode read after the first read wait states (WAITPAGE)		Number of wait states for asynchronous page mode read accesses after the first read:	0x1F
		0x0 - 0x1E	(n+ 1) CCLK cycle read access time. For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITPAGE + 1) x tCCLK.	
		0x1F	32 CCLK cycle read access time (POR reset value).	
31:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.25 Static Memory Write Delay registers (EMCStaticWaitwr0-3 - 0xFFE0 8214, 234, 254, 274)

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 5–69 shows the bit assignments for the EMCStaticWaitWr0-3 registers.

Table 69. Static Memory Write Delay registers0-3 (EMCStaticWaitWr - address 0xFFE0 8214, 0xFFE0 8234, 0xFFE0 8254, 0xFFE0 8274) bit description

Bit	Symbol	Value	Description	Reset Value
4:0	Write wait states (WAITWR)		SRAM wait state time for write accesses after the first read:	0x1F
		0x0 - 0x1E	(n + 2) CCLK cycle write access time. The wait state time for write accesses after the first read is WAITWR (n + 2) x tCCLK.	
		0x1F	33 CCLK cycle write access time (POR reset value).	
31:5	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.26 Static Memory Extended Wait register (EMCStaticExtendedWait - 0xFFE0 8880)

ExtendedWait (EW) bit in the EMCStaticConfig register is set. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. However, if necessary, these control bits can be altered during normal operation. This register is accessed with one wait state.

Table 5–70 shows the bit assignments for the EMCStaticExtendedWait0-3 registers.

Table 70. Static Memory Extended Wait register (EMCStaticExtendedWait - address 0xFFE0 8880) bit description

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

For example, for a static memory read/write transfer time of 16 µs, and a CCLK frequency of 50 MHz, the following value must be programmed into this register: (16 x 10-6 x 50 x 106) / 16 - 1 = 49

10.27 Static Memory Turn Round Delay registers (EMCStaticWaitTurn0-3 - 0xFFE0 8218, 238, 258, 278)

The EMCStaticWaitTurn0-3 registers enable you to program the number of bus turnaround cycles. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the EMC is idle, and then entering low-power, or disabled mode. These registers are accessed with one wait state.

Table 5–71 shows the bit assignments for the EMCStaticWaitTurn0-3 registers.

Table 71. Static Memory Trun Round Delay registers0-3 (EMCStaticWaitTurn0-3 - address 0xFFE0 8218, 0xFFE0 8238, 0xFFE0 8258, 0xFFE0 8278) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	Bus turnaround cycles (WAITTURN)	0x0 - 0xE	(n + 1) CCLK turnaround cycles. Bus turnaround time is (WAITTURN + 1) x tCCLK.	0xF
		0xF	16 CCLK turnaround cycles (POR reset value).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

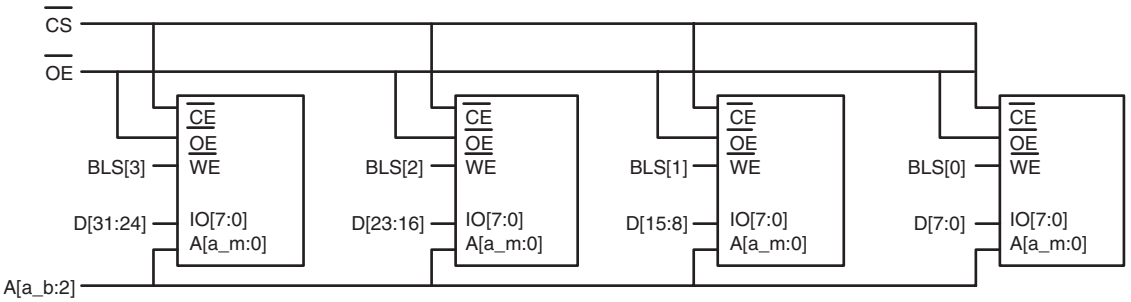
To prevent bus contention on the external memory data bus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses. The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses.

11. External memory interface

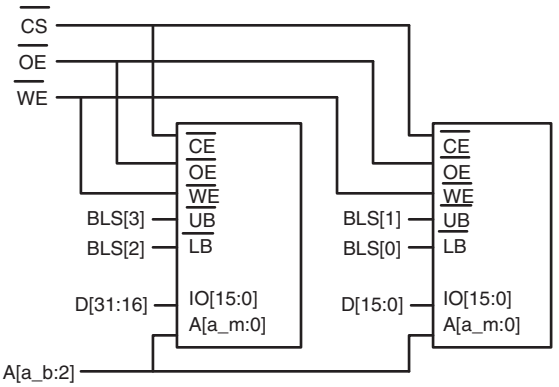
External memory interfacing depends on the bank width (32, 16 or 8 bit selected via MW bits in corresponding EMCStaticConfig register).

If a memory bank is configured to be 32 bits wide, address lines A0 and A1 can be used as non-address lines. If a memory bank is configured to 16 bits wide, A0 is not required. However, 8 bit wide memory banks do require all address lines down to A0. Configuring A1 and/or A0 line(s) to provide address or non-address function is accomplished using the Pin Function Select Register (see Section 9–4).

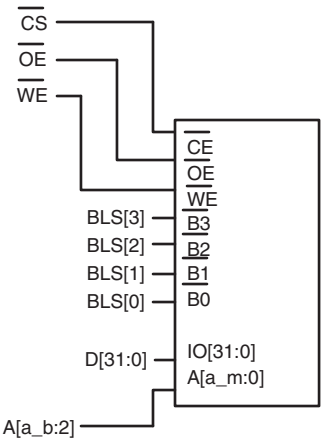
Symbol "a_b" in the following figures refers to the highest order address line in the data bus. Symbol "a_m" refers to the highest order address line of the memory chip used in the external memory interface.



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

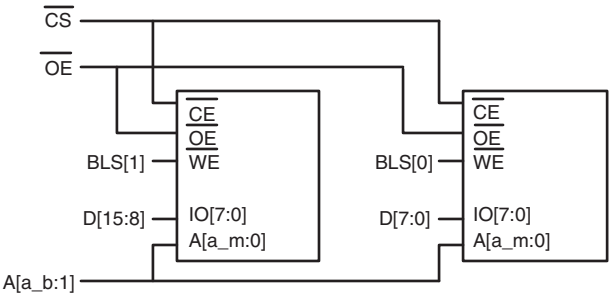


b. 32 bit wide memory bank interfaced to two 16 bit memory chips

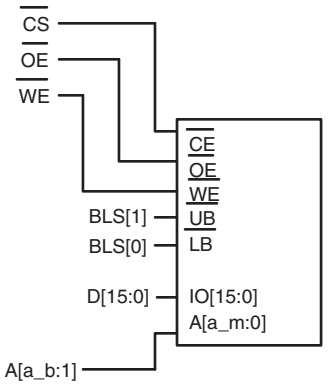


c. 32 bit wide memory bank interfaced to one 8 bit memory chip

Fig 11. 32 bit bank external memory interfaces (bits MW = 10)



a. 16 bit wide memory bank interfaced to two 8 bit memory chips



b. 16 bit wide memory bank interfaced to a 16 bit memory chip

Fig 12. 16 bit bank external memory interfaces (bits MW = 01)

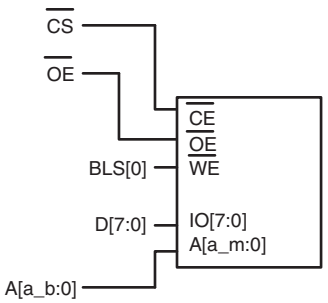


Fig 13. 8 bit bank external memory interface (bits MW = 00)

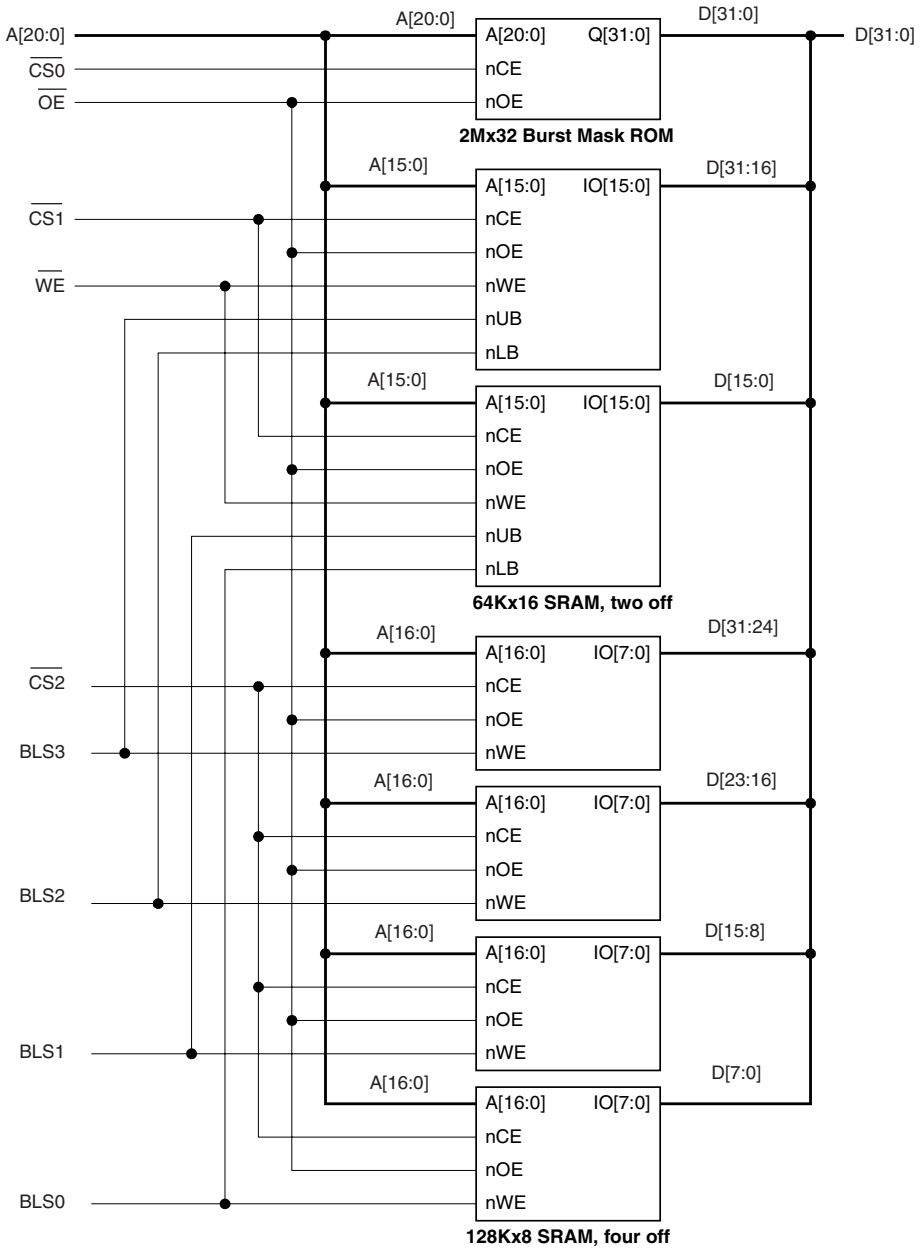


Fig 14. Typical memory configuration diagram

1. Introduction

The MAM block in the LPC2400 maximizes the performance of the ARM processor when it is running code in Flash memory using a single Flash bank.

2. Operation

Simply put, the Memory Accelerator Module (MAM) attempts to have the next ARM instruction that will be needed in its latches in time to prevent CPU fetch stalls. The LPC2400 uses one bank of Flash memory, compared to the two banks used on predecessor devices. It includes three 128 bit buffers called the Prefetch buffer, the Branch Trail Buffer and the data buffer. When an Instruction Fetch is not satisfied by either the Prefetch or Branch Trail buffer, nor has a prefetch been initiated for that line, the ARM is stalled while a fetch is initiated for the 128 bit line. If a prefetch has been initiated but not yet completed, the ARM is stalled for a shorter time. Unless aborted by a data access, a prefetch is initiated as soon as the Flash has completed the previous access. The prefetched line is latched by the Flash module, but the MAM does not capture the line in its prefetch buffer until the ARM core presents the address from which the prefetch has been made. If the core presents a different address from the one from which the prefetch has been made, the prefetched line is discarded.

The prefetch and Branch Trail buffers each include four 32 bit ARM instructions or eight 16 bit Thumb instructions. During sequential code execution, typically the prefetch buffer contains the current instruction and the entire Flash line that contains it.

The MAM uses the LPROT[0] line to differentiate between instruction and data accesses. Code and data accesses use separate 128 bit buffers. 3 of every 4 sequential 32 bit code or data accesses "hit" in the buffer without requiring a Flash access (7 of 8 sequential 16 bit accesses, 15 of every 16 sequential byte accesses). The fourth (eighth, 16th) sequential data access must access Flash, aborting any prefetch in progress. When a Flash data access is concluded, any prefetch that had been in progress is re-initiated.

Timing of Flash read operations is programmable and is described later in this section.

In this manner, there is no code fetch penalty for sequential instruction execution when the CPU clock period is greater than or equal to one fourth of the Flash access time. The average amount of time spent doing program branches is relatively small (less than 25%) and may be minimized in ARM (rather than Thumb) code through the use of the conditional execution feature present in all ARM instructions. This conditional execution may often be used to avoid small forward branches that would otherwise be necessary.

Branches and other program flow changes cause a break in the sequential flow of instruction fetches described above. The Branch Trail buffer captures the line to which such a non-sequential break occurs. If the same branch is taken again, the next instruction is taken from the Branch Trail buffer. When a branch outside the contents of the prefetch and Branch Trail buffer is taken, a stall of several clocks is needed to load the Branch Trail buffer. Subsequently, there will typically be no further instruction fetch delays until a new and different branch occurs.

If an attempt is made to write directly to the Flash memory, without using the normal Flash programming interface, the MAM generates a data abort.

3. Memory Acceleration Module blocks

The Memory Accelerator Module is divided into several functional blocks:

- A Flash Address Latch and an incrementor function to form prefetch addresses
- A 128 bit prefetch buffer and an associated Address latch and comparator
- A 128 bit Branch Trail buffer and an associated Address latch and comparator
- A 128 bit Data buffer and an associated Address latch and comparator
- Control logic
- Wait logic

Figure 6–15 shows a simplified block diagram of the Memory Accelerator Module data paths.

In the following descriptions, the term “fetch” applies to an explicit Flash read request from the ARM. “Pre-fetch” is used to denote a Flash read of instructions beyond the current processor fetch address.

3.1 Flash memory bank

There is one bank of Flash memory for the LPC2400 MAM.

Flash programming operations are not controlled by the MAM, but are handled as a separate function. A “boot block” sector contains Flash programming algorithms that may be called as part of the application program, and a loader that may be run to allow serial programming of the Flash memory.

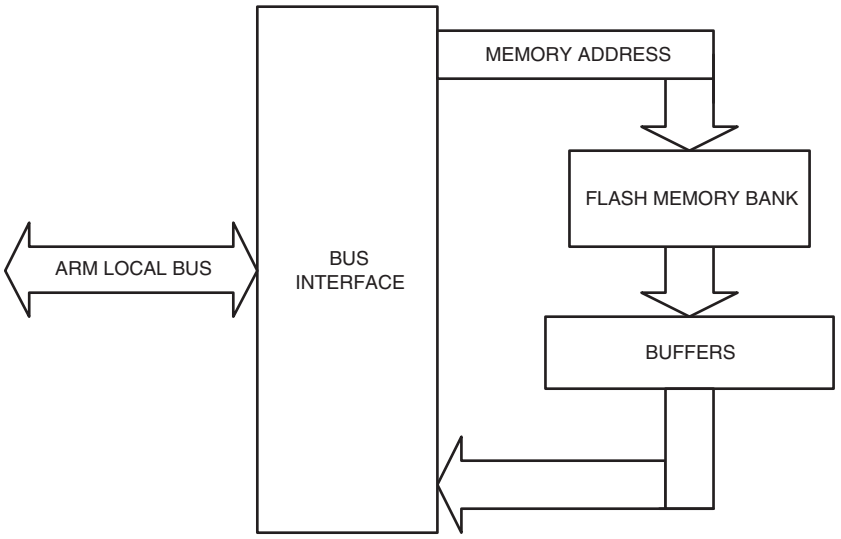


Fig 15. Simplified block diagram of the Memory Accelerator Module

3.2 Instruction latches and data latches

Code and Data accesses are treated separately by the Memory Accelerator Module. There is a 128 bit Latch, a 15 bit Address Latch, and a 15 bit comparator associated with each buffer (prefetch, branch trail, and data). Each 128 bit latch holds 4 words (4 ARM instructions, or 8 Thumb instructions).

Also associated with each buffer are 32 4:1 Multiplexers that select the requested word from the 128 bit line.

3.3 Flash programming Issues

Since the Flash memory does not allow accesses during programming and erase operations, it is necessary for the MAM to force the CPU to wait if a memory access to a Flash address is requested while the Flash module is busy. (This is accomplished by asserting the ARM7TDMI-S local bus signal CLKEN.) Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to ensure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the Flash memory.

In order to preclude the possibility of stale data being read from the Flash memory, the LPC2400 MAM holding latches are automatically invalidated at the beginning of any Flash programming or erase operation. Any subsequent read from a Flash address will cause a new fetch to be initiated after the Flash operation has completed.

4. Memory Accelerator Module operating modes

Three modes of operation are defined for the MAM, trading off performance for ease of predictability:

Mode 0: MAM off. All memory requests result in a Flash read operation (see note 2 below). There are no instruction prefetches.

Mode 1: MAM partially enabled. Sequential instruction accesses are fulfilled from the holding latches if the data is present. Instruction prefetch is enabled. Non-sequential instruction accesses initiate Flash read operations (see Table note 6–2). This means that all branches cause memory fetches. All data operations cause a Flash read because buffered data access timing is hard to predict and is very situation dependent.

Mode 2: MAM fully enabled. Any memory request (code or data) for a value that is contained in one of the corresponding holding latches is fulfilled from the latch. Instruction prefetch is enabled. Flash read operations are initiated for instruction prefetch and code or data values not available in the corresponding holding latches.

Table 72. MAM responses to program accesses of various types

Program Memory Request Type	MAM Mode		
	0	1	2
Sequential access, data in latches	Initiate Fetch[2]	Use Latched Data[1]	Use Latched Data[1]

Table 72. MAM responses to program accesses of various types

Program Memory Request Type	MAM Mode		
	0	1	2
Sequential access, data not in latches	Initiate Fetch	Initiate Fetch[1]	Initiate Fetch[1]
Non-sequential access, data in latches	Initiate Fetch[2]	Initiate Fetch[1][2]	Use Latched Data[1]
Non-sequential access, data not in latches	Initiate Fetch	Initiate Fetch[1]	Initiate Fetch[1]

[1] Instruction prefetch is enabled in modes 1 and 2.

[2] The MAM actually uses latched data if it is available, but mimics the timing of a Flash read operation. This saves power while resulting in the same execution timing. The MAM can truly be turned off by setting the fetch timing value in MAMTIM to one clock.

Table 73. MAM responses to data and DMA accesses of various types

Data Memory Request Type	MAM Mode		
	0	1	2
Sequential access, data in latches	Initiate Fetch[1]	Initiate Fetch[1]	Use Latched Data
Sequential access, data not in latches	Initiate Fetch	Initiate Fetch	Initiate Fetch
Non-sequential access, data in latches	Initiate Fetch[1]	Initiate Fetch[1]	Use Latched Data
Non-sequential access, data not in latches	Initiate Fetch	Initiate Fetch	Initiate Fetch

[1] The MAM actually uses latched data if it is available, but mimics the timing of a Flash read operation. This saves power while resulting in the same execution timing. The MAM can truly be turned off by setting the fetch timing value in MAMTIM to one clock.

5. MAM configuration

After reset the MAM defaults to the disabled state. Software can turn memory access acceleration on or off at any time. This allows most of an application to be run at the highest possible performance, while certain functions can be run at a somewhat slower but more predictable rate if more precise timing is required.

6. Register description

The MAM is controlled by the registers shown in Table 6–74. More detailed descriptions follow. Writes to any unused bits are ignored. A read of any unused bits will return a logic zero.

Table 74. Summary of Memory Acceleration Module registers

Name	Description	Access	Reset value[1]	Address
MAMCR	Memory Accelerator Module Control Register. Determines the MAM functional mode, that is, to what extent the MAM performance enhancements are enabled. See Table 6–75.	R/W	0x0	0xE01F C000
MAMTIM	Memory Accelerator Module Timing control. Determines the number of clocks used for Flash memory fetches (1 to 7 processor clocks).	R/W	0x07	0xE01F C004

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

7. MAM Control Register (MAMCR - 0xE01F C000)

Two configuration bits select the three MAM operating modes, as shown in Table 6–75.

Following any reset, MAM functions are disabled. Software can turn memory access acceleration on or off at any time allowing most of an application to be run at the highest possible performance, while certain functions can be run at a somewhat slower but more predictable rate if more precise timing is required.

Changing the MAM operating mode causes the MAM to invalidate all of the holding latches, resulting in new reads of Flash information as required. This guarantees synchronization of the MAM to CPU operation.

Table 75. MAM Control Register (MAMCR - address 0xE01F C000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MAM_mode _control		These bits determine the operating mode of the MAM.	0
		00	MAM functions disabled	
		01	MAM functions partially enabled	
		10	MAM functions fully enabled	
		11	Reserved. Not to be used in the application.	
7:2	-	-	Unused, always 0.	0

8. MAM Timing Register (MAMTIM - 0xE01F C004)

The MAM Timing register determines how many CCLK cycles are used to access the Flash memory. This allows tuning MAM timing to match the processor operating frequency. Flash access times from 1 clock to 7 clocks are possible. Single clock Flash accesses would essentially remove the MAM from timing calculations. In this case the MAM mode may be selected to optimize power usage.

Table 76. MAM Timing register (MAMTIM - address 0xE01F C004) bit description

Bit	Symbol	Value	Description	Reset value
2:0	MAM_fetch_cycle_timing		These bits set the duration of MAM fetch operations.	07
		000	0 - Reserved	
		001	1 - MAM fetch cycles are 1 processor clock (CCLK) in duration	
		010	2 - MAM fetch cycles are 2 CCLKs in duration	
		011	3 - MAM fetch cycles are 3 CCLKs in duration	
		100	4 - MAM fetch cycles are 4 CCLKs in duration	
		101	5 - MAM fetch cycles are 5 CCLKs in duration	
		110	6 - MAM fetch cycles are 6 CCLKs in duration	
		111	7 - MAM fetch cycles are 7 CCLKs in duration	
			Warning: These bits set the duration of MAM Flash fetch operations as listed here. Improper setting of this value may result in incorrect operation of the device.	
7:3	-	-	Unused, always 0	0

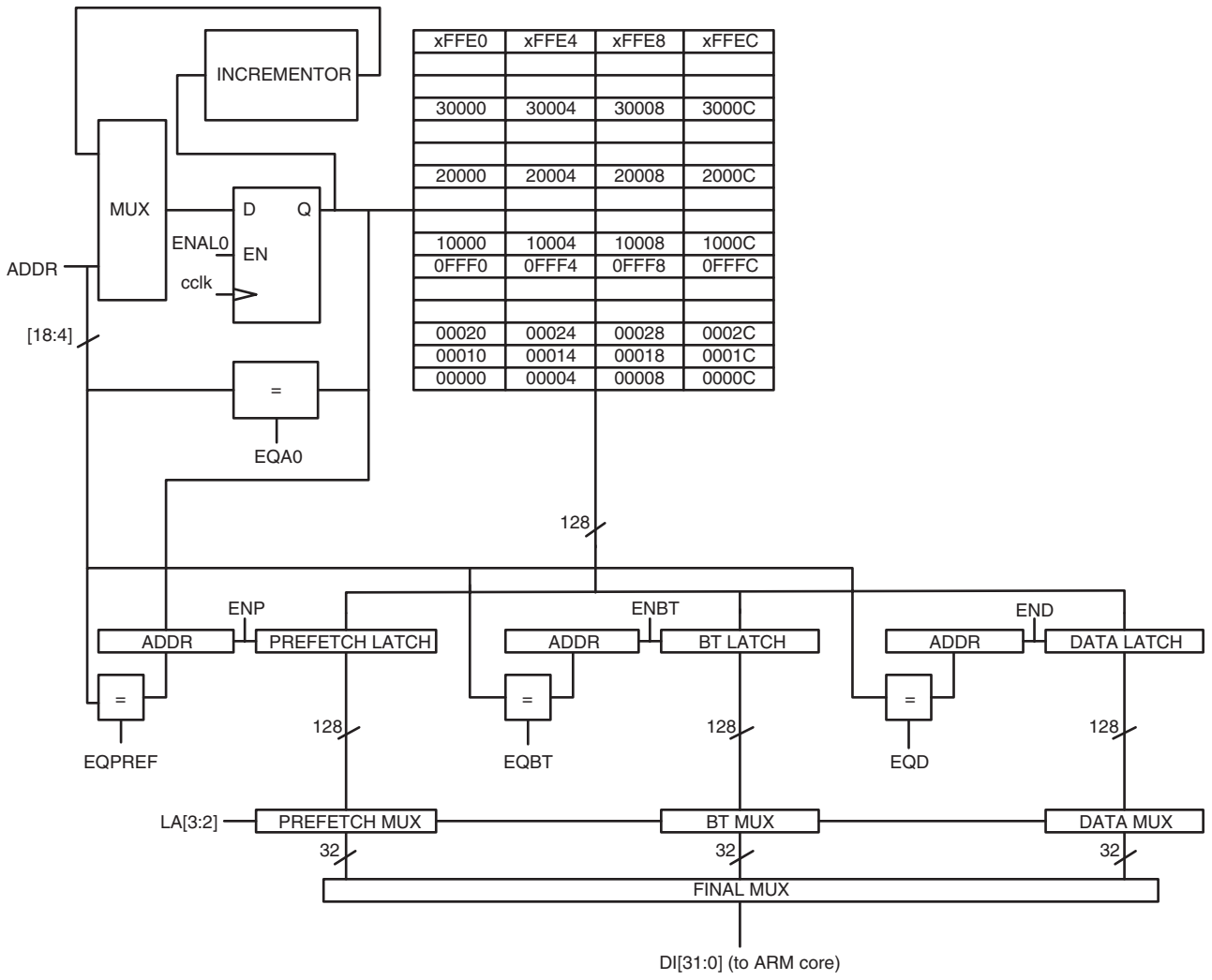


Fig 16. Block diagram of the Memory Accelerator Module

9. MAM usage notes

When changing MAM timing, the MAM must first be turned off by writing a zero to MAMCR. A new value may then be written to MAMTIM. Finally, the MAM may be turned on again by writing a value (1 or 2) corresponding to the desired operating mode to MAMCR.

For system clock slower than 20 MHz, MAMTIM can be 001. For system clock between 20 MHz and 40 MHz, Flash access time is suggested to be 2 CCLKs, while in systems with system clock faster than 40 MHz, 3 CCLKs are proposed.

1. Features

- ARM PrimeCell Vectored Interrupt Controller
- Mapped to AHB address space for fast access
- Supports 32 vectored IRQ interrupts
- 16 programmable interrupt priority levels
- Fixed hardware priority within each programmable priority level
- Hardware priority level masking
- Any input can be assigned as an FIQ interrupt
- Software interrupt generation

2. Description

The ARM processor core has two interrupt inputs called Interrupt Request (IRQ) and Fast Interrupt reQuest (FIQ). The Vectored Interrupt Controller (VIC) takes 32 interrupt request inputs and programmably assigns them as FIQ or vectored IRQ types. The programmable assignment scheme means that priorities of interrupts from the various peripherals can be dynamically assigned and adjusted.

Fast Interrupt reQuest (FIQ) requests have the highest priority. If more than one request is assigned to FIQ, the VIC ORs the requests to produce the FIQ signal to the ARM processor. The fastest possible FIQ latency is achieved when only one request is classified as FIQ, because then the FIQ service routine can simply start dealing with that device. But if more than one request is assigned to the FIQ class, the FIQ service routine can read a word from the VIC that identifies which FIQ source(s) is (are) requesting an interrupt.

Vectored IRQ's, which include all interrupt requests that are not classified as FIQs, have a programmable interrupt priority. When more than one interrupt is assigned the same priority and occur simultaneously, the one connected to the lowest numbered VIC channel (see Table 7–91 on page 95) will be serviced first.

The VIC ORs the requests from all of the vectored IRQs to produce the IRQ signal to the ARM processor. The IRQ service routine can start by reading a register from the VIC and jumping to the address supplied by that register.

3. Register description

The VIC implements the registers shown in Table 7–77. More detailed descriptions follow.

Table 77. VIC register map

Name	Description	Access	Reset value[1]	Address
VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.	RO	0	0xFFFF F000
VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ.	RO	0	0xFFFF F004
VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	-	0xFFFF F008
VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0	0xFFFF F00C
VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ.	R/W	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	WO	-	0xFFFF F014
VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0	0xFFFF F018
VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register.	WO	-	0xFFFF F01C
VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode.	R/W	0	0xFFFF F020
VICSWPriorityMask	Software Priority Mask Register. Allows masking individual interrupt priority levels in any combination.	R/W	0xFFFF	0xFFFF F024
VICVectAddr0	Vector address 0 register. Vector Address Registers 0-31 hold the addresses of the Interrupt Service routines (ISRs) for the 32 vectored IRQ slots.	R/W	0	0xFFFF F100
VICVectAddr1	Vector address 1 register.	R/W	0	0xFFFF F104
VICVectAddr2	Vector address 2 register.	R/W	0	0xFFFF F108
VICVectAddr3	Vector address 3 register.	R/W	0	0xFFFF F10C
VICVectAddr4	Vector address 4 register.	R/W	0	0xFFFF F110
VICVectAddr5	Vector address 5 register.	R/W	0	0xFFFF F114
VICVectAddr6	Vector address 6 register.	R/W	0	0xFFFF F118
VICVectAddr7	Vector address 7 register.	R/W	0	0xFFFF F11C
VICVectAddr8	Vector address 8 register.	R/W	0	0xFFFF F120
VICVectAddr9	Vector address 9 register.	R/W	0	0xFFFF F124
VICVectAddr10	Vector address 10 register.	R/W	0	0xFFFF F128
VICVectAddr11	Vector address 11 register.	R/W	0	0xFFFF F12C
VICVectAddr12	Vector address 12 register.	R/W	0	0xFFFF F130
VICVectAddr13	Vector address 13 register.	R/W	0	0xFFFF F134
VICVectAddr14	Vector address 14 register.	R/W	0	0xFFFF F138
VICVectAddr15	Vector address 15 register.	R/W	0	0xFFFF F13C
VICVectAddr16	Vector address 16 register.	R/W	0	0xFFFF F140
VICVectAddr17	Vector address 17 register.	R/W	0	0xFFFF F144
VICVectAddr18	Vector address 18 register.	R/W	0	0xFFFF F148

Table 77. VIC register map

Name	Description	Access	Reset value[1]	Address
VICVectAddr19	Vector address 19 register.	R/W	0	0xFFFF F14C
VICVectAddr20	Vector address 20 register.	R/W	0	0xFFFF F150
VICVectAddr21	Vector address 21 register.	R/W	0	0xFFFF F154
VICVectAddr22	Vector address 22 register.	R/W	0	0xFFFF F158
VICVectAddr23	Vector address 23 register.	R/W	0	0xFFFF F15C
VICVectAddr24	Vector address 24 register.	R/W	0	0xFFFF F160
VICVectAddr25	Vector address 25 register.	R/W	0	0xFFFF F164
VICVectAddr26	Vector address 26 register.	R/W	0	0xFFFF F168
VICVectAddr27	Vector address 27 register.	R/W	0	0xFFFF F16C
VICVectAddr28	Vector address 28 register.	R/W	0	0xFFFF F170
VICVectAddr29	Vector address 29 register.	R/W	0	0xFFFF F174
VICVectAddr30	Vector address 30 register.	R/W	0	0xFFFF F178
VICVectAddr31	Vector address 31 register.	R/W	0	0xFFFF F17C
VICVectPriority0	Vector priority 0 register. Vector Priority Registers 0-31. Each of these registers designates the priority of the corresponding vectored IRQ slot.	R/W	0xF	0xFFFF F200
VICVectPriority1	Vector priority 1 register.	R/W	0xF	0xFFFF F204
VICVectPriority2	Vector priority 2 register.	R/W	0xF	0xFFFF F208
VICVectPriority3	Vector priority 3 register.	R/W	0xF	0xFFFF F20C
VICVectPriority4	Vector priority 4 register.	R/W	0xF	0xFFFF F210
VICVectPriority5	Vector priority 5 register.	R/W	0xF	0xFFFF F214
VICVectPriority6	Vector priority 6 register.	R/W	0xF	0xFFFF F218
VICVectPriority7	Vector priority 7 register.	R/W	0xF	0xFFFF F21C
VICVectPriority8	Vector priority 8 register.	R/W	0xF	0xFFFF F220
VICVectPriority9	Vector priority 9 register.	R/W	0xF	0xFFFF F224
VICVectPriority10	Vector priority 10 register.	R/W	0xF	0xFFFF F228
VICVectPriority11	Vector priority 11 register.	R/W	0xF	0xFFFF F22C
VICVectPriority12	Vector priority 12 register.	R/W	0xF	0xFFFF F230
VICVectPriority13	Vector priority 13 register.	R/W	0xF	0xFFFF F234
VICVectPriority14	Vector priority 14 register.	R/W	0xF	0xFFFF F238
VICVectPriority15	Vector priority 15 register.	R/W	0xF	0xFFFF F23C
VICVectPriority16	Vector priority 16 register.	R/W	0xF	0xFFFF F240
VICVectPriority17	Vector priority 17 register.	R/W	0xF	0xFFFF F244
VICVectPriority18	Vector priority 18 register.	R/W	0xF	0xFFFF F248
VICVectPriority19	Vector priority 19 register.	R/W	0xF	0xFFFF F24C
VICVectPriority20	Vector priority 20 register.	R/W	0xF	0xFFFF F250
VICVectPriority21	Vector priority 21 register.	R/W	0xF	0xFFFF F254
VICVectPriority22	Vector priority 22 register.	R/W	0xF	0xFFFF F258
VICVectPriority23	Vector priority 23 register.	R/W	0xF	0xFFFF F25C
VICVectPriority24	Vector priority 24 register.	R/W	0xF	0xFFFF F260
VICVectPriority25	Vector priority 25 register.	R/W	0xF	0xFFFF F264

Table 77. VIC register map

Name	Description	Access	Reset value[1]	Address
VICVectPriority26	Vector priority 26 register.	R/W	0xF	0xFFFF F268
VICVectPriority27	Vector priority 27 register.	R/W	0xF	0xFFFF F26C
VICVectPriority28	Vector priority 28 register.	R/W	0xF	0xFFFF F270
VICVectPriority29	Vector priority 29 register.	R/W	0xF	0xFFFF F274
VICVectPriority30	Vector priority 30 register.	R/W	0xF	0xFFFF F278
VICVectPriority31	Vector priority 31 register.	R/W	0xF	0xFFFF F27C
VICAddress	Vector address register. When an IRQ interrupt occurs, the Vector Address Register holds the address of the currently active interrupt.	R/W	0	0xFFFF FF00

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

4. VIC registers

The following section describes the VIC registers in the order in which they are used in the VIC logic, from those closest to the interrupt request inputs to those most abstracted for use by software. For most people, this is also the best order to read about the registers when learning the VIC.

4.1 Software Interrupt Register (VICSoftInt - 0xFFFF F018)

The VICSoftInt register is used to generate software interrupts. The contents of this register are ORed with the 32 interrupt requests from the various peripherals, before any other logic is applied.

Table 78. Software Interrupt register (VICSoftInt - address 0xFFFF F018) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See Table 7–92 “Interrupt sources bit allocation table”.	0	Do not force the interrupt request with this bit number. Writing zeroes to bits in VICSoftInt has no effect, see VICSoftIntClear (Section 7–4.2).	0
		1	Force the interrupt request with this bit number.	

4.2 Software Interrupt Clear Register (VICSoftIntClear - 0xFFFF F01C)

The VICSoftIntClear register is a ’Write Only’ register. This register allows software to clear one or more bits in the Software Interrupt register, without having to first read it.

Table 79. Software Interrupt Clear register (VICSoftIntClear - address 0xFFFF F01C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See Table 7–92 “Interrupt sources bit allocation table”.	0	Writing a 0 leaves the corresponding bit in VICSoftInt unchanged.	0
		1	Writing a 1 clears the corresponding bit in the Software Interrupt register, removing any interrupt that may have been generated by that bit.	

4.3 Raw Interrupt Status Register (VICRawIntr - 0xFFFF F008)

This is a read only register. This register reads out the state of the 32 interrupt requests and software interrupts, regardless of enabling or classification.

Table 80. Raw Interrupt Status register (VICRawIntr - address 0xFFFF F008) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See Table 7–92 "Interrupt sources bit allocation table".	0	Neither the hardware nor software interrupt request with this bit number are asserted.	-
		1	The hardware or software interrupt request with this bit number is asserted.	

4.4 Interrupt Enable Register (VICIntEnable - 0xFFFF F010)

This is a read/write accessible register. This register controls which of the 32 combined hardware and software interrupt requests are enabled to contribute to FIQ or IRQ.

Table 81. Interrupt Enable register (VICIntEnable - address 0xFFFF F010) bit description

Bit	Symbol	Description	Reset value
31:0	See Table 7–92 "Interrupt sources bit allocation table".	When this register is read, 1s indicate interrupt requests or software interrupts that are enabled to contribute to FIQ or IRQ. When this register is written, ones enable interrupt requests or software interrupts to contribute to FIQ or IRQ, zeroes have no effect. See Section 7–4.5 "Interrupt Enable Clear Register (VICIntEnClear - 0xFFFF F014)" on page 92 and Table 7–82 below for how to disable interrupts.	0

4.5 Interrupt Enable Clear Register (VICIntEnClear - 0xFFFF F014)

This is a write only register. This register allows software to clear one or more bits in the Interrupt Enable register (see Section 7–4.4 "Interrupt Enable Register (VICIntEnable - 0xFFFF F010)" on page 92), without having to first read it.

Table 82. Interrupt Enable Clear register (VICIntEnClear - address 0xFFFF F014) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See Table 7–92 "Interrupt sources bit allocation table".	0	Writing a 0 leaves the corresponding bit in VICIntEnable unchanged.	-
		1	Writing a 1 clears the corresponding bit in the Interrupt Enable register, thus disabling interrupts for this request.	

4.6 Interrupt Select Register (VICIntSelect - 0xFFFF F00C)

This is a read/write accessible register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.

Table 83. Interrupt Select register (VICIntSelect - address 0xFFFF F00C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See Table 7–92 "Interrupt sources bit allocation table".	0	The interrupt request with this bit number is assigned to the IRQ category.	0
		1	The interrupt request with this bit number is assigned to the FIQ category.	

4.7 IRQ Status Register (VICIRQStatus - 0xFFFF F000)

This is a read only register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.

Table 84. IRQ Status register (VICIRQStatus - address 0xFFFF F000) bit description

Bit	Symbol	Description	Reset value
31:0	See Table 7–92 "Interrupt sources bit allocation table".	A bit read as 1 indicates a corresponding interrupt request being enabled, classified as IRQ, and asserted	0

4.8 FIQ Status Register (VICFIQStatus - 0xFFFF F004)

This is a read only register. This register reads out the state of those interrupt requests that are enabled and classified as FIQ. If more than one request is classified as FIQ, the FIQ service routine can read this register to see which request(s) is (are) active.

Table 85. FIQ Status register (VICFIQStatus - address 0xFFFF F004) bit description

Bit	Symbol	Description	Reset value
31:0	See Table 7–92 "Interrupt sources bit allocation table".	A bit read as 1 indicates a corresponding interrupt request being enabled, classified as IRQ, and asserted	0

4.9 Vector Address Registers 0-31 (VICVectAddr0-31 - 0xFFFF F100 to 17C)

These are read/write accessible registers. These registers hold the addresses of the Interrupt Service routines (ISRs) for the 32 vectored IRQ slots.

Table 86. Vector Address registers 0-31 (VICVectAddr0-31 - addresses 0xFFFF F100 to 0xFFFF F17C) bit description

Bit	Symbol	Description	Reset value
31:0	VICVectAddr	The VIC provides the contents of one of these registers in response to a read of the Vector Address register (VICAddress see Section 7–4.9). The contents of the specific VICVectAddr register (one of the 32 VICVectAddr registers) that corresponds to the interrupt that is to be serviced is read from VICAddress whenever an interrupt occurs.	0x0000 0000

4.10 Vector Priority Registers 0-31 (VICVectPriority0-31 - 0xFFFF F200 to 27C)

These registers select a priority level for the 32 vectored IRQs. There are 16 priority levels, corresponding to the values 0 through 15 decimal, of which 15 is the lowest priority. The reset value of these registers defaults all interrupt to the lowest priority, allowing a single write to elevate the priority of an individual interrupt.

Table 87. Vector Priority registers 0-31 (VICVectPriority0-31 - addresses 0xFFFF F200 to 0xFFFF F27C) bit description

Bit	Symbol	Description	Reset value
3:0	VICVectPriority	Selects one of 16 priority levels for the corresponding vectored interrupt.	0xF
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.11 Vector Address Register (VICAddress - 0xFFFF FF00)

When an IRQ interrupt occurs, the address of the Interrupt Service Routine (ISR) for the interrupt that is to be serviced can be read from this register. The address supplied is from one of the Vector Address Registers (VICVectAddr0-31).

Table 88. Vector Address register (VICAddress - address 0xFFFF FF00) bit description

Bit	Symbol	Description	Reset value
31:0	VICAddress	Contains the address of the ISR for the currently active interrupt. This register must be written (with any value) at the end of an ISR, to update the VIC priority hardware. Writing to the register at any other time can cause incorrect operation.	0

4.12 Software Priority Mask Register (VICSWPriorityMask - 0xFFFF F024)

The Software Priority Mask Register contains individual mask bits for the 16 interrupt priority levels.

Table 89. Software Priority Mask register (VICSWPriorityMask - address 0xFFFF F024) bit description

Bit	Symbol	Value	Description	Reset value
15:0	VICSWPriorityMask	0	Interrupt priority level is masked.	0xFFFF
		1	Interrupt priority level is not masked.	
31:16	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.13 Protection Enable Register (VICProtection - 0xFFFF F020)

This is a read/write accessible register. This one bit register controls access to the VIC registers by software running in User mode. The VICProtection register itself can only be accessed in privileged mode.

Table 90. Protection Enable register (VICProtection - address 0xFFFF F020) bit description

Bit	Symbol	Value	Description	Reset value
0	VIC_access	0	VIC registers can be accessed in User or privileged mode.	0
		1	The VIC registers can only be accessed in privileged mode.	
31:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5. Interrupt sources

Table 7–91 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. There is no significance or priority about what line is connected where, except for certain standards from ARM.

Table 91. Connection of interrupt sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Channel # and Hex Mask	
WDT	Watchdog Interrupt (WDINT)	0	0x0000 0001
-	Reserved for Software Interrupts only	1	0x0000 0002
ARM Core	Embedded ICE, DbgCommRx	2	0x0000 0004
ARM Core	Embedded ICE, DbgCommTX	3	0x0000 0008
TIMER0	Match 0 - 1 (MR0, MR1) Capture 0 - 1 (CR0, CR1)	4	0x0000 0010
TIMER1	Match 0 - 2 (MR0, MR1, MR2) Capture 0 - 1 (CR0, CR1)	5	0x0000 0020
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6	0x0000 0040

Table 91. Connection of interrupt sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Channel # and Hex Mask	
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Control Change	7	0x0000 0080
PWM0, PWM1	Match 0 - 6 of PWM0 Capture 0 of PWM0 Match 0 - 6 of PWM1 Capture 0-1 of PWM1	8	0x0000 0100
I^2C0	SI (state change)	9	0x0000 0200
SPI, SSP0	SPI Interrupt Flag of SPI (SPIF) Mode Fault of SPI0 (MODF) Tx FIFO half empty of SSP1 Rx FIFO half full of SSP1 Rx Timeout of SSP1 Rx Overrun of SSP1	10	0x0000 0400
SSP 1	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun	11	0x0000 0800
PLL	PLL Lock (PLOCK)	12	0x0000 1000
RTC	Counter Increment (RTCCIF) Alarm (RTCALF) Subsecond Int (RTCSSF)	13	0x0000 2000
System Control (External Interrupts)	External Interrupt 0 (EINT0)	14	0x0000 4000
	External Interrupt 1 (EINT1)	15	0x0000 8000
	External Interrupt 2 (EINT2)	16	0x0001 0000
	External Interrupt 3 (EINT3). **Note:** EINT3 channel is shared with GPIO interrupts	17	0x0002 0000
ADC0	A/D Converter 0 end of conversion	18	0x0004 0000
I^2C1	SI (state change)	19	0x0008 0000
BOD	Brown Out detect	20	0x0010 0000
Ethernet	WakeupInt, SoftInt, TxDoneInt, TxFinishedInt, TxErrorInt, TxUnderrunInt, RxDoneInt, RxFinishedInt, RxErrorInt, RxOverrunInt.	21	0x0020 0000
USB	USB_INT_REQ_LP, USB_INT_REQ_HP, USB_INT_REQ_DMA	22	0x0040 0000
CAN	CAN Common, CAN 0 Tx, CAN 0 Rx, CAN 1 Tx, CAN 1 Rx	23	0x0080 0000

Table 91. Connection of interrupt sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Channel # and Hex Mask	
SD/ MMC interface	RxDataAvlbl, TxDataAvlbl, RxFifoEmpty, TxFifoEmpty, RxFifoFull, TxFifoFull, RxFifoHalfFull, TxFifoHalfEmpty, RxActive, TxActive, CmdActive, DataBlockEnd, StartBitErr, DataEnd, CmdSent, CmdRespEnd, RxOverrun, TxUnderrun, DataTimeOut, CmdTimeOut, DataCrcFail, CmdCrcFail	24	0x0100 0000
GP DMA	IntStatus of DMA channel 0, IntStatus of DMA channel 1	25	0x0200 0000
Timer 2	Match 0-3 Capture 0-1	26	0x0400 0000
Timer 3	Match 0-3 Capture 0-1	27	0x0800 0000
UART 2	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	28	0x1000 0000
UART 3	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	29	0x2000 0000
I^2C2	SI (state change)	30	0x4000 0000
I^2S	irq_rx irq_tx	31	0x8000 0000

Table 92. Interrupt sources bit allocation table

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Symbol	I2S	I2C2	UART3	UART2	TIMER3	TIMER2	GPDMA	SD/MMC
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	CAN1&2	USB	Ethernet	BOD	I2C1	AD0	EINT3	EINT2
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	EINT1	EINT0	RTC	PLL	SSP1	SPI/SSP0	I2C0	PWM0&1
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT

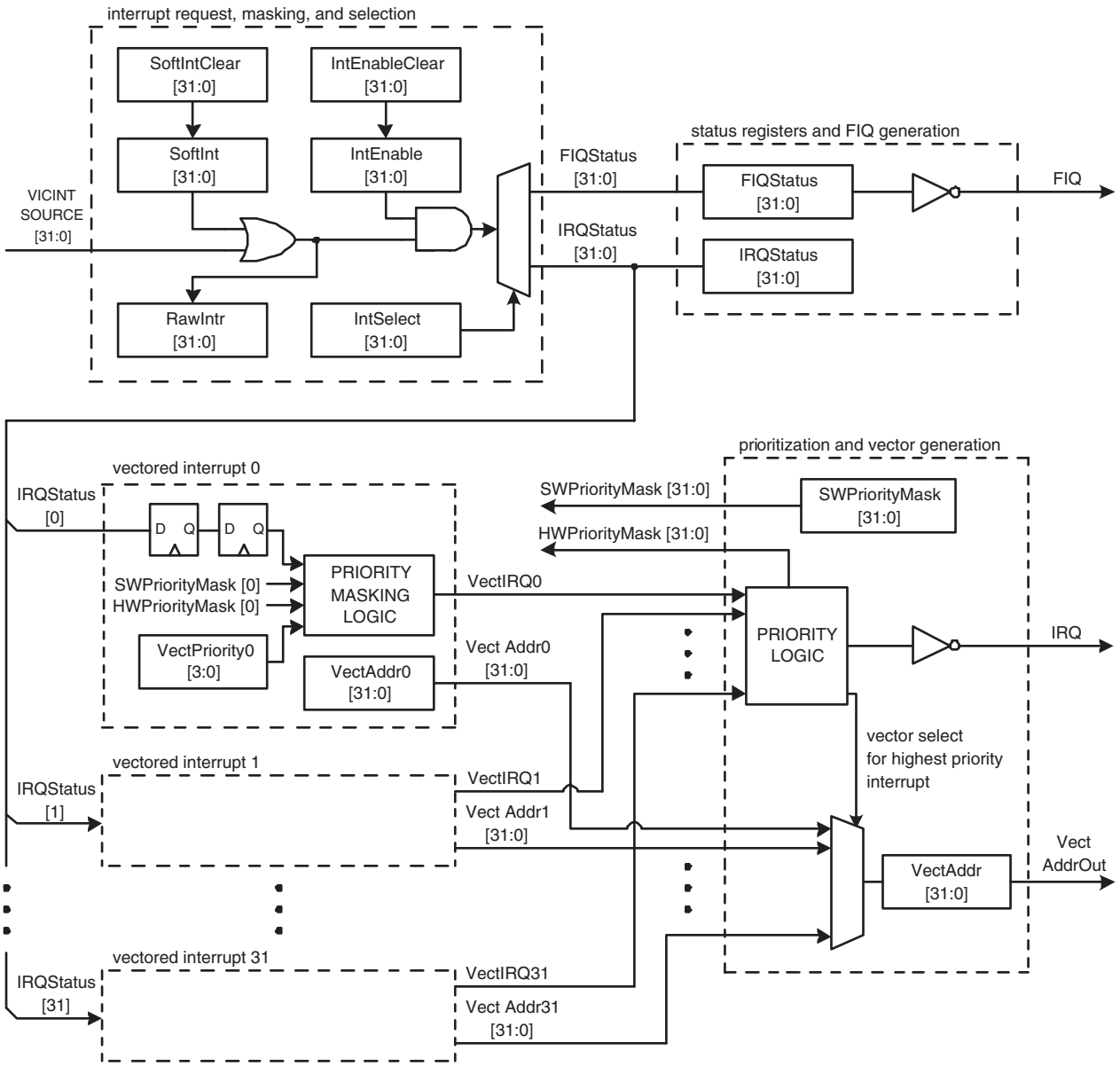


Fig 17. Block diagram of the Vectored Interrupt Controller

1. LPC2468 208-pin packages

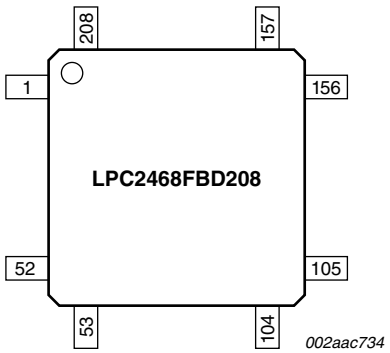


Fig 18. LPC2468 pinning LQFP208 package

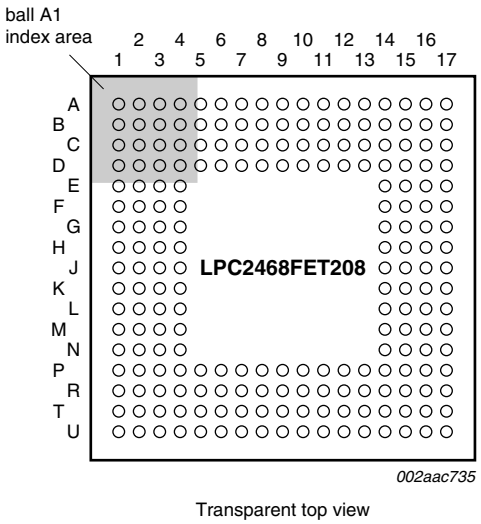


Fig 19. LPC2468 pinning TFBGA208 package

Table 93. Pin allocation table

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
Row A							
1	P3[27]/D27/ CAP1[0]/PWM1[4]	2	V_{SSIO}	3	P1[0]/ENET_TXD0	4	P4[31]/$\overline{CS1}$
5	P1[4]/ENET_TX_EN	6	P1[9]/ENET_RXD0	7	P1[14]/ENET_RX_ER	8	P1[15]/ ENET_REF_CLK/ ENET_RX_CLK
9	P1[17]/ENET_MDIO	10	P1[3]/ENET_TXD3/ MCICMD/PWM0[2]	11	P4[15]/A15	12	V_{SSIO}

Table 93. Pin allocation table *...continued*

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
13	P3[20]/D20/ PWM0[5]/DSR1	14	P1[11]/ENET_RXD2/ MCIDAT2/PWM0[6]	15	P0[8]/I2STX_WS/ MISO1/MAT2[2]	16	P1[12]/ENET_RXD3/ MCIDAT3/PCAP0[0]
17	P1[5]/ENET_TX_ER/ MCIPWR/PWM0[3]		-		-		-
Row B							
1	P3[2]/D2	2	P3[10]/D10	3	P3[1]/D1	4	P3[0]/D0
5	P1[1]/ENET_TXD1	6	V_{SSIO}	7	P4[30]/$\overline{CS0}$	8	P4[24]/$\overline{OE}$
9	P4[25]/$\overline{WE}$	10	P4[29]/BLS3/ MAT2[1]/RXD3	11	P1[6]/ENET_TX_CLK/ MCIDAT0/PWM0[4]	12	P0[4]/I2SRX_CLK/RD2/ CAP2[0]
13	$V_{DD(3V3)}$	14	P3[19]/D19/ PWM0[4]/DCD1	15	P4[14]/A14	16	P4[13]/A13
17	P2[0]/PWM1[1]/TXD1/ TRACECLK		-		-		-
Row C							
1	P3[13]/D13	2	TDI	3	RTCK	4	P0[2]/TXD0
5	P3[9]/D9	6	P3[22]/D22/ PCAP0[0]/RI1	7	P1[8]/ENET_CRS_DV/ ENET_CRS	8	P1[10]/ENET_RXD1
9	$V_{DD(3V3)}$	10	P3[21]/D21/ PWM0[6]/DTR1	11	P4[28]/BLS2/ MAT2[0]/TXD3	12	P0[5]/I2SRX_WS/TD2/ CAP2[1]
13	P0[7]/I2STX_CLK/SCK1 /MAT2[1]	14	P0[9]/I2STX_SDA/ MOSI1/MAT2[3]	15	P3[18]/D18/ PWM0[3]/CTS1	16	P4[12]/A12
17	$V_{DD(3V3)}$		-		-		-
Row D							
1	$\overline{TRST}$	2	P3[28]/D28/ CAP1[1]/PWM1[5]	3	TDO	4	P3[12]/D12
5	P3[11]/D11	6	P0[3]/RXD0	7	$V_{DD(3V3)}$	8	P3[8]/D8
9	P1[2]/ENET_TXD2/ MCICLK/PWM0[1]	10	P1[16]/ENET_MDC	11	$V_{DD(DCDC)(3V3)}$	12	V_{SSCORE}
13	P0[6]/I2SRX_SDA/ SSEL1/MAT2[0]	14	P1[7]/ENET_COL/ MCIDAT1/PWM0[5]	15	P2[2]/PWM1[3]/ CTS1/PIPESTAT1	16	P1[13]/ENET_RX_DV
17	P2[4]/PWM1[5]/ DSR1/TRACESYNC		-		-		-
Row E							
1	P0[26]/AD0[3]/ AOUT/RXD3	2	TCK	3	TMS	4	P3[3]/D3
14	P2[1]/PWM1[2]/RXD1/ PIPESTAT0	15	V_{SSIO}	16	P2[3]/PWM1[4]/ DCD1/PIPESTAT2	17	P2[6]/PCAP1[0]/ RI1/TRACEPKT1
Row F							
1	P0[25]/AD0[2]/ I2SRX_SDA/TXD3	2	P3[4]/D4	3	P3[29]/D29/ MAT1[0]/PWM1[6]	4	DBGEN
14	P4[11]/A11	15	P3[17]/D17/ PWM0[2]/RXD1	16	P2[5]/PWM1[6]/ DTR1/TRACEPKT0	17	P3[16]/D16/ PWM0[1]/TXD1
Row G							
1	P3[5]/D5	2	P0[24]/AD0[1]/ I2SRX_WS/CAP3[1]	3	$V_{DD(3V3)}$	4	V_{DDA}

Table 93. Pin allocation table *...continued*

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
14	NC	15	P4[27]/BLS1	16	P2[7]/RD2/ RTS1/TRACEPKT2	17	P4[10]/A10
Row H							
1	P0[23]/AD0[0]/ I2SRX_CLK/CAP3[0]	2	P3[14]/D14	3	P3[30]/D30/ MAT1[1]/RTS1	4	$V_{DD(DCDC)(3V3)}$
14	V_{SSIO}	15	P2[8]/TD2/ TXD2/TRACEPKT3	16	P2[9]/U1CONNECT/ RXD2/EXTIN0	17	P4[9]/A9
Row J							
1	P3[6]/D6	2	V_{SSA}	3	P3[31]/D31/MAT1[2]	4	NC
14	P0[16]/RXD1/ SSEL0/SSEL	15	P4[23]/A23/ RXD2/MOSI1	16	P0[15]/TXD1/ SCK0/SCK	17	P4[8]/A8
Row K							
1	VREF	2	RTCX1	3	$\overline{\text{RSTOUT}}$	4	V_{SSCORE}
14	P4[22]/A22/ TXD2/MISO1	15	P0[18]/DCD1/ MOSI0/MOSI	16	$V_{DD(3V3)}$	17	P0[17]/CTS1/ MISO0/MISO
Row L							
1	P3[7]/D7	2	RTCX2	3	V_{SSIO}	4	P2[30]/DQMOUT2/ MAT3[2]/SDA2
14	NC	15	P4[26]/BLS0	16	P4[7]/A7	17	P0[19]/DSR1/ MCICLK/SDA1
Row M							
1	P3[15]/D15	2	$\overline{\text{RESET}}$	3	VBAT	4	XTAL1
14	P4[6]/A6	15	P4[21]/A21/ SCL2/SSEL1	16	P0[21]/RI1/ MCIPWR/RD1	17	P0[20]/DTR1/ MCICMD/SCL1
Row N							
1	ALARM	2	P2[31]/DQMOUT3/ MAT3[3]/SCL2	3	P2[29]/DQMOUT1	4	XTAL2
14	P2[12]/$\overline{\text{EINT2}}$/ MCIDAT2/I2STX_WS	15	P2[10]/$\overline{\text{EINT0}}$	16	V_{SSIO}	17	P0[22]/RTS1/ MCIDAT0/TD1
Row P							
1	P1[31]/$\overline{\text{USB\_OVRCR2}}$/ SCK1/AD0[5]	2	P1[30]/USB_PWRD2/ V_{BUS}/AD0[4]	3	P2[27]/CKEOUT3/ MAT3[1]/MOSI0	4	P2[28]/DQMOUT0
5	P2[24]/CKEOUT0	6	$V_{DD(3V3)}$	7	P1[18]/USB_UP_LED1/ PWM1[1]/CAP1[0]	8	$V_{DD(3V3)}$
9	P1[23]/USB_RX_DP1/ PWM1[4]/MISO0	10	V_{SSCORE}	11	$V_{DD(DCDC)(3V3)}$	12	V_{SSIO}
13	P2[15]/$\overline{\text{CS3}}$/ CAP2[1]/SCL1	14	P4[17]/A17	15	P4[18]/A18	16	P4[19]/A19
17	$V_{DD(3V3)}$		-		-		-
Row R							
1	P0[12]/$\overline{\text{USB\_PPWR2}}$/ MISO1/AD0[6]	2	P0[13]/USB_UP_LED2/ MOSI1/AD0[7]	3	P0[28]/SCL0	4	P2[25]/CKEOUT1
5	P3[24]/D24/ CAP0[1]/PWM1[1]	6	P0[30]/USB_D−1	7	P2[19]/CLKOUT1	8	P1[21]/USB_TX_DM1/ PWM1[3]/SSEL0

Table 93. Pin allocation table *...continued*

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
9	V_{SSIO}	10	P1[26]/USB_SSPND1/ PWM1[6]/CAP0[0]	11	P2[16]/CAS	12	P2[14]/CS2/ CAP2[0]/SDA1
13	P2[17]/RAS	14	P0[11]/RXD2/SCL2/ MAT3[1]	15	P4[4]/A4	16	P4[5]/A5
17	P4[20]/A20/ SDA2/SCK1		-		-		-
Row T							
1	P0[27]/SDA0	2	P0[31]/USB_D+2	3	P3[26]/D26/ MAT0[1]/PWM1[3]	4	P2[26]/CKEOUT2/ MAT3[0]/MISO0
5	V_{SSIO}	6	P3[23]/D23/ CAP0[0]/PCAP1[0]	7	P0[14]/USB_HSTEN2/ USB_CONNECT2/ SSEL1	8	P2[20]/DYCS0
9	P1[24]/USB_RX_DM1/ PWM1[5]/MOSI0	10	P1[25]/USB_LS1/ USB_HSTEN1/MAT1[1]	11	P4[2]/A2	12	P1[27]/USB_INT1/ USB_OVRCR1/CAP0[1]
13	P1[28]/USB_SCL1/ PCAP1[0]/MAT0[0]	14	P0[1]/TD1/RXD3/SCL1	15	P0[10]/TXD2/SDA2/ MAT3[0]	16	P2[13]/EINT3/ MCIDAT3/I2STX_SDA
17	P2[11]/EINT1/ MCIDAT1/I2STX_CLK		-		-		-
Row U							
1	USB_D−2	2	P3[25]/D25/ MAT0[0]/PWM1[2]	3	P2[18]/CLKOUT0	4	P0[29]/USB_D+1
5	P2[23]/DYCS3/ CAP3[1]/SSEL0	6	P1[19]/USB_TX_E1/ USB_PPWR1/CAP1[1]	7	P1[20]/USB_TX_DP1/ PWM1[2]/SCK0	8	P1[22]/USB_RCV1/ USB_PWRD1/MAT1[0]
9	P4[0]/A0	10	P4[1]/A1	11	P2[21]/DYCS1	12	P2[22]/DYCS2/ CAP3[0]/SCK0
13	$V_{DD(3V3)}$	14	P1[29]/USB_SDA1/ PCAP1[1]/MAT0[1]	15	P0[0]/RD1/TXD/SDA1	16	P4[3]/A3
17	P4[16]/A16		-		-		-

2. Pin configuration

Table 94. Pin description

Symbol	Pin	Ball	Type	Description
P0[0] to P0[31]			I/O	**Port 0:** Port 0 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the Pin Connect block.
P0[0]/RD1/TXD3/ SDA1	94[1]	U15[1]	I/O	**P0[0] —** General purpose digital input/output pin.
			I	**RD1 —** CAN1 receiver input.
			O	**TXD3 —** Transmitter output for UART3.
			I/O	**SDA1 —** I^2C1 data input/output (this is not an open drain pin).
P0[1]/TD1/RXD3/ SCL1	96[1]	T14[1]	I/O	**P0[1] —** General purpose digital input/output pin.
			O	**TD1 —** CAN1 transmitter output.
			I	**RXD3 —** Receiver input for UART3.
			I/O	**SCL1 —** I^2C1 clock input/output (this is not an open drain pin).

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P0[2]/TXD0	202[1]	C4[1]	I/O	**P0[2] —** General purpose digital input/output pin.
			O	**TXD0 —** Transmitter output for UART0.
P0[3]/RXD0	204[1]	D6[1]	I/O	**P0[3] —** General purpose digital input/output pin.
			I	**RXD0 —** Receiver input for UART0.
P0[4]/ I2SRX_CLK/ RD2/CAP2[0]	168[1]	B12[1]	I/O	**P0[4] —** General purpose digital input/output pin.
			I/O	**I2SRX_CLK —** Receive Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
			I	**RD2 —** CAN2 receiver input.
			I	**CAP2[0] —** Capture input for Timer 2, channel 0.
P0[5]/ I2SRX_WS/ TD2/CAP2[1]	166[1]	C12[1]	I/O	**P0[5] —** General purpose digital input/output pin.
			I/O	**I2SRX_WS —** Receive Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
			O	**TD2 —** CAN2 transmitter output.
			I	**CAP2[1] —** Capture input for Timer 2, channel 1.
P0[6]/ I2SRX_SDA/ SSEL1/MAT2[0]	164[1]	D13[1]	I/O	**P0[6] —** General purpose digital input/output pin.
			I/O	**I2SRX_SDA —** Receive data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
			I/O	**SSEL1 —** Slave Select for SSP1.
			O	**MAT2[0] —** Match output for Timer 2, channel 0.
P0[7]/ I2STX_CLK/ SCK1/MAT2[1]	162[1]	C13[1]	I/O	**P0[7] —** General purpose digital input/output pin.
			I/O	**I2STX_CLK —** Transmit Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
			I/O	**SCK1 —** Serial Clock for SSP1.
			O	**MAT2[1] —** Match output for Timer 2, channel 1.
P0[8]/ I2STX_WS/ MISO1/MAT2[2]	160[1]	A15[1]	I/O	**P0[8] —** General purpose digital input/output pin.
			I/O	**I2STX_WS —** Transmit Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
			I/O	**MISO1 —** Master In Slave Out for SSP1.
			O	**MAT2[2] —** Match output for Timer 2, channel 2.
P0[9]/ I2STX_SDA/ MOSI1/MAT2[3]	158[1]	C14[1]	I/O	**P0[9] —** General purpose digital input/output pin.
			I/O	**I2STX_SDA —** Transmit data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
			I/O	**MOSI1 —** Master Out Slave In for SSP1.
			O	**MAT2[3] —** Match output for Timer 2, channel 3.
P0[10]/TXD2/ SDA2/MAT3[0]	98[1]	T15[1]	I/O	**P0[10] —** General purpose digital input/output pin.
			O	**TXD2 —** Transmitter output for UART2.
			I/O	**SDA2 —** I^2C2 data input/output (this is not an open drain pin).
			O	**MAT3[0] —** Match output for Timer 3, channel 0.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P0[11]/RXD2/ SCL2/MAT3[1]	100[1]	R14[1]	I/O	**P0[11] —** General purpose digital input/output pin.
			I	**RXD2 —** Receiver input for UART2.
			I/O	**SCL2 —** I^2C2 clock input/output (this is not an open drain pin).
			O	**MAT3[1] —** Match output for Timer 3, channel 1.
P0[12]/ USB_PPWR2/ MISO1/AD0[6]	41[2]	R1[2]	I/O	**P0[12] —** General purpose digital input/output pin.
			O	**USB_PPWR2 —** Port Power enable signal for USB port 2.
			I/O	**MISO1 —** Master In Slave Out for SSP1.
			I	**AD0[6] —** A/D converter 0, input 6.
P0[13]/ USB_UP_LED2/ MOSI1/AD0[7]	45[2]	R2[2]	I/O	**P0[13] —** General purpose digital input/output pin.
			O	**USB_UP_LED2 —** USB port 2 GoodLink LED indicator. It is LOW when device is configured (non-control endpoints enabled). It is HIGH when the device is not configured or during global suspend.
			I/O	**MOSI1 —** Master Out Slave In for SSP1.
			I	**AD0[7] —** A/D converter 0, input 7.
P0[14]/ USB_HSTEN2/ USB_CONNECT2/S SEL1	69[1]	T7[1]	I/O	**P0[14] —** General purpose digital input/output pin.
			O	**USB_HSTEN2 —** Host Enabled status for USB port 2.
			O	**USB_CONNECT2 —** Soft Connect control for USB port 2. Signal used to switch an external 1.5 kΩ resistor under software control. Used with the SoftConnect USB feature.
			I/O	**SSEL1 —** Slave Select for SSP1.
P0[15]/TXD1/ SCK0/SCK	128[1]	J16[1]	I/O	**P0[15] —** General purpose digital input/output pin.
			O	**TXD1 —** Transmitter output for UART1.
			I/O	**SCK0 —** Serial clock for SSP0.
			I/O	**SCK —** Serial clock for SPI.
P0[16]/RXD1/ SSEL0/SSEL	130[1]	J14[1]	I/O	**P0 [16] —** General purpose digital input/output pin.
			I	**RXD1 —** Receiver input for UART1.
			I/O	**SSEL0 —** Slave Select for SSP0.
			I/O	**SSEL —** Slave Select for SPI.
P0[17]/CTS1/ MISO0/MISO	126[1]	K17[1]	I/O	**P0[17] —** General purpose digital input/output pin.
			I	**CTS1 —** Clear to Send input for UART1.
			I/O	**MISO0 —** Master In Slave Out for SSP0.
			I/O	**MISO —** Master In Slave Out for SPI.
P0[18]/DCD1/ MOSI0/MOSI	124[1]	K15[1]	I/O	**P0[18] —** General purpose digital input/output pin.
			I	**DCD1 —** Data Carrier Detect input for UART1.
			I/O	**MOSI0 —** Master Out Slave In for SSP0.
			I/O	**MOSI —** Master Out Slave In for SPI.
P0[19]/DSR1/ MCICLK/SDA1	122[1]	L17[1]	I/O	**P0[19] —** General purpose digital input/output pin.
			I	**DSR1 —** Data Set Ready input for UART1.
			O	**MCICLK —** Clock output line for SD/MMC interface.
			I/O	**SDA1 —** I^2C1 data input/output (this is not an open drain pin).

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P0[20]/DTR1/ MCICMD/SCL1	120[1]	M17[1]	I/O	**P0[20] —** General purpose digital input/output pin.
			O	**DTR1 —** Data Terminal Ready output for UART1.
			I/O	**MCICMD —** Command line for SD/MMC interface.
			I/O	**SCL1 —** I^2C1 clock input/output (this is not an open drain pin).
P0[21]/RI1/ MCIPWR/RD1	118[1]	M16[1]	I/O	**P0[21] —** General purpose digital input/output pin.
			I	**RI1 —** Ring Indicator input for UART1.
			O	**MCIPWR —** Power Supply Enable for external SD/MMC power supply.
			I	**RD1 —** CAN1 receiver input.
P0[22]/RTS1/ MCIDAT0/TD1	116[1]	N17[1]	I/O	**P0[22] —** General purpose digital input/output pin.
			O	**RTS1 —** Request to Send output for UART1.
			I/O	**MCIDAT0 —** Data line 0 for SD/MMC interface.
			O	**TD1 —** CAN1 transmitter output.
P0[23]/AD0[0]/ I2SRX_CLK/ CAP3[0]	18[2]	H1[2]	I/O	**P0[23] —** General purpose digital input/output pin.
			I	**AD0[0] —** A/D converter 0, input 0.
			I/O	**I2SRX_CLK —** Receive Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
			I	**CAP3[0] —** Capture input for Timer 3, channel 0.
P0[24]/AD0[1]/ I2SRX_WS/ CAP3[1]	16[2]	G2[2]	I/O	**P0[24] —** General purpose digital input/output pin.
			I	**AD0[1] —** A/D converter 0, input 1.
			I/O	**I2SRX_WS —** Receive Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
			I	**CAP3[1] —** Capture input for Timer 3, channel 1.
P0[25]/AD0[2]/ I2SRX_SDA/ TXD3	14[2]	F1[2]	I/O	**P0[25] —** General purpose digital input/output pin.
			I	**AD0[2] —** A/D converter 0, input 2.
			I/O	**I2SRX_SDA —** Receive data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
			O	**TXD3 —** Transmitter output for UART3.
P0[26]/AD0[3]/ AOUT/RXD3	12[2][3]	E1[2][3]	I/O	**P0[26] —** General purpose digital input/output pin.
			I	**AD0[3] —** A/D converter 0, input 3.
			O	**AOUT —** D/A converter output.
			I	**RXD3 —** Receiver input for UART3.
P0[27]/SDA0	50[4]	T1[4]	I/O	**P0[27] —** General purpose digital input/output pin.
			I/O	**SDA0 —** I^2C0 data input/output. Open drain output (for I^2C-bus compliance).
P0[28]/SCL0	48[4]	R3[4]	I/O	**P0[28] —** General purpose digital input/output pin.
			I/O	**SCL0 —** I^2C0 clock input/output. Open drain output (for I^2C-bus compliance).
P0[29]/USB_D+1	61[5]	U4[5]	I/O	**P0[29] —** General purpose digital input/output pin.
			I/O	**USB_D+1 —** USB port 1 bidirectional D+ line.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P0[30]/USB_D−1	62[5]	R6[5]	I/O	**P0[30] —** General purpose digital input/output pin.
			I/O	**USB_D−1 —** USB port 1bidirectional D− line.
P0[31]/USB_D+2	51[5]	T2[5]	I/O	**P0[31] —** General purpose digital input/output pin.
			I/O	**USB_D+2 —** USB port 2 bidirectional D+ line.
P1[0] to P1[31]			I/O	**Port 1:** Port 1 is a 32 bit I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the Pin Connect block.
P1[0]/ ENET_TXD0	196[1]	A3[1]	I/O	**P1[0] —** General purpose digital input/output pin.
			O	**ENET_TXD0 —** Ethernet transmit data 0 (RMII/MII interface).
P1[1]/ ENET_TXD1	194[1]	B5[1]	I/O	**P1[1] —** General purpose digital input/output pin.
			O	**ENET_TXD1 —** Ethernet transmit data 1 (RMII/MII interface).
P1[2]/ ENET_TXD2/ MCICLK/ PWM0[1]	185[1]	D9[1]	I/O	**P1[2] —** General purpose digital input/output pin.
			O	**ENET_TXD2 —** Ethernet transmit data 2 (MII interface).
			O	**MCICLK —** Clock output line for SD/MMC interface.
			O	**PWM0[1] —** Pulse Width Modulator 0, output 1.
P1[3]/ ENET_TXD3/ MCICMD/ PWM0[2]	177[1]	A10[1]	I/O	**P1[3] —** General purpose digital input/output pin.
			O	**ENET_TXD3 —** Ethernet transmit data 3 (MII interface).
			I/O	**MCICMD —** Command line for SD/MMC interface.
			O	**PWM0[2] —** Pulse Width Modulator 0, output 2.
P1[4]/ ENET_TX_EN	192[1]	A5[1]	I/O	**P1[4] —** General purpose digital input/output pin.
			O	**ENET_TX_EN —** Ethernet transmit data enable (RMII/MII interface).
P1[5]/ ENET_TX_ER/ MCIPWR/ PWM0[3]	156[1]	A17[1]	I/O	**P1[5] —** General purpose digital input/output pin.
			O	**ENET_TX_ER —** Ethernet Trensmit Error (MII interface).
			O	**MCIPWR —** Power Supply Enable for external SD/MMC power supply.
			O	**PWM0[3] —** Pulse Width Modulator 0, output 3.
P1[6]/ ENET_TX_CLK/ MCIDAT0/ PWM0[4]	171[1]	B11[1]	I/O	**P1[6] —** General purpose digital input/output pin.
			I	**ENET_TX_CLK —** Ethernet Transmit Clock (MII interface).
			I/O	**MCIDAT0 —** Data line 0 for SD/MMC interface.
			O	**PWM0[4] —** Pulse Width Modulator 0, output 4.
P1[7]/ ENET_COL/ MCIDAT1/ PWM0[5]	153[1]	D14[1]	I/O	**P1[7] —** General purpose digital input/output pin.
			I	**ENET_COL —** Ethernet Collision detect (MII interface).
			I/O	**MCIDAT1 —** Data line 1 for SD/MMC interface.
			O	**PWM0[5] —** Pulse Width Modulator 0, output 5.
P1[8]/ ENET_CRS_DV/ ENET_CRS	190[1]	C7[1]	I/O	**P1[8] —** General purpose digital input/output pin.
			I	**ENET_CRS_DV/ENET_CRS —** Ethernet Carrier Sense/Data Valid (RMII interface)/ Ethernet Carrier Sense (MII interface).
P1[9]/ ENET_RXD0	188[1]	A6[1]	I/O	**P1[9] —** General purpose digital input/output pin.
			I	**ENET_RXD0 —** Ethernet receive data 0 (RMII/MII interface).
P1[10]/ ENET_RXD1	186[1]	C8[1]	I/O	**P1[10] —** General purpose digital input/output pin.
			I	**ENET_RXD1 —** Ethernet receive data 1 (RMII/MII interface).

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P1[11]/ ENET_RXD2/ MCIDAT2/ PWM0[6]	163[1]	A14[1]	I/O	**P1[11] —** General purpose digital input/output pin.
			I	**ENET_RXD2 —** Ethernet Receive Data 2 (MII interface).
			I/O	**MCIDAT2 —** Data line 2 for SD/MMC interface.
			O	**PWM0[6] —** Pulse Width Modulator 0, output 6.
P1[12]/ ENET_RXD3/ MCIDAT3/ PCAP0[0]	157[1]	A16[1]	I/O	**P1[12] —** General purpose digital input/output pin.
			I	**ENET_RXD3 —** Ethernet Receive Data (MII interface).
			I/O	**MCIDAT3 —** Data line 3 for SD/MMC interface.
			I	**PCAP0[0] —** Capture input for PWM0, channel 0.
P1[13]/ ENET_RX_DV	147[1]	D16[1]	I/O	**P1[13] —** General purpose digital input/output pin.
			I	**ENET_RX_DV —** Ethernet Receive Data Valid (MII interface).
P1[14]/ ENET_RX_ER	184[1]	A7[1]	I/O	**P1[14] —** General purpose digital input/output pin.
			I	**ENET_RX_ER —** Ethernet receive error (RMII/MII interface).
P1[15]/ ENET_REF_CLK/ ENET_RX_CLK	182[1]	A8[1]	I/O	**P1[15] —** General purpose digital input/output pin.
			I	**ENET_REF_CLK/ENET_RX_CLK —** Ethernet Reference Clock (RMII interface)/ Ethernet Receive Clock (MII interface).
P1[16]/ ENET_MDC	180[1]	D10[1]	I/O	**P1[16] —** General purpose digital input/output pin.
			I	**ENET_MDC —** Ethernet MIIM clock.
P1[17]/ ENET_MDIO	178[1]	A9[1]	I/O	**P1[17] —** General purpose digital input/output pin.
			I/O	**ENET_MDIO —** Ethernet MI data input and output.
P1[18]/ USB_UP_LED1/ PWM1[1]/ CAP1[0]	66[1]	P7[1]	I/O	**P1[18] —** General purpose digital input/output pin.
			O	**USB_UP_LED1 —** USB port 1 GoodLink LED indicator. It is LOW when device is configured (non-control endpoints enabled). It is HIGH when the device is not configured or during global suspend.
			O	**PWM1[1] —** Pulse Width Modulator 1, channel 1 output.
			I	**CAP1[0] —** Capture input for Timer 1, channel 0.
P1[19]/ $\overline{\text{USB\_TX\_E1}}$/ USB_PPWR1/ CAP1[1]	68[1]	U6[1]	I/O	**P1[19] —** General purpose digital input/output pin.
			O	**$\overline{\text{USB\_TX\_E1}}$ —** Transmit Enable signal for USB port 1 (OTG transceiver).
			O	**$\overline{\text{USB\_PPWR1}}$ —** Port Power enable signal for USB port 1.
			I	**CAP1[1] —** Capture input for Timer 1, channel 1.
P1[20]/ USB_TX_DP1/ PWM1[2]/SCK0	70[1]	U7[1]	I/O	**P1[20] —** General purpose digital input/output pin.
			O	**USB_TX_DP1 —** D+ transmit data for USB port 1 (OTG transceiver).
			O	**PWM1[2] —** Pulse Width Modulator 1, channel 2 output.
			I/O	**SCK0 —** Serial clock for SSP0.
P1[21]/ USB_TX_DM1/ PWM1[3]/SSEL0	72[1]	R8[1]	I/O	**P1[21] —** General purpose digital input/output pin.
			O	**USB_TX_DM1 —** D− transmit data for USB port 1 (OTG transceiver).
			O	**PWM1[3] —** Pulse Width Modulator 1, channel 3 output.
			I/O	**SSEL0 —** Slave Select for SSP0.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P1[22]/ USB_RCV1/ USB_PWRD1/ MAT1[0]	74[1]	U8[1]	I/O	**P1[22] —** General purpose digital input/output pin.
			I	**USB_RCV1 —** Differential receive data for USB port 1 (OTG transceiver).
			I	**USB_PWRD1 —** Power Status for USB port 1 (host power switch).
			O	**MAT1[0] —** Match output for Timer 1, channel 0.
P1[23]/ USB_RX_DP1/ PWM1[4]/MISO0	76[1]	P9[1]	I/O	**P1[23] —** General purpose digital input/output pin.
			I	**USB_RX_DP1 —** D+ receive data for USB port 1 (OTG transceiver).
			O	**PWM1[4] —** Pulse Width Modulator 1, channel 4 output.
			I/O	**MISO0 —** Master In Slave Out for SSP0.
P1[24]/ USB_RX_DM1/ PWM1[5]/MOSI0	78[1]	T9[1]	I/O	**P1[24] —** General purpose digital input/output pin.
			I	**USB_RX_DM1 —** D– receive data for USB port 1 (OTG transceiver).
			O	**PWM1[5] —** Pulse Width Modulator 1, channel 5 output.
			I/O	**MOSI0 —** Master Out Slave in for SSP0.
P1[25]/ $\overline{\text{USB\_LS1}}$/ USB_HSTEN1/ MAT1[1]	80[1]	T10[1]	I/O	**P1[25] —** General purpose digital input/output pin.
			O	**$\overline{\text{USB\_LS1}}$ —** Low Speed status for USB port 1 (OTG transceiver).
			O	**USB_HSTEN1 —** Host Enabled status for USB port 1.
			O	**MAT1[1] —** Match output for Timer 1, channel 1.
P1[26]/ $\overline{\text{USB\_SSPND1}}$/ PWM1[6]/ CAP0[0]	82[1]	R10[1]	I/O	**P1[26] —** General purpose digital input/output pin.
			O	**$\overline{\text{USB\_SSPND1}}$ —** USB port 1 Bus Suspend status (OTG transciever).
			O	**PWM1[6] —** Pulse Width Modulator 1, channel 6 output.
			I	**CAP0[0] —** Capture input for Timer 0, channel 0.
P1[27]/ $\overline{\text{USB\_INT1}}$/ $\overline{\text{USB\_OVRCR1}}$/ CAP0[1]	88[1]	T12[1]	I/O	**P1[27] —** General purpose digital input/output pin.
			I	**$\overline{\text{USB\_INT1}}$ —** USB port 1 OTG ATX interrupt (OTG transceiver).
			I	**$\overline{\text{USB\_OVRCR1}}$ —** USB port 1 Over-Current status.
			I	**CAP0[1] —** Capture input for Timer 0, channel 1.
P1[28]/ USB_SCL1/ PCAP1[0]/ MAT0[0]	90[1]	T13[1]	I/O	**P1[28] —** General purpose digital input/output pin.
			I/O	**USB_SCL1 —** USB port 1 I^2C serial clock (OTG transciever).
			I	**PCAP1[0] —** Capture input for PWM1, channel 0.
			O	**MAT0[0] —** Match output for Timer 0, channel 0.
P1[29]/ USB_SDA1/ PCAP1[1]/ MAT0[1]	92[1]	U14[1]	I/O	**P1[29] —** General purpose digital input/output pin.
			I/O	**USB_SDA1 —** USB port 1 I^2C serial data (OTG transciever).
			I	**PCAP1[1] —** Capture input for PWM1, channel 1.
			O	**MAT0[1] —** Match output for Timer 0, channel 0.
P1[30]/ USB_PWRD2/ V_{BUS}/AD0[4]	42[2]	P2[2]	I/O	**P1[30] —** General purpose digital input/output pin.
			I	**USB_PWRD2 —** Power Status for USB port 2.
			I	**V_{BUS} —** Indicates the presence of USB bus power. **Note:** This signal must be HIGH for USB reset to occur.
			I	**AD0[4] —** A/D converter 0, input 4.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P1[31]/ USB_OVRCR2/ SCK1/AD0[5]	40[2]	P1[2]	I/O	**P1[31] —** General purpose digital input/output pin.
			I	**USB_OVRCR2 —** Over-Current status for USB port 2.
			I/O	**SCK1 —** Serial Clock for SSP1.
			I	**AD0[5] —** A/D converter 0, input 5.
P2[0] to P2[31]			I/O	**Port 2:** Port 2 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 2 pins depends upon the pin function selected via the Pin Connect block.
P2[0]/PWM1[1]/ TXD1/ TRACECLK	154[1]	B17[1]	I/O	**P2[0] —** General purpose digital input/output pin.
			O	**PWM1[1] —** Pulse Width Modulator 1, channel 1 output.
			O	**TXD1 —** Transmitter output for UART1.
			O	**TRACECLK —** Trace Clock.
P2[1]/PWM1[2]/ RXD1/ PIPESTAT0	152[1]	E14[1]	I/O	**P2[1] —** General purpose digital input/output pin.
			O	**PWM1[2] —** Pulse Width Modulator 1, channel 2 output.
			I	**RXD1 —** Receiver input for UART1.
			O	**PIPESTAT0 —** Pipeline Status, bit 0.
P2[2]/PWM1[3]/ CTS1/ PIPESTAT1	150[1]	D15[1]	I/O	**P2[2] —** General purpose digital input/output pin.
			O	**PWM1[3] —** Pulse Width Modulator 1, channel 3 output.
			I	**CTS1 —** Clear to Send input for UART1.
			O	**PIPESTAT1 —** Pipeline Status, bit 1.
P2[3]/PWM1[4]/ DCD1/ PIPESTAT2	144[1]	E16[1]	I/O	**P2[3] —** General purpose digital input/output pin.
			O	**PWM1[4] —** Pulse Width Modulator 1, channel 4 output.
			I	**DCD1 —** Data Carrier Detect input for UART1.
			O	**PIPESTAT2 —** Pipeline Status, bit 2.
P2[4]/PWM1[5]/ DSR1/ TRACESYNC	142[1]	D17[1]	I/O	**P2[4] —** General purpose digital input/output pin.
			O	**PWM1[5] —** Pulse Width Modulator 1, channel 5 output.
			I	**DSR1 —** Data Set Ready input for UART1.
			O	**TRACESYNC —** Trace Synchronization.
P2[5]/PWM1[6]/ DTR1/ TRACEPKT0	140[1]	F16[1]	I/O	**P2[5] —** General purpose digital input/output pin.
			O	**PWM1[6] —** Pulse Width Modulator 1, channel 6 output.
			O	**DTR1 —** Data Terminal Ready output for UART1.
			O	**TRACEPKT0 —** Trace Packet, bit 0.
P2[6]/PCAP1[0]/RI1/ TRACEPKT1	138[1]	E17[1]	I/O	**P2[6] —** General purpose digital input/output pin.
			I	**PCAP1[0] —** Capture input for PWM1, channel 0.
			I	**RI1 —** Ring Indicator input for UART1.
			O	**TRACEPKT1 —** Trace Packet, bit 1.
P2[7]/RD2/ RTS1/ TRACEPKT2	136[1]	G16[1]	I/O	**P2[7] —** General purpose digital input/output pin.
			I	**RD2 —** CAN2 receiver input.
			O	**RTS1 —** Request to Send output for UART1.
			O	**TRACEPKT2 —** Trace Packet, bit 2.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P2[8]/TD2/ TXD2/ TRACEPKT3	134[1]	H15[1]	I/O	**P2[8] —** General purpose digital input/output pin.
			O	**TD2 —** CAN2 transmitter output.
			O	**TXD2 —** Transmitter output for UART2.
			O	**TRACEPKT3 —** Trace Packet, bit 3.
P2[9]/ USB_CONNECT1/ RXD2/ EXTIN0	132[1]	H16[1]	I/O	**P2[9] —** General purpose digital input/output pin.
			O	**USB_CONNECT1 —** USB1 Soft Connect control. Signal used to switch an external 1.5 kΩ resistor under the software control. Used with the SoftConnect USB feature.
			I	**RXD2 —** Receiver input for UART2.
			I	**EXTIN0 —** External Trigger Input.
P2[10]/$\overline{\text{EINT0}}$	110[6]	N15[6]	I/O	**P2[10] —** General purpose digital input/output pin. **Note:** LOW on this pin while RESET is LOW forces on-chip boot-loader to take over control of the part after a reset.
			I	**$\overline{\text{EINT0}}$ —** External interrupt 0 input.
P2[11]/$\overline{\text{EINT1}}$/ MCIDAT1/ I2STX_CLK	108[6]	T17[6]	I/O	**P2[11] —** General purpose digital input/output pin.
			I	**$\overline{\text{EINT1}}$ —** External interrupt 1 input.
			I/O	**MCIDAT1 —** Data line 1 for SD/MMC interface.
			I/O	**I2STX_CLK —** Transmit Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S-bus specification*.
P2[12]/$\overline{\text{EINT2}}$/ MCIDAT2/ I2STX_WS	106[6]	N14[6]	I/O	**P2[12] —** General purpose digital input/output pin.
			I	**$\overline{\text{EINT2}}$ —** External interrupt 2 input.
			I/O	**MCIDAT2 —** Data line 2 for SD/MMC interface.
			I/O	**I2STX_WS —** Transmit Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S-bus specification*.
P2[13]/$\overline{\text{EINT3}}$/ MCIDAT3/ I2STX_SDA	102[6]	T16[6]	I/O	**P2[13] —** General purpose digital input/output pin.
			I	**$\overline{\text{EINT3}}$ —** External interrupt 3 input.
			I/O	**MCIDAT3 —** Data line 3 for SD/MMC interface.
			I/O	**I2STX_SDA —** Transmit data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S-bus specification*.
P2[14]/$\overline{\text{CS2}}$/ CAP2[0]/SDA1	91[6]	R12[6]	I/O	**P2[14] —** General purpose digital input/output pin.
			O	**$\overline{\text{CS2}}$ —** LOW active Chip Select 2 signal.
			I	**CAP2[0] —** Capture input for Timer 2, channel 0.
			I/O	**SDA1 —** I^2C1 data input/output (this is not an open drain pin).
P2[15]/$\overline{\text{CS3}}$/ CAP2[1]/SCL1	99[6]	P13[6]	I/O	**P2[15] —** General purpose digital input/output pin.
			O	**$\overline{\text{CS3}}$ —** LOW active Chip Select 3 signal.
			I	**CAP2[1] —** Capture input for Timer 2, channel 1.
			I/O	**SCL1 —** I^2C1 clock input/output (this is not an open drain pin).
P2[16]/$\overline{\text{CAS}}$	87[1]	R11[1]	I/O	**P2[16] —** General purpose digital input/output pin.
			O	**$\overline{\text{CAS}}$ —** LOW active SDRAM Column Address Strobe.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P2[17]/$\overline{\text{RAS}}$	95[1]	R13[1]	I/O	**P2[17] —** General purpose digital input/output pin.
			O	**$\overline{\text{RAS}}$ —** LOW active SDRAM Row Address Strobe.
P2[18]/ CLKOUT0	59[1]	U3[1]	I/O	**P2[18] —** General purpose digital input/output pin.
			O	**CLKOUT0 —** SDRAM clock 0.
P2[19]/ CLKOUT1	67[1]	R7[1]	I/O	**P2[19] —** General purpose digital input/output pin.
			O	**CLKOUT1 —** SDRAM clock 1.
P2[20]/$\overline{\text{DYCS0}}$	73[1]	T8[1]	I/O	**P2[20] —** General purpose digital input/output pin.
			O	**$\overline{\text{DYCS0}}$ —** SDRAM chip select 0.
P2[21]/$\overline{\text{DYCS1}}$	81[1]	U11[1]	I/O	**P2[21] —** General purpose digital input/output pin.
			O	**$\overline{\text{DYCS1}}$ —** SDRAM chip select 1.
P2[22]/$\overline{\text{DYCS2}}$/ CAP3[0]/SCK0	85[1]	U12[1]	I/O	**P2[22] —** General purpose digital input/output pin.
			O	**$\overline{\text{DYCS2}}$ —** SDRAM chip select 2.
			I	**CAP3[0] —** Capture input for Timer 3, channel 0.
			I/O	**SCK0 —** Serial clock for SSP0.
P2[23]/$\overline{\text{DYCS3}}$/ CAP3[1]/SSEL0	64[1]	U5[1]	I/O	**P2[23] —** General purpose digital input/output pin.
			O	**$\overline{\text{DYCS3}}$ —** SDRAM chip select 3.
			I	**CAP3[1] —** Capture input for Timer 3, channel 1.
			I/O	**SSEL0 —** Slave Select for SSP0.
P2[24]/ CKEOUT0	53[1]	P5[1]	I/O	**P2[24] —** General purpose digital input/output pin.
			O	**CKEOUT0 —** SDRAM clock enable 0.
P2[25]/ CKEOUT1	54[1]	R4[1]	I/O	**P2[25] —** General purpose digital input/output pin.
			O	**CKEOUT1 —** SDRAM clock enable 1.
P2[26]/ CKEOUT2/ MAT3[0]/MISO0	57[1]	T4[1]	I/O	**P2[26] —** General purpose digital input/output pin.
			O	**CKEOUT2 —** SDRAM clock enable 2.
			O	**MAT3[0] —** Match output for Timer 3, channel 0.
			I/O	**MISO0 —** Master In Slave Out for SSP0.
P2[27]/ CKEOUT3/ MAT3[1]/MOSI0	47[1]	P3[1]	I/O	**P2[27] —** General purpose digital input/output pin.
			O	**CKEOUT3 —** SDRAM clock enable 3.
			O	**MAT3[1] —** Match output for Timer 3, channel 1.
			I/O	**MOSI0 —** Master Out Slave In for SSP0.
P2[28]/ DQMOUT0	49[1]	P4[1]	I/O	**P2[28] —** General purpose digital input/output pin.
			O	**DQMOUT0 —** Data mask 0 used with SDRAM and static devices.
P2[29]/ DQMOUT1	43[1]	N3[1]	I/O	**P2[29] —** General purpose digital input/output pin.
			O	**DQMOUT1 —** Data mask 1 used with SDRAM and static devices.
P2[30]/ DQMOUT2/ MAT3[2]/SDA2	31[1]	L4[1]	I/O	**P2[30] —** General purpose digital input/output pin.
			O	**DQMOUT2 —** Data mask 2 used with SDRAM and static devices.
			O	**MAT3[2] —** Match output for Timer 3, channel 2.
			I/O	**SDA2 —** I^2C2 data input/output (this is not an open drain pin).

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P2[31]/ DQMOUT3/ MAT3[3]/SCL2	39[1]	N2[1]	I/O	**P2[31] —** General purpose digital input/output pin.
			O	**DQMOUT3 —** Data mask 3 used with SDRAM and static devices.
			O	**MAT3[3] —** Match output for Timer 3, channel 3.
			I/O	**SCL2 —** I^2C2 clock input/output (this is not an open drain pin).
P3[0] to P3[31]			I/O	**Port 3:** Port 3 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 3 pins depends upon the pin function selected via the Pin Connect block.
P3[0]/D0	197[1]	B4[1]	I/O	**P3[0] —** General purpose digital input/output pin.
			I/O	**D0 —** External memory data line 0.
P3[1]/D1	201[1]	B3[1]	I/O	**P3[1] —** General purpose digital input/output pin.
			I/O	**D1 —** External memory data line 1.
P3[2]/D2	207[1]	B1[1]	I/O	**P3[2] —** General purpose digital input/output pin.
			I/O	**D2 —** External memory data line 2.
P3[3]/D3	3[1]	E4[1]	I/O	**P3[3] —** General purpose digital input/output pin.
			I/O	**D3 —** External memory data line 3.
P3[4]/D4	13[1]	F2[1]	I/O	**P3[4] —** General purpose digital input/output pin.
			I/O	**D4 —** External memory data line 4.
P3[5]/D5	17[1]	G1[1]	I/O	**P3[5] —** General purpose digital input/output pin.
			I/O	**D5 —** External memory data line 5.
P3[6]/D6	23[1]	J1[1]	I/O	**P3[6] —** General purpose digital input/output pin.
			I/O	**D6 —** External memory data line 6.
P3[7]/D7	27[1]	L1[1]	I/O	**P3[7] —** General purpose digital input/output pin.
			I/O	**D7 —** External memory data line 7.
P3[8]/D8	191[1]	D8[1]	I/O	**P3[8] —** General purpose digital input/output pin.
			I/O	**D8 —** External memory data line 8.
P3[9]/D9	199[1]	C5[1]	I/O	**P3[9] —** General purpose digital input/output pin.
			I/O	**D9 —** External memory data line 9.
P3[10]/D10	205[1]	B2[1]	I/O	**P3[10] —** General purpose digital input/output pin.
			I/O	**D10 —** External memory data line 10.
P3[11]/D11	208[1]	D5[1]	I/O	**P3[11] —** General purpose digital input/output pin.
			I/O	**D11 —** External memory data line 11.
P3[12]/D12	1[1]	D4[1]	I/O	**P3[12] —** General purpose digital input/output pin.
			I/O	**D12 —** External memory data line 12.
P3[13]/D13	7[1]	C1[1]	I/O	**P3[13] —** General purpose digital input/output pin.
			I/O	**D13 —** External memory data line 13.
P3[14]/D14	21[1]	H2[1]	I/O	**P3[14] —** General purpose digital input/output pin.
			I/O	**D14 —** External memory data line 14.
P3[15]/D15	28[1]	M1[1]	I/O	**P3[15] —** General purpose digital input/output pin.
			I/O	**D15 —** External memory data line 15.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P3[16]/D16/ PWM0[1]/TXD1	137[1]	F17[1]	I/O	**P3[16] —** General purpose digital input/output pin.
			I/O	**D16 —** External memory data line 16.
			O	**PWM0[1] —** Pulse Width Modulator 0, output 1.
			O	**TXD1 —** Transmitter output for UART1.
P3[17]/D17/ PWM0[2]/RXD1	143[1]	F15[1]	I/O	**P3[17] —** General purpose digital input/output pin.
			I/O	**D17 —** External memory data line 17.
			O	**PWM0[2] —** Pulse Width Modulator 0, output 2.
			I	**RXD1 —** Receiver input for UART1.
P3[18]/D18/ PWM0[3]/CTS1	151[1]	C15[1]	I/O	**P3[18] —** General purpose digital input/output pin.
			I/O	**D18 —** External memory data line 18.
			O	**PWM0[3] —** Pulse Width Modulator 0, output 3.
			I	**CTS1 —** Clear to Send input for UART1.
P3[19]/D19/ PWM0[4]/DCD1	161[1]	B14[1]	I/O	**P3[19] —** General purpose digital input/output pin.
			I/O	**D19 —** External memory data line 19.
			O	**PWM0[4] —** Pulse Width Modulator 0, output 4.
			I	**DCD1 —** Data Carrier Detect input for UART1.
P3[20]/D20/ PWM0[5]/DSR1	167[1]	A13[1]	I/O	**P3[20] —** General purpose digital input/output pin.
			I/O	**D20 —** External memory data line 20.
			O	**PWM0[5] —** Pulse Width Modulator 0, output 5.
			I	**DSR1 —** Data Set Ready input for UART1.
P3[21]/D21/ PWM0[6]/DTR1	175[1]	C10[1]	I/O	**P3[21] —** General purpose digital input/output pin.
			I/O	**D21 —** External memory data line 21.
			O	**PWM0[6] —** Pulse Width Modulator 0, output 6.
			O	**DTR1 —** Data Terminal Ready output for UART1.
P3[22]/D22/ PCAP0[0]/RI1	195[1]	C6[1]	I/O	**P3[22] —** General purpose digital input/output pin.
			I/O	**D22 —** External memory data line 22.
			I	**PCAP0[0] —** Capture input for PWM0, channel 0.
			I	**RI1 —** Ring Indicator input for UART1.
P3[23]/D23/ CAP0[0]/ PCAP1[0]	65[1]	T6[1]	I/O	**P3[23] —** General purpose digital input/output pin.
			I/O	**D23 —** External memory data line 23.
			I	**CAP0[0] —** Capture input for Timer 0, channel 0.
			I	**PCAP1[0] —** Capture input for PWM1, channel 0.
P3[24]/D24/ CAP0[1]/ PWM1[1]	58[1]	R5[1]	I/O	**P3[24] —** General purpose digital input/output pin.
			I/O	**D24 —** External memory data line 24.
			I	**CAP0[1] —** Capture input for Timer 0, channel 1.
			O	**PWM1[1] —** Pulse Width Modulator 1, output 1.
P3[25]/D25/ MAT0[0]/ PWM1[2]	56[1]	U2[1]	I/O	**P3[25] —** General purpose digital input/output pin.
			I/O	**D25 —** External memory data line 25.
			O	**MAT0[0] —** Match output for Timer 0, channel 0.
			O	**PWM1[2] —** Pulse Width Modulator 1, output 2.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P3[26]/D26/ MAT0[1]/ PWM1[3]	55[1]	T3[1]	I/O	**P3[26] —** General purpose digital input/output pin.
			I/O	**D26 —** External memory data line 26.
			O	**MAT0[1] —** Match output for Timer 0, channel 1.
			O	**PWM1[3] —** Pulse Width Modulator 1, output 3.
P3[27]/D27/ CAP1[0]/ PWM1[4]	203[1]	A1[1]	I/O	**P3[27] —** General purpose digital input/output pin.
			I/O	**D27 —** External memory data line 27.
			I	**CAP1[0] —** Capture input for Timer 1, channel 0.
			O	**PWM1[4] —** Pulse Width Modulator 1, output 4.
P3[28]/D28/ CAP1[1]/ PWM1[5]	5[1]	D2[1]	I/O	**P3[28] —** General purpose digital input/output pin.
			I/O	**D28 —** External memory data line 28.
			I	**CAP1[1] —** Capture input for Timer 1, channel 1.
			O	**PWM1[5] —** Pulse Width Modulator 1, output 5.
P3[29]/D29/ MAT1[0]/ PWM1[6]	11[1]	F3[1]	I/O	**P3[29] —** General purpose digital input/output pin.
			I/O	**D29 —** External memory data line 29.
			O	**MAT1[0] —** Match output for Timer 1, channel 0.
			O	**PWM1[6] —** Pulse Width Modulator 1, output 6.
P3[30]/D30/ MAT1[1]/ RTS1	19[1]	H3[1]	I/O	**P3[30] —** General purpose digital input/output pin.
			I/O	**D30 —** External memory data line 30.
			O	**MAT1[1] —** Match output for Timer 1, channel 1.
			O	**RTS1 —** Request to Send output for UART1.
P3[31]/D31/ MAT1[2]	25[1]	J3[1]	I/O	**P3[31] —** General purpose digital input/output pin.
			I/O	**D31 —** External memory data line 31.
			O	**MAT1[2] —** Match output for Timer 1, channel 2.
P4[0] to P4[31]			I/O	**Port 4:** Port 4 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 4 pins depends upon the pin function selected via the Pin Connect block.
P4[0]/A0	75[1]	U9[1]	I/O	**P4[0] —**]General purpose digital input/output pin.
			I/O	**A0 —** External memory address line 0.
P4[1]/A1	79[1]	U10[1]	I/O	**P4[1] —** General purpose digital input/output pin.
			I/O	**A1 —** External memory address line 1.
P4[2]/A2	83[1]	T11[1]	I/O	**P4[2] —** General purpose digital input/output pin.
			I/O	**A2 —** External memory address line 2.
P4[3]/A3	97[1]	U16[1]	I/O	**P4[3] —** General purpose digital input/output pin.
			I/O	**A3 —** External memory address line 3.
P4[4]/A4	103[1]	R15[1]	I/O	**P4[4] —** General purpose digital input/output pin.
			I/O	**A4 —** External memory address line 4.
P4[5]/A5	107[1]	R16[1]	I/O	**P4[5] —** General purpose digital input/output pin.
			I/O	**A5 —** External memory address line 5.
P4[6]/A6	113[1]	M14[1]	I/O	**P4[6] —** General purpose digital input/output pin.
			I/O	**A6 —** External memory address line 6.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P4[7]/A7	121[1]	L16[1]	I/O	**P4[7] —** General purpose digital input/output pin.
			I/O	**A7 —** External memory address line 7.
P4[8]/A8	127[1]	J17[1]	I/O	**P4[8] —** General purpose digital input/output pin.
			I/O	**A8 —** External memory address line 8.
P4[9]/A9	131[1]	H17[1]	I/O	**P4[9] —** General purpose digital input/output pin.
			I/O	**A9 —** External memory address line 9.
P4[10]/A10	135[1]	G17[1]	I/O	**P4[10] —** General purpose digital input/output pin.
			I/O	**A10 —** External memory address line 10.
P4[11]/A11	145[1]	F14[1]	I/O	**P4[11] —** General purpose digital input/output pin.
			I/O	**A11 —** External memory address line 11.
P4[12]/A12	149[1]	C16[1]	I/O	**P4[12] —** General purpose digital input/output pin.
			I/O	**A12 —** External memory address line 12.
P4[13]/A13	155[1]	B16[1]	I/O	**P4[13] —** General purpose digital input/output pin.
			I/O	**A13 —** External memory address line 13.
P4[14]/A14	159[1]	B15[1]	I/O	**P4[14] —** General purpose digital input/output pin.
			I/O	**A14 —** External memory address line 14.
P4[15]/A15	173[1]	A11[1]	I/O	**P4[15] —** General purpose digital input/output pin.
			I/O	**A15 —** External memory address line 15.
P4[16]/A16	101[1]	U17[1]	I/O	**P4[16] —** General purpose digital input/output pin.
			I/O	**A16 —** External memory address line 16.
P4[17]/A17	104[1]	P14[1]	I/O	**P4[17] —** General purpose digital input/output pin.
			I/O	**A17 —** External memory address line 17.
P4[18]/A18	105[1]	P15[1]	I/O	**P4[18] —** General purpose digital input/output pin.
			I/O	**A18 —** External memory address line 18.
P4[19]/A19	111[1]	P16[1]	I/O	**P4[19] —** General purpose digital input/output pin.
			I/O	**A19 —** External memory address line 19.
P4[20]/A20/ SDA2/SCK1	109[1]	R17[1]	I/O	**P4[20] —** General purpose digital input/output pin.
			I/O	**A20 —** External memory address line 20.
			I/O	**SDA2 —** I^2C2 data input/output (this is not an open drain pin).
			I/O	**SCK1 —** Serial Clock for SSP1.
P4[21]/A21/ SCL2/SSEL1	115[1]	M15[1]	I/O	**P4[21] —** General purpose digital input/output pin.
			I/O	**A21 —** External memory address line 21.
			I/O	**SCL2 —** I^2C2 clock input/output (this is not an open drain pin).
			I/O	**SSEL1 —** Slave Select for SSP1.
P4[22]/A22/ TXD2/MISO1	123[1]	K14[1]	I/O	**P4[22] —** General purpose digital input/output pin.
			I/O	**A22 —** External memory address line 22.
			O	**TXD2 —** Transmitter output for UART2.
			I/O	**MISO1 —** Master In Slave Out for SSP1.

Table 94. Pin description *…continued*

Symbol	Pin	Ball	Type	Description
P4[23]/A23/ RXD2/MOSI1	129[1]	J15[1]	I/O	**P4[23] —** General purpose digital input/output pin.
			I/O	**A23 —** External memory address line 23.
			I	**RXD2 —** Receiver input for UART2.
			I/O	**MOSI1 —** Master Out Slave In for SSP1.
P4[24]/$\overline{OE}$	183[1]	B8[1]	I/O	**P4[24] —** General purpose digital input/output pin.
			O	**$\overline{OE}$ —** LOW active Output Enable signal.
P4[25]/$\overline{WE}$	179[1]	B9[1]	I/O	**P4[25] —** General purpose digital input/output pin.
			O	**$\overline{WE}$ —** LOW active Write Enable signal.
P4[26]/BLS0	119[1]	L15[1]	I/O	**P4[26] —** General purpose digital input/output pin.
			O	**BLS0 —** LOW active Byte Lane select signal 0.
P4[27]/BLS1	139[1]	G15[1]	I/O	**P4[27] —** General purpose digital input/output pin.
			O	**BLS1 —** LOW active Byte Lane select signal 1.
P4[28]/BLS2/ MAT2[0]/TXD3	170[1]	C11[1]	I/O	**P4 [28] —** General purpose digital input/output pin.
			O	**BLS2 —** LOW active Byte Lane select signal 2.
			O	**MAT2[0] —** Match output for Timer 2, channel 0.
			O	**TXD3 —** Transmitter output for UART3.
P4[29]/BLS3/ MAT2[1]/RXD3	176[1]	B10[1]	I/O	**P4[29] —** General purpose digital input/output pin.
			O	**BLS3 —** LOW active Byte Lane select signal 3.
			O	**MAT2[1] —** Match output for Timer 2, channel 1.
			I	**RXD3 —** Receiver input for UART3.
P4[30]/$\overline{CS0}$	187[1]	B7[1]	I/O	**P4[30] —** General purpose digital input/output pin.
			O	**$\overline{CS0}$ —** LOW active Chip Select 0 signal.
P4[31]/$\overline{CS1}$	193[1]	A4[1]	I/O	**P4[31] —** General purpose digital input/output pin.
			O	**$\overline{CS1}$ —** LOW active Chip Select 1 signal.
ALARM	37[8]	N1[8]	O	**ALARM —** RTC controlled output. This is a 1.8 V pin. It goes HIGH when a RTC alarm is generated.
USB_D−2	52	U1	I/O	**USB_D−2 —** USB port 2 bidirectional D− line.
DBGEN	9[1]	F4[1]	I	**DBGEN —** JTAG interface control signal. Also used for boundary scanning.
TDO	2[1]	D3[1]	O	**TDO —** Test Data out for JTAG interface.
TDI	4[1]	C2[1]	I	**TDI —** Test Data in for JTAG interface.
TMS	6[1]	E3[1]	I	**TMS —** Test Mode Select for JTAG interface.
$\overline{TRST}$	8[1]	D1[1]	I	**$\overline{TRST}$ —** Test Reset for JTAG interface.
TCK	10[1]	E2[1]	I	**TCK —** Test Clock for JTAG interface.
RTCK	206[1]	C3[1]	I/O	**RTCK —** JTAG interface control signal. **Note:** LOW on this pin while $\overline{RESET}$ is LOW enables ETM pins (P2[9:0]) to operate as Trace port after reset.
$\overline{RSTOUT}$	29[1]	K3[1]	O	**$\overline{RSTOUT}$ —** This is a 1.8 V pin. LOW on this pin indicates LPC2468 being in Reset state.
$\overline{RESET}$	35[7]	M2[7]	I	**external reset input:** A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. TTL with hysteresis, 5 V tolerant.

Table 94. Pin description *...continued*

Symbol	Pin	Ball	Type	Description
XTAL1	44[8]	M4[8]	I	Input to the oscillator circuit and internal clock generator circuits.
XTAL2	46[8]	N4[8]	O	Output from the oscillator amplifier.
RTCX1	34[8]	K2[8]	I	Input to the RTC oscillator circuit.
RTCX2	36[8]	L2[8]	O	Output from the RTC oscillator circuit.
V_{SSIO}	33, 63, 77, 93, 114,133, 148,169, 189, 200[9]	L3, T5, R9,P12, N16, H14,E15 , A12, B6, A2[9]	I	**ground:** 0 V reference for the digital IO pins.
V_{SSCORE}	32, 84, 172[9]	K4, P10, D12[9]	I	**ground:** 0 V reference for the core.
V_{SSA}	22[10]	J2[10]	I	**analog ground:** 0 V reference. This should nominally be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{DD(3V3)}$	15, 60, 71, 89, 112,125, 146, 165,181, 198[11]	G3,P6, P8,U13, P17,K16 ,C17, B13,C9, D7[11]	I	**3.3 V supply voltage:** This is the power supply voltage for the I/O ports.
NC	30, 117, 141[12]	J4, L14, G14[12]	I	**Not Connected pins:** These pins must be left unconnected (floating).
$V_{DD(DCDC)(3V3)}$	26, 86, 174[13]	H4, P11, D11[13]	I	**3.3 V DC-to-DC converter supply voltage:** This is the power supply for the on-chip DC-to-DC converter.
V_{DDA}	20[14]	G4[14]	I	**analog 3.3 V pad supply voltage:** This should be nominally the same voltage as $V_{DD(3V3)}$ but should be isolated to minimize noise and error. This voltage is used to power the ADC and DAC.
VREF	24[14]	K1[14]	I	**ADC reference:** This should be nominally the same voltage as $V_{DD(3V3)}$ but should be isolated to minimize noise and error. The level on this pin is used as a reference for ADC and DAC.
VBAT	38[14]	M3[14]	I	**RTC power supply:** 3.3 V on this pin supplies the power to the RTC.

[1] 5 V tolerant pad providing digital I/O functions with TTL levels and hysteresis.

[2] 5 V tolerant pad providing digital I/O functions (with TTL levels and hysteresis) and analog input. When configured as a ADC input, digital section of the pad is disabled.

[3] 5 V tolerant pad providing digital I/O with TTL levels and hysteresis and analog output function. When configured as the DAC output, digital section of the pad is disabled.

[4] Open drain 5 V tolerant digital I/O I^2C-bus 400 kHz specification compatible pad. It requires an external pull-up to provide output functionality. When power is switched off, this pin connected to the I^2C-bus is floating and does not disturb the I^2C lines.

[5] Pad provides digital I/O and USB functions. It is designed in accordance with the *USB specification, revision 2.0* (Full-speed and Low-speed mode only).

[6] 5 V tolerant pad with 5 ns glitch filter providing digital I/O functions with TTL levels and hysteresis.

[7] 5 V tolerant pad with 20 ns glitch filter providing digital I/O function with TTL levels and hysteresis.

[8] Pad provides special analog functionality.

[9] Pad provides special analog functionality.

[10] Pad provides special analog functionality.

[11] Pad provides special analog functionality.

[12] Pad provides special analog functionality.

[13] Pad provides special analog functionality.

[14] Pad provides special analog functionality.

1. Features

Allows individual pin configuration.

2. Applications

The purpose of the Pin Connect Block is to configure the microcontroller pins to the desired functions.

3. Description

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

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

Selection of a single function on a port pin completely excludes all other functions otherwise available on the same pin.

4. Pin function select register values

The PINSEL registers control the functions of device pins as shown below. Pairs of bits in these registers correspond to specific device pins.

Table 95. Pin function select register bits

PINSEL0 to PINSEL9 Values	Function	Value after Reset
00	Primary (default) function, typically GPIO port	00
01	First alternate function	
10	Second alternate function	
11	Third alternate function	

The direction control bit in the GPIO registers is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically. Each derivative typically has a different pinout and therefore a different set of functions possible for each pin. Details for a specific derivative may be found in the appropriate data sheet.

5. Pin mode select register values

The PINMODE registers control the on-chip pull-up/pull-down resistor feature for all GPIO ports. Two bits are used to control a port pin.

Table 96. Pin Mode Select register bits

PINMODE0 to PINMODE9 Values	Function	Value after Reset
00	Pin has an on-chip pull-up resistor enabled.	00
01	Reserved. This value should not be used.	
10	Pin has neither pull-up nor pull-down resistor enabled.	
11	Pin has an on-chip pull-down resistor enabled.	

6. Register description

The Pin Control Module contains 11 registers as shown in Table 9–97 below.

Table 97. Pin Connect Block Register Map

Name	Description	Access	Reset Value[1]	Address
PINSEL0	Pin function select register 0.	R/W	0x0000 0000	0xE002 C000
PINSEL1	Pin function select register 1.	R/W	0x0000 0000	0xE002 C004
PINSEL2	Pin function select register 2.	R/W	0x0000 0000	0xE002 C008
PINSEL3	Pin function select register 3.	R/W	0x0000 0000	0xE002 C00C
PINSEL4	Pin function select register 4.	R/W	0x0000 0000	0xE002 C010
PINSEL5	Pin function select register 5.	R/W	0x0000 0000	0xE002 C014
PINSEL6	Pin function select register 6.	R/W	0x0000 0000	0xE002 C018
PINSEL7	Pin function select register 7.	R/W	0x0000 0000	0xE002 C01C
PINSEL8	Pin function select register 8.	R/W	0x0000 0000	0xE002 C020
PINSEL9	Pin function select register 9.	R/W	0x0000 0000	0xE002 C024
PINSEL10	Pin function select register 10.	R/W	0x0000 0000	0xE002 C028
PINMODE0	Pin mode select register 0.	R/W	0x0000 0000	0xE002 C040
PINMODE1	Pin mode select register 1.	R/W	0x0000 0000	0xE002 C044
PINMODE2	Pin mode select register 2.	R/W	0x0000 0000	0xE002 C048
PINMODE3	Pin mode select register 3.	R/W	0x0000 0000	0xE002 C04C
PINMODE4	Pin mode select register 4.	R/W	0x0000 0000	0xE002 C050
PINMODE5	Pin mode select register 5.	R/W	0x0000 0000	0xE002 C054
PINMODE6	Pin mode select register 6.	R/W	0x0000 0000	0xE002 C058
PINMODE7	Pin mode select register 7.	R/W	0x0000 0000	0xE002 C05C
PINMODE8	Pin mode select register 8.	R/W	0x0000 0000	0xE002 C060
PINMODE9	Pin mode select register 9.	R/W	0x0000 0000	0xE002 C064

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

Pin control module register reset values

On power-on reset and BOD reset, all registers in this module are reset to '0'.

On external reset and watchdog reset:

- The corresponding bits for P0[31:0], P1[31:0], P2[13:0] are always reset to '0'.
- For all the other bits:

– if the EMC_Reset_Disable = 1 (see Section 3–7 “Other system controls and status flags”), they retain their values for external memory interface
– else if the EMC_Reset_Disable = 0, they are reset to '0'.

6.1 Pin Function Select register 0 (PINSEL0 - 0xE002 C000)

The PINSEL0 register controls the functions of the pins according to the settings listed in Table 9–98. The direction control bit in the IO0DIR register (or the FIO0DIR register if the enhanced GPIO function is selected for port 0) is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 98. Pin function select register 0 (PINSEL0 - address 0xE002 C000) bit description

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0[0]	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0[1]	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0[2]	GPIO Port 0.2	TXD0	Reserved	Reserved	00
7:6	P0[3]	GPIO Port 0.3	RXD0	Reserved	Reserved	00
9:8	P0[4]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2[0]	00
11:10	P0[5]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2[1]	00
13:12	P0[6]	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2[0]	00
15:14	P0[7]	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2[1]	00
17:16	P0[8]	GPIO Port 0.8	I2STX_WS	MISO1	MAT2[2]	00
19:18	P0[9]	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2[3]	00
21:20	P0[10]	GPIO Port 0.10	TXD2	SDA2	MAT3[0]	00
23:22	P0[11]	GPIO Port 0.11	RXD2	SCL2	MAT3[1]	00
25:24	P0[12]	GPIO Port 0.12	$\overline{\text{USB\_PPWR2}}$	MISO1	AD0[6]	00
27:26	P0[13]	GPIO Port 0.13	USB_UP_LED2	MOSI1	AD0[7]	00
29:28	P0[14]	GPIO Port 0.14	$\overline{\text{USB\_HSTEN2}}$	$\overline{\text{USB\_CONNECT2}}$	SSEL1	00
31:30	P0[15]	GPIO Port 0.15	TXD1	SCK0	SCK	00

6.2 Pin Function Select Register 1 (PINSEL1 - 0xE002 C004)

The PINSEL1 register controls the functions of the pins as per the settings listed in Table 9–99. The direction control bit in the IO0DIR (or the FIO0DIR register if the enhanced GPIO function is selected for port 0) register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

Table 99. Pin function select register 1 (PINSEL1 - address 0xE002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0[16]	GPIO Port 0.16	RXD1	SSEL0	SSEL	00
3:2	P0[17]	GPIO Port 0.17	CTS1	MISO0	MISO	00
5:4	P0[18]	GPIO Port 0.18	DCD1	MOSI0	MOSI	00
7:6	P0[19]	GPIO Port 0.19	DSR1	MCICLK	SDA1	00
9:8	P0[20]	GPIO Port 0.20	DTR1	MCICMD	SCL1	00
11:10	P0[21]	GPIO Port 0.21	RI1	MCIPWR	RD1	00

Table 99. Pin function select register 1 (PINSEL1 - address 0xE002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
13:12	P0[22]	GPIO Port 0.22	RTS1	MCIDAT0	TD1	00
15:14	P0[23]	GPIO Port 0.23	AD0[0]	I2SRX_CLK	CAP3[0]	00
17:16	P0[24]	GPIO Port 0.24	AD0[1]	I2SRX_WS	CAP3[1]	00
19:18	P0[25]	GPIO Port 0.25	AD0[2]	I2SRX_SDA	TXD3	00
21:20	P0[26]	GPIO Port 0.26	AD0[3]	AOUT	RXD3	00
23:22	P0[27]	GPIO Port 0.27	SDA0	Reserved	Reserved	00
25:24	P0[28]	GPIO Port 0.28	SCL0	Reserved	Reserved	00
27:26	P0[29]	GPIO Port 0.29	USB_D+1	Reserved	Reserved	00
29:28	P0[30]	GPIO Port 0.30	USB_D–1	Reserved	Reserved	00
31:30	P0[31]	GPIO Port 0.31	USB_D+2	Reserved	Reserved	00

6.3 Pin Function Select register 2 (PINSEL2 - 0xE002 C008)

The PINSEL2 register controls the functions of the pins as per the settings listed in Table 9–100. The direction control bit in the IO1DIR register (or the FIO1DIR register if the enhanced GPIO function is selected for port 1) is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 100. Pin function select register 2 (PINSEL2 - address 0xE002 C008) bit description

PINSEL2	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P1[0]	GPIO Port 1.0	ENET_TXD0	Reserved	Reserved	00
3:2	P1[1]	GPIO Port 1.1	ENET_TXD1	Reserved	Reserved	00
5:4	P1[2]	GPIO Port 1.2	ENET_TXD2	MCICLK	PWM0[1]	00
7:6	P1[3]	GPIO Port 1.3	ENET_TXD3	MCICMD	PWM0[2]	00
9:8	P1[4]	GPIO Port 1.4	ENET_TX_EN	Reserved	Reserved	00
11:10	P1[5]	GPIO Port 1.5	ENET_TX_ER	MCIPWR	PWM0[3]	00
13:12	P1[6]	GPIO Port 1.6	ENET_TX_CLK	MCIDAT0	PWM0[4]	00
15:14	P1[7]	GPIO Port 1.7	ENET_COL	MCIDAT1	PWM0[5]	00
17:16	P1[8]	GPIO Port 1.8	ENET_CRS_DV/ ENET_CRS	Reserved	Reserved	00
19:18	P1[9]	GPIO Port 1.9	ENET_RXD0	Reserved	Reserved	00
21:20	P1[10]	GPIO Port 1.10	ENET_RXD1	Reserved	Reserved	00
23:22	P1[11]	GPIO Port 1.11	ENET_RXD2	MCIDAT2	PWM0[6]	00
25:24	P1[12]	GPIO Port 1.12	ENET_RXD3	MACIDAT3	PCAP0[0]	
27:26	P1[13]	GPIO Port 1.13	ENET_RX_DV	Reserved	Reserved	00
29:28	P1[14]	GPIO Port 1.14	ENET_RX_ER	Reserved	Reserved	00
31:30	P1[15]	GPIO Port 1.15	ENET_REF_CLK /ENET_RX_CLK	Reserved	Reserved	00

6.4 Pin Function Select Register 3 (PINSEL3 - 0xE002 C00C)

The PINSEL3 register controls the functions of the pins as per the settings listed in Table 9–101. The direction control bit in the IO1DIR register (or the FIO1DIR register if the enhanced GPIO function is selected for port 1) is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 101. Pin function select register 3 (PINSEL3 - address 0xE002 C00C) bit description

PINSEL3	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P1[16]	GPIO Port 1.16	ENET_MDC	Reserved	Reserved	00
3:2	P1[17]	GPIO Port 1.17	ENET_MDIO	Reserved	Reserved	00
5:4	P1[18]	GPIO Port 1.18	USB_UP_LED1	PWM1[1]	CAP1[0]	00
7:6	P1[19]	GPIO Port 1.19	$\overline{\text{USB\_TX\_E1}}$	$\overline{\text{USB\_PPWR1}}$	CAP1[1]	00
9:8	P1[20]	GPIO Port 1.20	USB_TX_DP1	PWM1[2]	SCK0	00
11:10	P1[21]	GPIO Port 1.21	USB_TX_DM1	PWM1[3]	SSEL0	00
13:12	P1[22]	GPIO Port 1.22	USB_RCV1	USB_PWRD1	MAT1[0]	00
15:14	P1[23]	GPIO Port 1.23	USB_RX_DP1	PWM1[4]	MISO0	00
17:16	P1[24]	GPIO Port 1.24	USB_RX_DM1	PWM1[5]	MOSI0	00
19:18	P1[25]	GPIO Port 1.25	$\overline{\text{USB\_LS1}}$	$\overline{\text{USB\_HSTEN1}}$	MAT1[1]	00
21:20	P1[26]	GPIO Port 1.26	$\overline{\text{USB\_SSPND1}}$	PWM1[6]	CAP0[0]	00
23:22	P1[27]	GPIO Port 1.27	$\overline{\text{USB\_INT1}}$	$\overline{\text{USB\_OVRCR1}}$	CAP0[1]	00
25:24	P1[28]	GPIO Port 1.28	USB_SCL1	PCAP1[0]	MAT0[0]	00
27:26	P1[29]	GPIO Port 1.29	USB_SDA1	PCAP1[1]	MAT0[1]	00
29:28	P1[30]	GPIO Port 1.30	USB_PWRD2	V_{BUS}	AD0[4]	00
31:30	P1[31]	GPIO Port 1.31	$\overline{\text{USB\_OVRCR2}}$	SCK1	AD0[5]	00

6.5 Pin Function Select Register 4 (PINSEL4 - 0xE002 C010)

The PINSEL4 register controls the functions of the pins as per the settings listed in Table 9–102. The direction control bit in the FIO2DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 102. Pin function select register 4 (PINSEL4 - address 0xE002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2[0]	GPIO Port 2.0	PWM1[1]	TXD1	TRACECLK[1]	00
3:2	P2[1]	GPIO Port 2.1	PWM1[2]	RXD1	PIPESTAT0[1]	00
5:4	P2[2]	GPIO Port 2.2	PWM1[3]	CTS1	PIPESTAT1[1]	00
7:6	P2[3]	GPIO Port 2.3	PWM1[4]	DCD1	PIPESTAT2[1]	00
9:8	P2[4]	GPIO Port 2.4	PWM1[5]	DSR1	TRACESYNC[1]	00
11:10	P2[5]	GPIO Port 2.5	PWM1[6]	DTR1	TRACEPKT0[1]	00
13:12	P2[6]	GPIO Port 2.6	PCAP1[0]	RI1	TRACEPKT1[1]	00
15:14	P2[7]	GPIO Port 2.7	RD2	RTS1	TRACEPKT2[1]	00
17:16	P2[8]	GPIO Port 2.8	TD2	TXD2	TRACEPKT3[1]	00

Table 102. Pin function select register 4 (PINSEL4 - address 0xE002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
19:18	P2[9]	GPIO Port 2.9	USB_CONNECT1	RXD2	EXTIN0[1]	00
21:20	P2[10]	GPIO Port 2.10	EINT0	Reserved	Reserved	00
23:22	P2[11]	GPIO Port 2.11	EINT1	MCIDAT1	I2STX_CLK	00
25:24	P2[12]	GPIO Port 2.12	EINT2	MCIDAT2	I2STX_WS	00
27:26	P2[13]	GPIO Port 2.13	EINT3	MCIDAT3	I2STX_SDA	00
29:28	P2[14]	GPIO Port 2.14	CS2	CAP2[0]	SDA1	00
31:30	P2[15]	GPIO Port 2.15	CS3	CAP2[1]	SCL1	00

[1] See Section 9–6.11 “Pin Function Select Register 10 (PINSEL10 - 0xE002 C028)”for details on using the ETM functionality.

6.6 Pin Function Select Register 5 (PINSEL5 - 0xE002 C014)

The PINSEL5 register controls the functions of the pins as per the settings listed in Table 9–103. The direction control bit in the FIO2DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 103. Pin function select register 5 (PINSEL5 - address 0xE002 C014) bit description

PINSEL5	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2[16]	GPIO Port 2.16	CAS	Reserved	Reserved	00
3:2	P2[17]	GPIO Port 2.17	RAS	Reserved	Reserved	00
5:4	P2[18]	GPIO Port 2.18	CLKOUT0	Reserved	Reserved	00
7:6	P2[19]	GPIO Port 2.19	CLKOUT1	Reserved	Reserved	00
9:8	P2[20]	GPIO Port 2.20	DYCS0	Reserved	Reserved	00
11:10	P2[21]	GPIO Port 2.21	DYCS1	Reserved	Reserved	00
13:12	P2[22]	GPIO Port 2.22	DYSC2	CAP3[0]	SCK0	00
15:14	P2[23]	GPIO Port 2.23	DYSC3	CAP3[1]	SSEL0	00
17:16	P2[24]	GPIO Port 2.24	CKEOUT0	Reserved	Reserved	00
19:18	P2[25]	GPIO Port 2.25	CKEOUT1	Reserved	Reserved	00
21:20	P2[26]	GPIO Port 2.26	CKEOUT2	MAT3[0]	MISO0	00
23:22	P2[27]	GPIO Port 2.27	CKEOUT3	MAT3[1]	MOSI0	00
25:24	P2[28]	GPIO Port 2.28	DQMOUT0	Reserved	Reserved	00
27:26	P2[29]	GPIO Port 2.29	DQMOUT1	Reserved	Reserved	00
29:28	P2[30]	GPIO Port 2.30	DQMOUT2	MAT3[2]	SDA2	00
31:30	P2[31]	GPIO Port 2.31	DQMOUT3	MAT3[3]	SCL2	00

6.7 Pin Function Select Register 6 (PINSEL6 - 0xE002 C018)

The PINSEL6 register controls the functions of the pins as per the settings listed in Table 9–104. The direction control bit in the FIO3DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 104. Pin function select register 6 (PINSEL6 - address 0xE002 C018) bit description

PINSEL6	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P3[0]	GPIO Port 3.0	D0	Reserved	Reserved	00
3:2	P3[1]	GPIO Port 3.1	D1	Reserved	Reserved	00
5:4	P3[2]	GPIO Port 3.2	D2	Reserved	Reserved	00
7:6	P3[3]	GPIO Port 3.3	D3	Reserved	Reserved	00
9:8	P3[4]	GPIO Port 3.4	D4	Reserved	Reserved	00
11:10	P3[5]	GPIO Port 3.5	D5	Reserved	Reserved	00
13:12	P3[6]	GPIO Port 3.6	D6	Reserved	Reserved	00
15:14	P3[7]	GPIO Port 3.7	D7	Reserved	Reserved	00
17:16	P3[8]	GPIO Port 3.8	D8	Reserved	Reserved	00
19:18	P3[9]	GPIO Port 3.9	D9	Reserved	Reserved	00
21:20	P3[10]	GPIO Port 3.10	D10	Reserved	Reserved	00
23:22	P3[11]	GPIO Port 3.11	D11	Reserved	Reserved	00
25:24	P3[12]	GPIO Port 3.12	D12	Reserved	Reserved	00
27:26	P3[13]	GPIO Port 3.13	D13	Reserved	Reserved	00
29:28	P3[14]	GPIO Port 3.14	D14	Reserved	Reserved	00
31:30	P3[15]	GPIO Port 3.15	D15	Reserved	Reserved	00

6.8 Pin Function Select Register 7 (PINSEL7 - 0xE002 C01C)

The PINSEL7 register controls the functions of the pins as per the settings listed in Table 9–105. The direction control bit in the FIO3DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 105. Pin function select register 7 (PINSEL7 - address 0xE002 C01C) bit description

PINSEL7	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P3[16]	GPIO Port 3.16	D16	PWM0[1]	TXD1	00
3:2	P3[17]	GPIO Port 3.17	D17	PWM0[2]	RXD1	00
5:4	P3[18]	GPIO Port 3.18	D18	PWM0[3]	CTS1	00
7:6	P3[19]	GPIO Port 3.19	D19	PWM0[4]	DCD1	00
9:8	P3[20]	GPIO Port 3.20	D20	PWM0[5]	DSR1	00
11:10	P3[21]	GPIO Port 3.21	D21	PWM0[6]	DR1	00
13:12	P3[22]	GPIO Port 3.22	D22	PCAP0[0]	RI1	00
15:14	P3[23]	GPIO Port 3.23	D23	CAP0[0]	PCAP1[0]	00
17:16	P3[24]	GPIO Port 3.24	D24	CAP0[1]	PWM1[1]	00
19:18	P3[25]	GPIO Port 3.25	D25	MAT0[0]	PWM1[2]	00
21:20	P3[26]	GPIO Port 3.26	D26	MAT0[1]	PWM1[3]	00
23:22	P3[27]	GPIO Port 3.27	D27	CAP1[0]	PWM1[4]	00
25:24	P3[28]	GPIO Port 3.28	D28	CAP1[1]	PWM1[5]	00

Table 105. Pin function select register 7 (PINSEL7 - address 0xE002 C01C) bit description

PINSEL7	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
27:26	P3[29]	GPIO Port 3.29	D29	MAT1[0]	PWM1[6]	00
29:28	P3[30]	GPIO Port 3.30	D30	MAT1[1]	RTS1	00
31:30	P3[31]	GPIO Port 3.31	D31	MAT1[2]	Reserved	00

6.9 Pin Function Select Register 8 (PINSEL8 - 0xE002 C020)

The PINSEL8 register controls the functions of the pins as per the settings listed in Table 9–106. The direction control bit in the FIO4DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 106. Pin function select register 8 (PINSEL8 - address 0xE002 C020) bit description

PINSEL8	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P4[0]	GPIO Port 4.0	A0	Reserved	Reserved	00
3:2	P4[1]	GPIO Port 4.1	A1	Reserved	Reserved	00
5:4	P4[2]	GPIO Port 4.2	A2	Reserved	Reserved	00
7:6	P4[3]	GPIO Port 4.3	A3	Reserved	Reserved	00
9:8	P4[4]	GPIO Port 4.4	A4	Reserved	Reserved	00
11:10	P4[5]	GPIO Port 4.5	A5	Reserved	Reserved	00
13:12	P4[6]	GPIO Port 4.6	A6	Reserved	Reserved	00
15:14	P4[7]	GPIO Port 4.7	A7	Reserved	Reserved	00
17:16	P4[8]	GPIO Port 4.8	A8	Reserved	Reserved	00
19:18	P4[9]	GPIO Port 4.9	A9	Reserved	Reserved	00
21:20	P4[10]	GPIO Port 4.10	A10	Reserved	Reserved	00
23:22	P4[11]	GPIO Port 4.11	A11	Reserved	Reserved	00
25:24	P4[12]	GPIO Port 4.12	A12	Reserved	Reserved	00
27:26	P4[13]	GPIO Port 4.13	A13	Reserved	Reserved	00
29:28	P4[14]	GPIO Port 4.14	A14	Reserved	Reserved	00
31:30	P4[15]	GPIO Port 4.15	A15	Reserved	Reserved	00

6.10 Pin Function Select Register 9 (PINSEL9 - 0xE002 C024)

The PINSEL9 register controls the functions of the pins as per the settings listed in Table 9–107. The direction control bit in the FIO4DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 107. Pin function select register 9 (PINSEL9 - address 0xE002 C024) bit description

PINSEL9	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P4[16]	GPIO Port 4.16	A16	Reserved	Reserved	00
3:2	P4[17]	GPIO Port 4.17	A17	Reserved	Reserved	00
5:4	P4[18]	GPIO Port 4.18	A18	Reserved	Reserved	00

Table 107. Pin function select register 9 (PINSEL9 - address 0xE002 C024) bit description

PINSEL9	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
7:6	P4[19]	GPIO Port 4.19	A19	Reserved	Reserved	00
9:8	P4[20]	GPIO Port 4.20	A20	SDA2	SCK1	00
11:10	P4[21]	GPIO Port 4.21	A21	SCL2	SSEL1	00
13:12	P4[22]	GPIO Port 4.22	A22	TXD2	MISO1	00
15:14	P4[23]	GPIO Port 4.23	A23	RXD2	MOSI1	00
17:16	P4[24]	GPIO Port 4.24	$\overline{OE}$	Reserved	Reserved	00
19:18	P4[25]	GPIO Port 4.25	$\overline{WE}$	Reserved	Reserved	00
21:20	P4[26]	GPIO Port 4.26	BLS0	Reserved	Reserved	00
23:22	P4[27]	GPIO Port 4.27	BLS1	Reserved	Reserved	00
25:24	P4[28]	GPIO Port 4.28	BLS2	MAT2[0]	TXD3	00
27:26	P4[29]	GPIO Port 4.29	BLS3	MAT2[1]	RXD3	00
29:28	P4[30]	GPIO Port 4.30	$\overline{CS0}$	Reserved	Reserved	00
31:30	P4[31]	GPIO Port 4.31	$\overline{CS1}$	Reserved	Reserved	00

6.11 Pin Function Select Register 10 (PINSEL10 - 0xE002 C028)

Only bit 3 of this register is used to control the ETM interface pins.

The value of the RTCK I/O pin is sampled when the external reset is asserted. When RTCK pin is low during external reset, bit 3 in PINSEL10 is set to enable the ETM interface pins. When RTCK pin is high during external reset, bit 3 in PINSEL10 is cleared to disable the ETM interface pins.

The ETM interface control pin can also be modified by the software.

Table 108. Pin function select register 10 (PINSEL10 - address 0xE002 C028) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved. Software should not write 1 to these bits.	NA
3	GPIO/TRACE		ETM interface pins control.	$\overline{RTCK}$, see the text above
		0	ETM interface is disabled.	
		1	ETM interface is enabled. ETM signals are available on the pins hosting them regardless of the PINSEL4 content.	
31:4	-	-	Reserved. Software should not write 1 to these bits.	NA

6.12 Pin Mode select register 0 (PINMODE0 - 0xE002 C040)

This register controls pull-up/pull-down resistor configuration for PORT0 pins 0 to 15.

Table 109. Pin Mode select register 0 (PINMODE0 - address 0xE002 C040) bit description

PINMODE0	Symbol	Value	Description	Reset value
1:0	P0.00MODE		PORT0 pin 0 on-chip pull-up/down resistor control.	00
		00	P0.00 pin has a pull-up resistor enabled.	
		01	Reserved. This value should not be used.	
		10	P0.00 pin has neither pull-up nor pull-down.	
		11	P0.00 has a pull-down resistor enabled.	
...				
31:30	P0.15MODE		PORT0 pin 15 on-chip pull-up/down resistor control.	00

6.13 Pin Mode select register 1 (PINMODE1 - 0xE002 C044)

This register controls pull-up/pull-down resistor configuration for PORT0 pins 16 to 26. For details see Section 9–5 “Pin mode select register values”.

Table 110. Pin Mode select register 1 (PINMODE1 - address 0xE002 C044) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P0.16MODE	PORT0 pin 16 on-chip pull-up/down resistor control.	00
...			
21:20	P0.26MODE	PORT0 pin 26 on-chip pull-up/down resistor control.	00
31:21	-	Reserved	

Remark: Pins P0.27 and P0.28 are dedicated I^2C open drain pins without pull-up/down. Pins P0.29, P0.30, P0.31 are USB specific pins without configurable pull-up or pull-down resistors.

6.14 Pin Mode select register 2 (PINMODE2 - 0xE002 C048)

This register controls pull-up/pull-down resistor configuration for PORT1 pins 0 to 15. For details see Section 9–5 “Pin mode select register values”.

Table 111. Pin Mode select register 2 (PINMODE2 - address 0xE002 C048) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P1.00MODE	PORT1 pin 0 on-chip pull-up/down resistor control.	00
...			
31:30	P1.15MODE	PORT1 pin 15 on-chip pull-up/down resistor control.	00

6.15 Pin Mode select register 3 (PINMODE3 - 0xE002 C04C)

This register controls pull-up/pull-down resistor configuration for PORT1 pins 16 to 31. For details see Section 9–5 “Pin mode select register values”.

Table 112. Pin Mode select register 3 (PINMODE3 - address 0xE002 C04C) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P1.16MODE	PORT1 pin 16 on-chip pull-up/down resistor control.	00
...			
31:30	P1.31MODE	PORT1 pin 31 on-chip pull-up/down resistor control.	00

6.16 Pin Mode select register 4 (PINMODE4 - 0xE002 C050)

This register controls pull-up/pull-down resistor configuration for PORT2 pins 0 to 15. For details see Section 9–5 “Pin mode select register values”.

Table 113. Pin Mode select register 4 (PINMODE4 - address 0xE002 C050) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P2.00MODE	PORT2 pin 0 on-chip pull-up/down resistor control.	00
...			
31:30	P2.15MODE	PORT2 pin 15 on-chip pull-up/down resistor control.	00

6.17 Pin Mode select register 5 (PINMODE5 - 0xE002 C054)

This register controls pull-up/pull-down resistor configuration for PORT2 pins 16 to 31. For details see Section 9–5 “Pin mode select register values”.

Table 114. Pin Mode select register 5 (PINMODE5 - address 0xE002 C054) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P2.16MODE	PORT2 pin 16 on-chip pull-up/down resistor control.	00
...			
31:30	P2.31MODE	PORT2 pin 31 on-chip pull-up/down resistor control.	00

6.18 Pin Mode select register 6 (PINMODE6 - 0xE002 C058)

This register controls pull-up/pull-down resistor configuration for PORT3 pins 0 to 15. For details see Section 9–5 “Pin mode select register values”.

Table 115. Pin Mode select register 6 (PINMODE6 - address 0xE002 C058) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P3.00MODE	PORT3 pin 0 on-chip pull-up/down resistor control.	00
...			
31:30	P3.15MODE	PORT3 pin 15 on-chip pull-up/down resistor control.	00

6.19 Pin Mode select register 7 (PINMODE7 - 0xE002 C05C)

This register controls pull-up/pull-down resistor configuration for PORT3 pins 16 to 31. For details see Section 9–5 “Pin mode select register values”.

Table 116. Pin Mode select register 7 (PINMODE7 - address 0xE002 C05C) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P3.16MODE	PORT3 pin 16 on-chip pull-up/down resistor control.	00
...			
31:30	P3.31MODE	PORT3 pin 31 on-chip pull-up/down resistor control.	00

6.20 Pin Mode select register 8 (PINMODE8 - 0xE002 C060)

This register controls pull-up/pull-down resistor configuration for PORT4 pins 0 to 15. For details see Section 9–5 “Pin mode select register values”.

Table 117. Pin Mode select register 8 (PINMODE8 - address 0xE002 C060) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P4.00MODE	PORT4 pin 0 on-chip pull-up/down resistor control.	00
...			
31:30	P4.15MODE	PORT4 pin 15 on-chip pull-up/down resistor control.	00

6.21 Pin Mode select register 9 (PINMODE9 - 0xE002 C064)

This register controls pull-up/pull-down resistor configuration for PORT4 pins 16 to 31. For details see Section 9–5 “Pin mode select register values”.

Table 118. Pin Mode select register 9 (PINMODE9 - address 0xE002 C064) bit description

PINMODE0	Symbol	Description	Reset value
1:0	P4.16MODE	PORT4 pin 16 on-chip pull-up/down resistor control.	00
...			
31:30	P4.31MODE	PORT4 pin 31 on-chip pull-up/down resistor control.	00

1. Features

1.1 Digital I/O ports

- GPIO PORT0 and PORT1 are ports accessible via either the group of registers providing enhanced features and accelerated port access or the legacy group of registers. PORT2/3/4 are accessed as fast ports only.
- Accelerated GPIO functions:
  - GPIO registers are relocated to the ARM local bus so that the fastest possible I/O timing can be achieved
  - Mask registers allow treating sets of port bits as a group, leaving other bits unchanged
  - All GPIO registers are byte and half-word addressable
  - Entire port value can be written in one instruction
- 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
- All I/O default to inputs after reset
- Backward compatibility with other earlier devices is maintained with legacy registers appearing at the original addresses on the APB bus

1.2 Interrupt generating digital ports

- PORT0 and PORT2 provide an interrupt for each port pin
- Each interrupt can be programmed to generate an interrupt on a rising edge, a falling edge, or both
- Edge detection is asynchronous, so may operate when clocks are not present, such as during Power Down mode. With this feature, level triggered interrupts are not needed
- Each enabled interrupt contributes to a Wakeup signal that can be used to bring the part out of Power Down mode
- Registers provide software a view of pending rising edge interrupts, pending falling edge interrupts, and overall pending GPIO interrupts
- GPIO0 and GPIO2 interrupts share the same VIC slot with the External Interrupt 3 event

2. Applications

- General purpose I/O
- Driving LEDs or other indicators
- Controlling off-chip devices

- Sensing digital inputs, detecting edges
- Bringing the part out of Power Down mode

3. Pin description

Table 119. GPIO pin description

Pin Name	Type	Description
P0[31:0] P1[31:0] P2[31:0] P3[31:0] P4[31:0]	Input/ Output	General purpose input/output. These are typically shared with other peripherals functions and will therefore not all be available in an application. Packaging options may affect the number of GPIOs available in a particular device. Some pins may be limited by requirements of the alternate functions of the pin. For example, I^2C0 pins are open-drain. Details may be found in the LPC2400 pin description in Section 8–2.

4. Register description

LPC2400 has up to five 32-bit General Purpose I/O ports. PORT0 and PORT1 are controlled via two groups of registers as shown in Table 10–120 and Table 10–121. Apart from them, LPC2400 can have three additional 32-bit ports, PORT2, PORT3 and PORT4. Details on a specific GPIO port usage can be found in the "Pin Configuration" chapter on page 102 and "Pin Connect Block" chapter on page 119.

Legacy registers shown in Table 10–120 allow backward compatibility with earlier family devices, using existing code. The functions and relative timing of older GPIO implementations is preserved. Only PORT0 and PORT1 can be controlled via the legacy port registers.

The registers in Table 10–121 represent the enhanced GPIO features available on all of the LPC2400's GPIO ports. These registers are located directly on the local bus of the CPU for the fastest possible read and write timing. They can be accessed as byte or half-word long data, too. A mask register allows access to a group of bits in a single GPIO port independently from other bits in the same port.

When PORT0 and PORT1 are used, user must select whether these ports will be accessed via registers that provide enhanced features or a legacy set of registers (see Section 3–7 "Other system controls and status flags" on page 26). While both of a port's fast and legacy GPIO registers are controlling the same physical pins, these two port control branches are mutually exclusive and operate independently. For example, changing a pin's output via a fast register will not be observable via the corresponding legacy register.

The following text will refer to the legacy GPIO as "the slow" GPIO, while GPIO equipped with the enhanced features will be referred as "the fast" GPIO.

Table 120. GPIO register map (legacy APB accessible registers)

Generic Name	Description	Access	Reset value[1]	PORTn Register Address & Name
IOPIN	GPIO Port Pin value register. The current state of the GPIO configured port pins can always be read from this register, regardless of pin direction. By writing to this register port's pins will be set to the desired level instantaneously.	R/W	NA	IO0PIN - 0xE002 8000 IO1PIN - 0xE002 8010
IOSET	GPIO Port Output Set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.	R/W	0x0	IO0SET - 0xE002 8004 IO1SET - 0xE002 8014
IODIR	GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0x0	IO0DIR - 0xE002 8008 IO1DIR - 0xE002 8018
IOCLR	GPIO Port Output Clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	WO	0x0	IO0CLR - 0xE002 800C IO1CLR - 0xE002 801C

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

Table 121. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORTn Register Address & Name
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0x0	FIO0DIR - 0x3FFF C000 FIO1DIR - 0x3FFF C020 FIO2DIR - 0x3FFF C040 FIO2DIR - 0x3FFF C060 FIO2DIR - 0x3FFF C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0x0	FIO0MASK - 0x3FFF C010 FIO1MASK - 0x3FFF C030 FIO2MASK - 0x3FFF C050 FIO3MASK - 0x3FFF C070 FIO4MASK - 0x3FFF C090
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC). The value read is masked by ANDing with inverted FIOMASK. Writing to this register places corresponding values in all bits enabled by zeros in FIOMASK. **Important:** if a FIOPIN register is read, its bit(s) masked with 1 in the FIOMASK register will be set to 0 regardless of the physical pin state.	R/W	0x0	FIO0PIN - 0x3FFF C014 FIO1PIN - 0x3FFF C034 FIO2PIN - 0x3FFF C054 FIO3PIN - 0x3FFF C074 FIO4PIN - 0x3FFF C094
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by 0 in FIOMASK can be altered.	R/W	0x0	FIO0SET - 0x3FFF C018 FIO1SET - 0x3FFF C038 FIO2SET - 0x3FFF C058 FIO3SET - 0x3FFF C078 FIO4SET - 0x3FFF C098
FIOCLR	Fast Port Output Clear register using FIOMASK0. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by 0 in FIOMASK0 can be altered.	WO	0x0	FIO0CLR - 0x3FFF C01C FIO1CLR - 0x3FFF C03C FIO2CLR - 0x3FFF C05C FIO3CLR - 0x3FFF C07C FIO4CLR - 0x3FFF C09C

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

Table 122. GPIO interrupt register map

Generic Name	Description	Access	Reset value[1]	PORTn Register Address & Name
IntEnR	GPIO Interrupt Enable for Rising edge.	R/W	0x0	IO0IntEnR - 0xE002 8090 IO2IntEnR - 0xE002 80B0
IntEnF	GPIO Interrupt Enable for Falling edge.	R/W	0x0	IO0IntEnR - 0xE002 8094 IO2IntEnR - 0xE002 80B4
IntStatR	GPIO Interrupt Status for Rising edge.	RO	0x0	IO0IntStatR - 0xE002 8084 IO2IntStatR - 0xE002 80A4
IntStatF	GPIO Interrupt Status for Falling edge.	RO	0x0	IO0IntStatF - 0xE002 8088 IO2IntStatF - 0xE002 80A8
IntClr	GPIO Interrupt Clear.	WO	0x0	IO0IntClr - 0xE002 808C IO2IntClr - 0xE002 80AC
IntStatus	GPIO overall Interrupt Status.	RO	0x00	IOIntStatus - 0xE002 8080

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

4.1 GPIO port Direction register IODIR and FIODIR(IO[0/1]DIR - 0xE002 80[0/1]8 and FIO[0/1/2/3/4]DIR - 0x3FFF C0[0/2/4/6/8]0)

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

Legacy registers are the IO0DIR and IO1DIR while the enhanced GPIO functions are supported via the FIO0DIR, FIO1DIR, FIO2DIR, FIO3DIR and FIO4DIR registers.

Table 123. GPIO port Direction register (IO0DIR - address 0xE002 8008 and IO1DIR - address 0xE002 8018) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xDIR or P1xDIR		Slow GPIO Direction PORTx control bits. Bit 0 in IOxDIR controls pin Px.0, bit 31 IOxDIR controls pin Px.31.	0x0
		0	Controlled pin is an input pin.	
		1	Controlled pin is an output pin.	

Table 124. Fast GPIO port Direction register (FIO[0/1/2/3/4]DIR - address 0x3FFF C0[0/2/4/6/8]0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP0xDIR FP1xDIR FP2xDIR FP3xDIR FP4xDIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.	0x0
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Aside from the 32-bit long and word only accessible FIODIR register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 10–125, too. Next to providing the same functions as the FIODIR register, these additional registers allow easier and faster access to the physical port pins.

Table 125. Fast GPIO port Direction control byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxDIR0	Fast GPIO Port x Direction control register 0. Bit 0 in FIOxDIR0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0DIR0 - 0x3FFF C000 FIO1DIR0 - 0x3FFF C020 FIO2DIR0 - 0x3FFF C040 FIO3DIR0 - 0x3FFF C060 FIO4DIR0 - 0x3FFF C080
FIOxDIR1	Fast GPIO Port x Direction control register 1. Bit 0 in FIOxDIR1 register corresponds to pin Px.8 ... bit 7 to pin Px.15.	8 (byte) R/W	0x00	FIO0DIR1 - 0x3FFF C001 FIO1DIR1 - 0x3FFF C021 FIO2DIR1 - 0x3FFF C041 FIO3DIR1 - 0x3FFF C061 FIO4DIR1 - 0x3FFF C081
FIO0DIR2	Fast GPIO Port x Direction control register 2. Bit 0 in FIOxDIR2 register corresponds to pin Px.16 ... bit 7 to pin Px.23.	8 (byte) R/W	0x00	FIO0DIR2 - 0x3FFF C002 FIO1DIR2 - 0x3FFF C022 FIO2DIR2 - 0x3FFF C042 FIO3DIR2 - 0x3FFF C062 FIO4DIR2 - 0x3FFF C082
FIOxDIR3	Fast GPIO Port x Direction control register 3. Bit 0 in FIOxDIR3 register corresponds to pin Px.24 ... bit 7 to pin Px.31.	8 (byte) R/W	0x00	FIO0DIR3 - 0x3FFF C003 FIO1DIR3 - 0x3FFF C023 FIO2DIR3 - 0x3FFF C043 FIO3DIR3 - 0x3FFF C063 FIO4DIR3 - 0x3FFF C083
FIOxDIRL	Fast GPIO Port x Direction control Lower half-word register. Bit 0 in FIOxDIRL register corresponds to pin Px.0 ... bit 15 to pin Px.15.	16 (half-word) R/W	0x0000	FIO0DIRL - 0x3FFF C000 FIO1DIRL - 0x3FFF C020 FIO2DIRL - 0x3FFF C040 FIO3DIRL - 0x3FFF C060 FIO4DIRL - 0x3FFF C080
FIOxDIRU	Fast GPIO Port x Direction control Upper half-word register. Bit 0 in FIOxDIRU register corresponds to Px.16 ... bit 15 to Px.31.	16 (half-word) R/W	0x0000	FIO0DIRU - 0x3FFF C002 FIO1DIRU - 0x3FFF C022 FIO2DIRU - 0x3FFF C042 FIO3DIRU - 0x3FFF C062 FIO4DIRU - 0x3FFF C082

4.2 GPIO port output Set register IOSET and FIOSET(IO[0/1]SET - 0xE002 80[0/1]4 and FIO[0/1/2/3/4]SET - 0x3FFF C0[1/3/5/7/9]8)

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

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

Legacy registers are the IO0SET and IO1SET while the enhanced GPIOs are supported via the FIO0SET, FIO1SET, FIO2SET, FIO3SET, and FIO4SET registers. Access to a port pin via the FIOSET register is conditioned by the corresponding bit of the FIOMASK register (see Section 10–4.5 “Fast GPIO port Mask register FIOMASK(FIO[0/1/2/3/4]MASK - 0x3FFF C0[1/3/5/7/9]0)”).

Table 126. GPIO port output Set register (IO0SET - address 0xE002 8004 and IO1SET - address 0xE002 8014) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xSET or P1xSET		Slow GPIO output value Set bits. Bit 0 in IOxSET controls pin Px.0, bit 31 in IOxSET controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to HIGH.	

Table 127. Fast GPIO port output Set register (FIO[0/1/2/3/4]SET - address 0x3FFF C0[1/3/5/7/9]8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP0xSET FP1xSET FP2xSET FP3xSET FP4xSET		Fast GPIO output value Set bits. Bit 0 in FIOxSET controls pin Px.0, bit 31 in FIOxSET controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to HIGH.	

Aside from the 32-bit long and word only accessible FIOSET register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 10–128, too. Next to providing the same functions as the FIOSET register, these additional registers allow easier and faster access to the physical port pins.

Table 128. Fast GPIO port output Set byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxSET0	Fast GPIO Port x output Set register 0. Bit 0 in FIOxSET0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0SET0 - 0x3FFF C018 FIO1SET0 - 0x3FFF C038 FIO2SET0 - 0x3FFF C058 FIO3SET0 - 0x3FFF C078 FIO4SET0 - 0x3FFF C098
FIOxSET1	Fast GPIO Port x output Set register 1. Bit 0 in FIOxSET1 register corresponds to pin Px.8 ... bit 7 to pin Px.15.	8 (byte) R/W	0x00	FIO0SET1 - 0x3FFF C019 FIO1SET1 - 0x3FFF C039 FIO2SET1 - 0x3FFF C059 FIO3SET1 - 0x3FFF C079 FIO4SET1 - 0x3FFF C099
FIOxSET2	Fast GPIO Port x output Set register 2. Bit 0 in FIOxSET2 register corresponds to pin Px.16 ... bit 7 to pin Px.23.	8 (byte) R/W	0x00	FIO0SET2 - 0x3FFF C01A FIO1SET2 - 0x3FFF C03A FIO2SET2 - 0x3FFF C05A FIO3SET2 - 0x3FFF C07A FIO4SET2 - 0x3FFF C09A

Table 128. Fast GPIO port output Set byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxSET3	Fast GPIO Port x output Set register 3. Bit 0 in FIOxSET3 register corresponds to pin Px.24 ... bit 7 to pin Px.31.	8 (byte) R/W	0x00	FIO0SET3 - 0x3FFF C01B FIO1SET3 - 0x3FFF C03B FIO2SET3 - 0x3FFF C05B FIO3SET3 - 0x3FFF C07B FIO4SET3 - 0x3FFF C09B
FIOxSETL	Fast GPIO Port x output Set Lower half-word register. Bit 0 in FIOxSETL register corresponds to pin Px.0 ... bit 15 to pin Px.15.	16 (half-word) R/W	0x0000	FIO0SETL - 0x3FFF C018 FIO1SETL - 0x3FFF C038 FIO2SETL - 0x3FFF C058 FIO3SETL - 0x3FFF C078 FIO4SETL - 0x3FFF C098
FIOxSETU	Fast GPIO Port x output Set Upper half-word register. Bit 0 in FIOxSETU register corresponds to Px.16 ... bit 15 to Px.31.	16 (half-word) R/W	0x0000	FIO0SETU - 0x3FFF C01A FIO1SETU - 0x3FFF C03A FIO2SETU - 0x3FFF C05A FIO3SETU - 0x3FFF C07A FIO4SETU - 0x3FFF C09A

4.3 GPIO port output Clear register IOCLR and FIOCLR (IO[0/1]CLR - 0xE002 80[0/1]C and FIO[0/1/2/3/4]CLR - 0x3FFF C0[1/3/5/7/9]C)

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

Legacy registers are the IO0CLR and IO1CLR while the enhanced GPIOs are supported via the FIO0CLR, FIO1CLR, FIO2CLR, FIO3CLR, and FIO4CLR registers. Access to a port pin via the FIOCLR register is conditioned by the corresponding bit of the FIOMASK register (see Section 10–4.5 “Fast GPIO port Mask register FIOMASK(FIO[0/1/2/3/4]MASK - 0x3FFF C0[1/3/5/7/9]0)”).

Table 129. GPIO port output Clear register (IO0CLR - address 0xE002 800C and IO1CLR - address 0xE002 801C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xCLR or P1xCLR		Slow GPIO output value Clear bits. Bit 0 in IOxCLR controls pin Px.0, bit 31 in IOxCLR controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to LOW.	

Table 130. Fast GPIO port output Clear register (FIO[0/1/2/3/4]CLR - address 0x3FFF C0[1/3/5/7/9]C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP0xCLR FP1xCLR FP2xCLR FP3xCLR FP4xCLR		Fast GPIO output value Clear bits. Bit 0 in FIOxCLR controls pin Px.0, bit 31 controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to LOW.	

Aside from the 32-bit long and word only accessible FIOCLR register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 10–131, too. Next to providing the same functions as the FIOCLR register, these additional registers allow easier and faster access to the physical port pins.

Table 131. Fast GPIO port output Clear byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxCLR0	Fast GPIO Port x output Clear register 0. Bit 0 in FIOxCLR0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) WO	0x00	FIO0CLR0 - 0x3FFF C01C FIO1CLR0 - 0x3FFF C03C FIO2CLR0 - 0x3FFF C05C FIO3CLR0 - 0x3FFF C07C FIO4CLR0 - 0x3FFF C09C
FIOxCLR1	Fast GPIO Port x output Clear register 1. Bit 0 in FIOxCLR1 register corresponds to pin Px.8 ... bit 7 to pin Px.15.	8 (byte) WO	0x00	FIO0CLR1 - 0x3FFF C01D FIO1CLR1 - 0x3FFF C03D FIO2CLR1 - 0x3FFF C05D FIO3CLR1 - 0x3FFF C07D FIO4CLR1 - 0x3FFF C09D
FIOxCLR2	Fast GPIO Port x output Clear register 2. Bit 0 in FIOxCLR2 register corresponds to pin Px.16 ... bit 7 to pin Px.23.	8 (byte) WO	0x00	FIO0CLR2 - 0x3FFF C01E FIO1CLR2 - 0x3FFF C03E FIO2CLR2 - 0x3FFF C05E FIO3CLR2 - 0x3FFF C07E FIO4CLR2 - 0x3FFF C09E
FIOxCLR3	Fast GPIO Port x output Clear register 3. Bit 0 in FIOxCLR3 register corresponds to pin Px.24 ... bit 7 to pin Px.31.	8 (byte) WO	0x00	FIO0CLR3 - 0x3FFF C01F FIO1CLR3 - 0x3FFF C03F FIO2CLR3 - 0x3FFF C05F FIO3CLR3 - 0x3FFF C07F FIO4CLR3 - 0x3FFF C09F
FIOxCLRL	Fast GPIO Port x output Clear Lower half-word register. Bit 0 in FIOxCLRL register corresponds to pin Px.0 ... bit 15 to pin Px.15.	16 (half-word) WO	0x0000	FIO0CLRL - 0x3FFF C01C FIO1CLRL - 0x3FFF C03C FIO2CLRL - 0x3FFF C05C FIO3CLRL - 0x3FFF C07C FIO4CLRL - 0x3FFF C09C
FIOxCLRU	Fast GPIO Port x output Clear Upper half-word register. Bit 0 in FIOxCLRU register corresponds to pin Px.16 ... bit 15 to Px.31.	16 (half-word) WO	0x0000	FIO0CLRU - 0x3FFF C01E FIO1CLRU - 0x3FFF C03E FIO2CLRU - 0x3FFF C05E FIO3CLRU - 0x3FFF C07E FIO4CLRU - 0x3FFF C09E

4.4 GPIO port Pin value register IOPIN and FIOPIN (IO[0/1]PIN - 0xE002 80[0/1]0 and FIO[0/1/2/3/4]PIN - 0x3FFF C0[1/3/5/7/9]4)

This register provides the value of port pins that are configured to perform only digital functions. The register will give the logic value of the pin regardless of whether the pin is configured for input or output, or as GPIO or an alternate digital function. As an example, a particular port pin may have GPIO input, GPIO output, UART receive, and PWM output as selectable functions. Any configuration of that pin will allow its current logic state to be read from the corresponding IOPIN register.

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

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

Legacy registers are the IO0PIN and IO1PIN while the enhanced GPIOs are supported via the FIO0PIN, FIO1PIN, FIO2PIN, FIO3PIN and FIO4PIN registers. Access to a port pin via the FIOPIN register is conditioned by the corresponding bit of the FIOMASK register (see Section 10–4.5 “Fast GPIO port Mask register FIOMASK(FIO[0/1/2/3/4]MASK - 0x3FFF C0[1/3/5/7/9]0)”).

Only pins masked with zeros in the Mask register (see Section 10–4.5 “Fast GPIO port Mask register FIOMASK(FIO[0/1/2/3/4]MASK - 0x3FFF C0[1/3/5/7/9]0)”) will be correlated to the current content of the Fast GPIO port pin value register.

Table 132. GPIO port Pin value register (IO0PIN - address 0xE002 8000 and IO1PIN - address 0xE002 8010) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xVAL or P1xVAL		Slow GPIO pin value bits. Bit 0 in IOxPIN corresponds to pin Px.0, bit 31 in IOxPIN corresponds to pin Px.31.	0x0
		0	Controlled pin output is set to LOW.	
		1	Controlled pin output is set to HIGH.	

Table 133. Fast GPIO port Pin value register (FIO[0/1/2/3/4]PIN - address 0x3FFF C0[1/3/5/7/9]4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP0xVAL FP1xVAL FP2xVAL FP3xVAL FP4xVAL		Fast GPIO output value Set bits. Bit 0 in FIOxCLR corresponds to pin Px.0, bit 31 in FIOxCLR corresponds to pin Px.31.	0x0
		0	Controlled pin output is set to LOW.	
		1	Controlled pin output is set to HIGH.	

Aside from the 32-bit long and word only accessible FIOPIN register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 10–134, too. Next to providing the same functions as the FIOPIN register, these additional registers allow easier and faster access to the physical port pins.

Table 134. Fast GPIO port Pin value byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxPIN0	Fast GPIO Port x Pin value register 0. Bit 0 in FIOxPIN0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0PIN0 - 0x3FFF C014 FIO1PIN0 - 0x3FFF C034 FIO2PIN0 - 0x3FFF C054 FIO3PIN0 - 0x3FFF C074 FIO4PIN0 - 0x3FFF C094
FIOxPIN1	Fast GPIO Port x Pin value register 1. Bit 0 in FIOxPIN1 register corresponds to pin Px.8 ... bit 7 to pin Px.15.	8 (byte) R/W	0x00	FIO0PIN1 - 0x3FFF C015 FIO1PIN1 - 0x3FFF C035 FIO2PIN1 - 0x3FFF C055 FIO3PIN1 - 0x3FFF C075 FIO4PIN1 - 0x3FFF C095
FIOxPIN2	Fast GPIO Port x Pin value register 2. Bit 0 in FIOxPIN2 register corresponds to pin Px.16 ... bit 7 to pin Px.23.	8 (byte) R/W	0x00	FIO0PIN2 - 0x3FFF C016 FIO1PIN2 - 0x3FFF C036 FIO2PIN2 - 0x3FFF C056 FIO3PIN2 - 0x3FFF C076 FIO4PIN2 - 0x3FFF C096
FIOxPIN3	Fast GPIO Port x Pin value register 3. Bit 0 in FIOxPIN3 register corresponds to pin Px.24 ... bit 7 to pin Px.31.	8 (byte) R/W	0x00	FIO0PIN3 - 0x3FFF C017 FIO1PIN3 - 0x3FFF C037 FIO2PIN3 - 0x3FFF C057 FIO3PIN3 - 0x3FFF C077 FIO4PIN3 - 0x3FFF C097
FIOxPINL	Fast GPIO Port x Pin value Lower half-word register. Bit 0 in FIOxPINL register corresponds to pin Px.0 ... bit 15 to pin Px.15.	16 (half-word) R/W	0x0000	FIO0PINL - 0x3FFF C014 FIO1PINL - 0x3FFF C034 FIO2PINL - 0x3FFF C054 FIO3PINL - 0x3FFF C074 FIO4PINL - 0x3FFF C094
FIOxPINU	Fast GPIO Port x Pin value Upper half-word register. Bit 0 in FIOxPINU register corresponds to pin Px.16 ... bit 15 to Px.31.	16 (half-word) R/W	0x0000	FIO0PINU - 0x3FFF C016 FIO1PINU - 0x3FFF C036 FIO2PINU - 0x3FFF C056 FIO3PINU - 0x3FFF C076 FIO4PINU - 0x3FFF C096

4.5 Fast GPIO port Mask register FIOMASK(FIO[0/1/2/3/4]MASK - 0x3FFF C0[1/3/5/7/9]0)

This register is available in the enhanced group of registers only. It is used to select port pins that will and will not be affected by write accesses to the FIOPIN, FIOSET or FIOCLR register. Mask register also filters out port's content when the FIOPIN register is read.

A zero in this register's bit enables an access to the corresponding physical pin via a read or write access. If a bit in this register is one, corresponding pin will not be changed with write access and if read, will not be reflected in the updated FIOPIN register. For software examples, see Section 10–5 "GPIO usage notes" on page 145

Table 135. Fast GPIO port Mask register (FIO[0/1/2/3/4]MASK - address 0x3FFF C0[1/3/5/7/9]0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP0xMASK, FP1xMASK, FP2xMASK, FP3xMASK FP4xMASK		Fast GPIO physical pin access control.	0x0
		0	Controlled pin is affected by writes to the port's FIOSET, FIOCLR, and FIOPIN register(s). Current state of the pin can be read from the FIOPIN register.	
		1	Controlled pin is not affected by writes into the port's FIOSET, FIOCLR and FIOPIN register(s). When the FIOPIN register is read, this bit will not be updated with the state of the physical pin.	

Aside from the 32-bit long and word only accessible FIOMASK register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 10–136, too. Next to providing the same functions as the FIOMASK register, these additional registers allow easier and faster access to the physical port pins.

Table 136. Fast GPIO port Mask byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxMASK0	Fast GPIO Port x Mask register 0. Bit 0 in FIOxMASK0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) R/W	0x0	FIO0MASK0 - 0x3FFF C010 FIO1MASK0 - 0x3FFF C030 FIO2MASK0 - 0x3FFF C050 FIO3MASK0 - 0x3FFF C070 FIO4MASK0 - 0x3FFF C090
FIOxMASK1	Fast GPIO Port x Mask register 1. Bit 0 in FIOxMASK1 register corresponds to pin Px.8 ... bit 7 to pin Px.15.	8 (byte) R/W	0x0	FIO0MASK1 - 0x3FFF C011 FIO1MASK1 - 0x3FFF C031 FIO2MASK1 - 0x3FFF C051 FIO3MASK1 - 0x3FFF C071 FIO4MASK1 - 0x3FFF C091
FIOxMASK2	Fast GPIO Port x Mask register 2. Bit 0 in FIOxMASK2 register corresponds to pin Px.16 ... bit 7 to pin Px.23.	8 (byte) R/W	0x0	FIO0MASK2 - 0x3FFF C012 FIO1MASK2 - 0x3FFF C032 FIO2MASK2 - 0x3FFF C052 FIO3MASK2 - 0x3FFF C072 FIO4MASK2 - 0x3FFF C092
FIOxMASK3	Fast GPIO Port x Mask register 3. Bit 0 in FIOxMASK3 register corresponds to pin Px.24 ... bit 7 to pin Px.31.	8 (byte) R/W	0x0	FIO0MASK3 - 0x3FFF C013 FIO1MASK3 - 0x3FFF C033 FIO2MASK3 - 0x3FFF C053 FIO3MASK3 - 0x3FFF C073 FIO4MASK3 - 0x3FFF C093
FIOxMASKL	Fast GPIO Port x Mask Lower half-word register. Bit 0 in FIOxMASKL register corresponds to pin Px.0 ... bit 15 to pin Px.15.	16 (half-word) R/W	0x0	FIO0MASKL - 0x3FFF C010 FIO1MASKL - 0x3FFF C030 FIO2MASKL - 0x3FFF C050 FIO3MASKL - 0x3FFF C070 FIO4MASKL - 0x3FFF C090
FIOxMASKU	Fast GPIO Port x Mask Upper half-word register. Bit 0 in FIOxMASKU register corresponds to pin Px.16 ... bit 15 to Px.31.	16 (half-word) R/W	0x0	FIO0MASKU - 0x3FFF C012 FIO1MASKU - 0x3FFF C032 FIO2MASKU - 0x3FFF C053 FIO3MASKU - 0x3FFF C072 FIO4MASKU - 0x3FFF C092

4.6 GPIO overall Interrupt Status register (IOIntStatus - 0xE002 8080)

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

Table 137. GPIO overall Interrupt Status register (IOIntStatus - address 0xE002 8080) bit description

Bit	Symbol	Value	Description	Reset value
0	P0Int		PORT0 GPIO interrupt pending.	0
		0	There are no pending interrupts on PORT0.	
		1	There is at least one pending interrupt on PORT0.	
1	-	-	Reserved. The value read from a reserved bit is not defined.	NA
2	P2Int		PORT2 GPIO interrupt pending.	0
		0	There are no pending interrupts on PORT2.	
		1	There is at least one pending interrupt on PORT2.	
31:2	-	-	Reserved. The value read from a reserved bit is not defined.	NA

4.7 GPIO Interrupt Enable for Rising edge register (IO0IntEnR - 0xE002 8090 and IO2IntEnR - 0xE002 B0)

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

Table 138. GPIO Interrupt Enable for Rising edge register (IO0IntEnR - address 0xE002 8090 and IO2IntEnR - address 0xE002 80B0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xER and P2xER		Enable Rising edge. Bit 0 in IOxIntEnR corresponds to pin Px.0, bit 31 in IOxIntEnR corresponds to pin Px.31.	0
		0	Rising edge interrupt is disabled on the controlled pin.	
		1	Rising edge interrupt is enabled on the controlled pin.	

4.8 GPIO Interrupt Enable for Falling edge register (IO0IntEnF - 0xE002 8094 and IO2IntEnF - 0xE002 B4)

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

Table 139. GPIO Interrupt Enable for Falling edge register (IO0IntEnF - address 0xE002 8094 and IO2IntEnF - address 0xE002 80B4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xEF and P2xEF		Enable Falling edge. Bit 0 in IOxIntEnF corresponds to pin Px.0, bit 31 in IOxIntEnF corresponds to pin Px.31.	0
		0	Falling edge interrupt is disabled on the controlled pin.	
		1	Falling edge interrupt is enabled on the controlled pin.	

4.9 GPIO Interrupt Status for Rising edge register (IO0IntStatR - 0xE002 8084 and IO2IntStatR - 0xE002 A4)

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

Table 140. GPIO Status for Rising edge register (IO0IntStatR - address 0xE002 8084 and IO2IntStatR - address 0xE002 80A4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xREI and P2xREI		Rising Edge Interrupt status. Bit 0 in IOxIntStatR corresponds to pin Px.0, bit 31 in IOxIntStatR corresponds to pin Px.31.	0
		0	Rising edge has not been detected on the corresponding pin.	
		1	An interrupt is generated due to a rising edge on the corresponding pin.	

4.10 GPIO Interrupt Status for Falling edge register (IO0IntStatF - 0xE002 8088 and IO2IntStatF - 0xE002 A8)

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

Table 141. GPIO Status for Falling edge register (IO0IntStatF - address 0xE002 8088 and IO2IntStatF - address 0xE002 80A8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xFEI and P2xFEI		Falling Edge Interrupt status. Bit 0 in IOxIntStatF corresponds to pin Px.0, bit 31 in IOxIntStatF corresponds to pin Px.31.	0
		0	Falling edge has not been detected on the corresponding pin.	
		1	An interrupt is generated due to a falling edge on the corresponding pin.	

4.11 GPIO Interrupt Clear register (IO0IntClr - 0xE002 808C and IO2IntClr - 0xE002 AC)

Writing a 1 into each bit in these write-only registers clears any interrupts for the corresponding GPIO port pin.

Table 142. GPIO Status for Falling edge register (IO0IntClr - address 0xE002 808C and IO2IntClr - address 0xE002 80AC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xCI and P2xCI		Clear GPIO port Interrupt. Bit 0 in IOxIntClr corresponds to pin Px.0, bit 31 in IOxIntClr corresponds to pin Px.31.	0
		0	Corresponding bit in IOxIntStatR and/or IOxIntStatF is unchanged.	
		1	Corresponding bit in IOxIntStatR and IOxStatF is cleared to 0.	

5. GPIO usage notes

5.1 Example 1: sequential accesses to IOSET and IOCLR affecting the same GPIO pin/bit

State of the output configured GPIO pin is determined by writes into the pin's port IOSET and IOCLR registers. Last of these accesses to the IOSET/IOCLR register will determine the final output of a pin.

In the example code:

```
IO0DIR = 0x0000 0080 ;pin P0.7 configured as output
IO0CLR = 0x0000 0080 ;P0.7 goes LOW
IO0SET = 0x0000 0080 ;P0.7 goes HIGH
IO0CLR = 0x0000 0080 ;P0.7 goes LOW
```

pin P0.7 is configured as an output (write to IO0DIR register). After this, P0.7 output is set to low (first write to IO0CLR register). Short high pulse follows on P0.7 (write access to IO0SET), and the final write to IO0CLR register sets pin P0.7 back to low level.

5.2 Example 2: an instantaneous output of 0s and 1s on a GPIO port

Write access to port's IOSET followed by write to the IOCLR register results with pins outputting 0s being slightly later then pins outputting 1s. There are systems that can tolerate this delay of a valid output, but for some applications simultaneous output of a binary content (mixed 0s and 1s) within a group of pins on a single GPIO port is required. This can be accomplished by writing to the port's IOPIN register.

Following code will preserve existing output on PORT0 pins P0.[31:16] and P0.[7:0] and at the same time set P0.[15:8] to 0xA5, regardless of the previous value of pins P0.[15:8]:

```
IO0PIN = (IO0PIN && 0xFFFF00FF) || 0x0000A500
```

The same outcome can be obtained using the fast port access.

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

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

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

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

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

```
FIO0PIN1  = 0xA5;
```

5.3 Writing to IOSET/IOCLR vs. IOPIN

Write to the IOSET/IOCLR register allows easy change of the port's selected output pin(s) to high/low level at a time. Only pin/bit(s) in the IOSET/IOCLR written with 1 will be set to high/low level, while those written as 0 will remain unaffected. However, by just writing to either IOSET or IOCLR register it is not possible to instantaneously output arbitrary binary data containing a mixture of 0s and 1s on a GPIO port.

Write to the IOPIN register enables instantaneous output of a desired content on the parallel GPIO. Binary data written into the IOPIN register will affect all output configured pins of that parallel port: 0s in the IOPIN will produce low level pin outputs and 1s in IOPIN will produce high level pin outputs. In order to change output of only a group of port's pins, application must logically AND readout from the IOPIN with mask containing 0s in bits corresponding to pins that will be changed, and 1s for all others. Finally, this result has to be logically ORred with the desired content and stored back into the IOPIN register. Example 2 from above illustrates output of 0xA5 on PORT0 pins 15 to 8 while preserving all other PORT0 output pins as they were before.

5.4 Output signal frequency considerations when using the legacy and enhanced GPIO registers

The enhanced features of the fast GPIO ports available on this microcontroller make GPIO pins more responsive to the code that has task of controlling them. In particular, software access to a GPIO pin is 3.5 times faster via the fast GPIO registers than it is when the legacy set of registers is used. As a result of the access speed increase, the maximum output frequency of the digital pin is increased 3.5 times, too. This tremendous increase of the output frequency is not always that visible when a plain C code is used, and a portion of an application handling the fast port output might have to be written in assembly code and executed in the ARM mode.

1. Introduction

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

The Ethernet block and the CPU share a dedicated AHB subsystem (AHB2) that is used to access the Ethernet SRAM for Ethernet data, control, and status information. All other AHB traffic in the LPC2400 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 External Memory Controller, as well as the SRAM located on AHB1, if is not being used by the USB block. However, using memory other than the Ethernet SRAM, especially off-chip memory, will slow Ethernet access to memory and increase the loading of AHB1.

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.

Table 143. 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 143. 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

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 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 FCS insertion (CRC) for transmit.
  - Selectable automatic transmit frame padding.
  - Over-length frame support for both transmit and receive allows any length frames.
  - Promiscuous receive mode.
  - Automatic collision backoff and frame retransmission.
  - Includes power management by clock switching.
  - Wake-on-LAN power management support allows system wake-up: using the receive filters or a magic frame detection filter.
- Physical interface:
  - Attachment of external PHY chip through standard Media Independent Interface (MII) or standard Reduced MII (RMII) interface, software selectable.
  - PHY register access is available via the Media Independent Interface Management (MIIM) interface.

3. Architecture and operation

Figure 11–20 shows the internal architecture of the Ethernet block.

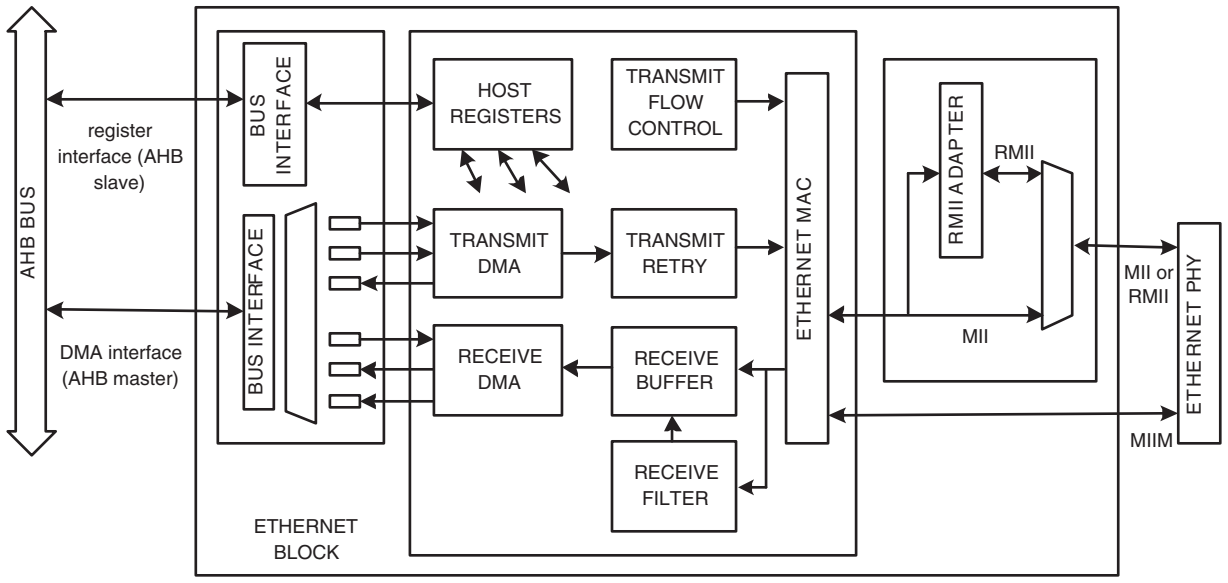


Fig 20. Ethernet block diagram

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 the Ethernet SRAM 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.

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 ARM7 CPU.

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

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

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.

6. Ethernet Packet

Figure 11–21 illustrates the different fields in an Ethernet packet.

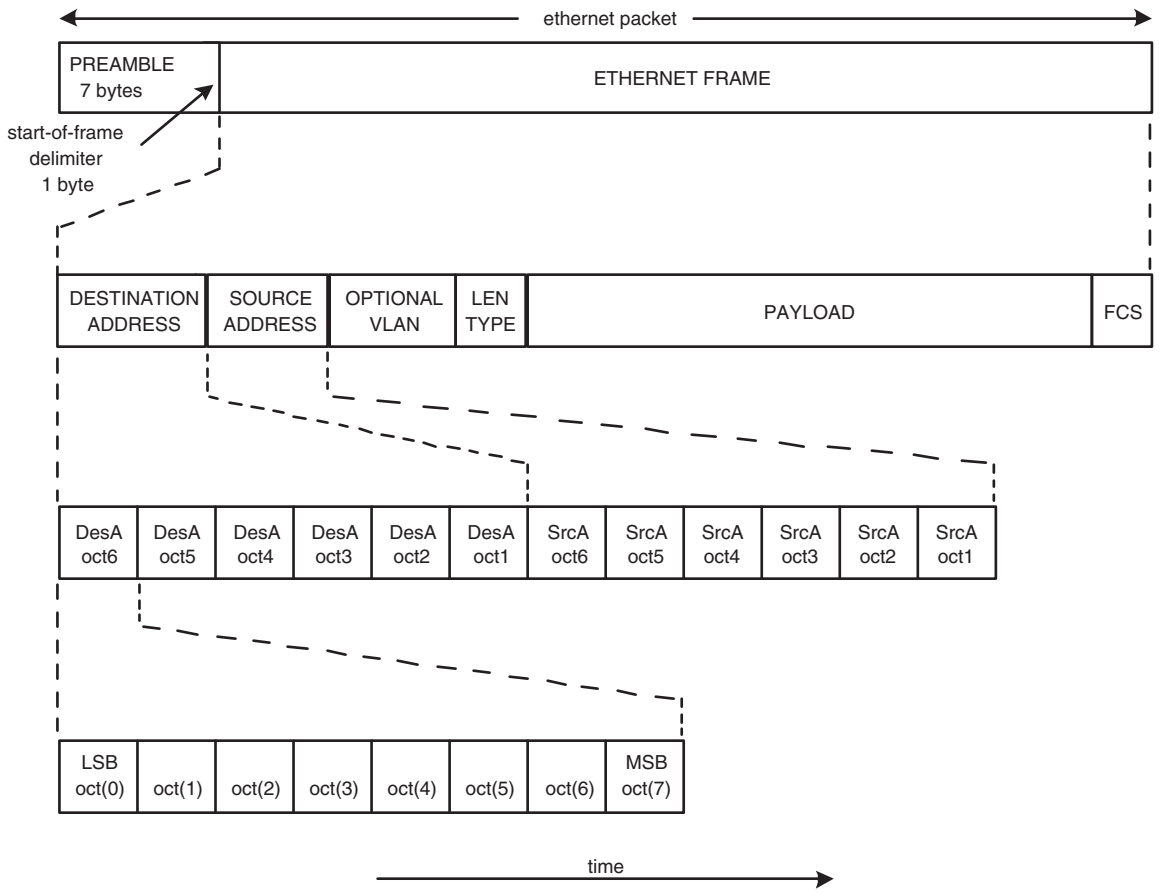


Fig 21. Ethernet packet fields

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.

7. Overview

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 prefetching and buffering.

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

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

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

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

The Ethernet block has both a standard 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) operating at 2.5 MHz.

7.2 Example PHY Devices

Some examples of compatible PHY devices are shown in Table 11–144.

Table 144. 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

8. Pin description

Table 11–146 shows the signals used for the Reduced Media Independent Interface (RMII) to the external PHY.

Table 145. 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 146. 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_TX_ER	Output	Transmit error.
ENET_TX_CLK	Input	Transmit clock.
ENET_RXD[1:0]	Input	Receive data, 2 bits.
ENET_RX_ER	Input	Receive error.
ENET_RX_DV	Input	Receive data valid.
ENET_CRS	Input	Carrier sense/data valid.
ENET_REF_CLK/ ENET_RX_CLK	Input	Reference clock

Table 11–147 shows the signals used for Media Independent Interface Management (MIIM) of the external PHY.

Table 147. Ethernet MIIM pin descriptions

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

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.

9.1 Register map

Table 11–148 lists the registers, register addresses and other basic information. The total AHB address space required is 4 kilobytes.

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

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

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

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

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

Table 148. Register definitions

Symbol	Address	R/W	Description
MAC registers			
MAC1	0xFFE0 0000	R/W	MAC configuration register 1.
MAC2	0xFFE0 0004	R/W	MAC configuration register 2.
IPGT	0xFFE0 0008	R/W	Back-to-Back Inter-Packet-Gap register.
IPGR	0xFFE0 000C	R/W	Non Back-to-Back Inter-Packet-Gap register.
CLRT	0xFFE0 0010	R/W	Collision window / Retry register.
MAXF	0xFFE0 0014	R/W	Maximum Frame register.
SUPP	0xFFE0 0018	R/W	PHY Support register.
TEST	0xFFE0 001C	R/W	Test register.
MCFG	0xFFE0 0020	R/W	MII Mgmt Configuration register.
MCMD	0xFFE0 0024	R/W	MII Mgmt Command register.
MADR	0xFFE0 0028	R/W	MII Mgmt Address register.
MWTD	0xFFE0 002C	WO	MII Mgmt Write Data register.
MRDD	0xFFE0 0030	RO	MII Mgmt Read Data register.
MIND	0xFFE0 0034	RO	MII Mgmt Indicators register.

Table 148. Register definitions

Symbol	Address	R/W	Description
-	0xFFE0 0038 to 0xFFE0 00FC	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
SA0	0xFFE0 0040	R/W	Station Address 0 register.
SA1	0xFFE0 0044	R/W	Station Address 1 register.
SA2	0xFFE0 0048	R/W	Station Address 2 register.
-	0xFFE0 004C to 0xFFE0 00FC	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
Control registers			
Command	0xFFE0 0100	R/W	Command register.
Status	0xFFE0 0104	RO	Status register.
RxDescriptor	0xFFE0 0108	R/W	Receive descriptor base address register.
RxStatus	0xFFE0 010C	R/W	Receive status base address register.
RxDescriptorNumber	0xFFE0 0110	R/W	Receive number of descriptors register.
RxProduceIndex	0xFFE0 0114	RO	Receive produce index register.
RxConsumeIndex	0xFFE0 0118	R/W	Receive consume index register.
TxDescriptor	0xFFE0 011C	R/W	Transmit descriptor base address register.
TxStatus	0xFFE0 0120	R/W	Transmit status base address register.
TxDescriptorNumber	0xFFE0 0124	R/W	Transmit number of descriptors register.
TxProduceIndex	0xFFE0 0128	R/W	Transmit produce index register.
TxConsumeIndex	0xFFE0 012C	RO	Transmit consume index register.
-	0xFFE0 0130 to 0xFFE0 0154	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
TSV0	0xFFE0 0158	RO	Transmit status vector 0 register.
TSV1	0xFFE0 015C	RO	Transmit status vector 1 register.
RSV	0xFFE0 0160	RO	Receive status vector register.
-	0xFFE0 0164 to 0xFFE0 016C	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
FlowControlCounter	0xFFE0 0170	R/W	Flow control counter register.
FlowControlStatus	0xFFE0 0174	RO	Flow control status register.
-	0xFFE0 0178 to 0xFFE0 01FC	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
Rx filter registers			
RxFliterCtrl	0xFFE0 0200		Receive filter control register.
RxFilterWoLStatus	0xFFE0 0204		Receive filter WoL status register.
RxFilterWoLClear	0xFFE0 0208		Receive filter WoL clear register.
-	0xFFE0 020C	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
HashFilterL	0xFFE0 0210		Hash filter table LSBs register.

Table 148. Register definitions

Symbol	Address	R/W	Description
HashFilterH	0xFFE0 0214		Hash filter table MSBs register.
-	0xFFE0 0218 to 0xFFE0 0FDC	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
Module control registers			
IntStatus	0xFFE0 0FE0	RO	Interrupt status register.
IntEnable	0xFFE0 0FE4	R/W	Interrupt enable register.
IntClear	0xFFE0 0FE8	WO	Interrupt clear register.
IntSet	0xFFE0 0FEC	WO	Interrupt set register.
-	0xFFE0 0FF0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
PowerDown	0xFFE0 0FF4	R/W	Power-down register.
-	0xFFE0 0FF8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

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

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

10. Ethernet MAC register definitions

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

10.1 MAC Configuration Register 1 (MAC1 - 0xFFE0 0000)

The MAC configuration register 1 (MAC1) has an address of 0xFFE0 0000. Its bit definition is shown in Table 11–149.

Table 149. MAC Configuration register 1 (MAC1 - address 0xFFE0 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

Table 149. MAC Configuration register 1 (MAC1 - address 0xFFE0 0000) bit description

Bit	Symbol	Function	Reset value
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

10.2 MAC Configuration Register 2 (MAC2 - 0xFFE0 0004)

The MAC configuration register 2 (MAC2) has an address of 0xFFE0 0004. Its bit definition is shown in Table 11–150.

Table 150. MAC Configuration register 2 (MAC2 - address 0xFFE0 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
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 11–152 - 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 11–152 - Pad Operation for more information on the various padding features. **Note:** This bit is ignored if PAD / CRC ENABLE is cleared.	0

Table 150. MAC Configuration register 2 (MAC2 - address 0xFFE0 0004) bit description

Bit	Symbol	Function	Reset value
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 11–152 - 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 151. Pad operation

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

10.3 Back-to-Back Inter-Packet-Gap Register (IPGT - 0xFFE0 0008)

The Back-to-Back Inter-Packet-Gap register (IPGT) has an address of 0xFFE0 0008. Its bit definition is shown in Table 11–152.

Table 152. Back-to-back Inter-packet-gap register (IPGT - address 0xFFE0 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

10.4 Non Back-to-Back Inter-Packet-Gap Register (IPGR - 0xFFE0 000C)

The Non Back-to-Back Inter-Packet-Gap register (IPGR) has an address of 0xFFE0 000C. Its bit definition is shown in Table 11–153.

Table 153. Non Back-to-back Inter-packet-gap register (IPGR - address 0xFFE0 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

10.5 Collision Window / Retry Register (CLRT - 0xFFE0 0010)

The Collision window / Retry register (CLRT) has an address of 0xFFE0 0010. Its bit definition is shown in Table 11–154.

Table 154. Collision Window / Retry register (CLRT - address 0xFFE0 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

10.6 Maximum Frame Register (MAXF - 0xFFE0 0014)

The Maximum Frame register (MAXF) has an address of 0xFFE0 0014. Its bit definition is shown in Table 11–155.

Table 155. Maximum Frame register (MAXF - address 0xFFE0 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

10.7 PHY Support Register (SUPP - 0xFFE0 0018)

The PHY Support register (SUPP) has an address of 0xFFE0 0018. The SUPP register provides additional control over the RMII interface. The bit definition of this register is shown in Table 11–156.

Table 156. PHY Support register (SUPP - address 0xFFE0 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.

10.8 Test Register (TEST - 0xFFE0 001C)

The Test register (TEST) has an address of 0xFFE0 001C. The bit definition of this register is shown in Table 11–157. These bits are used for testing purposes only.

Table 157. Test register (TEST - address 0xFFE0) 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

10.9 MII Mgmt Configuration Register (MCFG - 0xFFE0 0020)

The MII Mgmt Configuration register (MCFG) has an address of 0xFFE0 0020. The bit definition of this register is shown in Table 11–158.

Table 158. MII Mgmt Configuration register (MCFG - address 0xFFE0 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 12.5 MHz, however. Refer to Table 11–159 below for the definition of values for this field.	0
14:5	-	Unused	0x0
15	RESET MII MGMT	This bit resets the MII Management hardware.	0
31:16	-	Unused	0x0

Table 159. 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

10.10 MII Mgmt Command Register (MCMD - 0xFFE0 0024)

The MII Mgmt Command register (MCMD) has an address of 0xFFE0 0024. The bit definition of this register is shown in Table 11–160.

Table 160. MII Mgmt Command register (MCMD - address 0xFFE0 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

10.11 MII Mgmt Address Register (MADR - 0xFFE0 0028)

The MII Mgmt Address register (MADR) has an address of 0xFFE0 0028. The bit definition of this register is shown in Table 11–161.

Table 161. MII Mgmt Address register (MADR - address 0xFFE0 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

10.12 MII Mgmt Write Data Register (MWTD - 0xFFE0 002C)

The MII Mgmt Write Data register (MWTD) is a Write Only register with an address of 0xFFE0 002C. The bit definition of this register is shown in Table 11–162.

Table 162. MII Mgmt Write Data register (MWTD - address 0xFFE0 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

10.13 MII Mgmt Read Data Register (MRDD - 0xFFE0 0030)

The MII Mgmt Read Data register (MRDD) is a Read Only register with an address of 0xFFE0 0030. The bit definition of this register is shown in Table 11–163.

Table 163. MII Mgmt Read Data register (MRDD - address 0xFFE0 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

10.14 MII Mgmt Indicators Register (MIND - 0xFFE0 0034)

The MII Mgmt Indicators register (MIND) is a Read Only register with an address of 0xFFE0 0034. The bit definition of this register is shown in Table 11–164.

Table 164. MII Mgmt Indicators register (MIND - address 0xFFE0 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

10.15 Station Address 0 Register (SA0 - 0xFFE0 0040)

The Station Address 0 register (SA0) has an address of 0xFFE0 0040. The bit definition of this register is shown in Table 11–165.

Table 165. Station Address register (SA0 - address 0xFFE0 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 11–21.

10.16 Station Address 1 Register (SA1 - 0xFFE0 0044)

The Station Address 1 register (SA1) has an address of 0xFFE0 0044. The bit definition of this register is shown in Table 11–166.

Table 166. Station Address register (SA1 - address 0xFFE0 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 11–21.

10.17 Station Address 2 Register (SA2 - 0xFFE0 0048)

The Station Address 2 register (SA2) has an address of 0xFFE0 0048. The bit definition of this register is shown in Table 11–167.

Table 167. Station Address register (SA2 - address 0xFFE0 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 11–21.

11. Control register definitions

11.1 Command Register (Command - 0xFFE0 0100)

The Command register (Command) register has an address of 0xFFE0 0100. Its bit definition is shown in Table 11–168.

Table 168. Command register (Command - address 0xFFE0 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

Table 168. Command register (Command - address 0xFFE0 0100) bit description

Bit	Symbol	Function	Reset value
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.

11.2 Status Register (Status - 0xFFE0 0104)

The Status register (Status) is a Read Only register with an address of 0xFFE0 0104. Its bit definition is shown in Table 11–169.

Table 169. Status register (Status - address 0xFFE0 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.

11.3 Receive Descriptor Base Address Register (RxDescriptor - 0xFFE0 0108)

The Receive Descriptor base address register (RxDescriptor) has an address of 0xFFE0 0108. Its bit definition is shown in Table 11–170.

Table 170. Receive Descriptor Base Address register (RxDescriptor - address 0xFFE0 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.

11.4 Receive Status Base Address Register (RxStatus - 0xFFE0 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 171. receive Status Base Address register (RxStatus - address 0xFFE0 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’.

11.5 Receive Number of Descriptors Register (RxDescriptor - 0xFFE0 0110)

The Receive Number of Descriptors register (RxDescriptorNumber) has an address of 0xFFE0 0110. Its bit definition is shown in Table 11–172.

Table 172. Receive Number of Descriptors register (RxDescriptor - address 0xFFE0 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.

11.6 Receive Produce Index Register (RxProduceIndex - 0xFFE0 0114)

The Receive Produce Index register (RxProduceIndex) is a Read Only register with an address of 0xFFE0 0114. Its bit definition is shown in Table 11–173.

Table 173. Receive Produce Index register (RxProduceIndex - address 0xFFE0 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.

11.7 Receive Consume Index Register (RxConsumeIndex - 0xFFE0 0118)

The Receive consume index register (RxConsumeIndex) has an address of 0xFFE0 0118. Its bit definition is shown in Table 11–174.

Table 174. Receive Consume Index register (RXConsumeIndex - address 0xFFE0 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.

11.8 Transmit Descriptor Base Address Register (TxDescriptor - 0xFFE0 011C)

The Transmit Descriptor base address register (TxDescriptor) has an address of 0xFFE0 011C. Its bit definition is shown in Table 11–175.

Table 175. Transmit Descriptor Base Address register (TxDescriptor - address 0xFFE0 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.

11.9 Transmit Status Base Address Register (TxStatus - 0xFFE0 0120)

The Transmit Status base address register (TxStatus) has an address of 0xFFE0 0120. Its bit definition is shown in Table 11–176.

Table 176. Transmit Status Base Address register (TxStatus - address 0xFFE0 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.

11.10 Transmit Number of Descriptors Register (TxDescriptorNumber - 0xFFE0 0124)

The Transmit Number of Descriptors register (TxDescriptorNumber) has an address of 0xFFE0 0124. Its bit definition is shown in Table 11–177.

Table 177. Transmit Number of Descriptors register (TxDescriptorNumber - address 0xFFE0 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.

11.11 Transmit Produce Index Register (TxProduceIndex - 0xFFE0 0128)

The Transmit Produce Index register (TxProduceIndex) has an address of 0xFFE0 0128. Its bit definition is shown in Table 11–178.

Table 178. Transmit Produce Index register (TxProduceIndex - address 0xFFE0 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.

11.12 Transmit Consume Index Register (TxConsumeIndex - 0xFFE0 012C)

The Transmit Consume Index register (TxConsumeIndex) is a Read Only register with an address of 0xFFE0 012C. Its bit definition is shown in Table 11–179.

Table 179. Transmit Consume Index register (TxConsumeIndex - address 0xFFE0 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.

11.13 Transmit Status Vector 0 Register (TSV0 - 0xFFE0 0158)

The Transmit Status Vector 0 register (TSV0) is a Read Only register with an address of 0xFFE0 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 11–180 lists the bit definitions of the TSV0 register.

Table 180. Transmit Status Vector 0 register (TSV0 - address 0xFFE0 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

Table 180. Transmit Status Vector 0 register (TSV0 - address 0xFFE0 0158) bit description

Bit	Symbol	Function	Reset value
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.

11.14 Transmit Status Vector 1 Register (TSV1 - 0xFFE0 015C)

The Transmit Status Vector 1 register (TSV1) is a Read Only register with an address of 0xFFE0 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 11–181 lists the bit definitions of the TSV1 register.

Table 181. Transmit Status Vector 1 register (TSV1 - address 0xFFE0 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

11.15 Receive Status Vector Register (RSV - 0xFFE0 0160)

The Receive status vector register (RSV) is a Read Only register with an address of 0xFFE0 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 11–182 lists the bit definitions of the RSV register.

Table 182. Receive Status Vector register (RSV - address 0xFFE0 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
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.

11.16 Flow Control Counter Register (FlowControlCounter - 0xFFE0 0170)

The Flow Control Counter register (FlowControlCounter) has an address of 0xFFE0 0170. Table 11–183 lists the bit definitions of the register.

Table 183. Flow Control Counter register (FlowControlCounter - address 0xFFE0 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

11.17 Flow Control Status Register (FlowControlStatus - 0xFFE0 0174)

The Flow Control Status register (FlowControlStatus) is a Read Only register with an address of 0xFFE0 8174. Table 11–184 lists the bit definitions of the register.

Table 184. Flow Control Status register (FlowControlStatus - address 0xFFE0 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

12. Receive filter register definitions

12.1 Receive Filter Control Register (RxFilterCtrl - 0xFFE0 0200)

The Receive Filter Control register (RxFilterCtrl) has an address of 0xFFE0 0200. Table 11–185 lists the definition of the individual bits in the register.

Table 185. Receive Filter Control register (RxFilterCtrl - address 0xFFE0 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

Table 185. Receive Filter Control register (RxFilterCtrl - address 0xFFE0 0200) bit description

Bit	Symbol	Function	Reset value
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

12.2 Receive Filter WoL Status Register (RxFilterWoLStatus - 0xFFE0 0204)

The Receive Filter Wake-up on LAN Status register (RxFilterWoLStatus) is a Read Only register with an address of 0xFFE0 0204.

Table 11–186 lists the definition of the individual bits in the register.

Table 186. Receive Filter WoL Status register (RxFilterWoLStatus - address 0xFFE0 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.

12.3 Receive Filter WoL Clear Register (RxFilterWoLClear - 0xFFE0 0208)

The Receive Filter Wake-up on LAN Clear register (RxFilterWoLClear) is a Write Only register with an address of 0xFFE0 0208.

Table 11–187 lists the definition of the individual bits in the register.

Table 187. Receive Filter WoL Clear register (RxFilterWoLClear - address 0xFFE0 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.

12.4 Hash Filter Table LSBs Register (HashFilterL - 0xFFE0 0210)

The Hash Filter table LSBs register (HashFilterL) has an address of 0xFFE0 0210. Table 11–188 lists the bit definitions of the register. Details of Hash filter table use can be found in Section 11–16.10 "Receive filtering" on page 204.

Table 188. Hash Filter Table LSBs register (HashFilterL - address 0xFFE0 0210) bit description

Bit	Symbol	Function	Reset value
31:0	HashFilterL	Bit 31:0 of the imperfect filter hash table for receive filtering.	0x0

12.5 Hash Filter Table MSBs Register (HashFilterH - 0xFFE0 0214)

The Hash Filter table MSBs register (HashFilterH) has an address of 0xFFE0 0214. Table 11–189 lists the bit definitions of the register. Details of Hash filter table use can be found in Section 11–16.10 "Receive filtering" on page 204.

Table 189. Hash Filter MSBs register (HashFilterH - address 0xFFE0 0214) bit description

Bit	Symbol	Function	Reset value
31:0	HashFilterH	Bit 63:32 of the imperfect filter hash table for receive filtering.	0x0

13. Module control register definitions

13.1 Interrupt Status Register (IntStatus - 0xFFE0 0FE0)

The Interrupt Status register (IntStatus) is a Read Only register with an address of 0xFFE0 0FE0. The interrupt status register bit definition is shown in Table 11–190. 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 190. Interrupt Status register (IntStatus - address 0xFFE0 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.

13.2 Interrupt Enable Register (IntEnable - 0xFFE0 0FE4)

The Interrupt Enable register (IntEnable) has an address of 0xFFE0 0FE4. The interrupt enable register bit definition is shown in Table 11–191.

Table 191. Interrupt Enable register (intEnable - address 0xFFE0 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

13.3 Interrupt Clear Register (IntClear - 0xFFE0 0FE8)

The Interrupt Clear register (IntClear) is a Write Only register with an address of 0xFFE0 0FE8. The interrupt clear register bit definition is shown in Table 11–192.

Table 192. Interrupt Clear register (IntClear - address 0xFFE0 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
11:8	-	Unused	0x0

Table 192. Interrupt Clear register (IntClear - address 0xFFE0 0FE8) bit description

Bit	Symbol	Function	Reset value
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.

13.4 Interrupt Set Register (IntSet - 0xFFE0 0FEC)

The Interrupt Set register (IntSet) is a Write Only register with an address of 0xFFE0 0FEC. The interrupt set register bit definition is shown in Table 11–193.

Table 193. Interrupt Set register (IntSet - address 0xFFE0 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.

13.5 Power Down Register (PowerDown - 0xFFE0 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 0xFFE0 0FF4. The bit definition of the register is listed in Table 11–194.

Table 194. Power Down register (PowerDown - address 0xFFE0 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. 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.

14.1 Receive descriptors and statuses

Figure 11–22 depicts the layout of the receive descriptors in memory.

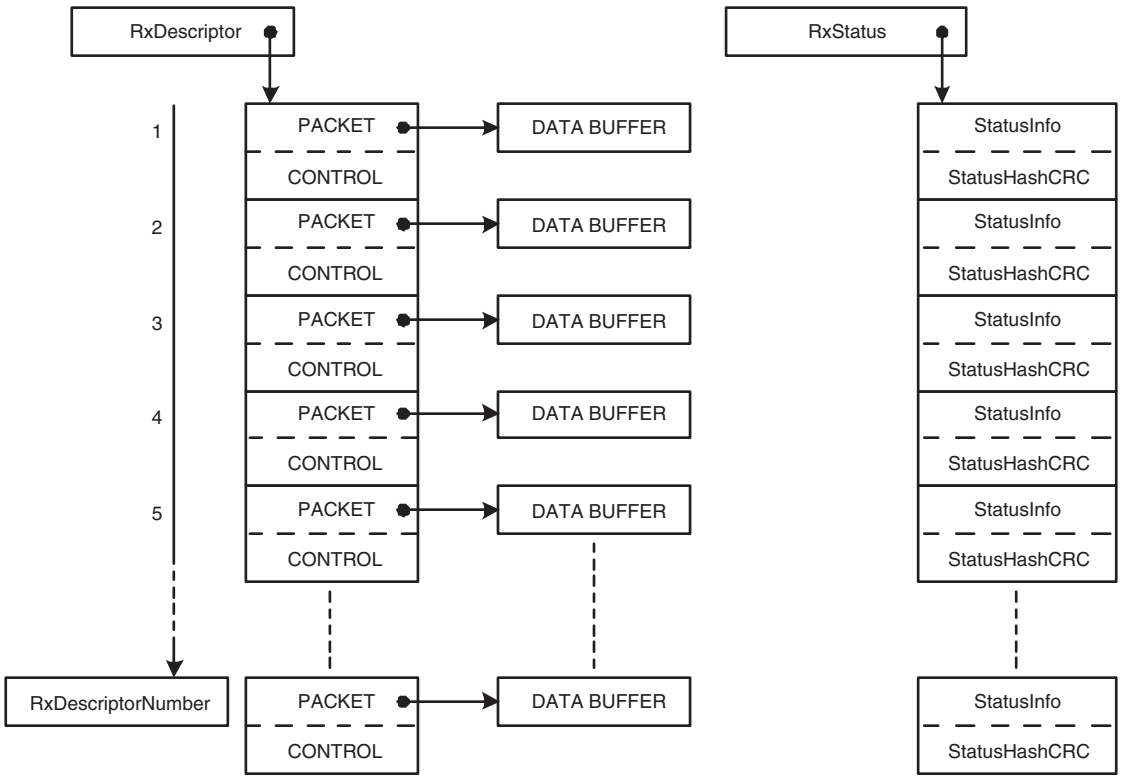


Fig 22. Receive descriptor memory layout

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 11–195.

Table 195. 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 11–196.

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 11–196.

Table 196. 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 11–197 lists the fields in the receive status elements from the status array.

Table 197. Receive Status Fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Receive status return flags, see Table 11–199.
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 11–198:

Table 198. 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 11–199 lists the bit definitions in the StatusInfo word.

Table 199. 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.

14.2 Transmit descriptors and statuses

Figure 11–23 depicts the layout of the transmit descriptors in memory.

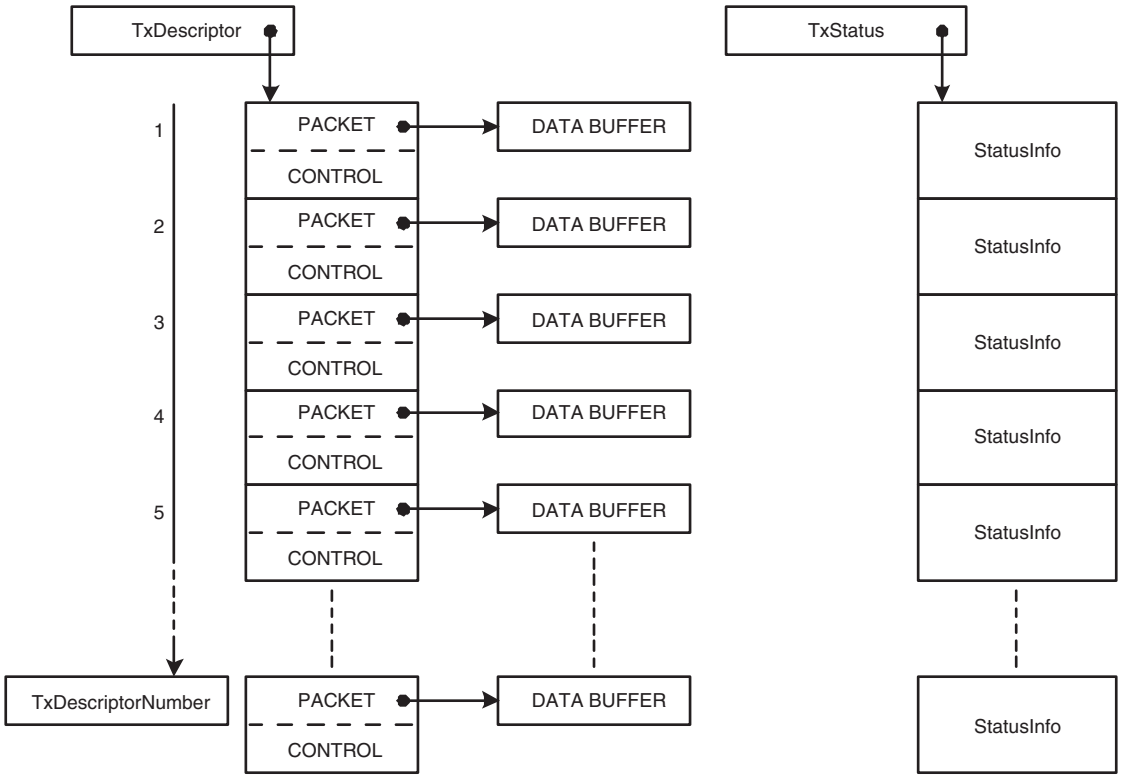


Fig 23. Transmit descriptor memory layout

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 11–200.

Table 200. 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 11–201.

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 11–201.

Table 201. 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 11–202 shows the one field transmit status.

Table 202. Transmit status fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Transmit status return flags, see Table 11–203.

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 11–203 lists the bit definitions in the StatusInfo word.

Table 203. 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.

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.

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.

To transmit a packet the software driver has to set up the appropriate Control registers and a descriptor to point to the packet data buffer before transferring the packet to hardware by incrementing the TxProduceIndex register. After transmission, hardware will increment TxConsumeIndex and optionally generate an interrupt.

The hardware will receive packets from the PHY and apply filtering as configured by the software driver. While receiving a packet the hardware will read a descriptor from memory to find the location of the associated receiver data buffer. Receive data is written in the data buffer and receive status is returned in the receive descriptor status word. Optionally an interrupt can be generated to notify software that a packet has been received. Note that the DMA manager will prefetch and buffer up to three descriptors.

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 11–148 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 11–148.
- If the PowerDown bit is set all accesses to AHB registers will result in an error response except for accesses to the PowerDown register.

16. Interrupts

The Ethernet block has a single interrupt request output to the CPU (via the Vectored 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.

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.

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
- 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.

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 11–24 illustrates the transmit process in an example transmitting uses a frame header of 8 bytes and a frame payload of 12 bytes.

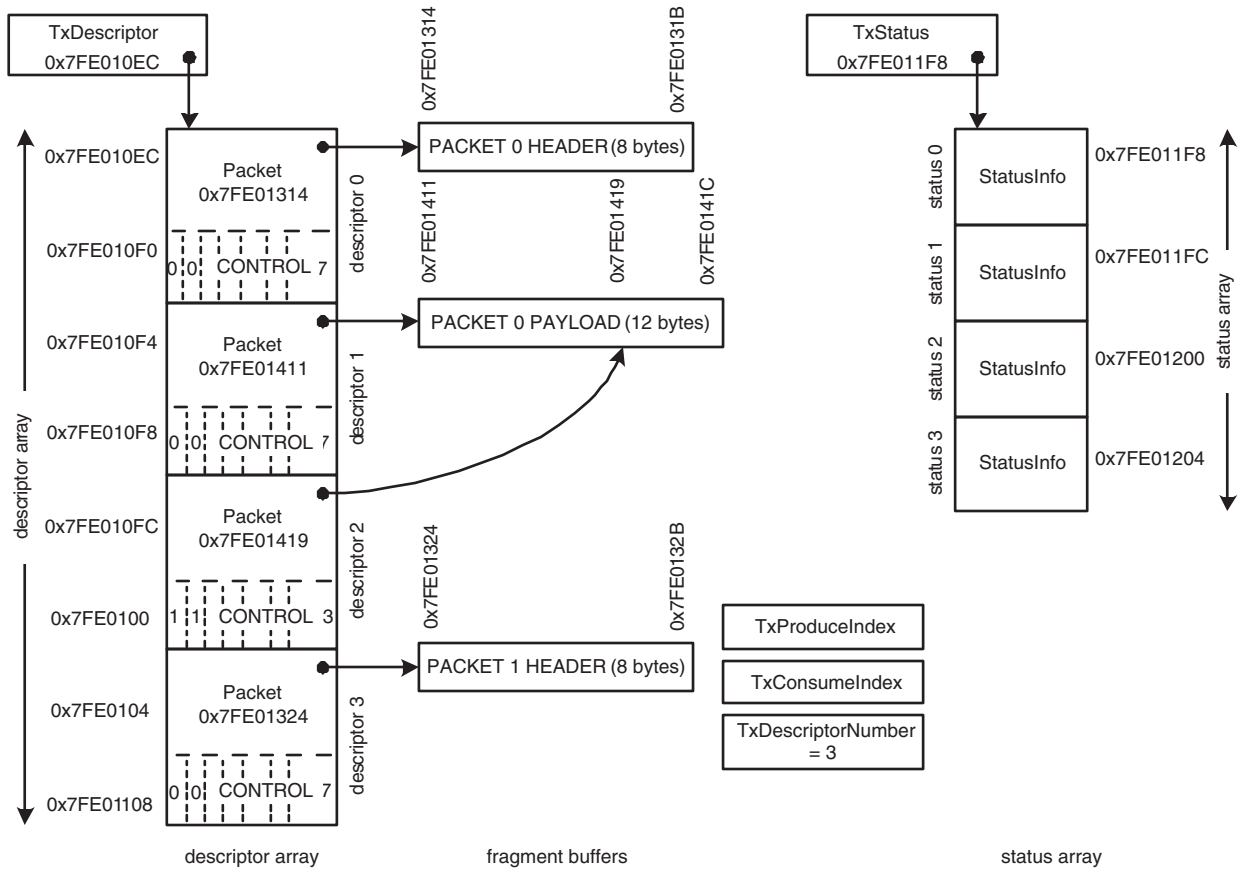


Fig 24. Transmit example memory and registers

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 (0x7FE0 10EC) to the TxDescriptor register and the base address of the status array

(0x7FE0 11F8) to the TxStatus register. The device driver writes the number of descriptors and statuses minus 1(3) to the TxDescriptorNumber register. The descriptors and statuses in the arrays need not be initialized, yet.

At this point, the transmit 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.

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 11–25 illustrates the receive process in an example receiving a frame of 19 bytes.

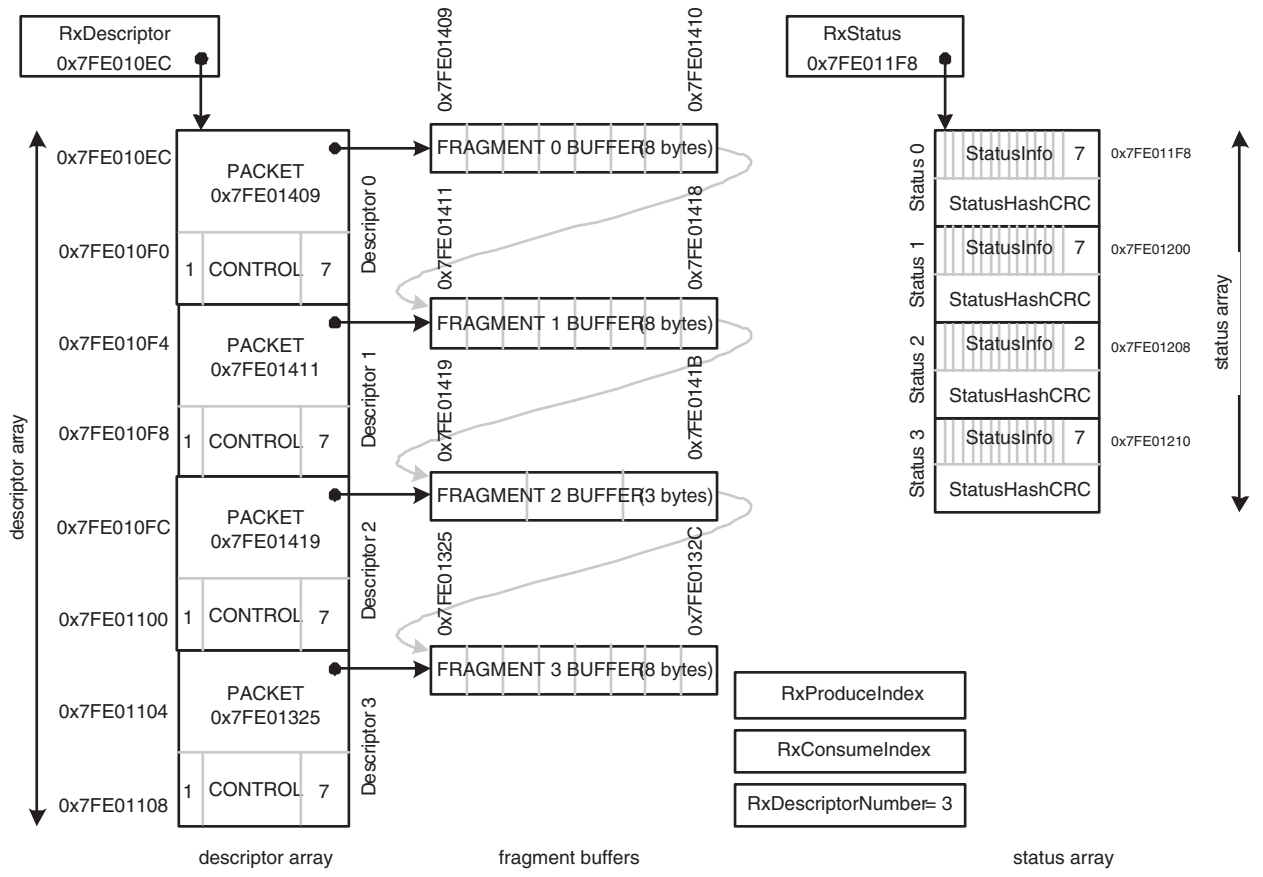


Fig 25. Receive Example Memory and Registers

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 (0xFEED B0EC) in the RxDescriptor register, and the base address of the status array (0xFEED B1F8) 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).

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.

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.

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.

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 11–26 illustrates the transmit flow control.

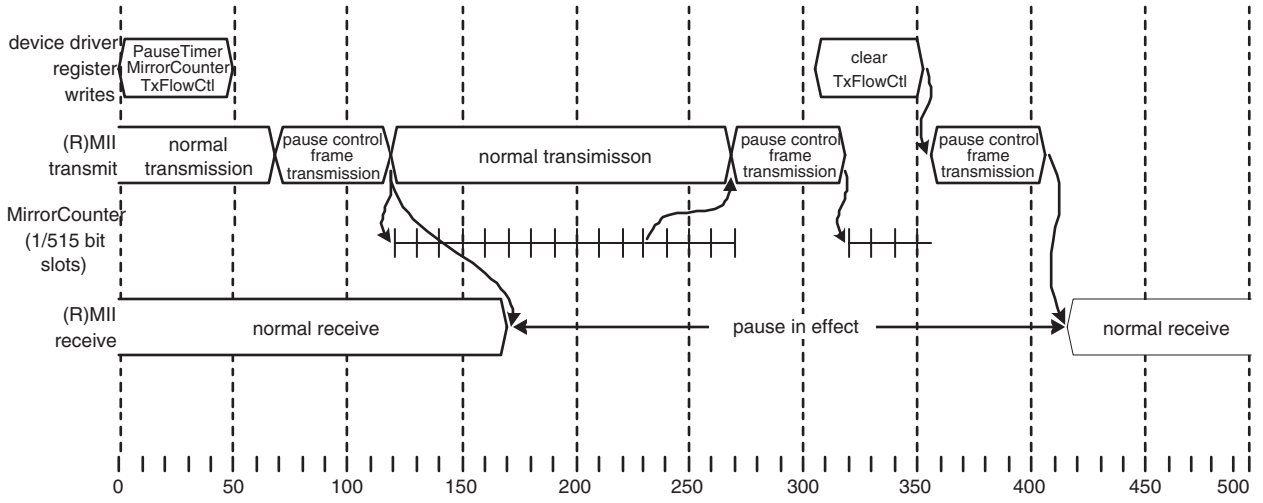


Fig 26. Transmit Flow Control

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.

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.

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 11–27 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.

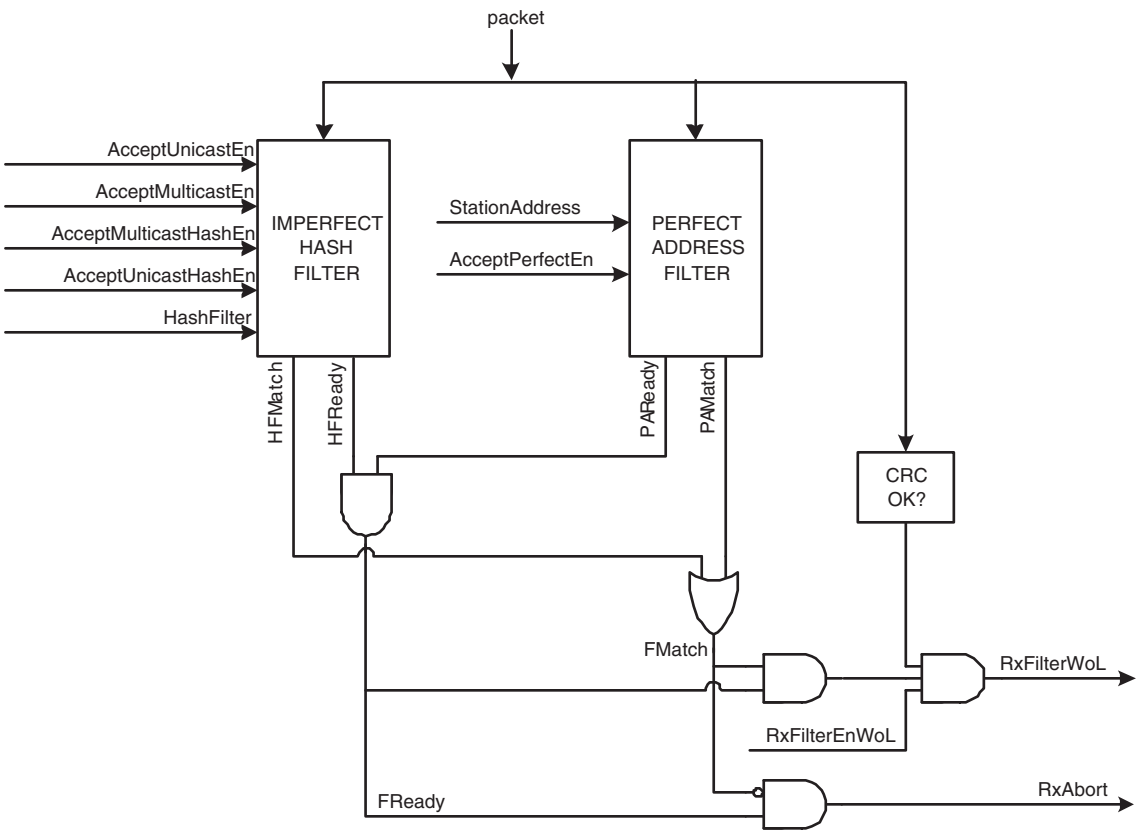


Fig 27. Receive filter block diagram

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.

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.

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 11–27 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 11–27: 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>
```

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 11–28 illustrates the state machine for the generation of the RxStatus bit.

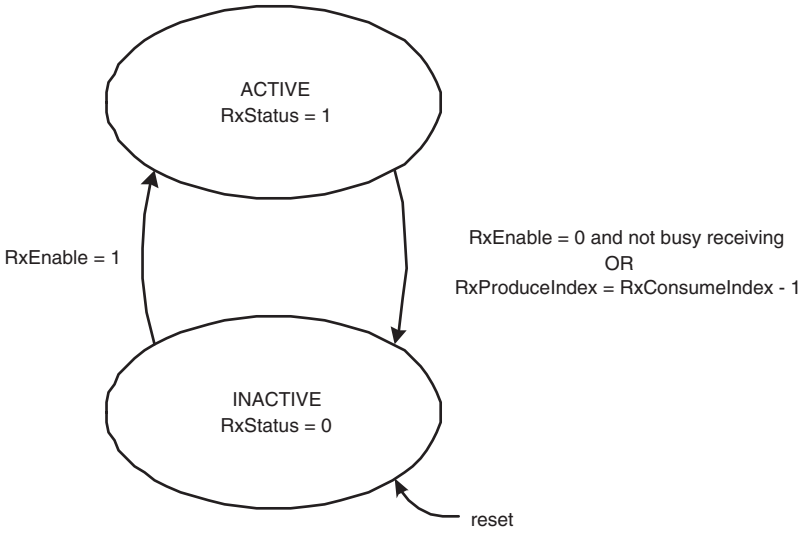


Fig 28. Receive Active/Inactive state machine

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 11–28, 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 11–29 illustrates the state machine for the generation of the TxStatus bit.

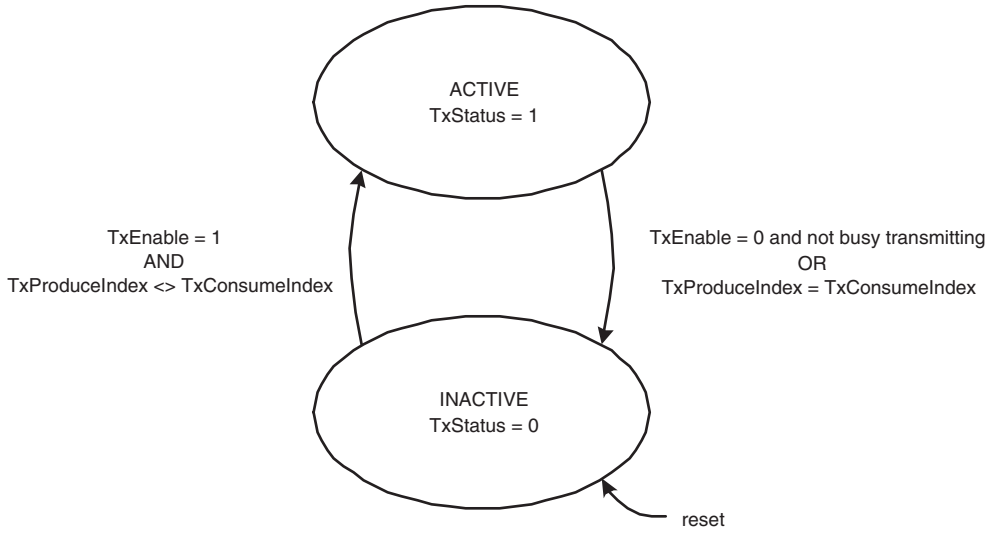


Fig 29. Transmit Active/Inactive state machine

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 11–29, 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.

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.

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.

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.

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.

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.

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).

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.

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 Mbps.

The Ethernet block initiates DMA accesses for the following cases:

- Tx descriptor read:
  - Transmit descriptors occupy 2 words (8 bytes) of memory and are read once for each use of a descriptor.
  - Two word read happens once every 64 bytes (16 words) of transmitted data.
  - This gives 1/8th of the data rate, which = 1.5625 Mbps.
- Rx descriptor read:
  - Receive descriptors occupy 2 words (8 bytes) of memory and are read once for each use of a descriptor.
  - Two word read happens once every 64 bytes (16 words) of received data.
  - This gives 1/8th of the data rate, which = 1.5625 Mbps.
- Tx status write:
  - Transmit status occupies 1 word (4 bytes) of memory and is written once for each use of a descriptor.
  - One word write happens once every 64 bytes (16 words) of transmitted data.
  - This gives 1/16th of the data rate, which = 0.7813 Mbps.
- Rx status write:
  - Receive status occupies 2 words (8 bytes) of memory and is written once for each use of a descriptor.
  - Two word write happens once every 64 bytes (16 words) of received data.
  - This gives 1/8 of the data rate, which = 1.5625 Mbps.
- Tx data read:
  - Data transmitted in an Ethernet frame, the size is variable.

- Basic Ethernet rate = 12.5 Mbps.

- Rx data write:
  - Data to be received in an Ethernet frame, the size is variable.
  - Basic Ethernet rate = 12.5 Mbps.

This gives a total rate of 30.5 Mbps for the traffic generated by the Ethernet DMA function.

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 Mbps.
- Access to registers in the Ethernet block:
  - The CPU must read the RxProduceIndex, TxConsumeIndex, and IntStatus registers, and both read and write the RxConsumeIndex and TxProduceIndex registers.
  - 7 word read/writes once every 64 bytes (16 words) of transmitted and received data.
  - This gives 7/16 of the data rate, which = 5.4688 Mbps.

This gives a total rate of 36 Mbps for the traffic generated by the Ethernet DMA function.

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 set at 60 MHz. All or nearly all of bus accesses related to the Ethernet will be word transfers. The raw AHB bandwidth can be approximated as 4 bytes per two system clocks, which equals 2 times the system clock rate. With a 60 MHz system clock, the bandwidth is 120 MB/s, giving about 55% utilization for Ethernet traffic during simultaneous transmit and receive operations.

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.

1. CAN controllers

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The CAN Controller is designed to provide a full implementation of the CAN-Protocol according to the CAN Specification Version 2.0B. Microcontrollers with this on-chip CAN controller are used to build powerful local networks by supporting distributed real-time control with a very high level of security. The applications are automotive, industrial environments, and high speed networks as well as low cost multiplex wiring. The result is a strongly reduced wiring harness and enhanced diagnostic and supervisory capabilities.

The CAN block is intended to support multiple CAN buses simultaneously, allowing the device to be used as a gateway, switch, or router among a number of CAN buses in various applications.

The CAN module consists of two elements: the controller and the Acceptance Filter. All registers and the RAM are accessed as 32 bit words.

2. Features

2.1 General CAN features

- Compatible with *CAN specification 2.0B, ISO 11898-1*.
- Multi-master architecture with non destructive bit-wise arbitration.
- Bus access priority determined by the message identifier (11-bit or 29-bit).
- Guaranteed latency time for high priority messages.
- Programmable transfer rate (up to 1 Mbit/s).
- Multicast and broadcast message facility.
- Data length from 0 up to 8 bytes.
- Powerful error handling capability.
- Non-return-to-zero (NRZ) coding/decoding with bit stuffing.

2.2 CAN controller features

- 2 CAN controllers and buses.
- Supports 11-bit identifier as well as 29-bit identifier.
- Double Receive Buffer and Triple Transmit Buffer.
- Programmable Error Warning Limit and Error Counters with read/write access.
- Arbitration Lost Capture and Error Code Capture with detailed bit position.
- Single Shot Transmission (no re-transmission).
- Listen Only Mode (no acknowledge, no active error flags).
- Reception of "own" messages (Self Reception Request).

2.3 Acceptance filter features

- Fast hardware implemented search algorithm supporting a large number of CAN identifiers.
- Global Acceptance Filter recognizes 11 and 29 bit Rx Identifiers for all CAN buses.
- Allows definition of explicit and groups for 11-bit and 29-bit CAN identifiers.
- Acceptance Filter can provide FullCAN-style automatic reception for selected Standard Identifiers.

3. Pin description

Table 204. CAN Pin descriptions

Pin Name	Type	Description
RD1, RD2	Input	**Serial Inputs.** From CAN transceivers.
TD1, TD2	Output	**Serial Outputs.** To CAN transceivers.

4. CAN controller architecture

The CAN Controller is a complete serial interface with both Transmit and Receive Buffers but without Acceptance Filter. CAN Identifier filtering is done for all CAN channels in a separate block (Acceptance Filter). Except for message buffering and acceptance filtering the functionality is similar to the PeliCAN concept.

The CAN Controller Block includes interfaces to the following blocks:

- APB Interface
- Acceptance Filter
- Vectored Interrupt Controller (VIC)
- CAN Transceiver
- Common Status Registers

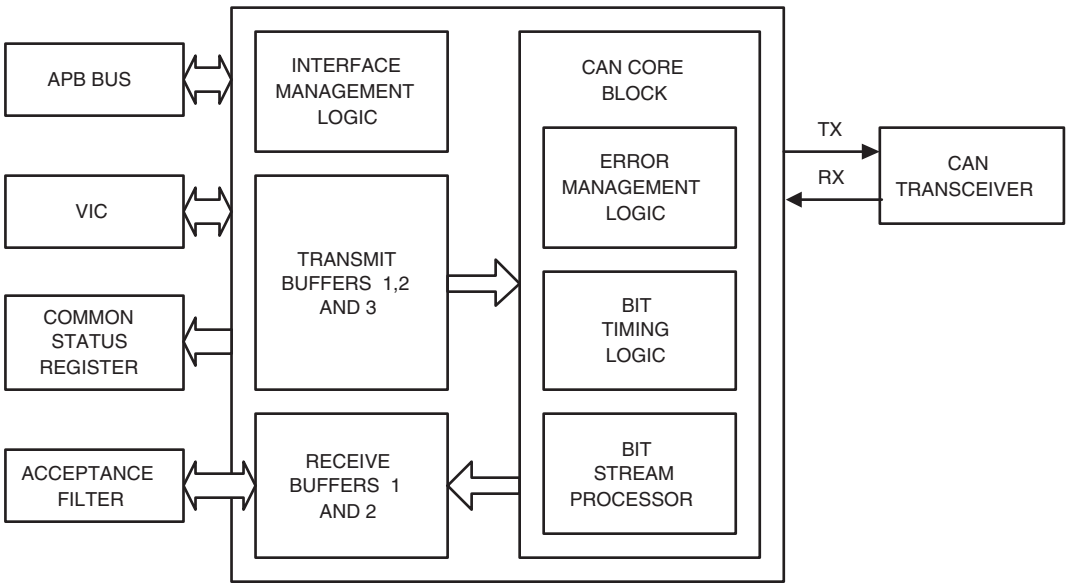


Fig 30. CAN controller block diagram

4.1 APB Interface Block (AIB)

The APB Interface Block provides access to all CAN Controller registers.

4.2 Interface Management Logic (IML)

The Interface Management Logic interprets commands from the CPU, controls internal addressing of the CAN Registers and provides interrupts and status information to the CPU.

4.3 Transmit Buffers (TXB)

The TXB represents a Triple Transmit Buffer, which is the interface between the Interface Management Logic (IML) and the Bit Stream Processor (BSP). Each Transmit Buffer is able to store a complete message which can be transmitted over the CAN network. This buffer is written by the CPU and read out by the BSP.

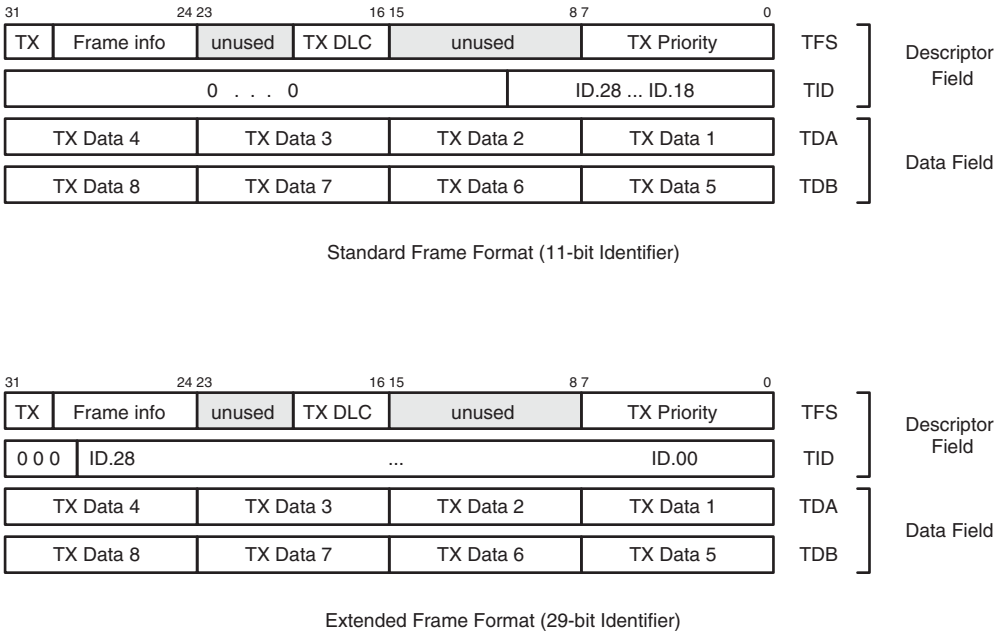


Fig 31. Transmit buffer layout for standard and extended frame format configurations

4.4 Receive Buffer (RXB)

The Receive Buffer (RXB) represents a CPU accessible Double Receive Buffer. It is located between the CAN Controller Core Block and APB Interface Block and stores all received messages from the CAN Bus line. With the help of this Double Receive Buffer concept the CPU is able to process one message while another message is being received.

The global layout of the Receive Buffer is very similar to the Transmit Buffer described earlier. Identifier, Frame Format, Remote Transmission Request bit and Data Length Code have the same meaning as described for the Transmit Buffer. In addition, the Receive Buffer includes an ID Index field (see Section 12–6.9.1 “ID index field”).

The received Data Length Code represents the real transmitted Data Length Code, which may be greater than 8 depending on transmitting CAN node. Nevertheless, the maximum number of received data bytes is 8. This should be taken into account by reading a message from the Receive Buffer. If there is not enough space for a new message within the Receive Buffer, the CAN Controller generates a Data Overrun condition when this message becomes valid and the acceptance test was positive. A message that is partly written into the Receive Buffer (when the Data Overrun situation occurs) is deleted. This situation is signalled to the CPU via the Status Register and the Data Overrun Interrupt, if enabled.

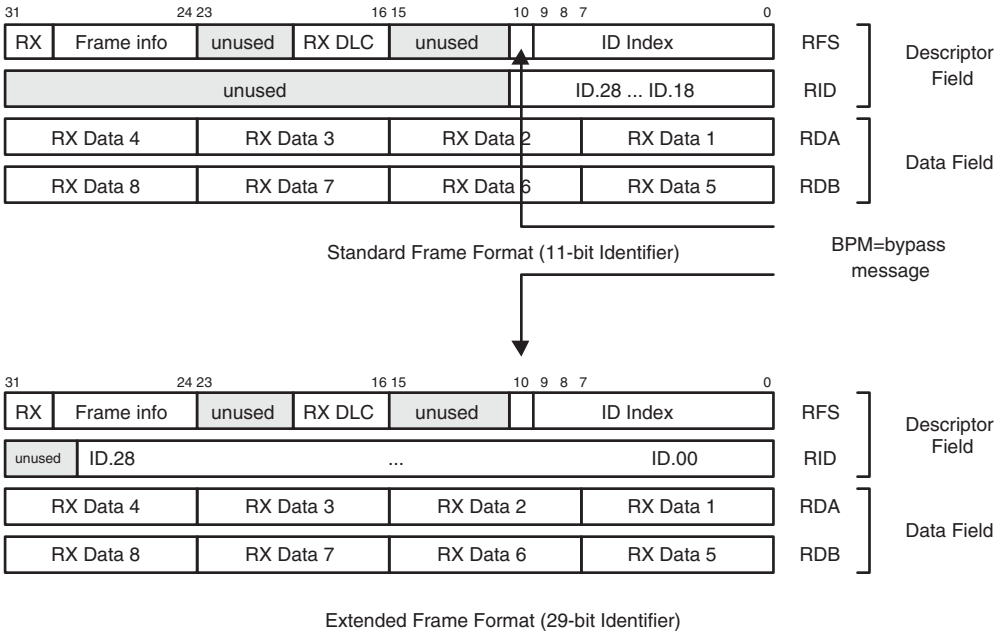


Fig 32. Receive buffer layout for standard and extended frame format configurations

4.5 Error Management Logic (EML)

The EML is responsible for the error confinement. It gets error announcements from the BSP and then informs the BSP and IML about error statistics.

4.6 Bit Timing Logic (BTL)

The Bit Timing Logic monitors the serial CAN Bus line and handles the Bus line related bit timing. It synchronizes to the bit stream on the CAN Bus on a "recessive" to "dominant" Bus line transition at the beginning of a message (hard synchronization) and re-synchronizes on further transitions during the reception of a message (soft synchronization). The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g. due to oscillator drifts) and to define the sample point and the number of samples to be taken within a bit time.

4.7 Bit Stream Processor (BSP)

The Bit Stream Processor is a sequencer, controlling the data stream between the Transmit Buffer, Receive Buffers and the CAN Bus. It also performs the error detection, arbitration, stuffing and error handling on the CAN Bus.

4.8 CAN controller self-tests

The CAN controller of the LPC2000 family supports two different options for self-tests:

- Global Self-Test (setting the self reception request bit in normal Operating Mode)
- Local Self-Test (setting the self reception request bit in Self Test Mode)

Both self-tests are using the ‘Self Reception’ feature of the CAN Controller. With the Self Reception Request, the transmitted message is also received and stored in the receive buffer. Therefore the acceptance filter has to be configured accordingly. As soon as the CAN message is transmitted, a transmit and a receive interrupt are generated, if enabled.

Global self test

A Global Self-Test can for example be used to verify the chosen configuration of the CAN Controller in a given CAN system. As shown in Figure 12–33, at least one other CAN node, which is acknowledging each CAN message has to be connected to the CAN bus.

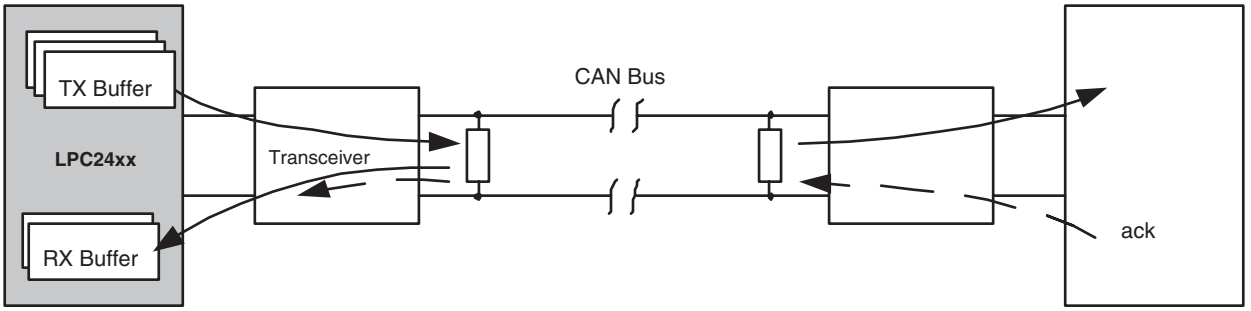


Fig 33. Global Self-Test (high-speed CAN Bus example)

Initiating a Global Self-Test is similar to a normal CAN transmission. In this case the transmission of a CAN message(s) is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer bits (STB3, STB2, STB1) in the CAN Controller Command register (CANCMR).

Local self test

The Local Self-Test perfectly fits for single node tests. In this case an acknowledge from other nodes is not needed. As shown in the Figure below, a CAN transceiver with an appropriate CAN bus termination has to be connected to the LPC. The CAN Controller has to be put into the 'Self Test Mode' by setting the STM bit in the CAN Controller Mode register (CANMOD). Hint: Setting the Self Test Mode bit (STM) is possible only when the CAN Controller is in Reset Mode.

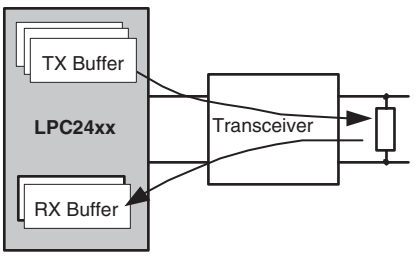


Fig 34. Local self test (high-speed CAN Bus example)

A message transmission is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer(s) (STB3, STB2, STB1).

5. Memory map of the CAN block

The CAN Controllers and Acceptance Filter occupy a number of APB slots, as follows:

Table 205. Memory map of the CAN block

Address Range	Used for
0xE003 8000 - 0xE003 87FF	Acceptance Filter RAM.
0xE003 C000 - 0xE003 C017	Acceptance Filter Registers.
0xE004 0000 - 0xE004 000B	Central CAN Registers.
0xE004 4000 - 0xE004 405F	CAN Controller 1 Registers.
0xE004 8000 - 0xE004 805F	CAN Controller 2 Registers.

6. CAN controller registers

CAN block implements the registers shown in Table 12–206 and Table 12–207. More detailed descriptions follow.

Table 206. CAN acceptance filter and central CAN registers

Name	Description	Access	Reset Value	Address
AFMR	Acceptance Filter Register	R/W	1	0xE003 C000
SFF_sa	Standard Frame Individual Start Address Register	R/W	0	0xE003 C004
SFF_GRP_sa	Standard Frame Group Start Address Register	R/W	0	0xE003 C008
EFF_sa	Extended Frame Start Address Register	R/W	0	0xE003 C00C
EFF_GRP_sa	Extended Frame Group Start Address Register	R/W	0	0xE003 C010
ENDofTable	End of AF Tables register	R/W	0	0xE003 C014
LUTerrAd	LUT Error Address register	RO	0	0xE003 C018
LUTerr	LUT Error Register	RO	0	0xE003 C01C
CANTxSR	CAN Central Transmit Status Register	RO	0x0003 0300	0xE004 0000
CANRxSR	CAN Central Receive Status Register	RO	0	0xE004 0004
CANMSR	CAN Central Miscellaneous Register	RO	0	0xE004 0008

Table 207. CAN1 and CAN2 controller register map

Generic Name	Description	Access	CAN1 Register Address & Name	CAN2 Register Address & Name
MOD	Controls the operating mode of the CAN Controller.	R/W	CAN1MOD - 0xE004 4000	CAN2MOD - 0xE004 8000
CMR	Command bits that affect the state of the CAN Controller	WO	CAN1CMR - 0xE004 4004	CAN2CMR - 0xE004 8004
GSR	Global Controller Status and Error Counters	RO[1]	CAN1GSR - 0xE004 4008	CAN2GSR - 0xE004 8008
ICR	Interrupt status, Arbitration Lost Capture, Error Code Capture	RO	CAN1ICR - 0xE004 400C	CAN2ICR - 0xE004 800C
IER	Interrupt Enable	R/W	CAN1IER - 0xE004 4010	CAN2IER - 0xE004 8010
BTR	Bus Timing	R/W[2]	CAN1BTR - 0xE004 4014	CAN2BTR - 0xE004 8014
EWL	Error Warning Limit	R/W[2]	CAN1EWL - 0xE004 4018	CAN2EWL - 0xE004 8018
SR	Status Register	RO	CAN1SR - 0xE004 401C	CAN2SR - 0xE004 801C
RFS	Receive frame status	R/W[2]	CAN1RFS - 0xE004 4020	CAN2RFS - 0xE004 8020

Table 207. CAN1 and CAN2 controller register map

Generic Name	Description	Access	CAN1 Register Address & Name	CAN2 Register Address & Name
RID	Received Identifier	R/W[2]	CAN1RID - 0xE004 4024	CAN2RID - 0xE004 8024
RDA	Received data bytes 1-4	R/W[2]	CAN1RDA - 0xE004 4028	CAN2RDA - 0xE004 8028
RDB	Received data bytes 5-8	R/W[2]	CAN1RDB - 0xE004 402C	CAN2RDB - 0xE004 802C
TFI1	Transmit frame info (Tx Buffer 1)	R/W	CAN1TFI1 - 0xE004 4030	CAN2TFI1 - 0xE004 8030
TID1	Transmit Identifier (Tx Buffer 1)	R/W	CAN1TID1 - 0xE004 4034	CAN2TID1 - 0xE004 8034
TDA1	Transmit data bytes 1-4 (Tx Buffer 1)	R/W	CAN1TDA1 - 0xE004 4038	CAN2TDA1 - 0xE004 8038
TDB1	Transmit data bytes 5-8 (Tx Buffer 1)	R/W	CAN1TDB1- 0xE004 403C CAN2TDB1- 0xE004 803C	CAN1TDB1 - 0xE004 403C CAN2TDB1 - 0xE004 803C
TFI2	Transmit frame info (Tx Buffer 2)	R/W	CAN1TFI2 - 0xE004 4040 CAN2TFI2 - 0xE004 8040	CAN1TFI2 - 0xE004 4040 CAN2TFI2 - 0xE004 8040
TID2	Transmit Identifier (Tx Buffer 2)	R/W	CAN1TID2 - 0xE004 4044 CAN2TID2 - 0xE004 8044	CAN1TID2 - 0xE004 4044 CAN2TID2 - 0xE004 8044
TDA2	Transmit data bytes 1-4 (Tx Buffer 2)	R/W	CAN1TDA2 - 0xE004 4048 CAN2TDA2 - 0xE004 8048	CAN1TDA2 - 0xE004 4048 CAN2TDA2 - 0xE004 8048
TDB2	Transmit data bytes 5-8 (Tx Buffer 2)	R/W	CAN1TDB2 - 0xE004 404C CAN2TDB2 - 0xE004 804C	CAN1TDB2 - 0xE004 404C CAN2TDB2 - 0xE004 804C
TFI3	Transmit frame info (Tx Buffer 3)	R/W	CAN1TFI3 - 0xE004 4050 CAN2TFI3 - 0xE004 8050	CAN1TFI3 - 0xE004 4050 CAN2TFI3 - 0xE004 8050
TID3	Transmit Identifier (Tx Buffer 3)	R/W	CAN1TID3 - 0xE004 4054 CAN2TID3 - 0xE004 8054	CAN1TID3 - 0xE004 4054 CAN2TID3 - 0xE004 8054
TDA3	Transmit data bytes 1-4 (Tx Buffer 3)	R/W	CAN1TDA3 - 0xE004 4058 CAN2TDA3 - 0xE004 8058	CAN1TDA3 - 0xE004 4058 CAN2TDA3 - 0xE004 8058
TDB3	Transmit data bytes 5-8 (Tx Buffer 3)	R/W	CAN1TDB3 - 0xE004 405C CAN2TDB3 - 0xE004 805C	CAN1TDB3 - 0xE004 405C CAN2TDB3 - 0xE004 805C

[1] The error counters can only be written when RM in CANMOD is 1.

[2] These registers can only be written when RM in CANMOD is 1.

The internal registers of each CAN Controller appear to the CPU as on-chip memory mapped peripheral registers. Because the CAN Controller can operate in different modes (Operating/Reset, see also Section 12–6.1 “Mode Register (CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000)”), one has to distinguish between different internal address definitions. Note that write access to some registers is only allowed in Reset Mode.

Table 208. CAN1 and CAN2 controller register map

Generic Name	Operating Mode		Reset Mode	
	Read	Write	Read	Write
MOD	Mode	Mode	Mode	Mode
CMR	0x00	Command	0x00	Command
GSR	Global Status and Error Counters	-	Global Status and Error Counters	Error Counters only
ICR	Interrupt and Capture	-	Interrupt and Capture	-
IER	Interrupt Enable	Interrupt Enable	Interrupt Enable	Interrupt Enable
BTR	Bus Timing	-	Bus Timing	Bus Timing
EWL	Error Warning Limit	-	Error Warning Limit	Error Warning Limit

Table 208. CAN1 and CAN2 controller register map

Generic Name	Operating Mode		Reset Mode	
	Read	Write	Read	Write
SR	Status	-	Status	-
RFS	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
RID	Rx Identifier	-	Rx Identifier	Rx Identifier
RDA	Rx Data	-	Rx Data	Rx Data
RDB	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
TFI1	Tx Info1	Tx Info	Tx Info	Tx Info
TID1	Tx Identifier	Tx Identifier	Tx Identifier	Tx Identifier
TDA1	Tx Data	Tx Data	Tx Data	Tx Data
TDB1	Tx Data	Tx Data	Tx Data	Tx Data

In the following register tables, the column “Reset Value” shows how a hardware reset affects each bit or field, while the column “RM Set” indicates how each bit or field is affected if software sets the RM bit, or RM is set because of a Bus-Off condition. Note that while hardware reset sets RM, in this case the setting noted in the “Reset Value” column prevails over that shown in the “RM Set” column, in the few bits where they differ. In both columns, X indicates the bit or field is unchanged.

6.1 Mode Register (CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000)

The contents of the Mode Register are used to change the behavior of the CAN Controller. Bits may be set or reset by the CPU that uses the Mode Register as a read/write memory. Reserved Bits are read as 0 and should be written as 0.

Table 209. Mode register (CAN1MOD - address 0xE004 4000, CAN2MOD - address 0xE004 8000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RM[1][6]		Reset Mode.	1	1
		0(normal)	The CAN Controller is in the Operating Mode, and certain registers can not be written.		
		1(reset)	CAN operation is disabled, writable registers can be written and the current transmission/reception of a message is aborted.		
1	LOM[3][2][6]		Listen Only Mode.	0	x
		0(normal)	The CAN controller acknowledges a successfully received message on the CAN bus. The error counters are stopped at the current value.		
		1(listen only)	The controller gives no acknowledgment, even if a message is successfully received. Messages cannot be sent, and the controller operates in “error passive” mode. This mode is intended for software bit rate detection and “hot plugging”.		
2	STM[3][6]		Self Test Mode.	0	x
		0(normal)	A transmitted message must be acknowledged to be considered successful.		
		1(self test)	The controller will consider a Tx message successful even if there is no acknowledgment received. In this mode a full node test is possible without any other active node on the bus using the SRR bit in CANxCMR.		

Table 209. Mode register (CAN1MOD - address 0xE004 4000, CAN2MOD - address 0xE004 8000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
3	TPM[4]		Transmit Priority Mode.	0	x
		0(CAN ID)	The transmit priority for 3 Transmit Buffers depends on the CAN Identifier.		
		1(local prio)	The transmit priority for 3 Transmit Buffers depends on the contents of the Tx Priority register within the Transmit Buffer.		
4	SM[5]		Sleep Mode.	0	0
		0(wake-up)	Normal operation.		
		1(sleep)	The CAN controller enters Sleep Mode if no CAN interrupt is pending and there is no bus activity. See the Sleep Mode description Section 12–7.2 on page 245.		
5	RPM		Receive Polarity Mode.	0	x
		0(low active)	RD input is active Low (dominant bit = 0).		
		1(high active)	RD input is active High (dominant bit = 1) -- reverse polarity.		
6	-	-	Reserved, user software should not write ones to reserved bits.	0	0
7	TM		Test Mode.	0	x
		0(disabled)	Normal operation.		
		1(enabled)	The TD pin will reflect the bit, detected on RD pin, with the next positive edge of the system clock.		

[1] During a Hardware reset or when the Bus Status bit is set '1' (Bus-Off), the Reset Mode bit is set '1' (present). After the Reset Mode bit is set '0' the CAN Controller will wait for:

- one occurrence of Bus-Free signal (11 recessive bits), if the preceding reset has been caused by a Hardware reset or a CPU-initiated reset.

- 128 occurrences of Bus-Free, if the preceding reset has been caused by a CAN Controller initiated Bus-Off, before re-entering the Bus-On mode.

[2] This mode of operation forces the CAN Controller to be error passive. Message Transmission is not possible. The Listen Only Mode can be used e.g. for software driven bit rate detection and "hot plugging".

[3] A write access to the bits MOD.1 and MOD.2 is possible only if the Reset Mode is entered previously.

[4] Transmit Priority Mode is explained in more detail in Section 12–4.3 “Transmit Buffers (TXB)”.

[5] The CAN Controller will enter Sleep Mode, if the Sleep Mode bit is set '1' (sleep), there is no bus activity, and none of the CAN interrupts is pending. Setting of SM with at least one of the previously mentioned exceptions valid will result in a wake-up interrupt. The CAN Controller will wake up if SM is set LOW (wake-up) or there is bus activity. On wake-up, a Wake-up Interrupt is generated. A sleeping CAN Controller which wakes up due to bus activity will not be able to receive this message until it detects 11 consecutive recessive bits (Bus-Free sequence). Note that setting of SM is not possible in Reset Mode. After clearing of Reset Mode, setting of SM is possible only when Bus-Free is detected again.

[6] The LOM and STM bits can only be written if the RM bit is 1 prior to the write operation.

6.2 Command Register (CAN1CMR - 0xE004 x004, CAN2CMR - 0xE004 8004)

Writing to this write-only register initiates an action within the transfer layer of the CAN Controller. Bits not listed should be written as 0. Reading this register yields zeroes.

At least one internal clock cycle is needed for processing between two commands.

Table 210. Command Register (CAN1CMR - address 0xE004 4004, CAN2CMR - address 0xE004 8004) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0[1][2]	TR		Transmission Request.	0	0
		0 (absent)	No transmission request.		
		1 (present)	The message, previously written to the CANxTFI, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer. If at two or all three of STB1, STB2 and STB3 bits are selected when TR=1 is written, Transmit Buffer will be selected based on the chosen priority scheme (for details see Section 12–4.3 “Transmit Buffers (TXB)”)		
1[1][3]	AT		Abort Transmission.	0	0
		0 (no action)	Do not abort the transmission.		
		1 (present)	if not already in progress, a pending Transmission Request for the selected Transmit Buffer is cancelled.		
2[4]	RRB		Release Receive Buffer.	0	0
		0 (no action)	Do not release the receive buffer.		
		1 (released)	The information in the Receive Buffer (consisting of CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers) is released, and becomes eligible for replacement by the next received frame. If the next received frame is not available, writing this command clears the RBS bit in the Status Register(s).		
3[5]	CDO		Clear Data Overrun.	0	0
		0 (no action)	Do not clear the data overrun bit.		
		1 (clear)	The Data Overrun bit in Status Register(s) is cleared.		
4[1][6]	SRR		Self Reception Request.	0	0
		0 (absent)	No self reception request.		
		1 (present)	The message, previously written to the CANxTFS, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer and received simultaneously. This differs from the TR bit above in that the receiver is not disabled during the transmission, so that it receives the message if its Identifier is recognized by the Acceptance Filter.		
5	STB1		Select Tx Buffer 1.	0	0
		0 (not selected)	Tx Buffer 1 is not selected for transmission.		
		1 (selected)	Tx Buffer 1 is selected for transmission.		
6	STB2		Select Tx Buffer 2.	0	0
		0 (not selected)	Tx Buffer 2 is not selected for transmission.		
		1 (selected)	Tx Buffer 2 is selected for transmission.		
7	STB3		Select Tx Buffer 3.	0	0
		0 (not selected)	Tx Buffer 3 is not selected for transmission.		
		1 (selected)	Tx Buffer 3 is selected for transmission.		

[1] - Setting the command bits TR and AT simultaneously results in transmitting a message once. No re-transmission will be performed in case of an error or arbitration lost (single shot transmission).
- Setting the command bits SRR and TR simultaneously results in sending the transmit message once using the self-reception feature. No re-transmission will be performed in case of an error or arbitration lost.
- Setting the command bits TR, AT and SRR simultaneously results in transmitting a message once as described for TR and AT. The moment the Transmit Status bit is set within the Status Register, the internal Transmission Request Bit is cleared automatically.
- Setting TR and SRR simultaneously will ignore the set SRR bit.

[2] If the Transmission Request or the Self-Reception Request bit was set '1' in a previous command, it cannot be cancelled by resetting the bits. The requested transmission may only be cancelled by setting the Abort Transmission bit.

[3] The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit a more urgent message before. A transmission already in progress is not stopped. In order to see if the original message has been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. This should be done after the Transmit Buffer Status bit has been set to '1' or a Transmit Interrupt has been generated.

[4] After reading the contents of the Receive Buffer, the CPU can release this memory space by setting the Release Receive Buffer bit '1'. This may result in another message becoming immediately available. If there is no other message available, the Receive Interrupt bit is reset. If the RRB command is given, it will take at least 2 internal clock cycles before a new interrupt is generated.

[5] This command bit is used to clear the Data Overrun condition signalled by the Data Overrun Status bit. As long as the Data Overrun Status bit is set no further Data Overrun Interrupt is generated.

[6] Upon Self Reception Request, a message is transmitted and simultaneously received if the Acceptance Filter is set to the corresponding identifier. A receive and a transmit interrupt will indicate correct self reception (see also Self Test Mode in Section 12–6.1 “Mode Register (CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000)”).

6.3 Global Status Register (CAN1GSR - 0xE004 x008, CAN2GSR - 0xE004 8008)

The content of the Global Status Register reflects the status of the CAN Controller. This register is read-only, except that the Error Counters can be written when the RM bit in the CANMOD register is 1. Bits not listed read as 0 and should be written as 0.

Table 211. Global Status Register (CAN1GSR - address 0xE004 4008, CAN2GSR - address 0xE004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS[1]		Receive Buffer Status.	0	0
		0 (empty)	No message is available.		
		1 (full)	At least one complete message is received by the Double Receive Buffer and available in the CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers. This bit is cleared by the Release Receive Buffer command in CANxCMR, if no subsequent received message is available.		
1	DOS[2]		Data Overrun Status.	0	0
		0 (absent)	No data overrun has occurred since the last Clear Data Overrun command was given/written to CANxCMR (or since Reset).		
		1 (overrun)	A message was lost because the preceding message to this CAN controller was not read and released quickly enough (there was not enough space for a new message in the Double Receive Buffer).		
2	TBS		Transmit Buffer Status.	1	1
		0 (locked)	At least one of the Transmit Buffers is not available for the CPU, i.e. at least one previously queued message for this CAN controller has not yet been sent, and therefore software should not write to the CANxTFI, CANxTID, CANxTDA, nor CANxTDB registers of that (those) Tx buffer(s).		
		1 (released)	All three Transmit Buffers are available for the CPU. No transmit message is pending for this CAN controller (in any of the 3 Tx buffers), and software may write to any of the CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS[3]		Transmit Complete Status.	1	x
		0 (incomplete)	At least one requested transmission has not been successfully completed yet.		
		1 (complete)	All requested transmission(s) has (have) been successfully completed.		

Table 211. Global Status Register (CAN1GSR - address 0xE004 4008, CAN2GSR - address 0xE004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
4	RS[4]		Receive Status.	1	0
		0 (idle)	The CAN controller is idle.		
		1 (receive)	The CAN controller is receiving a message.		
5	TS[4]		Transmit Status.	1	0
		0 (idle)	The CAN controller is idle.		
		1 (transmit)	The CAN controller is sending a message.		
6	ES[5]		Error Status.	0	0
		0 (ok)	Both error counters are below the Error Warning Limit.		
		1 (error)	One or both of the Transmit and Receive Error Counters has reached the limit set in the Error Warning Limit register.		
7	BS[6]		Bus Status.	0	0
		0 (Bus-On)	The CAN Controller is involved in bus activities		
		1 (Bus-Off)	The CAN controller is currently not involved/prohibited from bus activity because the Transmit Error Counter reached its limiting value of 255.		
15:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
23:16	RXERR	-	The current value of the Rx Error Counter (an 8 - bit value).	0	X
31:24	TXERR	-	The current value of the Tx Error Counter (an 8 - bit value).	0	X

[1] After reading all messages and releasing their memory space with the command 'Release Receive Buffer,' this bit is cleared.

[2] If there is not enough space to store the message within the Receive Buffer, that message is dropped and the Data Overrun condition is signalled to the CPU in the moment this message becomes valid. If this message is not completed successfully (e.g. because of an error), no overrun condition is signalled.

[3] The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' at least for one of the three Transmit Buffers. The Transmission Complete Status bit will remain '0' until all messages are transmitted successfully.

[4] If both the Receive Status and the Transmit Status bits are '0' (idle), the CAN-Bus is idle. If both bits are set, the controller is waiting to become idle again. After hardware reset 11 consecutive recessive bits have to be detected until idle status is reached. After Bus-off this will take 128 times of 11 consecutive recessive bits.

[5] Errors detected during reception or transmission will effect the error counters according to the CAN specification. The Error Status bit is set when at least one of the error counters has reached or exceeded the Error Warning Limit. An Error Warning Interrupt is generated, if enabled. The default value of the Error Warning Limit after hardware reset is 96 decimal, see also Section 12–6.7 “Error Warning Limit Register (CAN1EWL - 0xE004 4018, CAN2EWL - 0xE004 8018)”.

[6] Mode bit '1' (present) and an Error Warning Interrupt is generated, if enabled. Afterwards the Transmit Error Counter is set to '127', and the Receive Error Counter is cleared. It will stay in this mode until the CPU clears the Reset Mode bit. Once this is completed the CAN Controller will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) counting down the Transmit Error Counter. After that, the Bus Status bit is cleared (Bus-On), the Error Status bit is set '0' (ok), the Error Counters are reset, and an Error Warning Interrupt is generated, if enabled. Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery.

RX error counter

The RX Error Counter Register, which is part of the Status Register, reflects the current value of the Receive Error Counter. After hardware reset this register is initialized to 0. In Operating Mode this register appears to the CPU as a read only memory. A write access to this register is possible only in Reset Mode. If a Bus Off event occurs, the RX Error Counter is initialized to 0. As long as Bus Off is valid, writing to this register has no effect.The Rx Error Counter is determined as follows:

RX Error Counter = (CANxGSR AND 0x00FF0000) / 0x00010000

Note that a CPU-forced content change of the RX Error Counter is possible only if the Reset Mode was entered previously. An Error Status change (Status Register), an Error Warning or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

TX error counter

The TX Error Counter Register, which is part of the Status Register, reflects the current value of the Transmit Error Counter. In Operating Mode this register appears to the CPU as a read only memory. After hardware reset this register is initialized to '0'. A write access to this register is possible only in Reset Mode. If a bus-off event occurs, the TX Error Counter is initialized to 127 to count the minimum protocol-defined time (128 occurrences of the Bus-Free signal). Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery. If Bus Off is active, a write access to TXERR in the range of 0 to 254 clears the Bus Off Flag and the controller will wait for one occurrence of 11 consecutive recessive bits (bus free) after clearing of Reset Mode. The Tx error counter is determined as follows:

TX Error Counter = (CANxGSR AND 0xFF000000) / 0x01000000

Writing 255 to TXERR allows initiation of a CPU-driven Bus Off event. Note that a CPU-forced content change of the TX Error Counter is possible only if the Reset Mode was entered previously. An Error or Bus Status change (Status Register), an Error Warning, or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again. After leaving the Reset Mode, the new TX Counter content is interpreted and the Bus Off event is performed in the same way as if it was forced by a bus error event. That means, that the Reset Mode is entered again, the TX Error Counter is initialized to 127, the RX Counter is cleared, and all concerned Status and Interrupt Register Bits are set. Clearing of Reset Mode now will perform the protocol defined Bus Off recovery sequence (waiting for 128 occurrences of the Bus-Free signal). If the Reset Mode is entered again before the end of Bus Off recovery (TXERR>0), Bus Off keeps active and TXERR is frozen.

6.4 Interrupt and Capture Register (CAN1ICR - 0xE004 400C, CAN2ICR - 0xE004 800C)

Bits in this register indicate information about events on the CAN bus. This register is read-only. Bits not listed read as 0 and should be written as 0.

The Interrupt flags of the Interrupt and Capture Register allow the identification of an interrupt source. When one or more bits are set, a CAN interrupt will be indicated to the CPU. After this register is read from the CPU all interrupt bits are reset **except** of the Receive Interrupt bit. The Interrupt Register appears to the CPU as a read only memory.

Bits 1 thru 10 clear when they are read.

Bits 16-23 are captured when a bus error occurs. At the same time, if the BEIE bit in CANIER is 1, the BEI bit in this register is set, and a CAN interrupt can occur.

Bits 24-31 are captured when CAN arbitration is lost. At the same time, if the ALIE bit in CANIER is 1, the ALI bit in this register is set, and a CAN interrupt can occur. Once either of these bytes is captured, its value will remain the same until it is read, at which time it is released to capture a new value.

The clearing of bits 1 to 10 and the releasing of bits 16-23 and 24-31 all occur on any read from CANxICR, regardless of whether part or all of the register is read. This means that software should always read CANxICR as a word, and process and deal with all bits of the register as appropriate for the application.

Table 212. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RI[1]	0 (reset) 1 (set)	Receive Interrupt. This bit is set whenever the RBS bit in CANxSR and the RIE bit in CANxIER are both 1, indicating that a new message was received and stored in the Receive Buffer.	0	0
1	TI1	0 (reset) 1 (set)	Transmit Interrupt 1. This bit is set when the TBS1 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB1 was successfully transmitted or aborted), indicating that Transmit buffer 1 is available, and the TIE1 bit in CANxIER is 1.	0	0
2	EI	0 (reset) 1 (set)	Error Warning Interrupt. This bit is set on every change (set or clear) of either the Error Status or Bus Status bit in CANxSR and the EIE bit bit is set within the Interrupt Enable Register at the time of the change.	0	X
3	DOI	0 (reset) 1 (set)	Data Overrun Interrupt. This bit is set when the DOS bit in CANxSR goes from 0 to 1 and the DOIE bit in CANxIER is 1.	0	0
4	WUI[2]	0 (reset) 1 (set)	Wake-Up Interrupt. This bit is set if the CAN controller is sleeping and bus activity is detected and the WUIE bit in CANxIER is 1.	0	0
5	EPI	0 (reset) 1 (set)	Error Passive Interrupt. This bit is set if the EPIE bit in CANxIER is 1, and the CAN controller switches between Error Passive and Error Active mode in either direction. This is the case when the CAN Controller has reached the Error Passive Status (at least one error counter exceeds the CAN protocol defined level of 127) or if the CAN Controller is in Error Passive Status and enters the Error Active Status again.	0	0
6	ALI	0 (reset) 1 (set)	Arbitration Lost Interrupt. This bit is set if the ALIE bit in CANxIER is 1, and the CAN controller loses arbitration while attempting to transmit. In this case the CAN node becomes a receiver.	0	0
7	BEI	0 (reset) 1 (set)	Bus Error Interrupt -- this bit is set if the BEIE bit in CANxIER is 1, and the CAN controller detects an error on the bus.	0	X

Table 212. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
8	IDI	0 (reset) 1 (set)	ID Ready Interrupt -- this bit is set if the IDIE bit in CANxIER is 1, and a CAN Identifier has been received (a message was successfully transmitted or aborted). This bit is set whenever a message was successfully transmitted or aborted and the IDIE bit is set in the IER reg.	0	0
9	TI2	0 (reset) 1 (set)	Transmit Interrupt 2. This bit is set when the TBS2 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB2 was successfully transmitted or aborted), indicating that Transmit buffer 2 is available, and the TIE2 bit in CANxIER is 1.	0	0
10	TI3	0 (reset) 1 (set)	Transmit Interrupt 3. This bit is set when the TBS3 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB3 was successfully transmitted or aborted), indicating that Transmit buffer 3 is available, and the TIE3 bit in CANxIER is 1.	0	0
15:11	-	-	Reserved, user software should not write ones to reserved bits.	0	0

Table 212. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
20:16	ERRBIT 4:0[3]		Error Code Capture: when the CAN controller detects a bus error, the location of the error within the frame is captured in this field. The value reflects an internal state variable, and as a result is not very linear:	0	X
		00011	Start of Frame		
		00010	ID28 ... ID21		
		00110	ID20 ... ID18		
		00100	SRTR Bit		
		00101	IDE bit		
		00111	ID17 ... 13		
		01111	ID12 ... ID5		
		01110	ID4 ... ID0		
		01100	RTR Bit		
		01101	Reserved Bit 1		
		01001	Reserved Bit 0		
		01011	Data Length Code		
		01010	Data Field		
		01000	CRC Sequence		
		11000	CRC Delimiter		
		11001	Acknowledge Slot		
		11011	Acknowledge Delimiter		
		11010	End of Frame		
		10010	Intermission		
		10001	Active Error Flag		
		10110	Passive Error Flag		
		10011	Tolerate Dominant Bits		
		10111	Error Delimiter		
		11100	Overload flag		
21	ERRDIR		When the CAN controller detects a bus error, the direction of the current bit is captured in this bit.	0	X
		0	Error occurred during transmitting.		
		1	Error occurred during receiving.		
23:22	ERRC1:0		When the CAN controller detects a bus error, the type of error is captured in this field:	0	X
		00	Bit error		
		01	Form error		
		10	Stuff error		
		11	Other error		

Table 212. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
31:24	ALCBIT[4]	-	Each time arbitration is lost while trying to send on the CAN, the bit number within the frame is captured into this field. After the content of ALCBIT is read, the ALI bit is cleared and a new Arbitration Lost interrupt can occur.	0	X
		00	arbitration lost in the first bit (MS) of identifier		
		...	a		
		11	arbitration lost in SRTS bit (RTR bit for standard frame messages)		
		12	arbitration lost in IDE bit		
		13	arbitration lost in 12th bit of identifier (extended frame only)		
		...			
		30	arbitration lost in last bit of identifier (extended frame only)		
		31	arbitration lost in RTR bit (extended frame only)		

[1] The Receive Interrupt Bit is not cleared upon a read access to the Interrupt Register. Giving the Command “Release Receive Buffer” will clear RI temporarily. If there is another message available within the Receive Buffer after the release command, RI is set again. Otherwise RI remains cleared.

[2] A Wake-Up Interrupt is also generated if the CPU tries to set the Sleep bit while the CAN controller is involved in bus activities or a CAN Interrupt is pending. The WUI flag can also get asserted when the according enable bit WUIE is not set. In this case a Wake-Up Interrupt does not get asserted.

[3] Whenever a bus error occurs, the corresponding bus error interrupt is forced, if enabled. At the same time, the current position of the Bit Stream Processor is captured into the Error Code Capture Register. The content within this register is fixed until the user software has read out its content once. From now on, the capture mechanism is activated again, i.e. reading the CANxICR enables another Bus Error Interrupt.

[4] On arbitration lost, the corresponding arbitration lost interrupt is forced, if enabled. At that time, the current bit position of the Bit Stream Processor is captured into the Arbitration Lost Capture Register. The content within this register is fixed until the user application has read out its contents once. From now on, the capture mechanism is activated again.

6.5 Interrupt Enable Register (CAN1IER - 0xE004 4010, CAN2IER - 0xE004 8010)

This read/write register controls whether various events on the CAN controller will result in an interrupt or not. Bits 10:0 in this register correspond 1-to-1 with bits 10:0 in the CANxICR register. If a bit in the CANxIER register is 0 the corresponding interrupt is disabled; if a bit in the CANxIER register is 1 the corresponding source is enabled to trigger an interrupt.

Table 213. Interrupt Enable Register (CAN1IER - address 0xE004 4010, CAN2IER - address 0xE004 8010) bit description

Bit	Symbol	Function	Reset Value	RM Set
0	RIE	Receiver Interrupt Enable. When the Receive Buffer Status is 'full', the CAN Controller requests the respective interrupt.	0	X
1	TIE1	Transmit Interrupt Enable for Buffer1. When a message has been successfully transmitted out of TXB1 or Transmit Buffer 1 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
2	EIE	Error Warning Interrupt Enable. If the Error or Bus Status change (see Status Register), the CAN Controller requests the respective interrupt.	0	X
3	DOIE	Data Overrun Interrupt Enable. If the Data Overrun Status bit is set (see Status Register), the CAN Controller requests the respective interrupt.	0	X
4	WUIE	Wake-Up Interrupt Enable. If the sleeping CAN controller wakes up, the respective interrupt is requested.	0	X
5	EPIE	Error Passive Interrupt Enable. If the error status of the CAN Controller changes from error active to error passive or vice versa, the respective interrupt is requested.	0	X
6	ALIE	Arbitration Lost Interrupt Enable. If the CAN Controller has lost arbitration, the respective interrupt is requested.	0	X
7	BEIE	Bus Error Interrupt Enable. If a bus error has been detected, the CAN Controller requests the respective interrupt.	0	X
8	IDIE	ID Ready Interrupt Enable. When a CAN identifier has been received, the CAN Controller requests the respective interrupt.	0	X
9	TIE2	Transmit Interrupt Enable for Buffer2. When a message has been successfully transmitted out of TXB2 or Transmit Buffer 2 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
10	TIE3	Transmit Interrupt Enable for Buffer3. When a message has been successfully transmitted out of TXB3 or Transmit Buffer 3 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

6.6 Bus Timing Register (CAN1BTR - 0xE004 4014, CAN2BTR - 0xE004 8014)

This register controls how various CAN timings are derived from the APB clock. It defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). Furthermore, it defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. It can be read at any time but can only be written if the RM bit in CANmod is 1.

Table 214. Bus Timing Register (CAN1BTR - address 0xE004 4014, CAN2BTR - address 0xE004 8014) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
9:0	BRP		Baud Rate Prescaler. The APB clock is divided by (this value plus one) to produce the CAN clock.	0	X
13:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
15:14	SJW		The Synchronization Jump Width is (this value plus one) CAN clocks.	0	X
19:16	TESG1		The delay from the nominal Sync point to the sample point is (this value plus one) CAN clocks.	1100	X
22:20	TESG2		The delay from the sample point to the next nominal sync point is (this value plus one) CAN clocks. The nominal CAN bit time is (this value plus the value in TSEG1 plus 3) CAN clocks.	001	X
23	SAM		Sampling		
		0	The bus is sampled once (recommended for high speed buses)	0	X
		1	The bus is sampled 3 times (recommended for low to medium speed buses to filter spikes on the bus-line)		
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Baud rate prescaler

The period of the CAN system clock t_{SCL} is programmable and determines the individual bit timing. The CAN system clock t_{SCL} is calculated using the following equation:

(1)

$$t_{SCL} = t_{CANsuppliedCLK} \times (BRP + 1)$$

Synchronization jump width

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width t_{SJW} defines the maximum number of clock cycles a certain bit period may be shortened or lengthened by one re-synchronization:

(2)

$$t_{SJW} = t_{SCL} \times (SJW + 1)$$

Time segment 1 and time segment 2

Time segments TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

(3)

$$t_{SYNCSEG} = t_{SCL}$$

(4)

$$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$$

(5)

$$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$$

6.7 Error Warning Limit Register (CAN1EWL - 0xE004 4018, CAN2EWL - 0xE004 8018)

This register sets a limit on Tx or Rx errors at which an interrupt can occur. It can be read at any time but can only be written if the RM bit in CANmod is 1. The default value (after hardware reset) is 96.

Table 215. Error Warning Limit register (CAN1EWL - address 0xE004 4018, CAN2EWL - address 0xE004 8018) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	EWL	During CAN operation, this value is compared to both the Tx and Rx Error Counters. If either of these counter matches this value, the Error Status (ES) bit in CANSR is set.	96_{10} = 0x60	X

Note that a content change of the Error Warning Limit Register is possible only if the Reset Mode was entered previously. An Error Status change (Status Register) and an Error Warning Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

6.8 Status Register (CAN1SR - 0xE004 401C, CAN2SR - 0xE004 801C)

This register contains three status bytes in which the bits not related to transmission are identical to the corresponding bits in the Global Status Register, while those relating to transmission reflect the status of each of the 3 Tx Buffers.

Table 216. Status Register (CAN1SR - address 0xE004 401C, CAN2SR - address 0xE004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
1	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
2	TBS1[1]		Transmit Buffer Status 1.	1	1
		0(locked)	Software cannot access the Tx Buffer 1 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 1 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS1[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 1 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 1 has been successfully completed.		
4	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0

Table 216. Status Register (CAN1SR - address 0xE004 401C, CAN2SR - address 0xE004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	TS1		Transmit Status 1.	1	0
		0(idle)	There is no transmission from Tx Buffer 1.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 1.		
6	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
7	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
8	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
9	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
10	TBS2[1]		Transmit Buffer Status 2.	1	1
		0(locked)	Software cannot access the Tx Buffer 2 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 2 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
11	TCS2[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 2 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 2 has been successfully completed.		
12	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
13	TS2		Transmit Status 2.	1	0
		0(idle)	There is no transmission from Tx Buffer 2.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 2.		
14	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
15	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
16	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
17	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
18	TBS3[1]		Transmit Buffer Status 3.	1	1
		0(locked)	Software cannot access the Tx Buffer 3 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 3 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
19	TCS3[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 3 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 3 has been successfully completed.		
20	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
21	TS3		Transmit Status 3.	1	0
		0(idle)	There is no transmission from Tx Buffer 3.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 3.		

Table 216. Status Register (CAN1SR - address 0xE004 401C, CAN2SR - address 0xE004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
22	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
23	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

[1] If the CPU tries to write to this Transmit Buffer when the Transmit Buffer Status bit is '0' (locked), the written byte is not accepted and is lost without this being signalled.

[2] The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' for this TX buffer. The Transmission Complete Status bit remains '0' until a message is transmitted successfully.

6.9 Receive Frame Status Register (CAN1RFS - 0xE004 4020, CAN2RFS - 0xE004 8020)

This register defines the characteristics of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANxMOD is 1.

Table 217. Receive Frame Status register (CAN1RFS - address 0xE004 4020, CAN2RFS - address 0xE004 8020) bit description

Bit	Symbol	Function	Reset Value	RM Set
9:0	ID Index	If the BP bit (below) is 0, this value is the zero-based number of the Lookup Table RAM entry at which the Acceptance Filter matched the received Identifier. Disabled entries in the Standard tables are included in this numbering, but will not be matched. See Section 12–16 “Examples of acceptance filter tables and ID index values” on page 269 for examples of ID Index values.	0	X
10	BP	If this bit is 1, the current message was received in AF Bypass mode, and the ID Index field (above) is meaningless.	0	X
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
19:16	DLC	The field contains the Data Length Code (DLC) field of the current received message. When RTR = 0, this is related to the number of data bytes available in the CANRDA and CANRDB registers as follows: 0000-0111 = 0 to 7 bytes1000-1111 = 8 bytes With RTR = 1, this value indicates the number of data bytes requested to be sent back, with the same encoding.	0	X
29:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
30	RTR	This bit contains the Remote Transmission Request bit of the current received message. 0 indicates a Data Frame, in which (if DLC is non-zero) data can be read from the CANRDA and possibly the CANRDB registers. 1 indicates a Remote frame, in which case the DLC value identifies the number of data bytes requested to be sent using the same Identifier.	0	X
31	FF	A 0 in this bit indicates that the current received message included an 11 bit Identifier, while a 1 indicates a 29 bit Identifier. This affects the contents of the CANid register described below.	0	X

6.9.1 ID index field

The ID Index is a 10-bit field in the Info Register that contains the table position of the ID Look-up Table if the currently received message was accepted. The software can use this index to simplify message transfers from the Receive Buffer into the Shared Message Memory. Whenever bit 10 (BP) of the ID Index in the CANRFS register is 1, the current CAN message was received in acceptance filter bypass mode.

6.10 Receive Identifier Register (CAN1RID - 0xE004 4024, CAN2RID - 0xE004 8024)

This register contains the Identifier field of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANmod is 1. It has two different formats depending on the FF bit in CANRFS. See Table 12–206 for details on specific CAN channel register address.

Table 218. Receive Identifier Register (CAN1RID - address 0xE004 4024, CAN2RID - address 0xE004 8024) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11 bit Identifier field of the current received message. In CAN 2.0A, these bits are called ID10-0, while in CAN 2.0B they're called ID29-18.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 219. RX Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29 bit Identifier field of the current received message. In CAN 2.0B these bits are called ID29-0.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

6.11 Receive Data Register A (CAN1RDA - 0xE004 4028, CAN2RDA - 0xE004 8028)

This register contains the first 1-4 Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 12–206 for details on specific CAN channel register address.

Table 220. Receive Data register A (CAN1RDA - address 0xE004 4028, CAN2RDA - address 0xE004 8028) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If the DLC field in CANRFS ≥ 0001, this contains the first Data byte of the current received message.	0	X

Table 220. Receive Data register A (CAN1RDA - address 0xE004 4028, CAN2RDA - address 0xE004 8028) bit description

Bit	Symbol	Function	Reset Value	RM Set
15:8	Data 2	If the DLC field in CANRFS ≥ 0010, this contains the first Data byte of the current received message.	0	X
23:16	Data 3	If the DLC field in CANRFS ≥ 0011, this contains the first Data byte of the current received message.	0	X
31:24	Data 4	If the DLC field in CANRFS ≥ 0100, this contains the first Data byte of the current received message.	0	X

6.12 Receive Data Register B (CAN1RDB - 0xE004 402C, CAN2RDB - 0xE004 802C)

This register contains the 5th through 8th Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 12–206 for details on specific CAN channel register address.

Table 221. Receive Data register B (CAN1RDB - address 0xE004 402C, CAN2RDB - address 0xE004 802C) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If the DLC field in CANRFS ≥ 0101, this contains the first Data byte of the current received message.	0	X
15:8	Data 6	If the DLC field in CANRFS ≥ 0110, this contains the first Data byte of the current received message.	0	X
23:16	Data 7	If the DLC field in CANRFS ≥ 0111, this contains the first Data byte of the current received message.	0	X
31:24	Data 8	If the DLC field in CANRFS ≥ 1000, this contains the first Data byte of the current received message.	0	X

6.13 Transmit Frame Information Register (CAN1TFI[1/2/3] - 0xE004 40[30/40/50], CAN2TFI[1/2/3] - 0xE004 80[30/40/50])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the format of the next transmit message for that Tx buffer. Bits not listed read as 0 and should be written as 0.

The values for the reserved bits of the CANxTFI register in the Transmit Buffer should be set to the values expected in the Receive Buffer for an easy comparison, when using the Self Reception facility (self test), otherwise they are not defined.

The CAN Controller consist of three Transmit Buffers. Each of them has a length of 4 words and is able to store one complete CAN message as shown in Figure 12–31.

The buffer layout is subdivided into Descriptor and Data Field where the first word of the Descriptor Field includes the TX Frame Info that describes the Frame Format, the Data Length and whether it is a Remote or Data Frame. In addition, a TX Priority register allows the definition of a certain priority for each transmit message. Depending on the chosen Frame Format, an 11-bit identifier for Standard Frame Format (SFF) or an 29-bit identifier for Extended Frame Format (EFF) follows. Note that unused bits in the TID field have to be defined as 0. The Data Field in TDA and TDB contains up to eight data bytes.

Table 222. Transmit Frame Information Register (CAN1TFI[1/2/3] - address 0xE004 40[30/40/50], CAN2TFI[1/2/3] - 0xE004 80[30/40/50]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	PRIO	If the TPM (Transmit Priority Mode) bit in the CANxMOD register is set to 1, enabled Tx Buffers contend for the right to send their messages based on this field. The buffer with the lowest TX Priority value wins the prioritization and is sent first.		x
15:8	-	Reserved.	0	
19:16	DLC	Data Length Code. This value is sent in the DLC field of the next transmit message. In addition, if RTR = 0, this value controls the number of Data bytes sent in the next transmit message, from the CANxTDA and CANxTDB registers: 0000-0111 = 0-7 bytes 1xxx = 8 bytes	0	X
29:20	-	Reserved.	0	
30	RTR	This value is sent in the RTR bit of the next transmit message. If this bit is 0, the number of data bytes called out by the DLC field are sent from the CANxTDA and CANxTDB registers. If this bit is 1, a Remote Frame is sent, containing a request for that number of bytes.	0	X
31	FF	If this bit is 0, the next transmit message will be sent with an 11 bit Identifier (standard frame format), while if it's 1, the message will be sent with a 29 bit Identifier (extended frame format).	0	X

Automatic transmit priority detection

To allow uninterrupted streams of transmit messages, the CAN Controller provides Automatic Transmit Priority Detection for all Transmit Buffers. Depending on the selected Transmit Priority Mode, internal prioritization is based on the CAN Identifier or a user defined "local priority". If more than one message is enabled for transmission (TR=1) the internal transmit message queue is organized such as that the transmit buffer with the lowest CAN Identifier (TID) or the lowest "local priority" (TX Priority) wins the prioritization and is sent first. The result of the internal scheduling process is taken into account short before a new CAN message is sent on the bus. This is also true after the occurrence of a transmission error and right before a re-transmission.

Tx DLC

The number of bytes in the Data Field of a message is coded with the Data Length Code (DLC). At the start of a Remote Frame transmission the DLC is not considered due to the RTR bit being '1 ' (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the DLC must be specified correctly to avoid bus errors, if two CAN Controllers start a Remote Frame transmission with the same identifier simultaneously. For reasons of compatibility no DLC > 8 should be used. If a value greater than 8 is selected, 8 bytes are transmitted in the data frame with the Data Length Code specified in DLC. The range of the Data Byte Count is 0 to 8 bytes and is coded as follows:

$$DataByteCount = DLC \quad (6)$$

6.14 Transmit Identifier Register (CAN1TID[1/2/3] - 0xE004 40[34/44/54], CAN2TID[1/2/3] - 0xE004 80[34/44/54])

When the corresponding TBS bit in CANxSR is 1, software can write to one of these registers to define the Identifier field of the next transmit message. Bits not listed read as 0 and should be written as 0. The register assumes two different formats depending on the FF bit in CANTFI.

In Standard Frame Format messages, the CAN Identifier consists of 11 bits (ID.28 to ID.18), and in Extended Frame Format messages, the CAN identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit, and it is transmitted first on the bus during the arbitration process. The Identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority during the arbitration process.

Table 223. Transfer Identifier Register (CAN1TID[1/2/3] - address 0xE004 40[34/44/54], CAN2TID[1/2/3] - address 0xE004 80[34/44/54]) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11 bit Identifier to be sent in the next transmit message.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 224. Transfer Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29 bit Identifier to be sent in the next transmit message.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

6.15 Transmit Data Register A (CAN1TDA[1/2/3] - 0xE004 40[38/48/58], CAN2TDA[1/2/3] - 0xE004 80[38/48/58])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the first 1 - 4 data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 225. Transmit Data Register A (CAN1TDA[1/2/3] - address 0xE004 40[38/48/58], CAN2TDA[1/2/3] - address 0xE004 80[38/48/58]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If RTR = 0 and DLC ≥ 0001 in the corresponding CANxTFI, this byte is sent as the first Data byte of the next transmit message.	0	X
15;8	Data 2	If RTR = 0 and DLC ≥ 0010 in the corresponding CANxTFI, this byte is sent as the 2nd Data byte of the next transmit message.	0	X
23:16	Data 3	If RTR = 0 and DLC ≥ 0011 in the corresponding CANxTFI, this byte is sent as the 3rd Data byte of the next transmit message.	0	X
31:24	Data 4	If RTR = 0 and DLC ≥ 0100 in the corresponding CANxTFI, this byte is sent as the 4th Data byte of the next transmit message.	0	X

6.16 Transmit Data Register B (CAN1TDB[1/2/3] - 0xE004 40[3C/4C/5C], CAN2TDB[1/2/3] - 0xE004 80[3C/4C/5C])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the 5th through 8th data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 226. Transmit Data Register B (CAN1TDB[1/2/3] - address 0xE004 40[3C/4C/5C], CAN2TDB[1/2/3] - address 0xE004 80[3C/4C/5C]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If RTR = 0 and DLC ≥ 0101 in the corresponding CANTFI, this byte is sent as the 5th Data byte of the next transmit message.	0	X
15;8	Data 6	If RTR = 0 and DLC ≥ 0110 in the corresponding CANTFI, this byte is sent as the 6th Data byte of the next transmit message.	0	X
23:16	Data 7	If RTR = 0 and DLC ≥ 0111 in the corresponding CANTFI, this byte is sent as the 7th Data byte of the next transmit message.	0	X
31:24	Data 8	If RTR = 0 and DLC ≥ 1000 in the corresponding CANTFI, this byte is sent as the 8th Data byte of the next transmit message.	0	X

7. CAN controller operation

7.1 Error handling

The CAN Controllers count and handle transmit and receive errors as specified in CAN Spec 2.0B. The Transmit and Receive Error Counters are incriminated for each detected error and are decremented when operation is error-free. If the Transmit Error counter contains 255 and another error occurs, the CAN Controller is forced into a state called Bus-Off. In this state, the following register bits are set: BS in CANxSR, BEI and EI in CANxIR if these are enabled, and RM in CANxMOD. RM resets and disables much of the CAN Controller. Also at this time the Transmit Error Counter is set to 127 and the Receive Error Counter is cleared. Software must next clear the RM bit. Thereafter the Transmit Error Counter will count down 128 occurrences of the Bus Free condition (11 consecutive recessive bits). Software can monitor this countdown by reading the Tx Error Counter. When this countdown is complete, the CAN Controller clears BS and ES in CANxSR, and sets EI in CANxSR if EIE in IER is 1.

The Tx and Rx error counters can be written if RM in CANxMOD is 1. Writing 255 to the Tx Error Counter forces the CAN Controller to Bus-Off state. If Bus-Off (BS in CANxSR) is 1, writing any value 0 through 254 to the Tx Error Counter clears Bus-Off. When software clears RM in CANxMOD thereafter, only one Bus Free condition (11 consecutive recessive bits) is needed before operation resumes.

7.2 Sleep mode

The CAN Controller will enter sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. Software can only set SM when RM in the CAN Mode register is 0; it can also set the WUIE bit in the CAN Interrupt Enable register to enable an interrupt on any wake-up condition.

The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1) in response to a) a dominant bit on the CAN bus, or b) software clearing SM in the CAN Mode register. A sleeping CAN Controller, that wakes up in response to bus activity, is not able to receive an initial message, until after it detects Bus_Free (11 consecutive recessive bits). If an interrupt is pending or the CAN bus is active when software sets SM, the wake-up is immediate.

7.3 Interrupts

Each CAN Controller produces 3 interrupt requests, Receive, Transmit, and “other status”. The Transmit interrupt is the OR of the Transmit interrupts from the three Tx Buffers. Each Receive and Transmit interrupt request from each controller is assigned its own channel in the Vectored Interrupt Controller (VIC), and can have its own interrupt service routine. The “other status” interrupts from all of the CAN controllers, and the Acceptance Filter LUTerr condition, are ORed into one VIC channel.

7.4 Transmit priority

If the TPM bit in the CANxMOD register is 0, multiple enabled Tx Buffers contend for the right to send their messages based on the value of their CAN Identifier (TID). If TPM is 1, they contend based on the PRIO fields in bits 7:0 of their CANxTFS registers. In both cases the smallest binary value has priority. If two (or three) transmit-enabled buffers have the same smallest value, the lowest-numbered buffer sends first.

The CAN controller selects among multiple enabled Tx Buffers dynamically, just before it sends each message.

8. Centralized CAN registers

For easy and fast access, all CAN Controller Status bits from each CAN Controller Status register are bundled together. Each defined byte of the following registers contains one particular status bit from each of the CAN controllers, in its LS bits.

All Status registers are “read-only” and allow byte, half word and word access.

8.1 Central Transmit Status Register (CANTxSR - 0xE004 0000)

Table 227. Central Transit Status Register (CANTxSR - address 0xE004 0000) bit description

Bit	Symbol	Description	Reset Value
0	TS1	When 1, the CAN controller 1 is sending a message (same as TS in the CAN1GSR).	0
1	TS2	When 1, the CAN controller 2 is sending a message (same as TS in the CAN2GSR)	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	TBS1	When 1, all 3 Tx Buffers of the CAN1 controller are available to the CPU (same as TBS in CAN1GSR).	1
9	TBS2	When 1, all 3 Tx Buffers of the CAN2 controller are available to the CPU (same as TBS in CAN2GSR).	1

Table 227. Central Transit Status Register (CANTxSR - address 0xE004 0000) bit description

Bit	Symbol	Description	Reset Value
15:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	TCS1	When 1, all requested transmissions have been completed successfully by the CAN1 controller (same as TCS in CAN1GSR).	1
17:16	TCS2	When 1, all requested transmissions have been completed successfully by the CAN2 controller (same as TCS in CAN2GSR).	1
31:18	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.2 Central Receive Status Register (CANRxSR - 0xE004 0004)

Table 228. Central Receive Status Register (CANRxSR - address 0xE004 0004) bit description

Bit	Symbol	Description	Reset Value
0	RS1	When 1, CAN1 is receiving a message (same as RS in CAN1GSR).	0
1	RS2	When 1, CAN2 is receiving a message (same as RS in CAN2GSR).	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	RB1	When 1, a received message is available in the CAN1 controller (same as RBS in CAN1GSR).	0
9	RB2	When 1, a received message is available in the CAN2 controller (same as RBS in CAN2GSR).	0
15:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	DOS1	When 1, a message was lost because the preceding message to CAN1 controller was not read out quickly enough (same as DOS in CAN1GSR).	0
17:16	DOS2	When 1, a message was lost because the preceding message to CAN2 controller was not read out quickly enough (same as DOS in CAN2GSR).	0
31:18	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.3 Central Miscellaneous Status Register (CANMSR - 0xE004 0008)

Table 229. Central Miscellaneous Status Register (CANMSR - address 0xE004 0008) bit description

Bit	Symbol	Description	Reset Value
0	E1	When 1, one or both of the CAN1 Tx and Rx Error Counters has reached the limit set in the CAN1EWL register (same as ES in CAN1GSR)	0
1	E2	When 1, one or both of the CAN2 Tx and Rx Error Counters has reached the limit set in the CAN2EWL register (same as ES in CAN2GSR)	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 229. Central Miscellaneous Status Register (CANMSR - address 0xE004 0008) bit description

Bit	Symbol	Description	Reset Value
8	BS1	When 1, the CAN1 controller is currently involved in bus activities (same as BS in CAN1GSR).	0
9	BS2	When 1, the CAN2 controller is currently involved in bus activities (same as BS in CAN2GSR).	0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

9. Global acceptance filter

This block provides lookup for received Identifiers (called Acceptance Filtering in CAN terminology) for all the CAN Controllers. It includes a 512 × 32 (2 kB) RAM in which software maintains one to five tables of Identifiers. This RAM can contain up to 1024 Standard Identifiers or 512 Extended Identifiers, or a mixture of both types.

10. Acceptance filter modes

The Acceptance Filter can be put into different modes by setting the according AccOff, AccBP, and eFCAN bits in the Acceptance Filter Mode Register (Section 12–13.1 "Acceptance Filter Mode Register (AFMR - 0xE003 C000)"). During each mode the access to the Configuration Register and the ID Look-up table is handled differently.

Table 230. Acceptance filter modes and access control

Acceptance filter mode	Bit AccOff	Bit AccBP	Acceptance filter state	ID Look-up table RAM[1]	Acceptance filter config. registers	CAN controller message receive interrupt
Off Mode	1	0	reset & halted	r/w access from CPU	r/w access from CPU	no messages accepted
Bypass Mode	X	1	reset & halted	r/w access from CPU	r/w access from CPU	all messages accepted
Operating Mode and FullCAN Mode	0	0	running	read only from CPU[2]	access from Acceptance filter only	hardware acceptance filtering

[1] The whole ID Look-up Table RAM is only word accessible.

[2] During the Operating Mode of the Acceptance Filter the Look-up Table can be accessed only to disable or enable Messages.

A write access to all section configuration registers is only possible during the Acceptance Filter Off and Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

10.1 Acceptance filter Off mode

The Acceptance Filter Off Mode is typically used during initialization. During this mode an unconditional access to all registers and to the Look-up Table RAM is possible. With the Acceptance Filter Off Mode, CAN messages are not accepted and therefore not stored in the Receive Buffers of active CAN Controllers.

10.2 Acceptance filter Bypass mode

The Acceptance Filter Bypass Mode can be used for example to change the acceptance filter configuration during a running system, e.g. change of identifiers in the ID-Look-up Table memory. During this re-configuration, software acceptance filtering has to be used.

It is recommended to use the ID ready Interrupt (ID Index) and the Receive Interrupt (RI). In this mode all CAN message are accepted and stored in the Receive Buffers of active CAN Controllers.

10.3 Acceptance filter Operating mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 0.

10.4 FullCAN mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 1. More details on FullCAN mode are available in Section 12–15 “FullCAN mode”.

11. Sections of the ID look-up table RAM

Four 12-bit section configuration registers (SFF_sa, SFF_GRP_sa, EFF_sa, EFF_GRP_sa) are used to define the boundaries of the different identifier sections in the ID-Look-up Table Memory. The fifth 12-bit section configuration register, the End of Table address register (ENDofTable) is used to define the end of all identifier sections. The End of Table address is also used to assign the start address of the section where FullCAN Message Objects, if enabled are stored.

Table 231. Section configuration register settings

ID-Look up Table Section	Register	Value	Section status
FullCAN (Standard Frame Format) Identifier Section	SFF_sa	= 0x000	disabled
		> 0x000	enabled
Explicit Standard Frame Format Identifier Section	SFF_GRP_sa	= SFF_sa	disabled
		> SFF_sa	enabled
Group of Standard Frame Format Identifier Section	EFF_sa	= SFF_GRP_sa	disabled
		> SFF_GRP_sa	enabled
Explicit Extended Frame Format Identifier Section	EFF_GRP_sa	= EFF_sa	disabled
		> EFF_sa	enabled
Group of Extended Frame Format Identifier Section	ENDofTable	= EFF_GRP_sa	disabled
		> EFF_GRP_sa	enabled

12. ID look-up table RAM

The Whole ID Look-up Table RAM is only word accessible. A write access is only possible during the Acceptance Filter Off or Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

If Standard (11 bit) Identifiers are used in the application, at least one of 3 tables in Acceptance Filter RAM must not be empty. If the optional “fullCAN mode” is enabled, the first table contains Standard identifiers for which reception is to be handled in this mode. The next table contains individual Standard Identifiers and the third contains ranges of Standard Identifiers, for which messages are to be received via the CAN Controllers. The tables of fullCAN and individual Standard Identifiers must be arranged in ascending numerical order, one per halfword, two per word. Since each CAN bus has its own address map, each entry also contains the number of the CAN Controller (001-010) to which it applies.

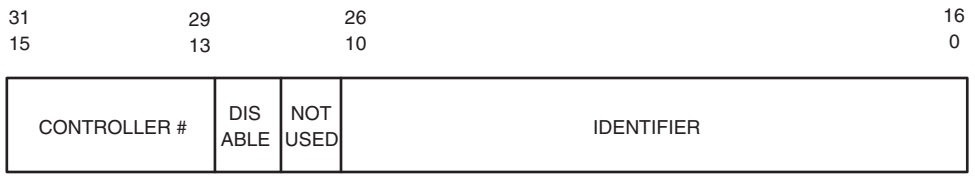


Fig 35. Entry in FullCAN and individual standard identifier tables

The table of Standard Identifier Ranges contains paired upper and lower (inclusive) bounds, one pair per word. These must also be arranged in ascending numerical order.

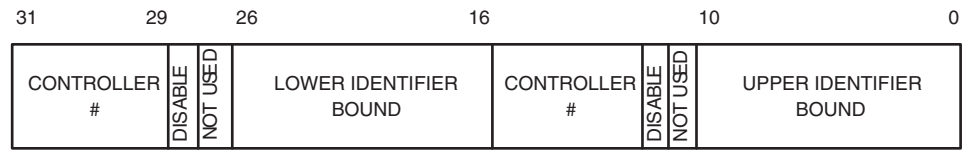


Fig 36. Entry in standard identifier range table

The disable bits in Standard entries provide a means to turn response, to particular CAN Identifiers or ranges of Identifiers, on and off dynamically. When the Acceptance Filter function is enabled, only the disable bits in Acceptance Filter RAM can be changed by software. Response to a range of Standard addresses can be enabled by writing 32 zero bits to its word in RAM, and turned off by writing 32 one bits (0xFFFF FFFF) to its word in RAM. Only the disable bits are actually changed. Disabled entries must maintain the ascending sequence of Identifiers.

If Extended (29 bit) Identifiers are used in the application, at least one of the other two tables in Acceptance Filter RAM must not be empty, one for individual Extended Identifiers and one for ranges of Extended Identifiers. The table of individual Extended Identifiers must be arranged in ascending numerical order.

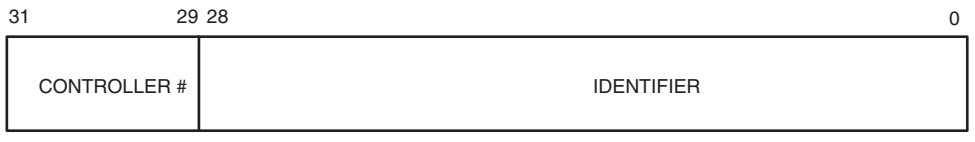


Fig 37. Entry in either extended identifier table

The table of ranges of Extended Identifiers must contain an even number of entries, of the same form as in the individual Extended Identifier table. Like the Individual Extended table, the Extended Range must be arranged in ascending numerical order. The first and second (3rd and 4th …) entries in the table are implicitly paired as an inclusive range of Extended addresses, such that any received address that falls in the inclusive range is received (accepted). Software must maintain the table to consist of such word pairs.

There is no facility to receive messages to Extended identifiers using the fullCAN method.

Five address registers point to the boundaries between the tables in Acceptance Filter RAM: fullCAN Standard addresses, Standard Individual addresses, Standard address ranges, Extended Individual addresses, and Extended address ranges. These tables must be consecutive in memory. The start of each of the latter four tables is implicitly the end of the preceding table. The end of the Extended range table is given in an End of Tables register. If the start address of a table equals the start of the next table or the End Of Tables register, that table is empty.

When the Receive side of a CAN controller has received a complete Identifier, it signals the Acceptance Filter of this fact. The Acceptance Filter responds to this signal, and reads the Controller number, the size of the Identifier, and the Identifier itself from the Controller. It then proceeds to search its RAM to determine whether the message should be received or ignored.

If fullCAN mode is enabled and the CAN controller signals that the current message contains a Standard identifier, the Acceptance Filter first searches the table of identifiers for which reception is to be done in fullCAN mode. Otherwise, or if the AF doesn't find a match in the fullCAN table, it searches its individual Identifier table for the size of Identifier signalled by the CAN controller. If it finds an equal match, the AF signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register.

If the Acceptance Filter does not find a match in the appropriate individual Identifier table, it then searches the Identifier Range table for the size of Identifier signalled by the CAN controller. If the AF finds a match to a range in the table, it similarly signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register. If the Acceptance Filter does not find a match in either the individual or Range table for the size of Identifier received, it signals the CAN controller to discard/ignore the received message.

13. Acceptance filter registers

13.1 Acceptance Filter Mode Register (AFMR - 0xE003 C000)

The AccBP and AccOff bits of the acceptance filter mode register are used for putting the acceptance filter into the Bypass and Off mode. The eFCAN bit of the mode register can be used to activate a FullCAN mode enhancement for received 11-bit CAN ID messages.

Table 232. Acceptance Filter Mode Register (AFMR - address 0xE003 C000) bit description

Bit	Symbol	Value	Description	Reset Value
0	AccOff[2]	1	if AccBP is 0, the Acceptance Filter is not operational. All Rx messages on all CAN buses are ignored.	1
1	AccBP[1]	1	All Rx messages are accepted on enabled CAN controllers. Software must set this bit before modifying the contents of any of the registers described below, and before modifying the contents of Lookup Table RAM in any way other than setting or clearing Disable bits in Standard Identifier entries. When both this bit and AccOff are 0, the Acceptance filter operates to screen received CAN Identifiers.	0
2	eFCAN[3]	0	Software must read all messages for all enabled IDs on all enabled CAN buses, from the receiving CAN controllers.	0
		1	The Acceptance Filter itself will take care of receiving and storing messages for selected Standard ID values on selected CAN buses. See Section 12–15 “FullCAN mode” on page 258.	
31:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Acceptance Filter Bypass Mode (AccBP): By setting the AccBP bit in the Acceptance Filter Mode Register, the Acceptance filter is put into the Acceptance Filter Bypass mode. During bypass mode, the internal state machine of the Acceptance Filter is reset and halted. All received CAN messages are accepted, and acceptance filtering can be done by software.

[2] Acceptance Filter Off mode (AccOff): After power-upon hardware reset, the Acceptance filter will be in Off mode, the AccOff bit in the Acceptance filter Mode register 0 will be set to 1. The internal state machine of the acceptance filter is reset and halted. If not in Off mode, setting the AccOff bit, either by hardware or by software, will force the acceptance filter into Off mode.

[3] FullCan Mode Enhancements: A FullCan mode for received CAN messages can be enabled by setting the eFCAN bit in the acceptance filter mode register.

13.2 Section configuration registers

The 10 bit section configuration registers are used for the ID look-up table RAM to indicate the boundaries of the different sections for explicit and group of CAN identifiers for 11 bit CAN and 29 bit CAN identifiers, respectively. The 10 bit wide section configuration registers allow the use of a 512x32 (2 kB) look-up table RAM. The whole ID Look-up Table RAM is only word accessible. All five section configuration registers contain APB addresses for the acceptance filter RAM and do not include the APB base address. A write access to all section configuration registers is only possible during the Acceptance filter off and Bypass modes. Read access is allowed in all acceptance filter modes.

13.3 Standard Frame Individual Start Address Register (SFF_sa - 0xE003 C004)

Table 233. Standard Frame Individual Start Address Register (SFF_sa - address 0xE003 C004) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	SFF_sa[1]	The start address of the table of individual Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the SFF_GRP_sa register described below. For compatibility with possible future devices, write zeroes in bits 31:11 and 1:0 of this register. If the eFCAN bit in the AFMR is 1, this value also indicates the size of the table of Standard IDs which the Acceptance Filter will search and (if found) automatically store received messages in Acceptance Filter RAM.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

13.4 Standard Frame Group Start Address Register (SFF_GRP_sa - 0xE003 C008)

Table 234. Standard Frame Group Start Address Register (SFF_GRP_sa - address 0xE003 C008) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	SFF_GRP_sa[1]	The start address of the table of grouped Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_sa register described below. The largest value that should be written to this register is 0x800, when only the Standard Individual table is used, and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

13.5 Extended Frame Start Address Register (EFF_sa - 0xE003 C00C)

Table 235. Extended Frame Start Address Register (EFF_sa - address 0xE003 C00C) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	EFF_sa[1]	The start address of the table of individual Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_GRP_sa register described below. The largest value that should be written to this register is 0x800, when both Extended Tables are empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:11 and 1:0 of this register.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

13.6 Extended Frame Group Start Address Register (EFF_GRP_sa - 0xE003 C010)

Table 236. Extended Frame Group Start Address Register (EFF_GRP_sa - address 0xE003 C010) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	Eff_GRP_sa[1]	The start address of the table of grouped Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the ENDofTable register described below. The largest value that should be written to this register is 0x800, when this table is empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

13.7 End of AF Tables Register (ENDofTable - 0xE003 C014)

Table 237. End of AF Tables Register (ENDofTable - address 0xE003 C014) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	EndofTable [1]	The address above the last active address in the last active AF table. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register. If the eFCAN bit in the AFMR is 0, the largest value that should be written to this register is 0x800, which allows the last word (address 0x7FC) in AF Lookup Table RAM to be used. If the eFCAN bit in the AFMR is 1, this value marks the start of the area of Acceptance Filter RAM, into which the Acceptance Filter will automatically receive messages for selected IDs on selected CAN buses. In this case, the maximum value that should be written to this register is 0x800 minus 6 times the value in SFF_sa. This allows 12 bytes of message storage between this address and the end of Acceptance Filter RAM, for each Standard ID that is specified between the start of Acceptance Filter RAM, and the next active AF table.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

13.8 Status registers

The look-up table error status registers, the error addresses, and the flag register provide information if a programming error in the look-up table RAM during the ID screening was encountered. The look-up table error address and flag register have only read access. If an error is detected, the LUTerror flag is set, and the LUTerrorAddr register provides the information under which address during an ID screening an error in the look-up table was encountered. Any read of the LUTerrorAddr Filter block can be used for a look-up table interrupt.

13.9 LUT Error Address Register (LUTerrAd - 0xE003 C018)

Table 238. LUT Error Address Register (LUTerrAd - address 0xE003 C018) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	LUTerrAd	It the LUT Error bit (below) is 1, this read-only field contains the address in AF Lookup Table RAM, at which the Acceptance Filter encountered an error in the content of the tables.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.10 LUT Error Register (LUTerr - 0xE003 C01C)

Table 239. LUT Error Register (LUTerr - address 0xE003 C01C) bit description

Bit	Symbol	Description	Reset Value
0	LUTerr	This read-only bit is set to 1 if the Acceptance Filter encounters an error in the content of the tables in AF RAM. It is cleared when software reads the LUTerrAd register. This condition is ORed with the “other CAN” interrupts from the CAN controllers, to produce the request for a VIC interrupt channel.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.11 Global FullCANInterrupt Enable register (FCANIE - 0xE003 C020)

A write access to the Global FullCAN Interrupt Enable register is only possible when the Acceptance Filter is in the off mode.

Table 240. Global FullCAN Enable register (FCANIE - address 0xE003 C020) bit description

Bit	Symbol	Description	Reset Value
0	FCANIE	Global FullCAN Interrupt Enable. When 1, this interrupt is enabled.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.12 FullCAN Interrupt and Capture registers (FCANIC0 - 0xE003 C024 and FCANIC1 - 0xE003 C028)

For detailed description on these two registers, see Section 12–15.2 “FullCAN interrupts”.

Table 241. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0xE003 C024) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd0	FullCan Interrupt Pending bit 0.	0
...	IntPndx (0<x<31)	FullCan Interrupt Pending bit x.	0
31	IntPnd31	FullCan Interrupt Pending bit 31.	0

Table 242. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0xE003 C028) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd32	FullCan Interrupt Pending bit 32.	0
...	IntPndx (32<x<63)	FullCan Interrupt Pending bit x.	0
31	IntPnd63	FullCan Interrupt Pending bit 63.	0

14. Configuration and search algorithm

The CAN Identifier Look-up Table Memory can contain explicit identifiers and groups of CAN identifiers for Standard and Extended CAN Frame Formats. They are organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

Every CAN identifier is linked to an ID Index number. In case of a CAN Identifier match, the matching ID Index is stored in the Identifier Index of the Frame Status Register (CANRFS) of the according CAN Controller.

14.1 Acceptance filter search algorithm

The identifier screening process of the acceptance filter starts in the following order:

1. FullCAN (Standard Frame Format) Identifier Section
2. Explicit Standard Frame Format Identifier Section
3. Group of Standard Frame Format Identifier Section
4. Explicit Extended Frame Format Identifier Section
5. Group of Extended Frame Format Identifier Section

Note: Only activated sections will take part in the screening process.

In cases where equal message identifiers of same frame format are defined in more than one section, the first match will end the screening process for this identifier.

For example, if the same Source CAN Channel in conjunction with the identifier is defined in the FullCAN, the Explicit Standard Frame Format and the Group of Standard Frame Format Identifier Sections, the screening will already be finished with the match in the FullCAN section.

In the example of Figure 12–38, Identifiers with their Source CAN Channel have been defined in the FullCAN, Explicit and Group of Standard Frame Format Identifier Sections. This example corresponds to a LPC2290 compatible part that would have 6 CAN controllers.

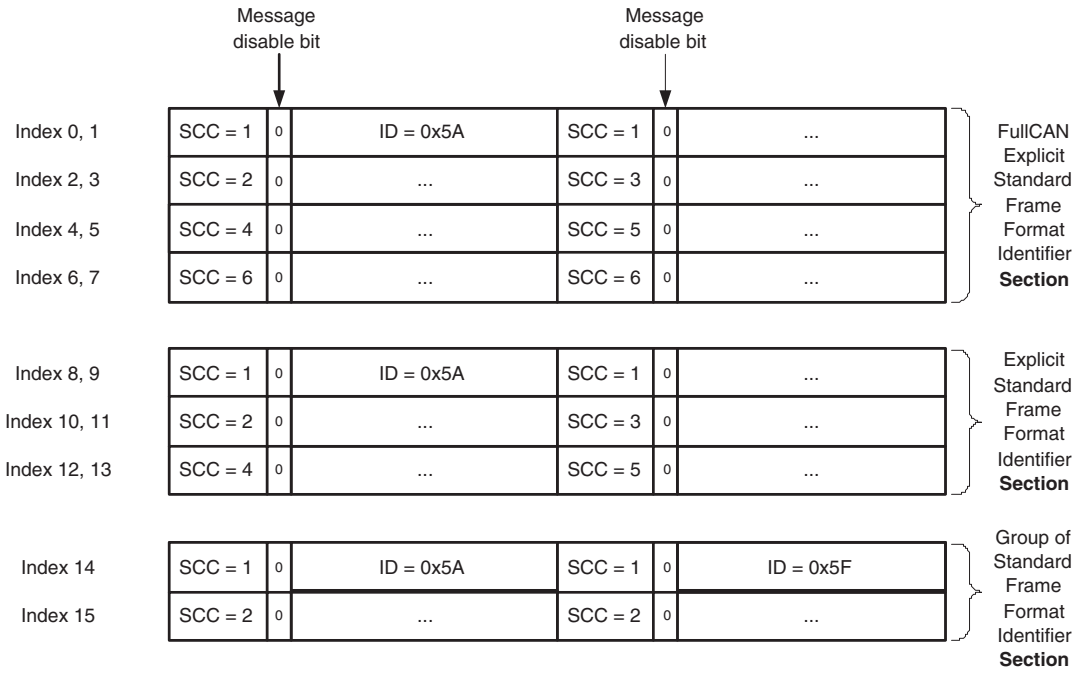


Fig 38. ID Look-up table example explaining the search algorithm

The identifier 0x5A of the CAN Controller 1 with the Source CAN Channel SCC = 1, is defined in all three sections. With this configuration incoming CAN messages on CAN Controller 1 with a 0x5A identifier will find a match in the FullCAN section.

It is possible to disable the '0x5A identifier' in the FullCAN section. With that, the screening process would be finished with the match in the Explicit Identifier Section.

The first group in the Group Identifier Section has been defined in that way, that incoming CAN messages with identifiers of 0x5A up to 0x5F are accepted on CAN Controller 1 with the Source CAN Channel SCC = 1. As stated above, the identifier 0x5A would find a match already in the FullCAN or in the Explicit Identifier section if enabled. The rest of the defined identifiers of this group (0x5B to 0x5F) will find a match in this Group Identifier Section.

This way the user can switch dynamically between different filter modes for same identifiers.

15. FullCAN mode

The FullCAN mode is based on capabilities provided by the CAN Gateway module used in the LPC2000 family of products. This block uses the Acceptance Filter to provide filtering for both CAN channels.

The concept of the CAN Gateway block is mainly based on a BasicCAN functionality. This concept fits perfectly in systems where a gateway is used to transfer messages or message data between different CAN channels. A BasicCAN device is generating a

receive interrupt whenever a CAN message is accepted and received. Software has to move the received message out of the receive buffer from the according CAN controller into the user RAM.

To cover dashboard like applications where the controller typically receives data from several CAN channels for further processing, the CAN Gateway block was extended by a so-called FullCAN receive function. This additional feature uses an internal message handler to move received FullCAN messages from the receive buffer of the according CAN controller into the FullCAN message object data space of Look-up Table RAM.

When fullCAN mode is enabled, the Acceptance Filter itself takes care of receiving and storing messages for selected Standard ID values on selected CAN buses, in the style of “FullCAN” controllers.

In order to set this bit and use this mode, two other conditions must be met with respect to the contents of Acceptance Filter RAM and the pointers into it:

- The Standard Frame Individual Start Address Register (SFF_sa) must be greater than or equal to the number of IDs for which automatic receive storage is to be done, times two. SFF_sa must be rounded up to a multiple of 4 if necessary.
- The EndOfTable register must be less than or equal to 0x800 minus 6 times the SFF_sa value, to allow 12 bytes of message storage for each ID for which automatic receive storage will be done.

When these conditions are met and eFCAN is set:

- The area between the start of Acceptance Filter RAM and the SFF_sa address, is used for a table of individual Standard IDs and CAN Controller/bus identification, sorted in ascending order and in the same format as in the Individual Standard ID table (see Figure 12–35 “Entry in FullCAN and individual standard identifier tables” on page 250). Entries can be marked as “disabled” as in the other Standard tables. If there are an odd number of “FullCAN” ID’s, at least one entry in this table must be so marked.
- The first (SFF_sa)/2 IDindex values are assigned to these automatically-stored ID’s. That is, IDindex values stored in the Rx Frame Status Register, for IDs not handled in this way, are increased by (SFF_sa)/2 compared to the values they would have when eFCAN is 0.
- When a Standard ID is received, the Acceptance Filter searches this table before the Standard Individual and Group tables.
- When a message is received for a controller and ID in this table, the Acceptance filter reads the received message out of the CAN controller and stores it in Acceptance Filter RAM, starting at (EndOfTable) + its IDindex*12.
- The format of such messages is shown in Table 12–243.

15.1 FullCAN message layout

Table 243. Format of automatically stored Rx messages

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FF	RTR	0000				SEM [1:0]		0000				DLC				00000					ID.28 ... ID.18										
+4	Rx Data 4								Rx Data 3								Rx Data 2								Rx Data 1							
+8	Rx Data 8								Rx Data 7								Rx Data 6								Rx Data 5							

The FF, RTR, and DLC fields are as described in Table 12–217.

Since the FullCAN message object section of the Look-up table RAM can be accessed both by the Acceptance Filter and the CPU, there is a method for insuring that no CPU reads from FullCAN message object occurs while the Acceptance Filter hardware is writing to that object.

For this purpose the Acceptance Filter uses a 3-state semaphore, encoded with the two semaphore bits SEM1 and SEM0 (see Table 12–243 “Format of automatically stored Rx messages”) for each message object. This mechanism provides the CPU with information about the current state of the Acceptance Filter activity in the FullCAN message object section.

The semaphore operates in the following manner:

Table 244. FullCAN semaphore operation

SEM1	SEM0	activity
0	1	Acceptance Filter is updating the content
1	1	Acceptance Filter has finished updating the content
0	0	CPU is in process of reading from the Acceptance Filter

Prior to writing the first data byte into a message object, the Acceptance Filter will write the FrameInfo byte into the according buffer location with SEM[1:0] = 01.

After having written the last data byte into the message object, the Acceptance Filter will update the semaphore bits by setting SEM[1:0] = 11.

Before reading a message object, the CPU should read SEM[1:0] to determine the current state of the Acceptance Filter activity therein. If SEM[1:0] = 01, then the Acceptance Filter is currently active in this message object. If SEM[1:0] = 11, then the message object is available to be read.

Before the CPU begins reading from the message object, it should clear SEM[1:0] = 00.

When the CPU is finished reading, it can check SEM[1:0] again. At the time of this final check, if SEM[1:0] = 01 or 11, then the Acceptance Filter has updated the message object during the time when the CPU reads were taking place, and the CPU should discard the data. If, on the other hand, SEM[1:0] = 00 as expected, then valid data has been successfully read by the CPU.

Figure 12–39 shows how software should use the SEM field to ensure that all three words read from the message are all from the same received message.

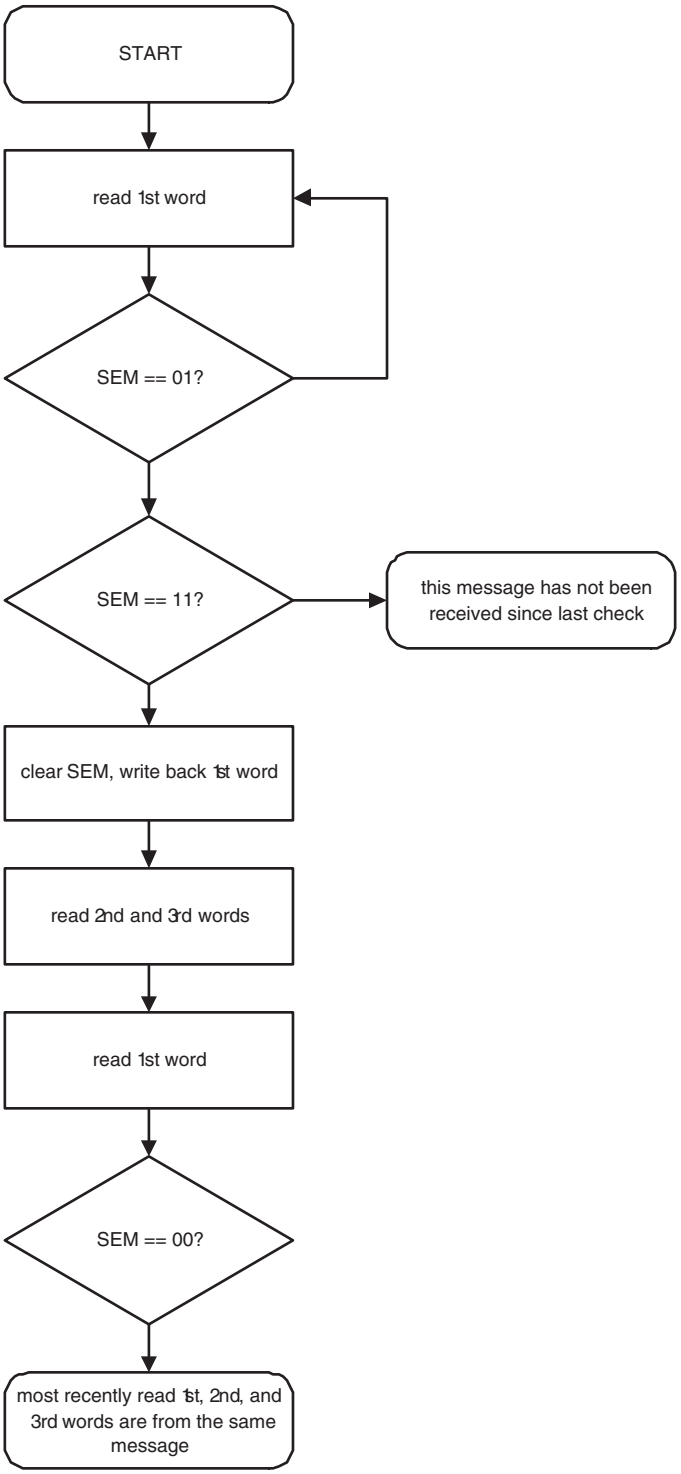


Fig 39. Semaphore procedure for reading an auto-stored message

15.2 FullCAN interrupts

The CAN Gateway Block contains a 2 kB ID Look-up Table RAM. With this size a maximum number of 146 FullCAN objects can be defined if the whole Look-up Table RAM is used for FullCAN objects only. Only the first 64 FullCAN objects can be configured to participate in the interrupt scheme. It is still possible to define more than 64 FullCAN objects. The only difference is, that the remaining FullCAN objects will not provide a FullCAN interrupt.

The FullCAN Interrupt Register-set contains interrupt flags (IntPndx) for (pending) FullCAN receive interrupts. As soon as a FullCAN message is received, the according interrupt bit (IntPndx) in the FCAN Interrupt Register gets asserted. In case that the Global FullCAN Interrupt Enable bit is set, the FullCAN Receive Interrupt is passed to the Vectored Interrupt Controller.

Application Software has to solve the following:

1. Index/Object number calculation based on the bit position in the FCANIC Interrupt Register for more than one pending interrupt.
2. Interrupt priority handling if more than one FullCAN receive interrupt is pending.

The software that covers the interrupt priority handling has to assign a receive interrupt priority to every FullCAN object. If more than one interrupt is pending, then the software has to decide, which received FullCAN object has to be served next.

To each FullCAN object a new FullCAN Interrupt Enable bit (FCANIntxEn) is added, so that it is possible to enable or disable FullCAN interrupts for each object individually. The new Message Lost flag (MsgLstx) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read by the CPU. The Interrupt Enable and the Message Lost bits reside in the existing Look-up Table RAM.

15.2.1 FullCAN message interrupt enable bit

In Figure 12–40 8 FullCAN Identifiers with their Source CAN Channel are defined in the FullCAN, Section. The new introduced FullCAN Message Interrupt enable bit can be used to enable for each FullCAN message an Interrupt.

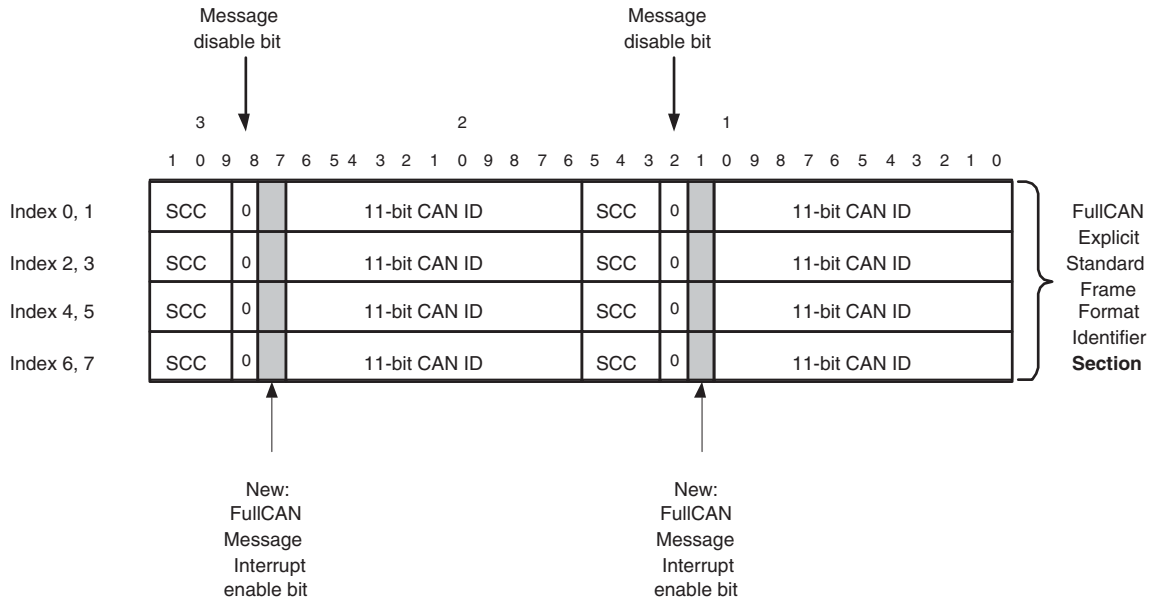


Fig 40. FullCAN section example of the ID look-up table

15.2.2 Message lost bit and CAN channel number

Figure 12–41 is the detailed layout structure of one FullCAN message stored in the FullCAN message object section of the Look-up Table.

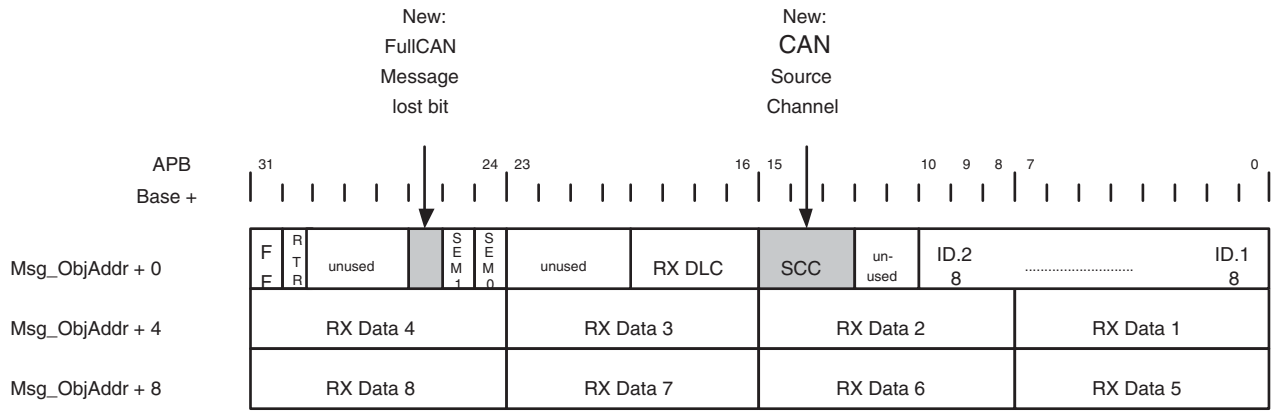


Fig 41. FullCAN message object layout

The new message lost bit (MsgLst) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read. For more information the CAN Source Channel (SCC) of the received FullCAN message is added to Message Object.

15.2.3 Setting the interrupt pending bits (IntPnd 63 to 0)

The interrupt pending bit (IntPndx) gets asserted in case of an accepted FullCAN message and if the interrupt of the according FullCAN Object is enabled (enable bit FCANIntxEn) is set).

During the **last write access** from the data storage of a FullCAN message object the interrupt pending bit of a FullCAN object (IntPndx) gets asserted.

15.2.4 Clearing the interrupt pending bits (IntPnd 63 to 0)

Each of the FullCAN Interrupt Pending requests gets cleared when the semaphore bits of a message object are cleared by Software (ARM CPU).

15.2.5 Setting the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets asserted in case of an accepted FullCAN message and when the FullCAN Interrupt of the same object is asserted already.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets asserted if the interrupt pending bit is set already.

15.2.6 Clearing the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets cleared when the FullCAN Interrupt of the same object is not asserted.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets cleared if the interrupt pending bit is not set.

15.3 Set and clear mechanism of the FullCAN interrupt

Special precaution is needed for the built-in set and clear mechanism of the FullCAN Interrupts. The following text illustrates how the already existing Semaphore Bits (see Section 12–15.1 “FullCAN message layout” for more details) and how the new introduced features (IntPndx, MsgLstx) will behave.

15.3.1 Scenario 1: Normal case, no message lost

Figure 12–42 below shows a typical “normal” scenario in which an accepted FullCAN message is stored in the FullCAN Message Object Section. After storage the message is read out by Software (ARM CPU).

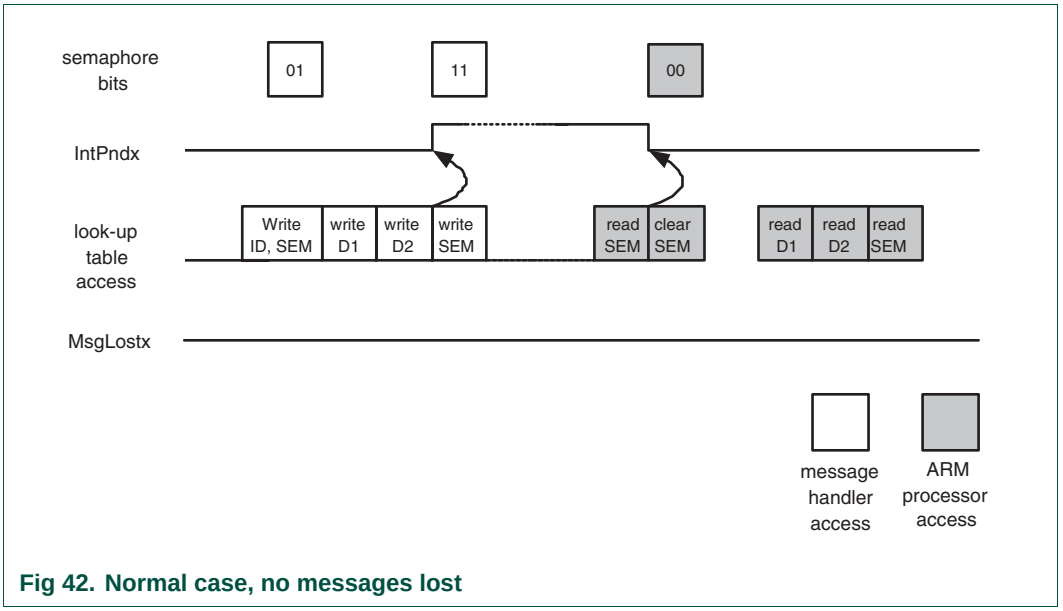


Fig 42. Normal case, no messages lost

15.3.2 Scenario 2: Message lost

In this scenario a first FullCAN Message is stored and read out by Software (1st Object write and read). In a second course a second message is stored (2nd Object write) but not read out before a third message gets stored (3rd Object write). Since the FullCAN Interrupt of that Object (IntPndx) is already asserted, the Message Lost Signal gets asserted.

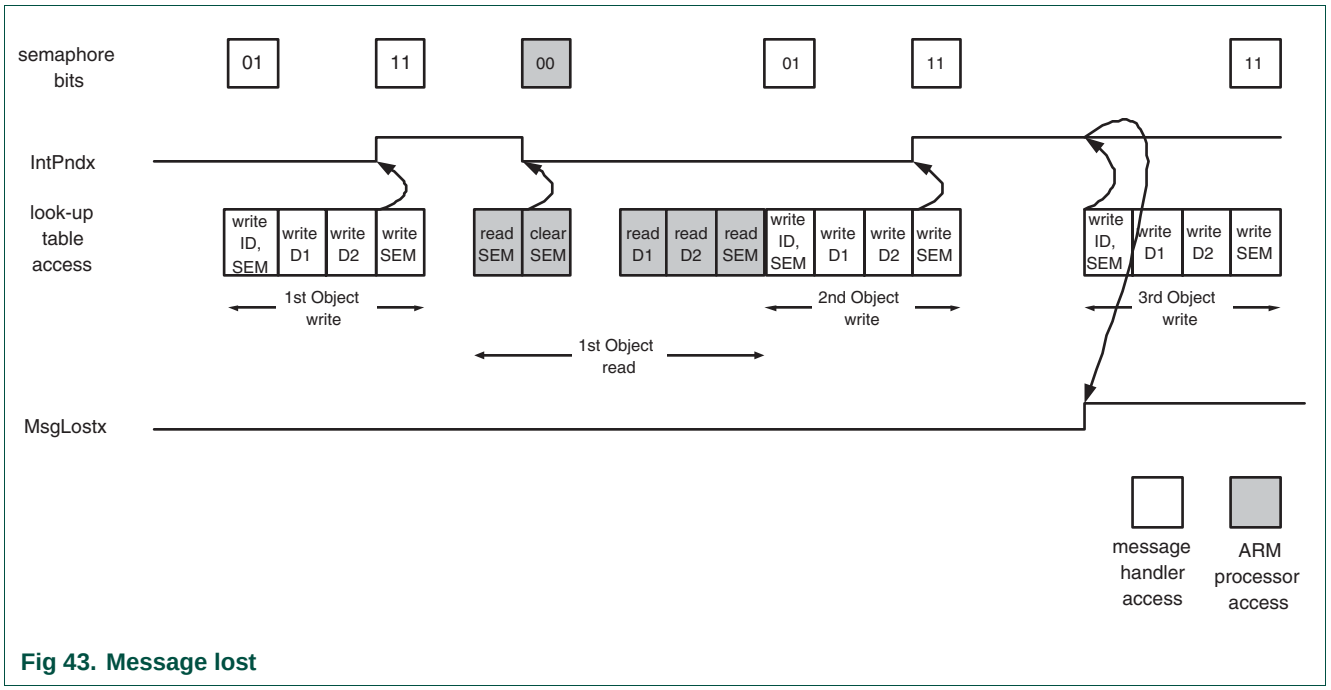


Fig 43. Message lost

15.3.3 Scenario 3: Message gets overwritten indicated by Semaphore bits

This scenario is a special case in which the lost message is indicated by the existing semaphore bits. The scenario is entered, if during a Software read of a message object another new message gets stored by the message handler. In this case, the FullCAN Interrupt bit gets set for a second time with the 2nd Object write.

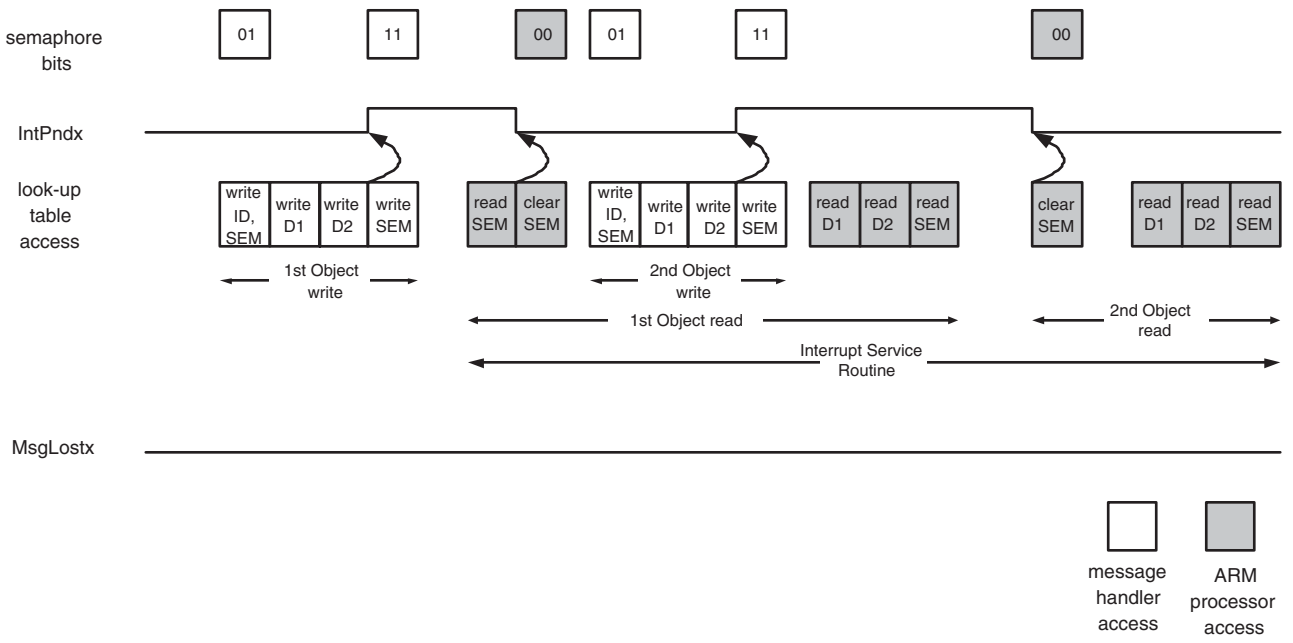


Fig 44. Message gets overwritten

15.3.4 Scenario 3.1: Message gets overwritten indicated by Semaphore bits and Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by the existing semaphore bits and by Message Lost.

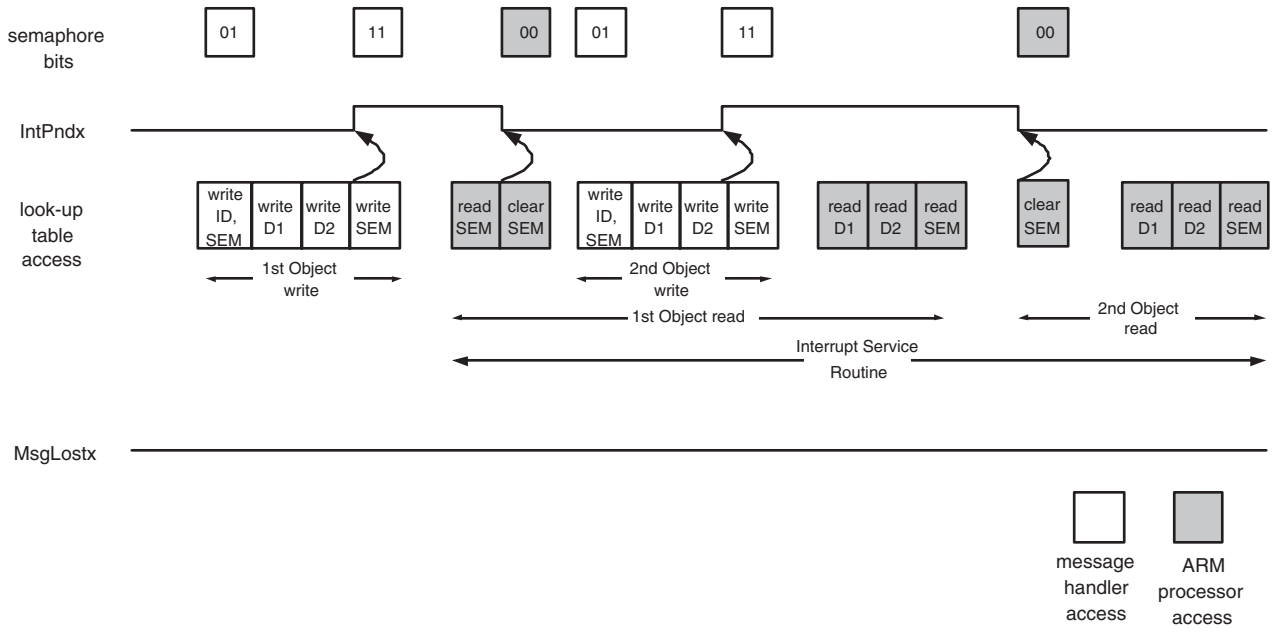


Fig 45. Message overwritten indicated by semaphore bits and message lost

15.3.5 Scenario 3.2: Message gets overwritten indicated by Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by Message Lost.

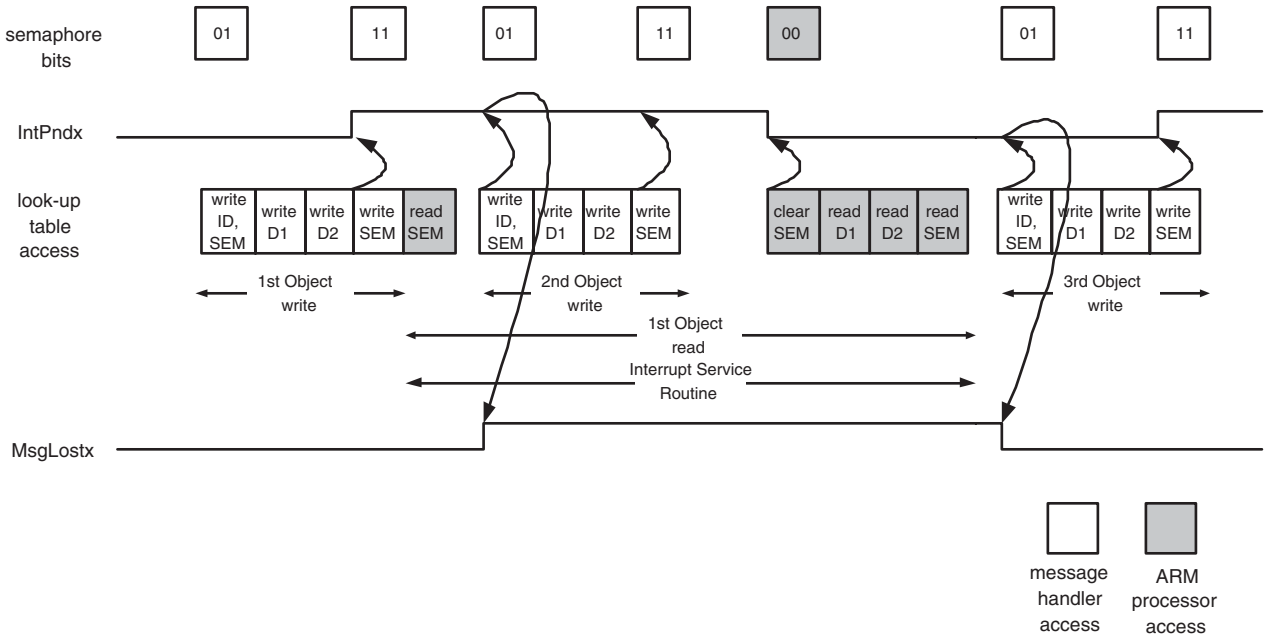


Fig 46. Message overwritten indicated by message lost

15.3.6 Scenario 4: Clearing Message Lost bit

This scenario is a special case in which the lost message bit of an object gets set during an overwrite of a none read message object (2nd Object write). The subsequent read out of that object by Software (1st Object read) clears the pending Interrupt. The 3rd Object write clears the Message Lost bit. Every “write ID, SEM” clears Message Lost bit if no pending Interrupt of that object is set.

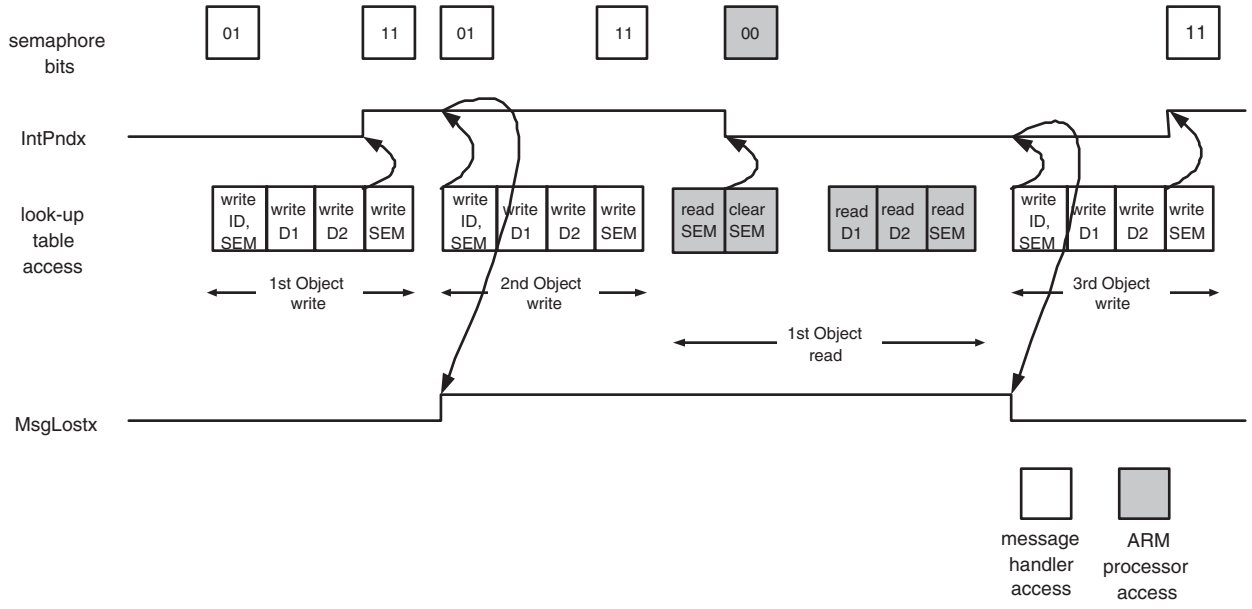


Fig 47. Clearing message lost

16. Examples of acceptance filter tables and ID index values

16.1 Example 1: only one section is used

```
SFF_sa     <    ENDofTable      OR
SFF_GRP_sa <    ENDofTable      OR
EFF_sa     <    ENDofTable      OR
EFF_GRP_sa <    ENDofTable
```

The start address of a section is lower than the end address of all programmed CAN identifiers.

16.2 Example 2: all sections are used

```
SFF_sa     <    SFF_GRP_sa      AND
SFF_GRP_sa <    EFF_sa          AND
EFF_sa     <    EFF_GRP_sa      AND
EFF_GRP_sa <    ENDofTable
```

In cases of a section not being used, the start address has to be set onto the value of the next section start address.

16.3 Example 3: more than one but not all sections are used

If the SFF group is not used, the start address of the SFF Group Section (SFF_GRP_sa register) has to be set to the same value of the next section start address, in this case the start address of the Explicit SFF Section (SFF_sa register).

In cases where explicit identifiers as well as groups of the identifiers are programmed, a CAN identifier search has to start in the explicit identifier section first. If no match is found, it continues the search in the group of identifier section. By this order it can be guaranteed that in case where an explicit identifier match is found, the succeeding software can directly proceed on this certain message whereas in case of a group of identifier match the succeeding software needs more steps to identify the message.

16.4 Configuration example 4

Suppose that the five Acceptance Filter address registers contain the values shown in the third column below. In this case each table contains the decimal number of words and entries shown in the next two columns, and the ID Index field of the CANRFS register can return the decimal values shown in the column ID Indexes for CAN messages whose Identifiers match the entries in that table.

Table 245. Example of Acceptance Filter Tables and ID index Values

Table	Register	Value	# Words	# Entire	ID Indexes
Standard Individual	SFF_sa	0x040	8_{10}	16_{10}	$0\text{-}15_{10}$
Standard Group	SFF_GRP_sa	0x060	4_{10}	4_{10}	$16\text{-}19_{10}$
Extended Individual	EFF_sa	0x070	8_{10}	16_{10}	$20\text{-}55_{10}$
Extended Group	EFF_GRP_sa	0x100	8_{10}	16_{10}	$56\text{-}57_{10}$
	ENDofTable	0x110			

16.5 Configuration example 5

Figure 12–48 below is a more detailed and graphic example of the address registers, table layout, and ID Index values. It shows:

- A Standard Individual table starting at the start of Acceptance Filter RAM and containing 26 Identifiers, followed by:
- A Standard Group table containing 12 ranges of Identifiers, followed by:
- An Extended Individual table containing 3 Identifiers, followed by:
- An Extended Group table containing 2 ranges of Identifiers.

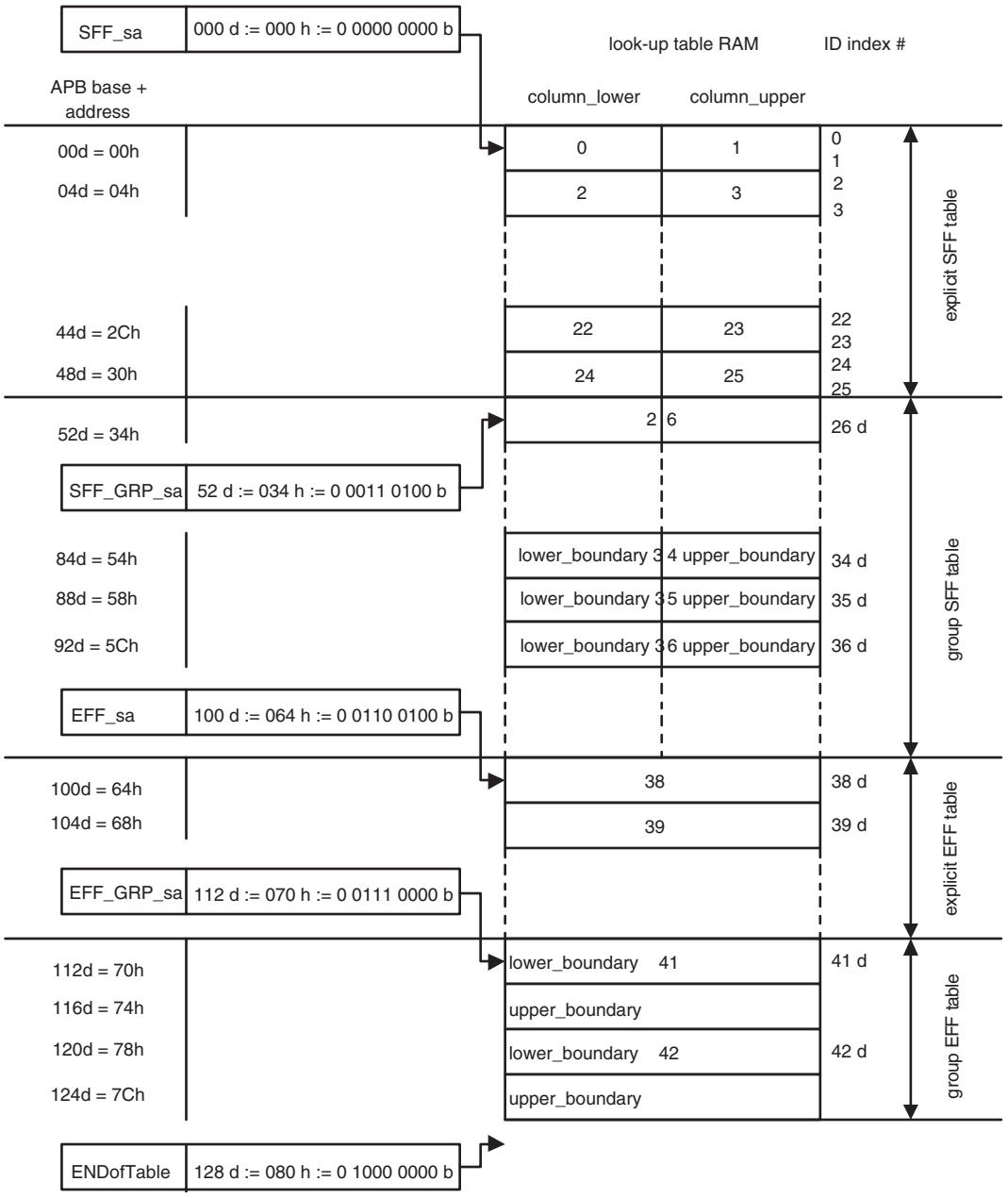


Fig 48. Detailed example of acceptance filter tables and ID index values

16.6 Configuration example 6

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 12–49.

Table 246. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	not activated
Explicit Standard Frame Format	activated
Group of Standard Frame Format	activated
Explicit Extended Frame Format	activated
Group of Extended Frame Format	activated

Explicit standard frame format identifier section (11-bit CAN ID):

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x00. The end of this section is defined in the SFF_GRP_sa register. In the Explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit.

Group of standard frame format identifier section (11-bit CAN ID):

The start address of the Group of Standard Frame Format section is defined with the SFF_GRP_sa register with the value of 0x10. The end of this section is defined with the EFF_sa register. In the Group of Standard Frame Format section two CAN Identifiers with the same Source CAN Channel (SCC) share one 32-bit word and represent a range of CAN Identifiers to be accepted. Bit 31 down to 16 represents the lower boundary and bit 15 down to 0 represents the upper boundary of the range of CAN Identifiers. All Identifiers within this range (including the boundary identifiers) will be accepted. A whole group can be disabled and not used by the acceptance filter by setting the message disable bit in the upper and lower boundary identifier. To provide memory space for four Groups of Standard Frame Format identifiers, the EFF_sa register value is set to 0x20. The identifier group with the Index 9 of this section is not used and therefore disabled.

Explicit extended frame format identifier section (29-bit CAN ID, Figure 12–49)

The start address of the Explicit Extended Frame Format section is defined with the EFF_sa register with the value of 0x20. The end of this section is defined with the EFF_GRP_sa register. In the explicit Extended Frame Format section only one CAN Identifier with its Source CAN Channel (SCC) is programmed per address line. To provide memory space for four Explicit Extended Frame Format identifiers, the EFF_GRP_sa register value is set to 0x30.

Group of extended frame format identifier section (29-bit CAN ID, Figure 12–49)

The start address of the Group of Extended Frame Format is defined with the EFF_GRP_sa register with the value of 0x30. The end of this section is defined with the End of Table address register (ENDofTable). In the Group of Extended Frame Format section the boundaries are programmed with a pair of address lines; the first is the lower boundary, the second the upper boundary. To provide memory space for two Groups of Extended Frame Format Identifiers, the ENDofTable register value is set to 0x40.

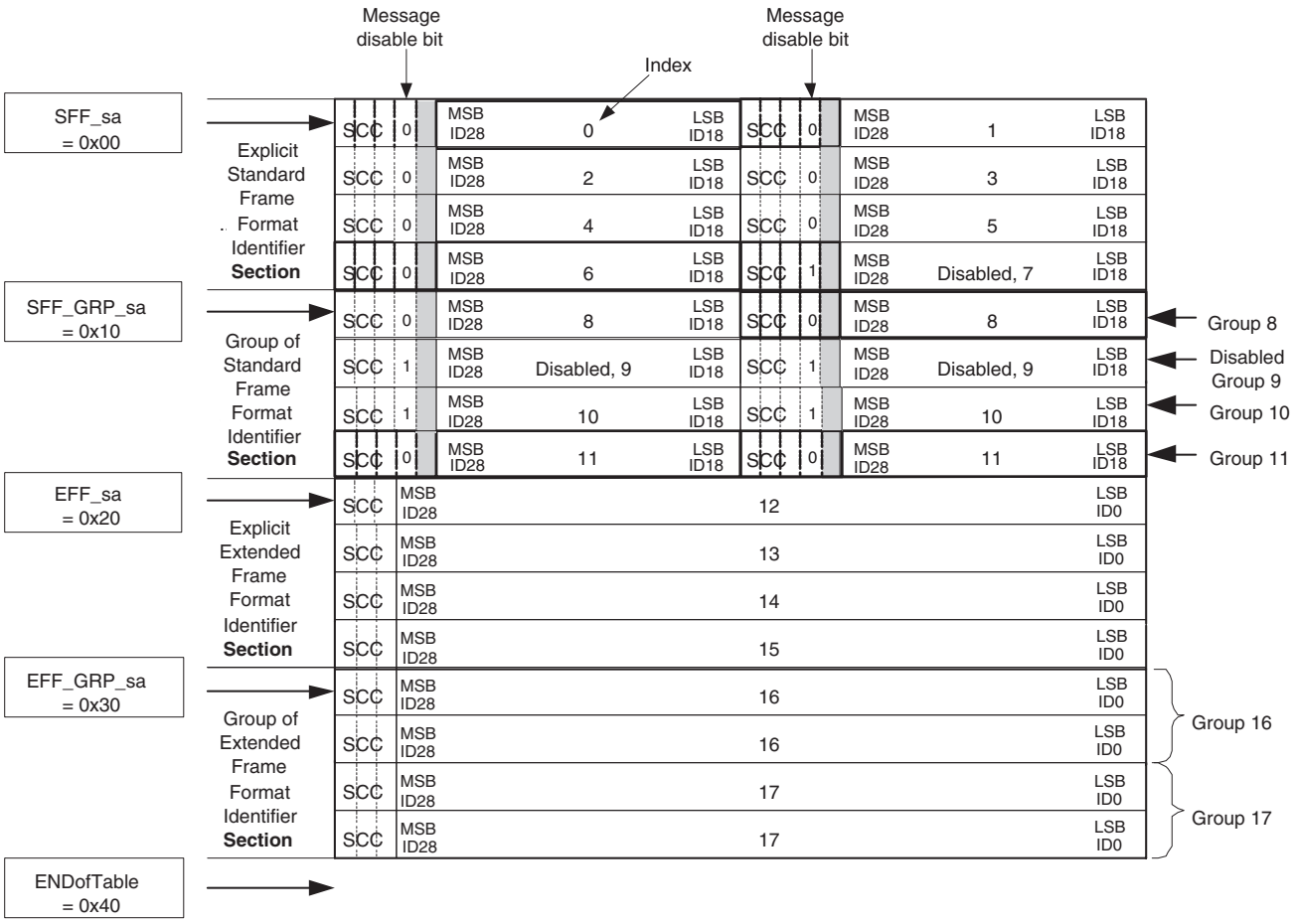


Fig 49. ID Look-up table configuration example (no FullCAN)

16.7 Configuration example 7

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 12–50.

This example uses a typical configuration in which FullCAN as well as Explicit Standard Frame Format messages are defined. As described in Section 12–14.1 “Acceptance filter search algorithm”, acceptance filtering takes place in a certain order. With the enabled FullCAN section, the identifier screening process of the acceptance filter starts always in the FullCAN section first, before it continues with the rest of enabled sections.e disabled.

Table 247. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	activated and enabled
Explicit Standard Frame Format	activated
Group of Standard Frame Format	not activated
Explicit Extended Frame Format	not activated
Group of Extended Frame Format	not activated

FullCAN explicit standard frame format identfier section (11-bit CAN ID)

The start address of the FullCAN Explicit Standard Frame Format Identifier section is (automatically) set to 0x00. The end of this section is defined in the SFF_sa register. In the FullCAN ID section only identifiers of FullCAN Object are stored for acceptance filtering. In this section two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. The FullCAN Object data for each defined identifier can be found in the FullCAN Message Object section. In case of an identifier match during the acceptance filter process, the received FullCAN message object data is moved from the Receive Buffer of the appropriate CAN Controller into the FullCAN Message Object section. To provide memory space for eight FullCAN, Explicit Standard Frame Format identifiers, the SFF_sa register value is set to 0x10. The identifier with the Index 1 of this section is not used and therefore disabled.

Explicit standard frame format identifier section (11-bit CAN ID)

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x10. The end of this section is defined in the End of Table address register (ENDofTable). In the explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channel (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. To provide memory space for eight Explicit Standard Frame Format identifiers, the ENDofTable register value is set to 0x20.

FullCAN message object data section

The start address of the FullCAN Message Object Data section is defined with the ENDofTable register. The number of enabled FullCAN identifiers is limited to the available memory space in the FullCAN Message Object Data section. Each defined FullCAN Message needs three address lines for the Message Data in the FullCAN Message Object Data section. The FullCAN Message Object section is organized in that way, that each Index number of the FullCAN Identifier section corresponds to a Message Object Number in the FullCAN Message Object section.

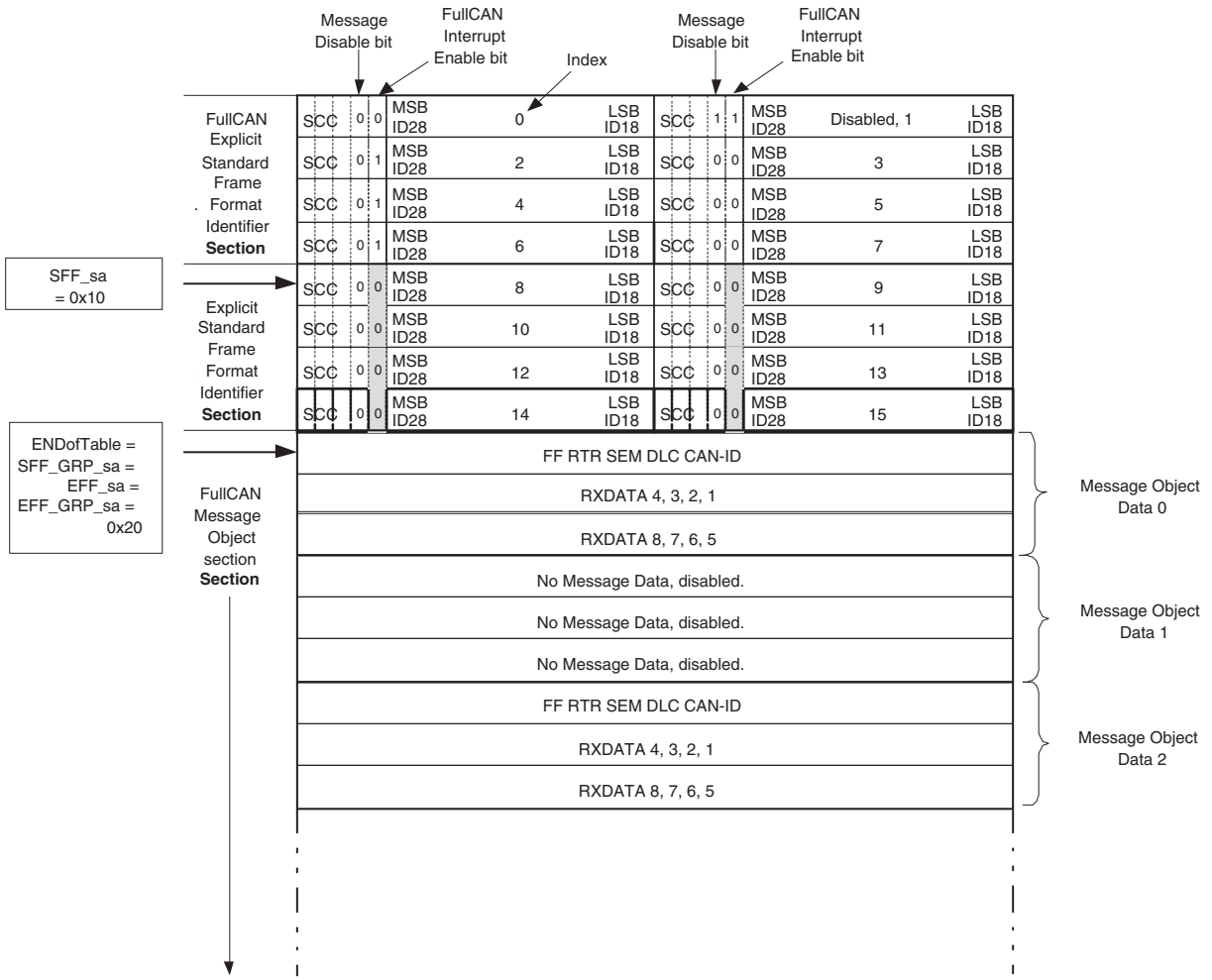


Fig 50. ID Look-up table configuration example (FullCAN activated and enabled)

16.8 Look-up table programming guidelines

All identifier sections of the ID Look-up Table have to be programmed in such a way, that each active section is organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

In cases, where a syntax error in the ID Look-up Table is encountered, the Look-up Table address of the incorrect line is made available in the Look-up Table Error Address Register (LUTerrAd).

The reporting process in the Look-up Table Error Address Register (LUTerrAd) is a “run-time” process. Only those address lines with syntax error are reported, which were passed through the acceptance filtering process.

The following general rules for programming the Look-up Table apply:

- Each section has to be organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) in conjunction with the CAN Identifier (there is no exception for disabled identifiers).
- The upper and lower bound in a Group of Identifiers definition has to be from the same Source CAN Channel.
- To disable a Group of Identifiers the message disable bit has to be set for both, the upper and lower bound.

1. Introduction

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 1 ms frames. Each frame contains a Start-Of-Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 16 logical or 32 physical endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device. Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the rate of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

For more information on the Universal Serial Bus, see the USB Implementers Forum website.

The USB device controller on the LPC2400 enables full-speed (12 Mb/s) data exchange with a USB host controller.

Table 248. USB related acronyms, abbreviations, and definitions used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-performance bus
ATLE	Auto Transfer Length Extraction
ATX	Analog Transceiver
DD	DMA Descriptor
DDP	DMA Description Pointer
DMA	Direct Memory Access
EOP	End-Of-Packet
EP	Endpoint
EP_RAM	Endpoint RAM
FS	Full Speed
LED	Light Emitting Diode
LS	Low Speed
MPS	Maximum Packet Size
NAK	Negative Acknowledge
PLL	Phase Locked Loop
RAM	Random Access Memory
SOF	Start-Of-Frame
SIE	Serial Interface Engine

Table 248. 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

2. Features

- Fully compliant with the USB 2.0 specification (full speed).
- Supports 32 physical (16 logical) endpoints.
- Supports Control, Bulk, Interrupt and Isochronous endpoints.
- Scalable realization of endpoints at run time.
- Endpoint maximum packet size selection (up to USB maximum specification) by software at run time.
- Supports SoftConnect and GoodLink features.
- Supports DMA transfers on all non-control endpoints.
- Allows dynamic switching between CPU controlled and DMA modes.
- Double buffer implementation for Bulk and Isochronous endpoints.

3. Fixed endpoint configuration

Table 13–249 shows the supported endpoint configurations. Endpoints are realized and configured at run time using the Endpoint realization registers, documented in Section 13–8.5 “Endpoint realization registers”.

Table 249. 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

Table 249. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Packet size (bytes)	Double buffer
8	16	Bulk	Out	8, 16, 32, 64	Yes
8	17	Bulk	In	8, 16, 32, 64	Yes
9	18	Isochronous	Out	1 to 1023	Yes
9	19	Isochronous	In	1 to 1023	Yes
10	20	Interrupt	Out	1 to 64	No
10	21	Interrupt	In	1 to 64	No
11	22	Bulk	Out	8, 16, 32, 64	Yes
11	23	Bulk	In	8, 16, 32, 64	Yes
12	24	Isochronous	Out	1 to 1023	Yes
12	25	Isochronous	In	1 to 1023	Yes
13	26	Interrupt	Out	1 to 64	No
13	27	Interrupt	In	1 to 64	No
14	28	Bulk	Out	8, 16, 32, 64	Yes
14	29	Bulk	In	8, 16, 32, 64	Yes
15	30	Bulk	Out	8, 16, 32, 64	Yes
15	31	Bulk	In	8, 16, 32, 64	Yes

4. Functional description

The architecture of the USB device controller is shown below in Figure 13–51.

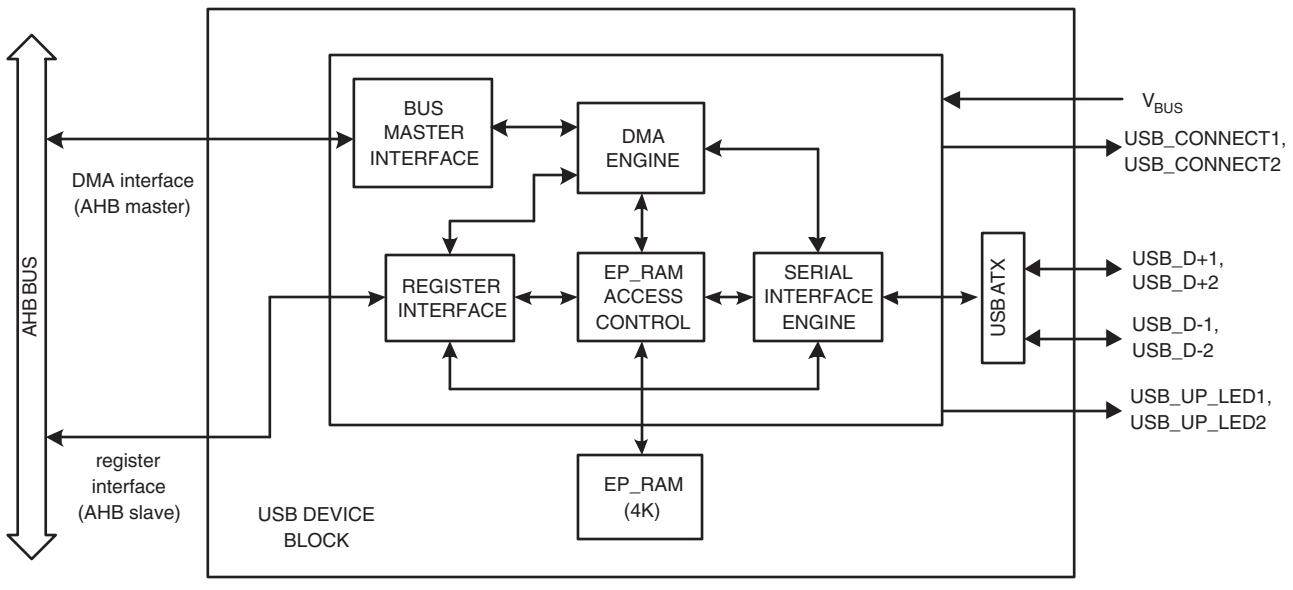


Fig 51. USB device controller block diagram

4.1 Analog transceiver

The USB Device Controller has a built-in analog transceiver (ATX). The USB ATX sends/receives the bi-directional D+ and D- signals of the USB bus.

4.2 Serial Interface Engine (SIE)

The SIE implements the full USB protocol layer. It is completely hardwired for speed and needs no firmware intervention. It handles transfer of data between the endpoint buffers in EP_RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

4.3 Endpoint RAM (EP_RAM)

Each endpoint buffer is implemented as an SRAM based FIFO. The SRAM dedicated for this purpose is called the EP_RAM. Each realized endpoint has a reserved space in the EP_RAM. The total EP_RAM space required depends on the number of realized endpoints, the maximum packet size of the endpoint, and whether the endpoint supports double buffering.

4.4 EP_RAM access control

The EP_RAM Access Control logic handles transfer of data from/to the EP_RAM and the three sources that can access it: the CPU (via the Register Interface), the SIE, and the DMA Engine.

4.5 DMA engine and bus master interface

When enabled for an endpoint, the DMA Engine transfers data between RAM on the AHB bus and the endpoint's buffer in EP_RAM. A single DMA channel is shared between all endpoints. When transferring data, the DMA Engine functions as a master on the AHB bus through the bus master interface.

4.6 Register interface

The Register Interface allows the CPU to control the operation of the USB Device Controller. It also provides a way to write transmit data to the controller and read receive data from the controller.

4.7 SoftConnect

The connection to the USB is accomplished by bringing D+ (for a full-speed device) HIGH through a 1.5 kOhm pull-up resistor. The SoftConnect feature can be used to allow software to finish its initialization sequence before deciding to establish connection to the USB. Re-initialization of the USB bus connection can also be performed without having to unplug the cable.

To use the SoftConnect feature, the CONNECT signal should control an external switch that connects the 1.5 kOhm resistor between D+ and +3.3V. Software can then control the CONNECT signal by writing to the CON bit using the SIE Set Device Status command.

4.8 GoodLink

Good USB connection indication is provided through GoodLink technology. When the device is successfully enumerated and configured, the LED indicator will be permanently ON. During suspend, the LED will be OFF.

This feature provides a user-friendly indicator on the status of the USB device. It is a useful field diagnostics tool to isolate faulty equipment.

To use the GoodLink feature the UP_LED signal should control an LED. The UP_LED signal is controlled using the SIE Configure Device command.

5. Operational overview

Transactions on the USB bus transfer data between device endpoints and the host. The direction of a transaction is defined with respect to the host. OUT transactions transfer data from the host to the device. IN transactions transfer data from the device to the host. All transactions are initiated by the host controller.

For an OUT transaction, the USB ATX receives the bi-directional D+ and D- signals of the USB bus. The Serial Interface Engine (SIE) receives the serial data from the ATX and converts it into a parallel data stream. The parallel data is written to the corresponding endpoint buffer in the EP_RAM.

For IN transactions, the SIE reads the parallel data from the endpoint buffer in EP_RAM, converts it into serial data, and transmits it onto the USB bus using the USB ATX.

Once data has been received or sent, the endpoint buffer can be read or written. How this is accomplished depends on the endpoint's type and operating mode. The two operating modes for each endpoint are Slave (CPU-controlled) mode, and DMA mode.

In Slave mode, the CPU transfers data between RAM and the endpoint buffer using the Register Interface. See Section 13–12 "Slave mode operation" for a detailed description of this mode.

In DMA mode, the DMA transfers data between RAM and the endpoint buffer. See Section 13–13 "DMA operation" for a detailed description of this mode.

6. Pin description

The device controller can access two USB ports indicated by suffixes 1 and 2 in the USB pin names and referred to as USB port 1 (U1) and USB port 2 (U2) in the following text.

Table 250. USB device pin description

Name	Direction	Description
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding PINSEL register, it is driven HIGH internally.
USB_CONNECT1, USB_CONNECT2	O	SoftConnect control signal.
USB_UP_LED1, USB_UP_LED2	O	GoodLink LED control signal.
USB_D+1, USB_D+2	I/O	Positive differential data.
USB_D−1, USB_D−2	I/O	Negative differential data.

6.1 USB device usage note

The USB device interface can be routed to either USB port1 (using USB_CONNECT1, USB_UP_LED1, USB_D+1, USB_D−1) or USB port2 (using USB_CONNECT2, USB_UP_LED2, USB_D+2, USB_D−2) to allow for more versatile pin multiplexing (see Section 13–8.1.1 “USB Port Select register (USBPortSel - 0xFFE0 C110)”).

To use both ports for USB transfer, port1 has to be configured as host and port2 has to be configured as device. See Section 15–5.3 “Connecting USB as one port host and one port device” on page 345 for details.

7. Clocking and power management

This section describes the clocking and power management features of the USB Device Controller.

7.1 Power requirements

The USB protocol insists on power management by the device. This becomes very critical if the device draws power from the bus (bus-powered device). The following constraints should be met by a bus-powered device:

1. A device in the non-configured state should draw a maximum of 100 mA from the bus.
2. A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
3. A suspended device can draw a maximum of 500 μA.

7.2 Clocks

The USB device controller clocks are shown in Table 13–251

Table 251. USB device controller clock sources

Clock source	Description
AHB master clock	Clock for the AHB master bus interface and DMA
AHB slave clock	Clock for the AHB slave interface
usbclk	48 MHz clock from the USB clock divider, used to recover the 12 MHz clock from the USB bus

7.3 Power management support

To help conserve power, the USB device controller automatically disables the AHB master clock and usbclk when not in use.

When the USB Device Controller goes into the suspend state (bus is idle for 3 ms), the usbclk input to the device controller is automatically disabled, helping to conserve power. However, if software wishes to access the device controller registers, usbclk must be active. To allow access to the device controller registers while in the suspend state, the USBClkCtrl and USBClkSt registers are provided.

When software wishes to access the device controller registers, it should first ensure usbclk is enabled by setting DEV_CLK_EN in the USBClkCtrl register, and then poll the corresponding DEV_CLK_ON bit in USBClkSt until set. Once set, usbclk will remain enabled until DEV_CLK_EN is cleared by software.

When a DMA transfer occurs, the device controller automatically turns on the AHB master clock. Once asserted, it remains active for a minimum of 2 ms (2 frames), to help ensure that DMA throughput is not affected by turning off the AHB master clock. 2 ms after the last DMA access, the AHB master clock is automatically disabled to help conserve power. If desired, software also has the capability of forcing this clock to remain enabled using the USBClkCtrl register.

Note that the AHB slave clock is always enabled as long as the PCUSB bit of PCONP is set. When the device controller is not in use, all of the device controller clocks may be disabled by clearing PCUSB.

The USB_NEED_CLK signal is used to facilitate going into and waking up from chip Power-down mode. USB_NEED_CLK is asserted if any of the bits of the USBClkSt register are asserted.

After entering the suspend state with DEV_CLK_EN and AHB_CLK_EN cleared, the DEV_CLK_ON and AHB_CLK_ON will be cleared when the corresponding clock turns off. When both bits are zero, USB_NEED_CLK will be low, indicating that the chip can be put into Power-down mode by writing to the PCON register. The status of USB_NEED_CLK can be read from the USBIntSt register.

Any bus activity in the suspend state will cause the USB_NEED_CLK signal to be asserted. When the USB is configured to be a wakeup source from power-down (USBWAKE bit set in the INTWAKE register), the assertion of USB_NEED_CLK causes the chip to wake up from Power-down mode.

7.4 Remote wake-up

The USB device controller supports software initiated remote wake-up. Remote wake-up involves resume signaling on the USB bus initiated from the device. This is done by clearing the SUS bit in the SIE Set Device Status register. Before writing into the register, all the clocks to the device controller have to be enabled using the USBClkCtrl register.

8. Register description

Table 13–252 shows the USB Device Controller registers directly accessible by the CPU. The Serial Interface Engine (SIE) has other registers that are indirectly accessible via the SIE command registers. See Section 13–10 “Serial interface engine command description” for more info.

Table 252. USB device register map

Name	Description	Access	Reset value[1]	Address
Port select register				
USBPortSel	USB Port Select	R/W	0x0000 0000	0xFFE0 C110
Clock control registers				
USBClkCtrl	USB Clock Control	R/W	0x0000 0000	0xFFE0 CFF4
USBClkSt	USB Clock Status	RO	0x0000 0000	0xFFE0 CFF8
Device interrupt registers				
USBIntSt	USB Interrupt Status	R/W	0x8000 0000	0xE01F C1C0
USBDevIntSt	USB Device Interrupt Status	RO	0x0000 0010	0xFFE0 C200
USBDevIntEn	USB Device Interrupt Enable	R/W	0x0000 0000	0xFFE0 C204
USBDevIntClr	USB Device Interrupt Clear	WO	0x0000 0000	0xFFE0 C208
USBDevIntSet	USB Device Interrupt Set	WO	0x0000 0000	0xFFE0 C20C
USBDevIntPri	USB Device Interrupt Priority	WO	0x00	0xFFE0 C22C
Endpoint interrupt registers				
USBEpIntSt	USB Endpoint Interrupt Status	RO	0x0000 0000	0xFFE0 C230
USBEpIntEn	USB Endpoint Interrupt Enable	R/W	0x0000 0000	0xFFE0 C234
USBEpIntClr	USB Endpoint Interrupt Clear	WO	0x0000 0000	0xFFE0 C238
USBEpIntSet	USB Endpoint Interrupt Set	WO	0x0000 0000	0xFFE0 C23C
USBEpIntPri	USB Endpoint Priority	WO[2]	0x0000 0000	0xFFE0 C240
Endpoint realization registers				
USBReEp	USB Realize Endpoint	R/W	0x0000 0003	0xFFE0 C244
USBEpInd	USB Endpoint Index	WO[2]	0x0000 0000	0xFFE0 C248
USBMaxPSize	USB MaxPacketSize	R/W	0x0000 0008	0xFFE0 C24C
USB transfer registers				
USBRxData	USB Receive Data	RO	0x0000 0000	0xFFE0 C218
USBRxPLen	USB Receive Packet Length	RO	0x0000 0000	0xFFE0 C220
USBTxData	USB Transmit Data	WO[2]	0x0000 0000	0xFFE0 C21C
USBTxPLen	USB Transmit Packet Length	WO[2]	0x0000 0000	0xFFE0 C224
USBCtrl	USB Control	R/W	0x0000 0000	0xFFE0 C228
SIE Command registers				
USBCmdCode	USB Command Code	WO[2]	0x0000 0000	0xFFE0 C210
USBCmdData	USB Command Data	RO	0x0000 0000	0xFFE0 C214
DMA registers				
USBDMARSt	USB DMA Request Status	RO	0x0000 0000	0xFFE0 C250
USBDMARClr	USB DMA Request Clear	WO[2]	0x0000 0000	0xFFE0 C254
USBDMARSet	USB DMA Request Set	WO[2]	0x0000 0000	0xFFE0 C258

Table 252. USB device register map

Name	Description	Access	Reset value[1]	Address
USBUDCAH	USB UDCA Head	R/W	0x0000 0000	0xFFE0 C280
USBEpDMASt	USB Endpoint DMA Status	RO	0x0000 0000	0xFFE0 C284
USBEpDMAEn	USB Endpoint DMA Enable	WO[2]	0x0000 0000	0xFFE0 C288
USBEpDMADis	USB Endpoint DMA Disable	WO[2]	0x0000 0000	0xFFE0 C28C
USBDMAIntSt	USB DMA Interrupt Status	RO	0x0000 0000	0xFFE0 C290
USBDMAIntEn	USB DMA Interrupt Enable	R/W	0x0000 0000	0xFFE0 C294
USBEoTIntSt	USB End of Transfer Interrupt Status	RO	0x0000 0000	0xFFE0 C2A0
USBEoTIntClr	USB End of Transfer Interrupt Clear	WO[2]	0x0000 0000	0xFFE0 C2A4
USBEoTIntSet	USB End of Transfer Interrupt Set	WO[2]	0x0000 0000	0xFFE0 C2A8
USBNDDRIntSt	USB New DD Request Interrupt Status	RO	0x0000 0000	0xFFE0 C2AC
USBNDDRIntClr	USB New DD Request Interrupt Clear	WO[2]	0x0000 0000	0xFFE0 C2B0
USBNDDRIntSet	USB New DD Request Interrupt Set	WO[2]	0x0000 0000	0xFFE0 C2B4
USBSysErrIntSt	USB System Error Interrupt Status	RO	0x0000 0000	0xFFE0 C2B8
USBSysErrIntClr	USB System Error Interrupt Clear	WO[2]	0x0000 0000	0xFFE0 C2BC
USBSysErrIntSet	USB System Error Interrupt Set	WO[2]	0x0000 0000	0xFFE0 C2C0

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

[2] Reading WO register will return an invalid value.

8.1 Port select register

8.1.1 USB Port Select register (USBPortSel - 0xFFE0 C110)

This register selects the USB port pins the USB device signals are routed to. USBPortSel is a read/write register.

Table 253. USB Port Select register (USBPortSel - address 0xFFE0 C110) bit description

Bit	Symbol	Value	Description	Reset value
1:0	PORTSEL	0x0	The USB device controller signals are mapped to the U1 port: USB_CONNECT1, USB_UP_LED1, USB_D+1, USB_D−1.	0
		0x3	The USB device controller signals are mapped to the U2 port:USB_CONNECT2, USB_UP_LED2, USB_D+2, USB_D−2	
31:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.2 Clock control registers

8.2.1 USB Clock Control register (USBClkCtrl - 0xFFE0 CFF4)

This register controls the clocking of the USB Device Controller. Whenever software wants to access the device controller registers, both DEV_CLK_EN and AHB_CLK_EN must be set. The PORTSEL_CLK_EN bit need only be set when accessing the USBPortSel register.

The software does not have to repeat this exercise for every register access, provided that the corresponding USBClkCtrl bits are already set. Note that this register is functional only when the PCUSB bit of PCONP is set; when PCUSB is cleared, all clocks to the device controller are disabled irrespective of the contents of this register. USBClkCtrl is a read/write register.

Table 254. USBClkCtrl register (USBClkCtrl - address 0xFFE0 CFF4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	DEV_CLK_EN	Device clock enable. Enables the usbclk input to the device controller	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PORTSEL_CLK_EN	Port select register clock enable.	NA
4	AHB_CLK_EN	AHB clock enable	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.2.2 USB Clock Status register (USBClkSt - 0xFFE0 CFF8)

This register holds the clock availability status. The bits of this register are ORed together to form the USB_NEED_CLK signal. When enabling a clock via USBClkCtrl, software should poll the corresponding bit in USBClkSt. If it is set, then software can go ahead with the register access. Software does not have to repeat this exercise for every access, provided that the USBClkCtrl bits are not disturbed. USBClkSt is a read only register.

Table 255. USB Clock Status register (USBClkSt - 0xFFE0 CFF8) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	DEV_CLK_ON	Device clock on. The usbclk input to the device controller is active.	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PORTSEL_CLK_ON	Port select register clock on.	NA
4	AHB_CLK_ON	AHB clock on.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.3 Device interrupt registers

8.3.1 USB Interrupt Status register (USBIntSt - 0xE01F C1C0)

The USB Device Controller has three interrupt lines. This register allows software to determine their status with a single read operation. All three interrupt lines are ORed together to a single channel of the vectored interrupt controller. This register also contains the USB_NEED_CLK status and EN_USB_INTS control bits. USBIntSt is a read/write register.

Table 256. USB Interrupt Status register (USBIntSt - address 0xE01F C1C0) bit description

Bit	Symbol	Description	Reset value
0	USB_INT_REQ_LP	Low priority interrupt line status. This bit is read only.	0
1	USB_INT_REQ_HP	High priority interrupt line status. This bit is read only.	0
2	USB_INT_REQ_DMA	DMA interrupt line status. This bit is read only.	0
7:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	USB_NEED_CLK	USB need clock indicator. This bit is set to 1 when USB activity or a change of state on the USB data pins is detected, and it indicates that a PLL supplied clock of 48 MHz is needed. Once USB_NEED_CLK becomes one, it it resets to zero 5 ms after the last packet has been received/sent, or 2 ms after the Suspend Change (SUS_CH) interrupt has occurred. A change of this bit from 0 to 1 can wake up the microcontroller if activity on the USB bus is selected to wake up the part from the Power Down mode (see Section 4–7.8 “Interrupt Wakeup Register (INTWAKE - 0xE01F C144)” for details). Also see Section 4–5.10 “PLL and Power-down mode” and Section 4–7.9 “Power Control for Peripherals register (PCONP - 0xE01F COC4)” for considerations about the PLL and invoking the Power Down mode. This bit is read only.	0
30:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31	EN_USB_INTS	Enable all USB interrupts. When this bit is cleared, the Vectored Interrupt Controller does not see the ORed output of the USB interrupt lines.	1

8.3.2 USB Device Interrupt Status register (USBDevIntSt - 0xFFE0 C200)

The USBDevIntSt register holds the status of each interrupt. A 0 indicates no interrupt and 1 indicates the presence of the interrupt. USBDevIntSt is a read only register.

Table 257. USB Device Interrupt Status register (USBDevIntSt - address 0xFFE0 C200) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	23	22	21	20	19	18	17	16
Symbol	-	-	-	-	-	-	-	-
Bit	15	14	13	12	11	10	9	8
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	7	6	5	4	3	2	1	0
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 258. USB Device Interrupt Status register (USBDevIntSt - address 0xFFE0 C200) bit description

Bit	Symbol	Description	Reset value
0	FRAME	The frame interrupt occurs every 1 ms. This is used in isochronous packet transfers.	0
1	EP_FAST	Fast endpoint interrupt. If an Endpoint Interrupt Priority register (USBEpIntPri) bit is set, the corresponding endpoint interrupt will be routed to this bit.	0
2	EP_SLOW	Slow endpoints interrupt. If an Endpoint Interrupt Priority Register (USBEpIntPri) bit is not set, the corresponding endpoint interrupt will be routed to this bit.	0
3	DEV_STAT	Set when USB Bus reset, USB suspend change or Connect change event occurs. Refer to Section 13–10.6 “Set Device Status (Command: 0xFE, Data: write 1 byte)” on page 313.	0
4	CCEMPTY	The command code register (USBCmdCode) is empty (New command can be written).	1
5	CDFULL	Command data register (USBCmdData) is full (Data can be read now).	0
6	RxENDPKT	The current packet in the endpoint buffer is transferred to the CPU.	0
7	TxENDPKT	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen).	0
8	EP_RLZED	Endpoints realized. Set when Realize Endpoint register (USBReEp) or MaxPacketSize register (USBMaxPSize) is updated and the corresponding operation is completed.	0
9	ERR_INT	Error Interrupt. Any bus error interrupt from the USB device. Refer to Section 13–10.9 “Read Error Status (Command: 0xFB, Data: read 1 byte)” on page 315	0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.3.3 USB Device Interrupt Enable register (USBDevIntEn - 0xFFE0 C204)

Writing a one to a bit in this register enables the corresponding bit in USBDevIntSt to generate an interrupt on one of the interrupt lines when set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri. USBDevIntEn is a read/write register.

Table 259. USB Device Interrupt Enable register (USBDevIntEn - address 0xFFE0 C204) bit allocation

Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 260. USB Device Interrupt Enable register (USBDevIntEn - address 0xFFE0 C204) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBDevIntEn bit allocation table above	0	No interrupt is generated.	0
		1	An interrupt will be generated when the corresponding bit in the Device Interrupt Status (USBDevIntSt) register (Table 13–257) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	

8.3.4 USB Device Interrupt Clear register (USBDevIntClr - 0xFFE0 C208)

Writing one to a bit in this register clears the corresponding bit in USBDevIntSt. Writing a zero has no effect.

Remark: Before clearing the EP_SLOW or EP_FAST interrupt bits, the corresponding endpoint interrupts in USBEpIntSt should be cleared.

USBDevIntClr is a write only register.

Table 261. USB Device Interrupt Clear register (USBDevIntClr - address 0xFFE0 C208) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 262. USB Device Interrupt Clear register (USBDevIntClr - address 0xFFE0 C208) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBDevIntClr bit allocation table above	0	No effect.	0
		1	The corresponding bit in USBDevIntSt (Section 13–8.3.2) is cleared.	

8.3.5 USB Device Interrupt Set register (USBDevIntSet - 0xFFE0 C20C)

Writing one to a bit in this register sets the corresponding bit in the USBDevIntSt. Writing a zero has no effect

USBDevIntSet is a write only register.

Table 263. USB Device Interrupt Set register (USBDevIntSet - address 0xFFE0 C20C) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	-	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	7	6	5	4	3	2	1	0
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 264. USB Device Interrupt Set register (USBDevIntSet - address 0xFFE0 C20C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBDevIntSet bit allocation table above	0	No effect.	0
		1	The corresponding bit in USBDevIntSt (Section 13–8.3.2) is set.	

8.3.6 USB Device Interrupt Priority register (USBDevIntPri - 0xFFE0 C22C)

Writing one to a bit in this register causes the corresponding interrupt to be routed to the USB_INT_REQ_HP interrupt line. Writing zero causes the interrupt to be routed to the USB_INT_REQ_LP interrupt line. Either the EP_FAST or FRAME interrupt can be routed to USB_INT_REQ_HP, but not both. If the software attempts to set both bits to one, no interrupt will be routed to USB_INT_REQ_HP. USBDevIntPri is a write only register.

Table 265. USB Device Interrupt Priority register (USBDevIntPri - address 0xFFE0 C22C) bit description

Bit	Symbol	Value	Description	Reset value
0	FRAME	0	FRAME interrupt is routed to USB_INT_REQ_LP.	0
		1	FRAME interrupt is routed to USB_INT_REQ_HP.	
1	EP_FAST	0	EP_FAST interrupt is routed to USB_INT_REQ_LP.	0
		1	EP_FAST interrupt is routed to USB_INT_REQ_HP.	
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.4 Endpoint interrupt registers

The registers in this group facilitate handling of endpoint interrupts. Endpoint interrupts are used in Slave mode operation.

8.4.1 USB Endpoint Interrupt Status register (USBEpIntSt - 0xFFE0 C230)

Each physical non-isochronous endpoint is represented by a bit in this register to indicate that it has generated an interrupt. All non-isochronous OUT endpoints generate an interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet is successfully transmitted, or when a NAK handshake is sent on the bus and the interrupt on NAK feature is enabled (see Section 13–10.3 “Set Mode (Command: 0xF3, Data: write 1 byte)” on page 312). A bit set to one in this register causes either the EP_FAST or EP_SLOW bit of USBDevIntSt to be set depending on the value of the coreesponding bit of USBEpDevIntPri. USBEpIntSt is a read only register.

Note that for Isochronous endpoints, handling of packet data is done when the FRAME interrupt occurs.

Table 266. USB Endpoint Interrupt Status register (USBEpIntSt - address 0xFFE0 C230) bit allocation

Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 267. USB Endpoint Interrupt Status register (USBEpIntSt - address 0xFFE0 C230) bit description

Bit	Symbol	Description	Reset value
0	EP0RX	Endpoint 0, Data Received Interrupt bit.	0
1	EP0TX	Endpoint 0, Data Transmitted Interrupt bit or sent a NAK.	0
2	EP1RX	Endpoint 1, Data Received Interrupt bit.	0
3	EP1TX	Endpoint 1, Data Transmitted Interrupt bit or sent a NAK.	0
4	EP2RX	Endpoint 2, Data Received Interrupt bit.	0
5	EP2TX	Endpoint 2, Data Transmitted Interrupt bit or sent a NAK.	0
6	EP3RX	Endpoint 3, Isochronous endpoint.	NA
7	EP3TX	Endpoint 3, Isochronous endpoint.	NA
8	EP4RX	Endpoint 4, Data Received Interrupt bit.	0
9	EP4TX	Endpoint 4, Data Transmitted Interrupt bit or sent a NAK.	0
10	EP5RX	Endpoint 5, Data Received Interrupt bit.	0
11	EP5TX	Endpoint 5, Data Transmitted Interrupt bit or sent a NAK.	0
12	EP6RX	Endpoint 6, Isochronous endpoint.	NA
13	EP6TX	Endpoint 6, Isochronous endpoint.	NA
14	EP7RX	Endpoint 7, Data Received Interrupt bit.	0
15	EP7TX	Endpoint 7, Data Transmitted Interrupt bit or sent a NAK.	0
16	EP8RX	Endpoint 8, Data Received Interrupt bit.	0
17	EP8TX	Endpoint 8, Data Transmitted Interrupt bit or sent a NAK.	0
18	EP9RX	Endpoint 9, Isochronous endpoint.	NA
19	EP9TX	Endpoint 9, Isochronous endpoint.	NA
20	EP10RX	Endpoint 10, Data Received Interrupt bit.	0
21	EP10TX	Endpoint 10, Data Transmitted Interrupt bit or sent a NAK.	0
22	EP11RX	Endpoint 11, Data Received Interrupt bit.	0
23	EP11TX	Endpoint 11, Data Transmitted Interrupt bit or sent a NAK.	0
24	EP12RX	Endpoint 12, Isochronous endpoint.	NA
25	EP12TX	Endpoint 12, Isochronous endpoint.	NA
26	EP13RX	Endpoint 13, Data Received Interrupt bit.	0
27	EP13TX	Endpoint 13, Data Transmitted Interrupt bit or sent a NAK.	0
28	EP14RX	Endpoint 14, Data Received Interrupt bit.	0

Table 267. USB Endpoint Interrupt Status register (USBEpIntSt - address 0xFFE0 C230) bit description

Bit	Symbol	Description	Reset value
29	EP14TX	Endpoint 14, Data Transmitted Interrupt bit or sent a NAK.	0
30	EP15RX	Endpoint 15, Data Received Interrupt bit.	0
31	EP15TX	Endpoint 15, Data Transmitted Interrupt bit or sent a NAK.	0

8.4.2 USB Endpoint Interrupt Enable register (USBEpIntEn - 0xFFE0 C234)

Setting a bit to 1 in this register causes the corresponding bit in USBEpIntSt to be set when an interrupt occurs for the associated endpoint. Setting a bit to 0 causes the corresponding bit in USBDMARSt to be set when an interrupt occurs for the associated endpoint. USBEpIntEn is a read/write register.

Table 268. USB Endpoint Interrupt Enable register (USBEpIntEn - address 0xFFE0 C234) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 269. USB Endpoint Interrupt Enable register (USBEpIntEn - address 0xFFE0 C234) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntEn bit allocation table above	0	The corresponding bit in USBDMARSt is set when an interrupt occurs for this endpoint.	0
		1	The corresponding bit in USBEpIntSt is set when an interrupt occurs for this endpoint. Implies Slave mode for this endpoint.	

8.4.3 USB Endpoint Interrupt Clear register (USBEpIntClr - 0xFFE0 C238)

Writing a one to this a bit in this register causes the SIE Select Endpoint/Clear Interrupt command to be executed (Table 13–313) for the corresponding physical endpoint. Writing zero has no effect. Before executing the Select Endpoint/Clear Interrupt command, the CDFULL bit in USBDevIntSt is cleared by hardware. On completion of the command, the CDFULL bit is set, USBCmdData contains the status of the endpoint, and the corresponding bit in USBEpIntSt is cleared.

Notes:

- When clearing interrupts using USBEpIntClr, software should wait for CDFULL to be set to ensure the corresponding interrupt has been cleared before proceeding.
- While setting multiple bits in USBEpIntClr simultaneously is possible, it is not recommended; only the status of the endpoint corresponding to the least significant interrupt bit cleared will be available at the end of the operation.
- Alternatively, the SIE Select Endpoint/Clear Interrupt command can be directly invoked using the SIE command registers, but using USBEpIntClr is recommended because of its ease of use.

Each physical endpoint has its own reserved bit in this register. The bit field definition is the same as that of USBEpIntSt shown in Table 13–266 . USBEpIntClr is a write only register.

Table 270. USB Endpoint Interrupt Clear register (USBEpIntClr - address 0xFFE0 C238) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 271. USB Endpoint Interrupt Clear register (USBEpIntClr - address 0xFFE0 C238) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntClr bit allocation table above	0	No effect.	0
		1	Clears the corresponding bit in USBEpIntSt, by executing the SIE Select Endpoint/Clear Interrupt command for this endpoint.	

8.4.4 USB Endpoint Interrupt Set register (USBEpIntSet - 0xFFE0 C23C)

Writing a one to a bit in this register sets the corresponding bit in USBEpIntSt. Writing zero has no effect. Each endpoint has its own bit in this register. USBEpIntSet is a write only register.

Table 272. USB Endpoint Interrupt Set register (USBEpIntSet - address 0xFFE0 C23C) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 273. USB Endpoint Interrupt Set register (USBEpIntSet - address 0xFFE0 C23C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntSet bit allocation table above	0	No effect.	0
		1	Sets the corresponding bit in USBEpIntSt.	

8.4.5 USB Endpoint Interrupt Priority register (USBEpIntPri - 0xFFE0 C240)

This register determines whether an endpoint interrupt is routed to the EP_FAST or EP_SLOW bits of USBDevIntSt. If a bit in this register is set to one, the interrupt is routed to EP_FAST, if zero it is routed to EP_SLOW. Routing of multiple endpoints to EP_FAST or EP_SLOW is possible.

Note that the USBDevIntPri register determines whether the EP_FAST interrupt is routed to the USB_INT_REQ_HP or USB_INT_REQ_LP interrupt line.

USBEpIntPri is a write only register.

Table 274. USB Endpoint Interrupt Priority register (USBEpIntPri - address 0xFFE0 C240) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	E14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 275. USB Endpoint Interrupt Priority register (USBEpIntPri - address 0xFFE0 C240) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntPri bit allocation table above	0	The corresponding interrupt is routed to the EP_SLOW bit of USBDevIntSt	0
		1	The corresponding interrupt is routed to the EP_FAST bit of USBDevIntSt	

8.5 Endpoint realization registers

The registers in this group allow realization and configuration of endpoints at run time.

8.5.1 EP RAM requirements

The USB device controller uses a RAM based FIFO for each endpoint buffer. The RAM dedicated for this purpose is called the Endpoint RAM (EP_RAM). Each endpoint has space reserved in the EP_RAM. The EP_RAM space required for an endpoint depends on its MaxPacketSize and whether it is double buffered. 32 words of EP_RAM are used by the device for storing the endpoint buffer pointers. The EP_RAM is word aligned but the MaxPacketSize is defined in bytes hence the RAM depth has to be adjusted to the next word boundary. Also, each buffer has one word header showing the size of the packet length received.

The EP_ RAM space (in words) required for the physical endpoint can be expressed as

$$EPRAMspace = \left(\frac{MaxPacketSize + 3}{4} + 1\right) \times dbstatus$$

where dbstatus = 1 for a single buffered endpoint and 2 for double a buffered endpoint.

Since all the realized endpoints occupy EP_RAM space, the total EP_RAM requirement is

$$TotalEPRAMspace = 32 + \sum_{n=0}^{N} EPRAMspace(n)$$

where N is the number of realized endpoints. Total EP_RAM space should not exceed 4096 bytes (4 kB, 1 kwords).

8.5.2 USB Realize Endpoint register (USBReEp - 0xFFE0 C244)

Writing one to a bit in this register causes the corresponding endpoint to be realized. Writing zeros causes it to be unrealized. This register returns to its reset state when a bus reset occurs. USBReEp is a read/write register.

Table 276. USB Realize Endpoint register (USBReEp - address 0xFFE0 C244) bit allocation
Reset value: 0x0000 0003

Bit	31	30	29	28	27	26	25	24
Symbol	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

Table 277. USB Realize Endpoint register (USBReEp - address 0xFFE0 C244) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint EP0 is not realized.	1
		1	Control endpoint EP0 is realized.	
1	EP1	0	Control endpoint EP1 is not realized.	1
		1	Control endpoint EP1 is realized.	
31:2	EPxx	0	Endpoint EPxx is not realized.	0
		1	Endpoint EPxx is realized.	

On reset, only the control endpoints are realized. Other endpoints, if required, are realized by programming the corresponding bits in USBReEp. To calculate the required EP_RAM space for the realized endpoints, see Section 13–8.5.1.

Realization of endpoints is a multi-cycle operation. Pseudo code for endpoint realization is shown below.

```
Clear EP_RLZED bit in USBDevIntSt;

for every endpoint to be realized,
{
     /* OR with the existing value of the Realize Endpoint register */
     USBReEp |= (UInt32) ((0x1 << endpt));
     /* Load Endpoint index Reg with physical endpoint no.*/
     USBEpIn = (UInt32) endpointnumber;

     /* load the max packet size Register */
     USBEpMaxPSize = MPS;

     /* check whether the EP_RLZED bit in the Device Interrupt Status register is set
     */
     while (!(USBDevIntSt & EP_RLZED))
     {
          /* wait until endpoint realization is complete */
     }
     /* Clear the EP_RLZED bit */
     Clear EP_RLZED bit in USBDevIntSt;
}
```

The device will not respond to any transactions to unrealized endpoints. The SIE Configure Device command will only cause realized and enabled endpoints to respond to transactions. For details see Table 13–308.

8.5.3 USB Endpoint Index register (USBEpIn - 0xFFE0 C248)

Each endpoint has a register carrying the MaxPacketSize value for that endpoint. This is in fact a register array. Hence before writing, this register is addressed through the USBEpIn register.

The USBEpIn register will hold the physical endpoint number. Writing to USBMaxPSize will set the array element pointed to by USBEpIn. USBEpIn is a write only register.

Table 278. USB Endpoint Index register (USBEpIn - address 0xFFE0 C248) bit description

Bit	Symbol	Description	Reset value
4:0	PHY_EP	Physical endpoint number (0-31)	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

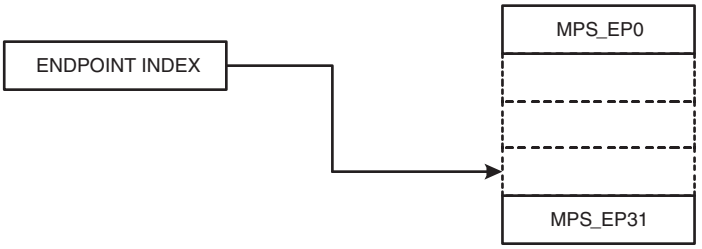
8.5.4 USB MaxPacketSize register (USBMaxPSize - 0xFFE0 C24C)

On reset, the control endpoint is assigned the maximum packet size of 8 bytes. Other endpoints are assigned 0. Modifying USBMaxPSize will cause the endpoint buffer addresses within the EP_RAM to be recalculated. This is a multi-cycle process. At the end, the EP_RLZED bit will be set in USBDevIntSt (Table 13–257). USBMaxPSize array indexing is shown in Figure 13–52. USBMaxPSize is a read/write register.

Table 279. USB MaxPacketSize register (USBMaxPSize - address 0xFFE0 C24C) bit description

Bit	Symbol	Description	Reset value
9:0	MPS	The maximum packet size value.	0x008[1]
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Reset value for EP0 and EP1. All other endpoints have a reset value of 0x0.



The Endpoint Index is set via the USBEpIn register. MPS_EP0 to MPS_EP31 are accessed via the USBMaxPSize register.

Fig 52. USB MaxPacketSize register array indexing

8.6 USB transfer registers

The registers in this group are used for transferring data between endpoint buffers and RAM in Slave mode operation. See Section 13–12 “Slave mode operation”.

8.6.1 USB Receive Data register (USBRxData - 0xFFE0 C218)

For an OUT transaction, the CPU reads the endpoint buffer data from this register. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. On reading this register, data from the selected endpoint buffer is fetched. The data is in little endian format: the first byte received from the USB bus will be available in the least significant byte of USBRxData. USBRxData is a read only register.

Table 280. USB Receive Data register (USBRxData - address 0xFFE0 C218) bit description

Bit	Symbol	Description	Reset value
31:0	RX_DATA	Data received.	0x0000 0000

8.6.2 USB Receive Packet Length register (USBRxPLen - 0xFFE0 C220)

This register contains the number of bytes remaining in the endpoint buffer for the current packet being read via the USBRxData register, and a bit indicating whether the packet is valid or not. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. This register is updated on each read of the USBRxData register. USBRxPLen is a read only register.

Table 281. USB Receive Packet Length register (USBRxPlen - address 0xFFE0 C220) bit description

Bit	Symbol	Value	Description	Reset value
9:0	PKT_LNGTH	-	The remaining number of bytes to be read from the currently selected endpoint's buffer. When this field decrements to 0, the RxENDPKT bit will be set in USBDevIntSt.	0
10	DV		Data valid. This bit is useful for isochronous endpoints. Non-isochronous endpoints do not raise an interrupt when an erroneous data packet is received. But invalid data packet can be produced with a bus reset. For isochronous endpoints, data transfer will happen even if an erroneous packet is received. In this case DV bit will not be set for the packet.	0
		0	Data is invalid.	
		1	Data is valid.	
11	PKT_RDY	-	The PKT_LNGTH field is valid and the packet is ready for reading.	0
31:12	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.6.3 USB Transmit Data register (USBTxData - 0xFFE0 C21C)

For an IN transaction, the CPU writes the endpoint data into this register. Before writing to this register, the WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately, and the packet length should be written to the USBTxPlen register. On writing this register, the data is written to the selected endpoint buffer. The data is in little endian format: the first byte sent on the USB bus will be the least significant byte of USBTxData. USBTxData is a write only register.

Table 282. USB Transmit Data register (USBTxData - address 0xFFE0 C21C) bit description

Bit	Symbol	Description	Reset value
31:0	TX_DATA	Transmit Data.	0x0000 0000

8.6.4 USB Transmit Packet Length register (USBTxPLen - 0xFFE0 C224)

This register contains the number of bytes transferred from the CPU to the selected endpoint buffer. Before writing data to USBTxData, software should first write the packet length (≤ MaxPacketSize) to this register. After each write to USBTxData, hardware decrements USBTxPLen by 4. The WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set to select the desired endpoint buffer before starting this process.

For data buffers larger than the endpoint's MaxPacketSize, software should submit data in packets of MaxPacketSize, and send the remaining extra bytes in the last packet. For example, if the MaxPacketSize is 64 bytes and the data buffer to be transferred is of length 130 bytes, then the software sends two 64-byte packets and the remaining 2 bytes in the last packet. So, a total of 3 packets are sent on USB. USBTxPLen is a write only register.

Table 283. USB Transmit Packet Length register (USBTxPLen - address 0xFFE0 C224) bit description

Bit	Symbol	Value	Description	Reset value
9:0	PKT_LNGTH	-	The remaining number of bytes to be written to the selected endpoint buffer. This field is decremented by 4 by hardware after each write to USBTxData. When this field decrements to 0, the TxENDPKT bit will be set in USBDevIntSt.	0x000
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.6.5 USB Control register (USBCtrl - 0xFFE0 C228)

This register controls the data transfer operation of the USB device. It selects the endpoint buffer that is accessed by the USBRxData and USBTxData registers, and enables reading and writing them. USBCtrl is a read/write register.

Table 284. USB Control register (USBCtrl - address 0xFFE0 C228) bit description

Bit	Symbol	Value	Description	Reset value
0	RD_EN		Read mode control. Enables reading data from the OUT endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBRxData register. This bit is cleared by hardware when the last word of the current packet is read from USBRxData.	0
		0	Read mode is disabled.	
		1	Read mode is enabled.	
1	WR_EN		Write mode control. Enables writing data to the IN endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBTxData register. This bit is cleared by hardware when the number of bytes in USBTxLen have been sent.	0
		0	Write mode is disabled.	
		1	Write mode is enabled.	
5:2	LOG_ENDPOINT	-	Logical Endpoint number.	0x0
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.7 SIE command code registers

The SIE command code registers are used for communicating with the Serial Interface Engine. See Section 13–10 “Serial interface engine command description” for more information.

8.7.1 USB Command Code register (USBCmdCode - 0xFFE0 C210)

This register is used for sending the command and write data to the SIE. The commands written here are propagated to the SIE and executed there. After executing the command, the register is empty, and the CCEMPTY bit of USBDevIntSt register is set. See Section 13–10 for details. USBCmdCode is a write only register.

Table 285. USB Command Code register (USBCmdCode - address 0xFFE0 C210) bit description

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:8	CMD_PHASE		The command phase:	0x00
		0x01	Read	
		0x02	Write	
		0x05	Command	
23:16	CMD_CODE/ CMD_WDATA		This is a multi-purpose field. When CMD_PHASE is Command or Read, this field contains the code for the command (CMD_CODE). When CMD_PHASE is Write, this field contains the command write data (CMD_WDATA).	0x00
31:24	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.7.2 USB Command Data register (USBCmdData - 0xFFE0 C214)

This register contains the data retrieved after executing a SIE command. When the data is ready to be read, the CD_FULL bit of the USBDevIntSt register is set. See Table 13–257 for details. USBCmdData is a read only register.

Table 286. USB Command Data register (USBCmdData - address 0xFFE0 C214) bit description

Bit	Symbol	Description	Reset value
7:0	CMD_RDATA	Command Read Data.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.8 DMA registers

The registers in this group are used for the DMA mode of operation (see Section 13–13 "DMA operation")

8.8.1 USB DMA Request Status register (USBDMARSt - 0xFFE0 C250)

A bit in this register associated with a non-isochronous endpoint is set by hardware when an endpoint interrupt occurs (see the description of USBEpIntSt) and the corresponding bit in USBEpIntEn is 0. A bit associated with an isochronous endpoint is set when the corresponding bit in USBEpIntEn is 0 and a FRAME interrupt occurs. A set bit serves as a flag for the DMA engine to start the data transfer if the DMA is enabled for the corresponding endpoint in the USBEpDMASt register. The DMA cannot be enabled for control endpoints (EP0 and EP1). USBDMARSt is a read only register.

Table 287. USB DMA Request Status register (USBDMARSt - address 0xFFE0 C250) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16

Bit	15	14	13	12	11	10	9	8
Symbol	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
Bit	7	6	5	4	3	2	1	0
Symbol	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

Table 288. USB DMA Request Status register (USBDMARSt - address 0xFFE0 C250) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and EP0 bit must be 0).	0
1	EP1	0	Control endpoint IN (DMA cannot be enabled for this endpoint and EP1 bit must be 0).	0
31:2	EPxx		Endpoint xx (2 ≤ xx ≤ 31) DMA request.	0
		0	DMA not requested by endpoint xx.	
		1	DMA requested by endpoint xx.	

[1] DMA can not be enabled for this endpoint and the corresponding bit in the USBDMARSt must be 0.

8.8.2 USB DMA Request Clear register (USBDMARClr - 0xFFE0 C254)

Writing one to a bit in this register will clear the corresponding bit in the USBDMARSt register. Writing zero has no effect.

This register is intended for initialization prior to enabling the DMA for an endpoint. When the DMA is enabled for an endpoint, hardware clears the corresponding bit in USBDMARSt on completion of a packet transfer. Therefore, software should not clear the bit using this register while the endpoint is enabled for DMA operation.

USBDMARClr is a write only register.

The USBDMARClr bit allocation is identical to the USBDMARSt register (Table 13–287).

Table 289. USB DMA Request Clear register (USBDMARClr - address 0xFFE0 C254) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).	0
1	EP1	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0
31:2	EPxx		Clear the endpoint xx (2 ≤ xx ≤ 31) DMA request.	0
		0	No effect.	
		1	Clear the corresponding bit in USBDMARSt.	

8.8.3 USB DMA Request Set register (USBDMARSet - 0xFFE0 C258)

Writing one to a bit in this register sets the corresponding bit in the USBDMARSt register. Writing zero has no effect.

This register allows software to raise a DMA request. This can be useful when switching from Slave to DMA mode of operation for an endpoint: if a packet to be processed in DMA mode arrives before the corresponding bit of USBEpIntEn is cleared, the DMA request is not raised by hardware. Software can then use this register to manually start the DMA transfer.

Software can also use this register to initiate a DMA transfer to proactively fill an IN endpoint buffer before an IN token packet is received from the host.

USBDMARSet is a write only register.

The USBDMARSet bit allocation is identical to the USBDMARSt register (Table 13–287).

Table 290. USB DMA Request Set register (USBDMARSet - address 0xFFE0 C258) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).	0
1	EP1	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0
31:2	EPxx		Set the endpoint xx (2 ≤ xx ≤ 31) DMA request.	0
		0	No effect.	
		1	Set the corresponding bit in USBDMARSt.	

8.8.4 USB UDCA Head register (USBUDCAH - 0xFFE0 C280)

The UDCA (USB Device Communication Area) Head register maintains the address where the UDCA is located in the USB RAM. Refer to Section 13–13.2 “USB device communication area” and Section 13–13.4 “The DMA descriptor” for more details on the UDCA and DMA descriptors. USBUDCAH is a read/write register.

Table 291. USB UDCA Head register (USBUDCAH - address 0xFFE0 C280) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved. Software should not write ones to reserved bits. The UDCA is aligned to 128-byte boundaries.	0x00
31:7	UDCA_ADDR	Start address of the UDCA.	0

8.8.5 USB EP DMA Status register (USBEpDMASt - 0xFFE0 C284)

Bits in this register indicate whether DMA operation is enabled for the corresponding endpoint. A DMA transfer for an endpoint can start only if the corresponding bit is set in this register. USBEpDMASt is a read only register.

Table 292. USB EP DMA Status register (USBEpDMASt - address 0xFFE0 C284) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0_DMA_ENABLE	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_ENABLE bit must be 0).	0
1	EP1_DMA_ENABLE	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_ENABLE bit must be 0).	0
31:2	EPxx_DMA_ENABLE		endpoint xx (2 ≤ xx ≤ 31) DMA enabled bit.	0
		0	The DMA for endpoint EPxx is disabled.	
		1	The DMA for endpoint EPxx is enabled.	

8.8.6 USB EP DMA Enable register (USBEpDMAEn - 0xFFE0 C288)

Writing one to a bit to this register will enable the DMA operation for the corresponding endpoint. Writing zero has no effect.The DMA cannot be enabled for control endpoints EP0 and EP1. USBEpDMAEn is a write only register.

Table 293. USB EP DMA Enable register (USBEpDMAEn - address 0xFFE0 C288) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0_DMA_ENABLE	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_ENABLE bit value must be 0).	0
1	EP1_DMA_ENABLE	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_ENABLE bit must be 0).	0
31:2	EPxx_DMA_ENABLE		Endpoint xx($2 \le xx \le 31$) DMA enable control bit.	0
		0	No effect.	
		1	Enable the DMA operation for endpoint EPxx.	

8.8.7 USB EP DMA Disable register (USBEpDMADis - 0xFFE0 C28C)

Writing a one to a bit in this register clears the corresponding bit in USBEpDMASt. Writing zero has no effect on the corresponding bit of USBEpDMASt. Any write to this register clears the internal DMA_PROCEED flag. Refer to Section 13–13.5.4 "Optimizing descriptor fetch" for more information on the DMA_PROCEED flag. If a DMA transfer is in progress for an endpoint when its corresponding bit is cleared, the transfer is completed before the DMA is disabled. When an error condition is detected during a DMA transfer, the corresponding bit is cleared by hardware. USBEpDMADis is a write only register.

Table 294. USB EP DMA Disable register (USBEpDMADis - address 0xFFE0 C28C) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0_DMA_DISABLE	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_DISABLE bit value must be 0).	0
1	EP1_DMA_DISABLE	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_DISABLE bit value must be 0).	0
31:2	EPxx_DMA_DISABLE		Endpoint xx ($2 \le xx \le 31$) DMA disable control bit.	0
		0	No effect.	
		1	Disable the DMA operation for endpoint EPxx.	

8.8.8 USB DMA Interrupt Status register (USBDMAIntSt - 0xFFE0 C290)

Each bit of this register reflects whether any of the 32 bits in the corresponding interrupt status register are set. USBDMAIntSt is a read only register.

Table 295. USB DMA Interrupt Status register (USBDMAIntSt - address 0xFFE0 C290) bit description

Bit	Symbol	Value	Description	Reset value
0	EOT		End of Transfer Interrupt bit.	0
		0	All bits in the USBEoTIntSt register are 0.	
		1	At least one bit in the USBEoTIntSt is set.	
1	NDDR		New DD Request Interrupt bit.	0
		0	All bits in the USBNDDRIntSt register are 0.	
		1	At least one bit in the USBNDDRIntSt is set.	
2	ERR		System Error Interrupt bit.	0
		0	All bits in the USBSysErrIntSt register are 0.	
		1	At least one bit in the USBSysErrIntSt is set.	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.8.9 USB DMA Interrupt Enable register (USBDMAIntEn - 0xFFE0 C294)

Writing a one to a bit in this register enables the corresponding bit in USBDMAIntSt to generate an interrupt on the USB_INT_REQ_DMA interrupt line when set. USBDMAIntEn is a read/write register.

Table 296. USB DMA Interrupt Enable register (USBDMAIntEn - address 0xFFE0 C294) bit description

Bit	Symbol	Value	Description	Reset value
0	EOT		End of Transfer Interrupt enable bit.	0
		0	The End of Transfer Interrupt is disabled.	
		1	The End of Transfer Interrupt is enabled.	
1	NDDR		New DD Request Interrupt enable bit.	0
		0	The New DD Request Interrupt is disabled.	
		1	The New DD Request Interrupt is enabled.	
2	ERR		System Error Interrupt enable bit.	0
		0	The System Error Interrupt is disabled.	
		1	The System Error Interrupt is enabled.	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.8.10 USB End of Transfer Interrupt Status register (USBEoTIntSt - 0xFFE0 C2A0)

When the DMA transfer completes for the current DMA descriptor, either normally (descriptor is retired) or because of an error, the bit corresponding to the endpoint is set in this register. The cause of the interrupt is recorded in the DD_status field of the descriptor. USBEoTIntSt is a read only register.

Table 297. USB End of Transfer Interrupt Status register (USBEoTIntSt - address 0xFFE0 C2A0s) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Endpoint xx (2 ≤ xx ≤ 31) End of Transfer Interrupt request.	0
		0	There is no End of Transfer interrupt request for endpoint xx.	
		1	There is an End of Transfer Interrupt request for endpoint xx.	

8.8.11 USB End of Transfer Interrupt Clear register (USBEoTIntClr - 0xFFE0 C2A4)

Writing one to a bit in this register clears the corresponding bit in the USBEoTIntSt register. Writing zero has no effect. USBEoTIntClr is a write only register.

Table 298. USB End of Transfer Interrupt Clear register (USBEoTIntClr - address 0xFFE0 C2A4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Clear endpoint xx (2 ≤ xx ≤ 31) End of Transfer Interrupt request.	0
		0	No effect.	
		1	Clear the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	

8.8.12 USB End of Transfer Interrupt Set register (USBEoTIntSet - 0xFFE0 C2A8)

Writing one to a bit in this register sets the corresponding bit in the USBEoTIntSt register. Writing zero has no effect. USBEoTIntSet is a write only register.

Table 299. USB End of Transfer Interrupt Set register (USBEoTIntSet - address 0xFFE0 C2A8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Set endpoint xx (2 ≤ xx ≤ 31) End of Transfer Interrupt request.	0
		0	No effect.	
		1	Set the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	

8.8.13 USB New DD Request Interrupt Status register (USBNDDRIntSt - 0xFFE0 C2AC)

A bit in this register is set when a transfer is requested from the USB device and no valid DD is detected for the corresponding endpoint. USBNDDRIntSt is a read only register.

Table 300. USB New DD Request Interrupt Status register (USBNDDRIntSt - address 0xFFE0 C2AC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request.	0
		0	There is no new DD interrupt request for endpoint xx.	
		1	There is a new DD interrupt request for endpoint xx.	

8.8.14 USB New DD Request Interrupt Clear register (USBNDDRIntClr - 0xFFE0 C2B0)

Writing one to a bit in this register clears the corresponding bit in the USBNDDRIntSt register. Writing zero has no effect. USBNDDRIntClr is a write only register.

Table 301. USB New DD Request Interrupt Clear register (USBNDDRIntClr - address 0xFFE0 C2B0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Clear endpoint xx ($2 \le xx \le 31$) new DD interrupt request.	0
		0	No effect.	
		1	Clear the EPxx new DD interrupt request in the USBNDDRIntSt register.	

8.8.15 USB New DD Request Interrupt Set register (USBNDDRIntSet - 0xFFE0 C2B4)

Writing one to a bit in this register sets the corresponding bit in the USBNDDRIntSt register. Writing zero has no effect. USBNDDRIntSet is a write only register

Table 302. USB New DD Request Interrupt Set register (USBNDDRIntSet - address 0xFFE0 C2B4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Set endpoint xx ($2 \le xx \le 31$) new DD interrupt request.	0
		0	No effect.	
		1	Set the EPxx new DD interrupt request in the USBNDDRIntSt register.	

8.8.16 USB System Error Interrupt Status register (USBSysErrIntSt - 0xFFE0 C2B8)

If a system error (AHB bus error) occurs when transferring the data or when fetching or updating the DD the corresponding bit is set in this register. USBSysErrIntSt is a read only register.

Table 303. USB System Error Interrupt Status register (USBSysErrIntSt - address 0xFFE0 C2B8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Endpoint xx ($2 \le xx \le 31$) System Error Interrupt request.	0
		0	There is no System Error Interrupt request for endpoint xx.	
		1	There is a System Error Interrupt request for endpoint xx.	

8.8.17 USB System Error Interrupt Clear register (USBSysErrIntClr - 0xFFE0 C2BC)

Writing one to a bit in this register clears the corresponding bit in the USBSysErrIntSt register. Writing zero has no effect. USBSysErrIntClr is a write only register.

Table 304. USB System Error Interrupt Clear register (USBSysErrIntClr - address 0xFFE0 C2BC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Clear endpoint xx (2 ≤ xx ≤ 31) System Error Interrupt request.	0
		0	No effect.	
		1	Clear the EPxx System Error Interrupt request in the USBSysErrIntSt register.	

8.8.18 USB System Error Interrupt Set register (USBSysErrIntSet - 0xFFE0 C2C0)

Writing one to a bit in this register sets the corresponding bit in the USBSysErrIntSt register. Writing zero has no effect. USBSysErrIntSet is a write only register.

Table 305. USB System Error Interrupt Set register (USBSysErrIntSet - address 0xFFE0 C2C0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Set endpoint xx (2 ≤ xx ≤ 31) System Error Interrupt request.	0
		0	No effect.	
		1	Set the EPxx System Error Interrupt request in the USBSysErrIntSt register.	

9. Interrupt handling

This section describes how an interrupt event on any of the endpoints is routed to the Vectored Interrupt Controller (VIC). For a diagram showing interrupt event handling, see Figure 13–53.

All non-isochronous OUT endpoints (control, bulk, and interrupt endpoints) generate an interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet has been succesfully transmitted or when a NAK signal is sent and interrupts on NAK are enabled by the SIE Set Mode command, see Section 13–10.3. For isochronous endpoints, a frame interrupt is generated every 1 ms.

The interrupt handling is different for Slave and DMA mode.

Slave mode

If an interrupt event occurs on an endpoint and the endpoint interrupt is enabled in the USBEpIntEn register, the corresponding status bit in the USBEpIntSt is set. For non-isochronous endpoints, all endpoint interrupt events are divided into two types by the corresponding USBEpIntPri[n] registers: fast endpoint interrupt events and slow endpoint interrupt events. All fast endpoint interrupt events are ORed and routed to bit EP_FAST in the USBDevIntSt register. All slow endpoint interrupt events are ORed and routed to the EP_SLOW bit in USBDevIntSt.

For isochronous endpoints, the FRAME bit in USBDevIntSt is set every 1 ms.

The USBDevIntSt register holds the status of all endpoint interrupt events as well as the status of various other interrupts (see Section 13–8.3.2). By default, all interrupts (if enabled in USBDevIntEn) are routed to the USB_INT_REQ_LP bit in the USBIntSt

register to request low priority interrupt handling. However, the USBDevIntPri register can route either the FRAME or the EP_FAST bit to the USB_INT_REQ_HP bit in the USBIntSt register.

Only one of the EP_FAST and FRAME interrupt events can be routed to the USB_INT_REQ_HP bit. If routing both bits to USB_INT_REQ_HP is attempted, both interrupt events are routed to USB_INT_REQ_LP.

Slow endpoint interrupt events are always routed directly to the USB_INT_REQ_LP bit for low priority interrupt handling by software.

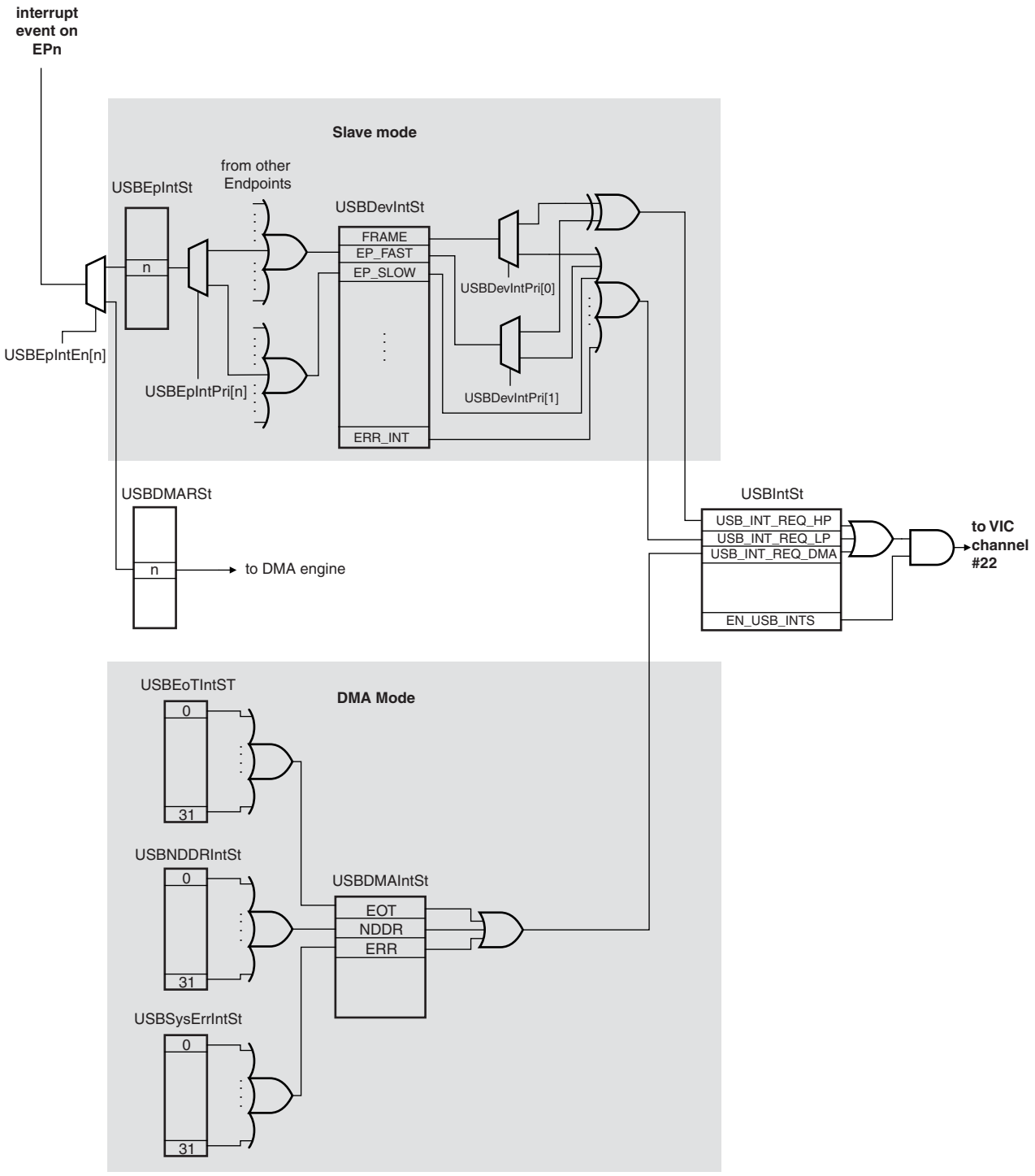
The final interrupt signal to the VIC is gated by the EN_USB_INTS bit in the USBIntSt register. The USB interrupts are routed to VIC channel #22 only if EN_USB_INTS is set.

DMA mode

If an interrupt event occurs on a non-control endpoint and the endpoint interrupt is not enabled in the USBEpIntEn register, the corresponding status bit in the USBDMARSt is set by hardware. This serves as a flag for the DMA engine to transfer data if DMA transfer is enabled for the corresponding endpoint in the USBEpDMASt register.

Three types of interrupts can occur for each endpoint for data transfers in DMA mode: End of transfer interrupt , new DD request interrupt, and system error interrupt. These interrupt events set a bit for each endpoint in the respective registers USBEoTIntSt, USBNDDRIntSt, and USBSysErrIntSt. The End of transfer interrupts from all endpoints are then Ored and routed to the EOT bit in USBDMAIntSt. Likewise, all New DD request interrupts and system error interrupt events are routed to the NDDR and ERR bits respectively in the USBDMAStInt register.

The EOT, NDDR, and ERR bits (if enabled in USBDMAIntEn) are ORed to set the USB_INT_REQ_DMA bit in the USBIntSt register. If the EN_USB_INTS bit is set in USBIntSt, the interrupt is routed to VIC channel #22.



For simplicity, USBDevIntEn and USBDMAIntEn are not shown.

Fig 53. Interrupt event handling

10. Serial interface engine command description

The functions and registers of the Serial Interface Engine (SIE) are accessed using commands, which consist of a command code followed by optional data bytes (read or write action). The USBCmdCode (Table 13–285) and USBCmdData (Table 13–286) registers are used for these accesses.

A complete access consists of two phases:

1. **Command phase:** the USBCmdCode register is written with the CMD_PHASE field set to the value 0x05 (Command), and the CMD_CODE field set to the desired command code. On completion of the command, the CCEMPTY bit of USBDevIntSt is set.
2. **Data phase (optional):** for writes, the USBCmdCode register is written with the CMD_PHASE field set to the value 0x01 (Write), and the CMD_WDATA field set with the desired write data. On completion of the write, the CCEMPTY bit of USBDevIntSt is set. For reads, USBCmdCode register is written with the CMD_PHASE field set to the value 0x02 (Read), and the CMD_CODE field set with command code the read corresponds to. On completion of the read, the CDFULL bit of USBDevInSt will be set, indicating the data is available for reading in the USBCmdData register. In the case of multi-byte registers, the least significant byte is accessed first.

An overview of the available commands is given in Table 13–306.

Here is an example of the Read Current Frame Number command (reading 2 bytes):

```
USBDevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL
USBCmdCode = 0x00F50500;       // CMD_CODE=0xF5, CMD_PHASE=0x05(Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY interrupt bit.
USBCmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
USBDevIntClr = 0x20;           // Clear CDFULL.
CurFrameNum = USBCmdData;      // Read Frame number LSB byte.
USBCmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
Temp = USBCmdData;             // Read Frame number MSB byte
USBDevIntClr = 0x20;           // Clear CDFULL interrupt bit.
CurFrameNum = CurFrameNum | (Temp << 8);
```

Here is an example of the Set Address command (writing 1 byte):

```
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x00D00500;       // CMD_CODE=0xD0, CMD_PHASE=0x05(Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x008A0100;       // CMD_WDATA=0x8A(DEV_EN=1, DEV_ADDR=0xA),
                               // CMD_PHASE=0x01(Write)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
```

Table 306. SIE command code table

Command name	Recipient	Code (Hex)	Data phase
Device commands			
Set Address	Device	D0	Write 1 byte
Configure Device	Device	D8	Write 1 byte
Set Mode	Device	F3	Write 1 byte
Read Current Frame Number	Device	F5	Read 1 or 2 bytes
Read Test Register	Device	FD	Read 2 bytes
Set Device Status	Device	FE	Write 1 byte
Get Device Status	Device	FE	Read 1 byte
Get Error Code	Device	FF	Read 1 byte
Read Error Status	Device	FB	Read 1 byte
Endpoint Commands			
Select Endpoint	Endpoint 0	00	Read 1 byte (optional)
	Endpoint 1	01	Read 1 byte (optional)
	Endpoint xx	xx	Read 1 byte (optional)
Select Endpoint/Clear Interrupt	Endpoint 0	40	Read 1 byte
	Endpoint 1	41	Read 1 byte
	Endpoint xx	xx + 40	Read 1 byte
Set Endpoint Status	Endpoint 0	40	Write 1 byte
	Endpoint 1	41	Write 1 byte
	Endpoint xx	xx + 40	Write 1 byte
Clear Buffer	Selected Endpoint	F2	Read 1 byte (optional)
Validate Buffer	Selected Endpoint	FA	None

10.1 Set Address (Command: 0xD0, Data: write 1 byte)

The Set Address command is used to set the USB assigned address and enable the (embedded) function. The address set in the device will take effect after the status stage of the control transaction. After a bus reset, DEV_ADDR is set to 0x00, and DEV_EN is set to 1. The device will respond to packets for function address 0x00, endpoint 0 (default endpoint).

Table 307. Device Set Address Register bit description

Bit	Symbol	Description	Reset value
6:0	DEV_ADDR	Device address set by the software. After a bus reset this field is set to 0x00.	0x00
7	DEV_EN	Device Enable. After a bus reset this bit is set to 1. 0: Device will not respond to any packets. 1: Device will respond to packets for function address DEV_ADDR.	0

10.2 Configure Device (Command: 0xD8, Data: write 1 byte)

A value of 1 written to the register indicates that the device is configured and all the enabled non-control endpoints will respond. Control endpoints are always enabled and respond even if the device is not configured, in the default state.

Table 308. Configure Device Register bit description

Bit	Symbol	Description	Reset value
0	CONF_DEVICE	Device is configured. All enabled non-control endpoints will respond. This bit is cleared by hardware when a bus reset occurs. When set, the UP_LED signal is driven LOW if the device is not in the suspended state (SUS=0).	
7:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.3 Set Mode (Command: 0xF3, Data: write 1 byte)

Table 309. Set Mode Register bit description

Bit	Symbol	Value	Description	Reset value
0	AP_CLK		Always PLL Clock.	0
		0	USB_NEED_CLK is functional; the 48 MHz clock can be stopped when the device enters suspend state.	
		1	USB_NEED_CLK is fixed to 1; the 48 MHz clock cannot be stopped when the device enters suspend state.	
1	INAK_CI		Interrupt on NAK for Control IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
2	INAK_CO		Interrupt on NAK for Control OUT endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
3	INAK_II		Interrupt on NAK for Interrupt IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
4	INAK_IO[1]		Interrupt on NAK for Interrupt OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
5	INAK_BI		Interrupt on NAK for Bulk IN endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
6	INAK_BO[2]		Interrupt on NAK for Bulk OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
7	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] This bit should be reset to 0 if the DMA is enabled for any of the Interrupt OUT endpoints.

[2] This bit should be reset to 0 if the DMA is enabled for any of the Bulk OUT endpoints.

10.4 Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes)

Returns the frame number of the last successfully received SOF. The frame number is eleven bits wide. The frame number returns least significant byte first. In case the user is only interested in the lower 8 bits of the frame number, only the first byte needs to be read.

- In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF.
- In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.

10.5 Read Test Register (Command: 0xFD, Data: read 2 bytes)

The test register is 16 bits wide. It returns the value of 0xA50F if the USB clocks (usbclk and AHB slave clock) are running.

10.6 Set Device Status (Command: 0xFE, Data: write 1 byte)

The Set Device Status command sets bits in the Device Status Register.

Table 310. Set Device Status Register bit description

Bit	Symbol	Value	Description	Reset value
0	CON		The Connect bit indicates the current connect status of the device. It controls the CONNECT output pin, used for SoftConnect. Reading the connect bit returns the current connect status. This bit is cleared by hardware when the V_{BUS} status input is LOW for more than 3 ms. The 3 ms delay filters out temporary dips in the V_{BUS} voltage.	0
		0	Writing a 0 will make the CONNECT pin go HIGH.	
		1	Writing a 1 will make the CONNECT pin go LOW..	
1	CON_CH		Connect Change.	0
		0	This bit is cleared when read.	
		1	This bit is set when the device's pull-up resistor is disconnected because V_{BUS} disappeared. The DEV_STAT interrupt is generated when this bit is 1.	
2	SUS		Suspend: The Suspend bit represents the current suspend state. When the device is suspended (SUS = 1) and the CPU writes a 0 into it, the device will generate a remote wakeup. This will only happen when the device is connected (CON = 1). When the device is not connected or not suspended, writing a 0 has no effect. Writing a 1 to this bit has no effect.	0
		0	This bit is reset to 0 on any activity.	
		1	This bit is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 ms.	

Table 310. Set Device Status Register bit description

Bit	Symbol	Value	Description	Reset value
3	SUS_CH		Suspend (SUS) bit change indicator. The SUS bit can toggle because: • The device goes into the suspended state. • The device is disconnected. • The device receives resume signalling on its upstream port. This bit is cleared when read.	0
		0	SUS bit not changed.	
		1	SUS bit changed. At the same time a DEV_STAT interrupt is generated.	
4	RST		Bus Reset bit. On a bus reset, the device will automatically go to the default state. In the default state: • Device is unconfigured. • Will respond to address 0. • Control endpoint will be in the Stalled state. • All endpoints are unrealized except control endpoints EP0 and EP1. • Data toggling is reset for all endpoints. • All buffers are cleared. • There is no change to the endpoint interrupt status. • DEV_STAT interrupt is generated. Note: Bus resets are ignored when the device is not connected (CON=0).	0
		0	This bit is cleared when read.	
		1	This bit is set when the device receives a bus reset. A DEV_STAT interrupt is generated.	
7:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.7 Get Device Status (Command: 0xFE, Data: read 1 byte)

The Get Device Status command returns the Device Status Register. Reading the device status returns 1 byte of data. The bit field definition is same as the Set Device Status Register as shown in Table 13–310.

Remark: To ensure correct operation, the DEV_STAT bit of USBDevIntSt must be cleared before executing the Get Device Status command.

10.8 Get Error Code (Command: 0xFF, Data: read 1 byte)

Different error conditions can arise inside the SIE. The Get Error Code command returns the last error code that occurred. The 4 least significant bits form the error code.

Table 311. Get Error Code Register bit description

Bit	Symbol	Value	Description	Reset value
3:0	EC		Error Code.	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.	
4	EA	-	The Error Active bit will be reset once this register is read.	
7:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.9 Read Error Status (Command: 0xFB, Data: read 1 byte)

This command reads the 8-bit Error register from the USB device. This register records which error events have recently occurred in the SIE. If any of these bits are set, the ERR_INT bit of USBDevIntSt is set. The error bits are cleared after reading this register.

Table 312. Read Error Status Register bit description

Bit	Symbol	Description	Reset value
0	PID_ERR	PID encoding error or Unknown PID or Token CRC.	0
1	UEPKT	Unexpected Packet - any packet sequence violation from the specification.	0
2	DCRC	Data CRC error.	0
3	TIMEOUT	Time out error.	0
4	EOP	End of packet error.	0
5	B_OVRN	Buffer Overrun.	0
6	BTSTF	Bit stuff error.	0
7	TGL_ERR	Wrong toggle bit in data PID, ignored data.	0

10.10 Select Endpoint (Command: 0x00 - 0x1F, Data: read 1 byte (optional))

The Select Endpoint command initializes an internal pointer to the start of the selected buffer in EP_RAM. Optionally, this command can be followed by a data read, which returns some additional information on the packet(s) in the endpoint buffer(s). The command code of the Select Endpoint command is equal to the physical endpoint number. In the case of a single buffered endpoint the B_2_FULL bit is not valid.

Table 313. Select Endpoint Register bit description

Bit	Symbol	Value	Description	Reset value
0	FE		Full/Empty. This bit indicates the full or empty status of the endpoint buffer(s). For IN endpoints, the FE bit gives the ANDed result of the B_1_FULL and B_2_FULL bits. For OUT endpoints, the FE bit gives ORed result of the B_1_FULL and B_2_FULL bits. For single buffered endpoints, this bit simply reflects the status of B_1_FULL.	0
		0	For an IN endpoint, at least one write endpoint buffer is empty.	
		1	For an OUT endpoint, at least one endpoint read buffer is full.	
1	ST		Stalled endpoint indicator.	0
		0	The selected endpoint is not stalled.	
		1	The selected endpoint is stalled.	
2	STP		SETUP bit: the value of this bit is updated after each successfully received packet (i.e. an ACKed package on that particular physical endpoint).	0
		0	The STP bit is cleared by doing a Select Endpoint/Clear Interrupt on this endpoint.	
		1	The last received packet for the selected endpoint was a SETUP packet.	
3	PO		Packet over-written bit.	0
		0	The PO bit is cleared by the ‘Select Endpoint/Clear Interrupt’ command.	
		1	The previously received packet was over-written by a SETUP packet.	
4	EPN		EP NAKed bit indicates sending of a NAK. If the host sends an OUT packet to a filled OUT buffer, the device returns NAK. If the host sends an IN token packet to an empty IN buffer, the device returns NAK.	0
		0	The EPN bit is reset after the device has sent an ACK after an OUT packet or when the device has seen an ACK after sending an IN packet.	
		1	The EPN bit is set when a NAK is sent and the interrupt on NAK feature is enabled.	
5	B_1_FULL		The buffer 1 status.	0
		0	Buffer 1 is empty.	
		1	Buffer 1 is full.	

Table 313. Select Endpoint Register bit description

Bit	Symbol	Value	Description	Reset value
6	B_2_FULL		The buffer 2 status.	0
		0	Buffer 2 is empty.	
		1	Buffer 2 is full.	
7	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.11 Select Endpoint/Clear Interrupt (Command: 0x40 - 0x5F, Data: read 1 byte)

Commands 0x40 to 0x5F are identical to their Select Endpoint equivalents, with the following differences:

- They clear the bit corresponding to the endpoint in the USBEpIntSt register.
- In case of a control OUT endpoint, they clear the STP and PO bits in the corresponding Select Endpoint Register.
- Reading one byte is obligatory.

Remark: This command may be invoked by using the USBCmdCode and USBCmdData registers, or by setting the corresponding bit in USBEpIntClr. For ease of use, using the USBEpIntClr register is recommended.

10.12 Set Endpoint Status (Command: 0x40 - 0x55, Data: write 1 byte (optional))

The Set Endpoint Status command sets status bits 7:5 and 0 of the endpoint. The Command Code of Set Endpoint Status is equal to the sum of 0x40 and the physical endpoint number in hex. Not all bits can be set for all types of endpoints.

Table 314. Set Endpoint Status Register bit description

Bit	Symbol	Value	Description	Reset value
0	ST		Stalled endpoint bit. A Stalled control endpoint is automatically unstalled when it receives a SETUP token, regardless of the content of the packet. If the endpoint should stay in its stalled state, the CPU can stall it again by setting this bit. When a stalled endpoint is unstalled - either by the Set Endpoint Status command or by receiving a SETUP token - it is also re-initialized. This flushes the buffer: in case of an OUT buffer it waits for a DATA 0 PID; in case of an IN buffer it writes a DATA 0 PID. There is no change of the interrupt status of the endpoint. When already unstalled, writing a zero to this bit initializes the endpoint. When an endpoint is stalled by the Set Endpoint Status command, it is also re-initialized.	0
		0	The endpoint is unstalled.	
		1	The endpoint is stalled.	
4:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 314. Set Endpoint Status Register bit description

Bit	Symbol	Value	Description	Reset value
5	DA		Disabled endpoint bit.	0
		0	The endpoint is enabled.	
		1	The endpoint is disabled.	
6	RF_MO		Rate Feedback Mode.	0
		0	Interrupt endpoint is in the Toggle mode.	
		1	Interrupt endpoint is in the Rate Feedback mode. This means that transfer takes place without data toggle bit.	
7	CND_ST		Conditional Stall bit.	0
		0	Unstalls both control endpoints.	
		1	Stall both control endpoints, unless the STP bit is set in the Select Endpoint register. It is defined only for control OUT endpoints.	

10.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional))

When an OUT packet sent by the host has been received successfully, an internal hardware FIFO status Buffer_Full flag is set. All subsequent packets will be refused by returning a NAK. When the device software has read the data, it should free the buffer by issuing the Clear Buffer command. This clears the internal Buffer_Full flag. When the buffer is cleared, new packets will be accepted.

When bit 0 of the optional data byte is 1, the previously received packet was over-written by a SETUP packet. The Packet over-written bit is used only in control transfers. According to the USB specification, a SETUP packet should be accepted irrespective of the buffer status. The software should always check the status of the PO bit after reading the SETUP data. If it is set then it should discard the previously read data, clear the PO bit by issuing a Select Endpoint/Clear Interrupt command, read the new SETUP data and again check the status of the PO bit.

See Section 13–12 “Slave mode operation” for a description of when this command is used.

Table 315. Clear Buffer Register bit description

Bit	Symbol	Value	Description	Reset value
0	PO		Packet over-written bit. This bit is only applicable to the control endpoint EP0.	0
		0	The previously received packet is intact.	
		1	The previously received packet was over-written by a later SETUP packet.	
7:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.14 Validate Buffer (Command: 0xFA, Data: none)

When the CPU has written data into an IN buffer, software should issue a Validate Buffer command. This tells hardware that the buffer is ready for sending on the USB bus. Hardware will send the contents of the buffer when the next IN token packet is received.

Internally, there is a hardware FIFO status flag called Buffer_Full. This flag is set by the Validate Buffer command and cleared when the data has been sent on the USB bus and the buffer is empty.

A control IN buffer cannot be validated when its corresponding OUT buffer has the Packet Over-written (PO) bit (see the Clear Buffer Register) set or contains a pending SETUP packet. For the control endpoint the validated buffer will be invalidated when a SETUP packet is received.

See Section 13–12 “Slave mode operation” for a description of when this command is used.

11. USB device controller initialization

The LPC2400 USB device controller initialization includes the following steps:

1. Enable the device controller by setting the PCUSB bit of PCONP.
2. Configure and enable the PLL and Clock Dividers to provide 48 MHz for usbclk, and the desired frequency for cclk. For correct operation of synchronization logic in the device controller, the minimum cclk frequency is 18 MHz. For the procedure for determining the PLL setting and configuration, see Section 4–5.12 “Procedure for determining PLL settings”.
3. Enable the device controller clocks by setting DEV_CLK_EN and AHB_CLK_EN bits in the USBClkCtrl register. Poll the respective clock bits in the USBClkSt register until they are set.
4. Select the desired USB port pins using the USBPortSel register. The PORTSEL_CLK_EN bit must be set in USBClkCtrl before accessing USBPortSel and should be cleared after accessing USBPortSel.
5. Enable the USB pin functions by writing to the corresponding PINSEL register.
6. Disable the pull-up resistor on the V_{BUS} pin using the corresponding PINMODE register.
7. Set USBEpIn and USBMaxPSize registers for EP0 and EP1, and wait until the EP_RLZED bit in USBDevIntSt is set so that EP0 and EP1 are realized.
8. Enable endpoint interrupts (Slave mode):
   - Clear all endpoint interrupts using USBEpIntClr.
   - Clear any device interrupts using USBDevIntClr.
   - Enable Slave mode for the desired endpoints by setting the corresponding bits in USBEpIntEn.
   - Set the priority of each enabled interrupt using USBEpIntPri.
   - Configure the desired interrupt mode using the SIE Set Mode command.
   - Enable device interrupts using USBDevIntEn (normally DEV_STAT, EP_SLOW, and possibly EP_FAST).
9. Configure the DMA (DMA mode):
   - Disable DMA operation for all endpoints using USBEpDMADis.
   - Clear any pending DMA requests using USBDMARClr.

- Clear all DMA interrupts using USBEoTIntClr, USBNDDRIntClr, and USBSysErrIntClr.
- Prepare the UDCA in system memory.
- Write the desired address for the UDCA to USBUDCAH (for example 0x7FD0 0000).
- Enable the desired endpoints for DMA operation using USBEpDMAEn.
- Set EOT, DDR, and ERR bits in USBDMAIntEn.

10. Install USB interrupt handler in the VIC by writing its address to the corresponding VICVectAddr register and enabling the USB interrupt in the VICIntEnable register.
11. Set default USB address to 0x0 and DEV_EN to 1 using the SIE Set Address command. A bus reset will also cause this to happen.
12. Set CON bit to 1 to make CONNECT active using the SIE Set Device Status command.

The configuration of the endpoints varies depending on the software application. By default, all the endpoints are disabled except control endpoints EP0 and EP1. Additional endpoints are enabled and configured by software after a SET_CONFIGURATION or SET_INTERFACE device request is received from the host.

12. Slave mode operation

In Slave mode, the CPU transfers data between RAM and the endpoint buffer using the Register Interface.

12.1 Interrupt generation

In slave mode, data packet transfer between RAM and an endpoint buffer can be initiated in response to an endpoint interrupt. Endpoint interrupts are enabled using the USBEpIntEn register, and are observable in the USBEpIntSt register.

All non-isochronous OUT endpoints generate an endpoint interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet is successfully transmitted, or when a NAK handshake is sent on the bus and the interrupt on NAK feature is enabled.

For Isochronous endpoints, transfer of data is done when the FRAME interrupt (in USBDevIntSt) occurs.

12.2 Data transfer for OUT endpoints

When the software wants to read the data from an endpoint buffer it should set the RD_EN bit and program LOG_ENDPOINT with the desired endpoint number in the USBCtrl register. The control logic will fetch the packet length to the USBRxPLen register, and set the PKT_RDY bit (Table 13–281).

Software can now start reading the data from the USBRxData register (Table 13–280). When the end of packet is reached, the RD_EN bit is cleared, and the RxENDPKT bit is set in the USBDevSt register. Software now issues a Clear Buffer (refer to Table 13–315) command. The endpoint is now ready to accept the next packet. For OUT isochronous

endpoints, the next packet will be received irrespective of whether the buffer has been cleared. Any data not read from the buffer before the end of the frame is lost. See Section 13–14 “Double buffered endpoint operation” for more details.

If the software clears RD_EN before the entire packet is read, reading is terminated, and the data remains in the endpoint’s buffer. When RD_EN is set again for this endpoint, the data will be read from the beginning.

12.3 Data transfer for IN endpoints

When writing data to an endpoint buffer, WR_EN (Section 13–8.6.5 “USB Control register (USBCtrl - 0xFFE0_C228)”) is set and software writes to the number of bytes it is going to send in the packet to the USBTxPLen register (Section 13–8.6.4). It can then write data continuously in the USBTxData register.

When the the number of bytes programmed in USBTxPLen have been written to USBTxData, the WR_EN bit is cleared, and the TxENDPKT bit is set in the USBDevIntSt register. Software issues a Validate Buffer (Section 13–10.14 “Validate Buffer (Command: 0xFA, Data: none)”) command. The endpoint is now ready to send the packet. For IN isochronous endpoints, the data in the buffer will be sent only if the buffer is validated before the next FRAME interrupt occurs; otherwise, an empty packet will be sent in the next frame. If the software clears WR_EN before the entire packet is written, writing will start again from the beginning the next time WR_EN is set for this endpoint.

Both RD_EN and WR_EN can be high at the same time for the same logical endpoint. Interleaved read and write operation is possible.

13. DMA operation

In DMA mode, the DMA transfers data between RAM and the endpoint buffer.

The following sections discuss DMA mode operation. Background information is given in sections Section 13–13.2 “USB device communication area” and Section 13–13.3 “Triggering the DMA engine”. The fields of the DMA Descriptor are described in section Section 13–13.4 “The DMA descriptor”. The last three sections describe DMA operation: Section 13–13.5 “Non-isochronous endpoint operation”, Section 13–13.6 “Isochronous endpoint operation”, and Section 13–13.7 “Auto Length Transfer Extraction (ATLE) mode operation”.

13.1 Transfer terminology

Within this section three types of transfers are mentioned:

1. USB transfers – transfer of data over the USB bus. The USB 2.0 specification refers to these simply as transfers. Within this section they are referred to as USB transfers to distinguish them from DMA transfers. A USB transfer is composed of transactions. Each transaction is composed of packets.
2. DMA transfers – the transfer of data between an endpoint buffer and system memory (RAM).
3. Packet transfers – in this section, a packet transfer refers to the transfer of a packet of data between an endpoint buffer and system memory (RAM). A DMA transfer is composed of one or more packet transfers.

13.2 USB device communication area

The CPU and DMA controller communicate through a common area of memory, called the USB Device Communication Area, or UDCA. The UDCA is a 32-word array of DMA Descriptor Pointers (DDPs), each of which corresponds to a physical endpoint. Each DDP points to the start address of a DMA Descriptor, if one is defined for the endpoint. DDPs for unrealized endpoints and endpoints disabled for DMA operation are ignored and can be set to a NULL (0x0) value.

The start address of the UDCA is stored in the USBUDCAH register. The UDCA can reside at any 128-byte boundary of RAM that is accessible to both the CPU and DMA controller.

Figure 36 illustrates the UDCA and its relationship to the UDCA Head (USBUDCAH) register and DMA Descriptors.

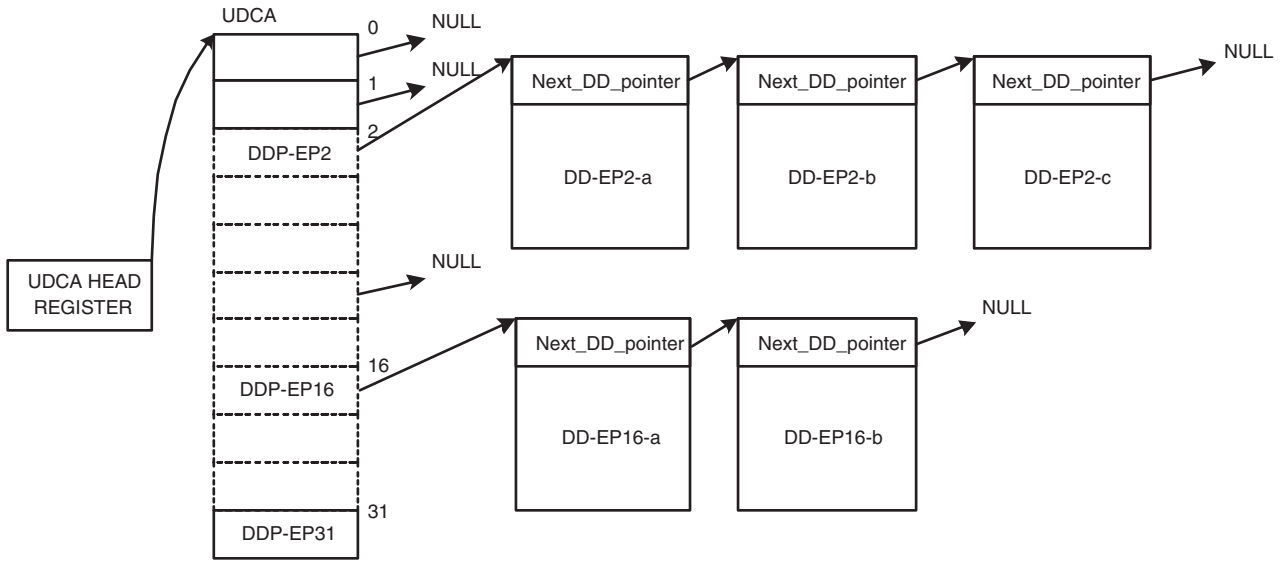


Fig 54. UDCA Head register and DMA Descriptors

13.3 Triggering the DMA engine

An endpoint raises a DMA request when Slave mode is disabled by setting the corresponding bit in the USBEpIntEn register to 0 (Section 13–8.4.2) and an endpoint interrupt occurs (see Section 13–8.8.1 “USB DMA Request Status register (USBDMARSt - 0xFFE0_C250)”).

A DMA transfer for an endpoint starts when the endpoint is enabled for DMA operation in USBEpDMASt, the corresponding bit in USBDMARSt is set, and a valid DD is found for the endpoint.

All endpoints share a single DMA channel to minimize hardware overhead. If more than one DMA request is active in USBDMARSt, the endpoint with the lowest physical endpoint number is processed first.

In DMA mode, the bits corresponding to Interrupt on NAK for Bulk OUT and Interrupt OUT endpoints (INAK_BO and INAK_IO) should be set to 0 using the SIE Set Mode command (Section 13–10.3).

13.4 The DMA descriptor

DMA transfers are described by a data structure called the DMA Descriptor (DD).

DDs are placed in the USB RAM. These descriptors can be located anywhere in the USB RAM at word-aligned addresses. USB RAM is part of the system memory that is used for the USB purposes. It is located at address 0x7FD0 0000 and is 8 kB in size.

DDs for non-isochronous endpoints are four words long. DDs for isochronous endpoints are five words long.

The parameters associated with a DMA transfer are:

- The start address of the DMA buffer
- The length of the DMA buffer
- The start address of the next DMA descriptor
- Control information
- Count information (number of bytes transferred)
- Status information

Table 13–316 lists the DMA descriptor fields.

Table 316. DMA descriptor

Word position	Access (H/W)	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[1]	R/W	31:16	DMA_buffer_length This value is specified in bytes for non-isochronous endpoints and in number of packets for isochronous endpoints.
2	R/W	R/W	31:0	DMA_buffer_start_addr

Table 316. DMA descriptor

Word position	Access (H/W)	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 0000): 0000 - NotServiced 0001 - BeingServiced 0010 - NormalCompletion 0011 - DataUnderrun (short packet) 1000 - DataOverrun 1001 - SystemError
	W	R/I	5	Packet_valid (To be initialized to 0)
	W	R/I	6	LS_byte_extracted (ATLE mode) (To be initialized to 0)
	W	R/I	7	MS_byte_extracted (ATLE mode) (To be initialized to 0)
	R	W	13:8	Message_length_position (ATLE mode)
	-	-	15:14	Reserved
	R/W	R/I	31:16	Present_DMA_count (To be initialized to 0)
4	R/W	R/W	31:0	Isochronous_packetsize_memory_address

[1] Write only in ATLE mode

Legend: R - Read; W - Write; I - Initialize

13.4.1 Next_DD_pointer

Pointer to the memory location from where the next DMA descriptor will be fetched.

13.4.2 DMA_mode

Specifies the DMA mode of operation. Two modes have been defined: Normal and Automatic Transfer Length Extraction (ATLE) mode. In normal mode, software initializes the DMA_buffer_length for OUT endpoints. In ATLE mode, the DMA_buffer_length is extracted from the incoming data. See Section 13–13.7 “Auto Length Transfer Extraction (ATLE) mode operation” on page 330 for more details.

13.4.3 Next_DD_valid

This bit indicates whether the software has prepared the next DMA descriptor. If set, the DMA engine fetches the new descriptor when it is finished with the current one.

13.4.4 Isochronous_endpoint

When set, this bit indicates that the descriptor belongs to an isochronous endpoint. Hence 5 words have to be read when fetching it.

13.4.5 Max_packet_size

The maximum packet size of the endpoint. This parameter is used while transferring the data for IN endpoints from the memory. It is used for OUT endpoints to detect the short packet. This is applicable to non-isochronous endpoints only. This field should be set to the same MPS value that is assigned for the endpoint using the USBMaxPSize register.

13.4.6 DMA_buffer_length

This indicates the depth of the DMA buffer allocated for transferring the data. The DMA engine will stop using this descriptor when this limit is reached and will look for the next descriptor.

In Normal mode operation, software sets this value for both IN and OUT endpoints. In ATLE mode operation, software sets this value for IN endpoints only. For OUT endpoints, hardware sets this value using the extracted length of the data stream.

For isochronous endpoints, DMA_buffer_length is specified in number of packets, for non-isochronous endpoints in bytes.

13.4.7 DMA_buffer_start_addr

The address where the data is read from or written to. This field is updated each time the DMA engine finishes transferring a packet.

13.4.8 DD_retired

This bit is set by hardware when the DMA engine finishes the current descriptor. This happens when the end of the buffer is reached, a short packet is transferred (non-isochronous endpoints), or an error condition is detected.

13.4.9 DD_status

The status of the DMA transfer is encoded in this field. The following codes are defined:

- **NotServiced** - No packet has been transferred yet.
- **BeingServiced** - At least one packet is transferred.
- **NormalCompletion** - The DD is retired because the end of the buffer is reached and there were no errors. The DD_retired bit is also set.
- **DataUnderrun** - Before reaching the end of the DMA buffer, the USB transfer is terminated because a short packet is received. The DD_retired bit is also set.
- **DataOverrun** - The end of the DMA buffer is reached in the middle of a packet transfer. This is an error situation. The DD_retired bit is set. The present DMA count field is equal to the value of DMA_buffer_length. The packet must be re-transmitted from the endpoint buffer in another DMA transfer. The corresponding EPxx_DMA_ENABLE bit in USBEpDMASt is cleared.
- **SystemError** - The DMA transfer being serviced is terminated because of an error on the AHB bus. The DD_retired bit is not set in this case. The corresponding EPxx_DMA_ENABLE in USBEpDMASt is cleared. Since a system error can happen while updating the DD, the DD fields in RAM may be unreliable.

13.4.10 Packet_valid

This bit is used for isochronous endpoints. It indicates whether the last packet transferred to the memory is received with errors or not. This bit is set if the packet is valid, i.e., it was received without errors. See Section 13–13.6 “Isochronous endpoint operation” on page 328 for isochronous endpoint operation.

This bit is unnecessary for non-isochronous endpoints because a DMA request is generated only for packets without errors, and thus Packet_valid will always be set when the request is generated.

13.4.11 LS_byte_extracted

Used in ATLE mode. When set, this bit indicates that the Least Significant Byte (LSB) of the transfer length has been extracted. The extracted size is reflected in the DMA_buffer_length field, bits 23:16.

13.4.12 MS_byte_extracted

Used in ATLE mode. When set, this bit indicates that the Most Significant Byte (MSB) of the transfer size has been extracted. The size extracted is reflected in the DMA_buffer_length field, bits 31:24. Extraction stops when LS_Byte_extracted and MS_byte_extracted bits are set.

13.4.13 Present_DMA_count

The number of bytes transferred by the DMA engine. The DMA engine updates this field after completing each packet transfer.

For isochronous endpoints, Present_DMA_count is the number of packets transferred; for non-isochronous endpoints, Present_DMA_count is the number of bytes.

13.4.14 Message_length_position

Used in ATLE mode. This field gives the offset of the message length position embedded in the incoming data packets. This is applicable only for OUT endpoints. Offset 0 indicates that the message length starts from the first byte of the first packet.

13.4.15 Isochronous_packetsize_memory_address

The memory buffer address where the packet size information along with the frame number has to be transferred or fetched. See Figure 13–55. This is applicable to isochronous endpoints only.

13.5 Non-isochronous endpoint operation

13.5.1 Setting up DMA transfers

Software prepares the DMA Descriptors (DDs) for those physical endpoints to be enabled for DMA transfer. These DDs are present in the USB RAM. The start address of the first DD is programmed into the DMA Description pointer (DDP) location for the corresponding endpoint in the UDCA. Software then sets the EPxx_DMA_ENABLE bit for this endpoint in the USBEpDMAEn register (Section 13–8.8.6).The DMA_mode bit field in the descriptor is set to ‘00’ for normal mode operation. All other DD fields are initialized as specified in Table 13–316.

DMA operation is not supported for physical endpoints 0 and 1 (default control endpoints).

13.5.2 Finding DMA Descriptor

When there is a trigger for a DMA transfer for an endpoint, the DMA engine will first determine whether a new descriptor has to the fetched or not. A new descriptor does not have to be fetched if the last packet transferred was for the same endpoint and the DD is not yet in the retired state. An internal flag called DMA_PROCEED is used to identify this condition (see Section 13–13.5.4 “Optimizing descriptor fetch” on page 327).

If a new descriptor has to be read, the DMA engine will calculate the location of the DDP for this endpoint and will fetch the start address of the DD from this location. A DD start address at location zero is considered invalid. In this case the NDDR interrupt is raised. All other word-aligned addresses are considered valid.

When the DD is fetched, the DD status word (word 3) is read first and the status of the DD_retired bit is checked. If not set, DDP points to a valid DD. If DD_retired is set, the DMA engine will read the control word (word 1) of the DD.

If Next_DD_valid bit is set, the DMA engine will fetch the Next_DD_pointer field (word 0) of the DD and load it to the DDP. The new DDP is written to the UDCA area.

The full DD (4 words) will then be fetched from the address in the DDP. The DD will give the details of the DMA transfer to be done. The DMA engine will load its hardware resources with the information fetched from the DD (start address, DMA count etc.).

If Next_DD_valid is not set and DD_retired bit is set, the DMA engine raises the NDDR interrupt for this endpoint and clears the corresponding EPxx_DMA_ENABLE bit.

13.5.3 Transferring the data

For OUT endpoints, the current packet is read from the EP_RAM by the DMA Engine and transferred to the USB RAM memory locations starting from DMA_buffer_start_addr. For IN endpoints, the data is fetched from the USB RAM at DMA_buffer_start_addr and written to the EP_RAM. The DMA_buffer_start_addr and Present_DMA_count fields are updated after each packet is transferred.

13.5.4 Optimizing descriptor fetch

A DMA transfer normally involves multiple packet transfers. Hardware will not re-fetch a new DD from memory unless the endpoint changes. To indicate an ongoing multi-packet transfer, hardware sets an an internal flag called DMA_PROCEED.

The DMA_PROCEED flag is cleared after the required number of bytes specified in the DMA_buffer_length field is transferred. It is also cleared when the software writes into the USBEpDMADis register. The ability to clear the DMA_PROCEED flag allows software to to force the DD to be re-fetched for the next packet transfer. Writing all zeros into the USBEpDMADis register clears the DMA_PROCEED flag without disabling DMA operation for any endpoint.

13.5.5 Ending the packet transfer

On completing a packet transfer, the DMA engine writes back the DD with updated status information to the same memory location from where it was read. The DMA_buffer_start_addr, Present_DMA_count, and the DD_status fields in the DD are updated.

A DD can have the following types of completion:

Normal completion - If the current packet is fully transferred and the Present_DMA_count field equals the DMA_buffer_length, the DD has completed normally. The DD will be written back to memory with DD_retired set and DD_status set to NormalCompletion. The EOT interrupt is raised for this endpoint.

USB transfer end completion - If the current packet is fully transferred and its size is less than the Max_packet_size field, and the end of the DMA buffer is still not reached, the USB transfer end completion occurs. The DD will be written back to the memory with DD_retired set and DD_Status set to the DataUnderrun completion code. The EOT interrupt is raised for this endpoint.

Error completion - If the current packet is partially transferred i.e. the end of the DMA buffer is reached in the middle of the packet transfer, an error situation occurs. The DD is written back with DD_retired set and DD_status set to the DataOverrun status code. The EOT interrupt is raised for this endpoint and the corresponding bit in USBEpDMASt register is cleared. The packet will be re-sent from the endpoint buffer to memory when the corresponding EPxx_DMA_ENABLE bit is set again using the USBEpDMAEn register.

13.5.6 No_Packet DD

For an IN transfer, if the system does not have any data to send for a while, it can respond to an NDDR interrupt by programming a No_Packet DD. This is done by setting both the Max_packet_size and DMA_buffer_length fields in the DD to 0. On processing a No_Packet DD, the DMA engine clears the DMA request bit in USBDMARSt corresponding to the endpoint without transferring a packet. The DD is retired with a status code of NormalCompletion. This can be repeated as often as necessary. The device will respond to IN token packets on the USB bus with a NAK until a DD with a data packet is programmed and the DMA transfers the packet into the endpoint buffer.

13.6 Isochronous endpoint operation

For isochronous endpoints, the packet size can vary for each packet. There is one packet per isochronous endpoint for each frame.

13.6.1 Setting up DMA transfers

Software sets the isochronous endpoint bit to 1 in the DD, and programs the initial value of the Isochronous_packetsize_memory_address field. All other fields are initialized the same as for non-isochronous endpoints.

For isochronous endpoints, the DMA_buffer_length and Present_DMA_count fields are in frames rather than bytes.

13.6.2 Finding the DMA Descriptor

Finding the descriptors is done in the same way as that for a non-isochronous endpoint.

A DMA request will be placed for DMA-enabled isochronous endpoints on every FRAME interrupt. On processing the request, the DMA engine will fetch the descriptor and if Isochronous_endpoint is set, will fetch the Isochronous_packetsize_memory_address from the fifth word of the DD.

13.6.3 Transferring the Data

The data is transferred to or from the memory location DMA_buffer_start_addr. After the end of the packet transfer the Present_DMA_count value is incremented by 1.

The isochronous packet size is stored in memory as shown in figure 32. Each word in the packet size memory shown is divided into fields: Frame_number (bits 31 to 17), Packet_valid (bit 16), and Packet_length (bits 15 to 0). The space allocated for the packet size memory for a given DD should be DMA_buffer_length words in size – one word for each packet to transfer.

OUT endpoints

At the completion of each frame, the packet size is written to the address location in Isochronous_packet_size_memory_address, and Isochronous_packet_size_memory_address is incremented by 4.

IN endpoints

Only the Packet_length field of the isochronous packet size word is used. For each frame, an isochronous data packet of size specified by this field is transferred from the USB device to the host, and Isochronous_packet_size_memory_address is incremented by 4 at the end of the packet transfer. If Packet_length is zero, an empty packet will be sent by the USB device.

13.6.4 DMA descriptor completion

DDs for isochronous endpoints can only end with a status code of NormalCompletion since there is no short packet on Isochronous endpoints, and the USB transfer continues indefinitely until a SystemError occurs. There is no DataOverrun detection for isochronous endpoints.

13.6.5 Isochronous OUT Endpoint Operation Example

Assume that an isochronous endpoint is programmed for the transfer of 10 frames and that the transfer begins when the frame number is 21. After transferring four frames with packet sizes of 10,15, 8 and 20 bytes without errors, the descriptor and memory map appear as shown in Figure 13–55.

The_total_number_of_bytes_transferred = 0x0A + 0x0F + 0x08 + 0x14 = 0x35.

The Packet_valid bit (bit 16) of all the words in the packet length memory is set to 1.

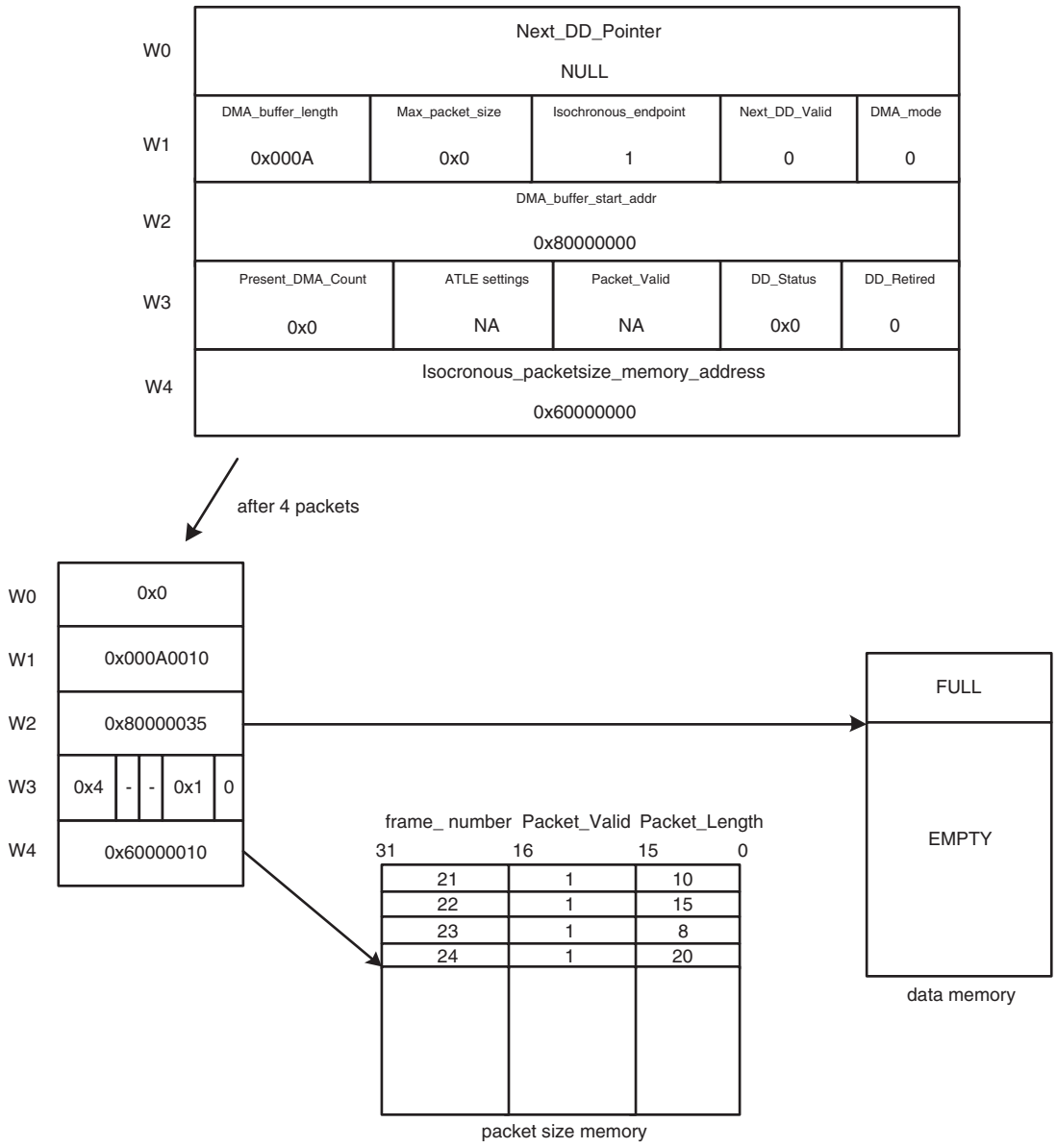


Fig 55. Isochronous OUT endpoint operation example

13.7 Auto Length Transfer Extraction (ATLE) mode operation

Some host drivers such as NDIS (Network Driver Interface Specification) host drivers are capable of concatenating small USB transfers (delta transfers) to form a single large USB transfer. For OUT USB transfers, the device hardware has to break up this concatenated transfer back into the original delta transfers and transfer them to separate DMA buffers. This is achieved by setting the DMA mode to Auto Transfer Length Extraction (ATLE) mode in the DMA descriptor. ATLE mode is supported for Bulk endpoints only.

OUT transfers in ATLE mode

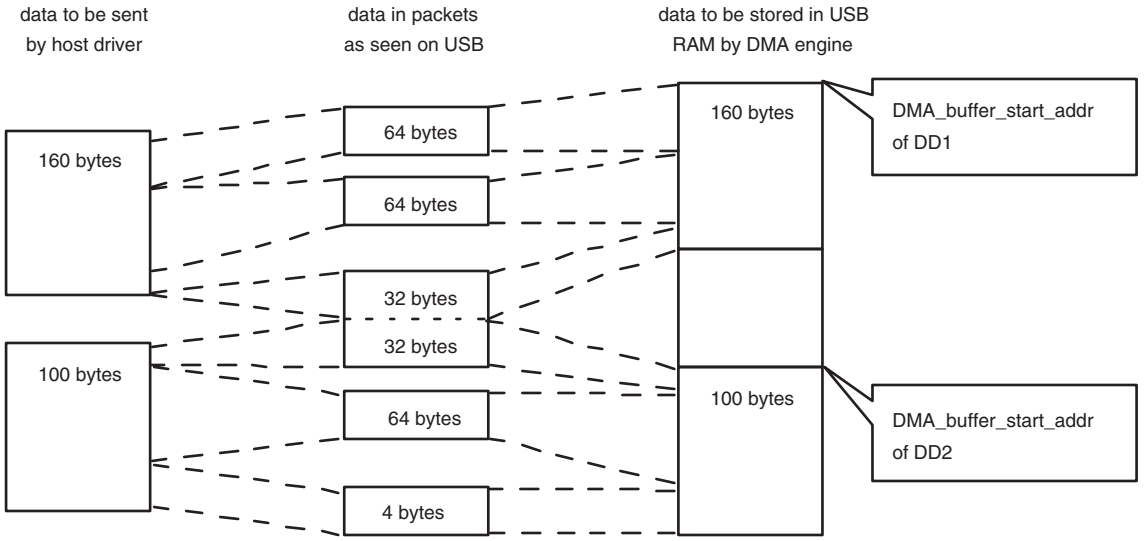


Fig 56. Data transfer in ATLE mode

Figure 13–56 shows a typical OUT USB transfer in ATLE mode, where the host concatenates two USB transfers of 160 bytes and 100 bytes, respectively. Given a MaxPacketSize of 64, the device hardware interprets this USB transfer as four packets of 64 bytes and a short packet of 4 bytes. The third and fourth packets are concatenated. Note that in Normal mode, the USB transfer would be interpreted as packets of 64, 64, 32, and 64 and 36 bytes.

It is now the responsibility of the DMA engine to separate these two USB transfers and put them in the memory locations in the DMA_buffer_start_addr field of DMA Descriptor 1 (DD1) and DMA Descriptor 2 (DD2).

Hardware reads the two-byte-wide DMA_buffer_length at the offset (from the start of the USB transfer) specified by Message_length_position from the incoming data packets and writes it in the DMA_buffer_length field of the DD. To ensure that both bytes of the DMA_buffer_length are extracted in the event they are split between two packets, the flags LS_byte_extracted and MS_byte_extracted are set by hardware after the respective byte is extracted. After the extraction of the MS byte, the DMA transfer continues as in the normal mode.

The flags LS_byte_extracted and MS_byte_extracted are set to 0 by software when preparing a new DD. Therefore, once a DD is retired, the transfer length is extracted again for the next DD.

If DD1 is retired during the transfer of a concatenated packet (such as the third packet in Figure 13–56), and DD2 is not programmed (Next_DD_valid field of DD1 is 0), then DD1 is retired with DD_status set to the DataOverrun status code. This is treated as an error condition and the corresponding EPxx_DMA_ENABLE bit of USBEpDMASt is cleared by hardware.

In ATLE mode, the last buffer length to be transferred always ends with a short or empty packet indicating the end of the USB transfer. If the concatenated transfer lengths are such that the USB transfer ends on a MaxPacketSize packet boundary, the (NDIS) host will send an empty packet to mark the end of the USB transfer.

IN transfers in ATLE mode

For IN USB transfers from the device to the host, DMA_buffer_length is set by the device software as in normal mode.

In ATLE mode, the device concatenates data from multiple DDs to form a single USB transfer. If a DD is retired in the middle of a packet (packet size is less than MaxPacketSize), the next DD referenced by Next_DD_pointer is fetched, and the remaining bytes to form a packet of MaxPacketSize are transferred from the next DD's buffer.

If the next DD is not programmed (i.e. Next_DD_valid field in DD is 0), and the DMA buffer length for the current DD has completed before the MaxPacketSize packet boundary, then the available bytes from current DD are sent as a short packet on USB, which marks the end of the USB transfer for the host.

If the last buffer length completes on a MaxPacketSize packet boundary, the device software must program the next DD with DMA_buffer_length field 0, so that an empty packet is sent by the device to mark the end of the USB transfer for the host.

13.7.1 Setting up the DMA transfer

For OUT endpoints, the host hardware needs to set the field Message_length_position in the DD. This indicates the start location of the message length in the incoming data packets. Also the device software has to set the DMA_buffer_length field to 0 for OUT endpoints because this field is updated by the device hardware after the extraction of the buffer length.

For IN endpoints, descriptors are set in the same way as in normal mode operation.

Since a single packet can be split between two DDs, software should always keep two DDs ready, except for the last DMA transfer which ends with a short or empty packet.

13.7.2 Finding the DMA Descriptor

DMA descriptors are found in the same way as the normal mode operation.

13.7.3 Transferring the Data

OUT endpoints

If the LS_byte_extracted or MS_byte_extracted bit in the status field is not set, the hardware will extract the transfer length from the data stream and program DMA_buffer_length. Once the extraction is complete both the LS_byte_extracted and MS_byte_extracted bits will be set.

IN endpoints

The DMA transfer proceeds as in normal mode and continues until the number of bytes transferred equals the DMA_buffer_length.

13.7.4 Ending the packet transfer

The DMA engine proceeds with the transfer until the number of bytes specified in the field DMA_buffer_length is transferred to or from the USB RAM. Then the EOT interrupt will be generated. If this happens in the middle of the packet, the linked DD will get loaded and the remaining part of the packet gets transferred to or from the address pointed by the new DD.

OUT endpoints

If the linked DD is not valid and the packet is partially transferred to memory, the DD ends with DataOverrun status code set, and the DMA will be disabled for this endpoint. Otherwise DD_status will be updated with the NormalCompletion status code.

IN endpoints

If the linked DD is not valid and the packet is partially transferred to USB, the DD ends with a status code of NormalCompletion in the DD_status field. This situation corresponds to the end of the USB transfer, and the packet will be sent as a short packet. Also, when the linked DD is valid and buffer length is 0, an empty packet will be sent to indicate the end of the USB transfer.

14. Double buffered endpoint operation

The Bulk and Isochronous endpoints of the USB Device Controller are double buffered to increase data throughput.

When a double-buffered endpoint is realized, enough space for both endpoint buffers is automatically allocated in the EP_RAM. See Section 13–8.5.1.

For the following discussion, the endpoint buffer currently accessible to the CPU or DMA engine for reading or writing is said to be the active buffer.

14.1 Bulk endpoints

For Bulk endpoints, the active endpoint buffer is switched by the SIE Clear Buffer or Validate Buffer commands.

The following example illustrates how double buffering works for a Bulk OUT endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty, and that the active buffer is B_1.

1. The host sends a data packet to the endpoint. The device hardware puts the packet into B_1, and generates an endpoint interrupt.
2. Software clears the endpoint interrupt and begins reading the packet data from B_1. While B_1 is still being read, the host sends a second packet, which device hardware places in B_2, and generates an endpoint interrupt.
3. Software is still reading from B_1 when the host attempts to send a third packet. Since both B_1 and B_2 are full, the device hardware responds with a NAK.
4. Software finishes reading the first packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.

5. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. Software finds that the active buffer (B_2) has data (FE=1). Software clears the endpoint interrupt and begins reading the contents of B_2.
6. The host resends the third packet which device hardware places in B_1. An endpoint interrupt is generated.
7. Software finishes reading the second packet from B_2 and sends a SIE Clear Buffer command to free B_2 to receive another packet. B_1 becomes the active buffer. Software waits for the next endpoint interrupt to occur (it already has been generated back in step 6).
8. Software responds to the endpoint interrupt by clearing it and begins reading the third packet from B_1.
9. Software finishes reading the third packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.
10. Software tests the FE bit and finds that the active buffer (B_2) is empty (FE=0).
11. Both B_1 and B_2 are empty. Software waits for the next endpoint interrupt to occur. The active buffer is now B_2. The next data packet sent by the host will be placed in B_2.

The following example illustrates how double buffering works for a Bulk IN endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty and that the active buffer is B_1. The interrupt on NAK feature is enabled.

1. The host requests a data packet by sending an IN token packet. The device responds with a NAK and generates an endpoint interrupt.
2. Software clears the endpoint interrupt. The device has three packets to send. Software fills B_1 with the first packet and sends a SIE Validate Buffer command. The active buffer is switched to B_2.
3. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. It finds that B_2 is empty (FE=0) and fills B_2 with the second packet. Software sends a SIE Validate Buffer command, and the active buffer is switched to B_1.
4. Software waits for the endpoint interrupt to occur.
5. The device successfully sends the packet in B_1 and clears the buffer. An endpoint interrupt occurs.
6. Software clears the endpoint interrupt. Software fills B_1 with the third packet and validates it using the SIE Validate Buffer command. The active buffer is switched to B_2.
7. The device successfully sends the second packet from B_2 and generates an endpoint interrupt.
8. Software has no more packets to send, so it simply clears the interrupt.
9. The device successfully sends the third packet from B_1 and generates an endpoint interrupt.
10. Software has no more packets to send, so it simply clears the interrupt.

11. Both B_1 and B_2 are empty, and the active buffer is B_2. The next packet written by software will go into B_2.

In DMA mode, switching of the active buffer is handled automatically in hardware. For Bulk IN endpoints, proactively filling an endpoint buffer to take advantage of the double buffering can be accomplished by manually starting a packet transfer using the USBDMARSet register.

14.2 Isochronous endpoints

For isochronous endpoints, the active data buffer is switched by hardware when the FRAME interrupt occurs. The SIE Clear Buffer and Validate Buffer commands do not cause the active buffer to be switched.

Double buffering allows the software to make full use of the frame interval writing or reading a packet to or from the active buffer, while the packet in the other buffer is being sent or received on the bus.

For an OUT isochronous endpoint, any data not read from the active buffer before the end of the frame is lost when it switches.

For an IN isochronous endpoint, if the active buffer is not validated before the end of the frame, an empty packet is sent on the bus when the active buffer is switched, and its contents will be overwritten when it becomes active again.

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 317. 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

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 wakeup 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.

1.2 Architecture

The architecture of the USB host controller is shown below in Figure 14–57.

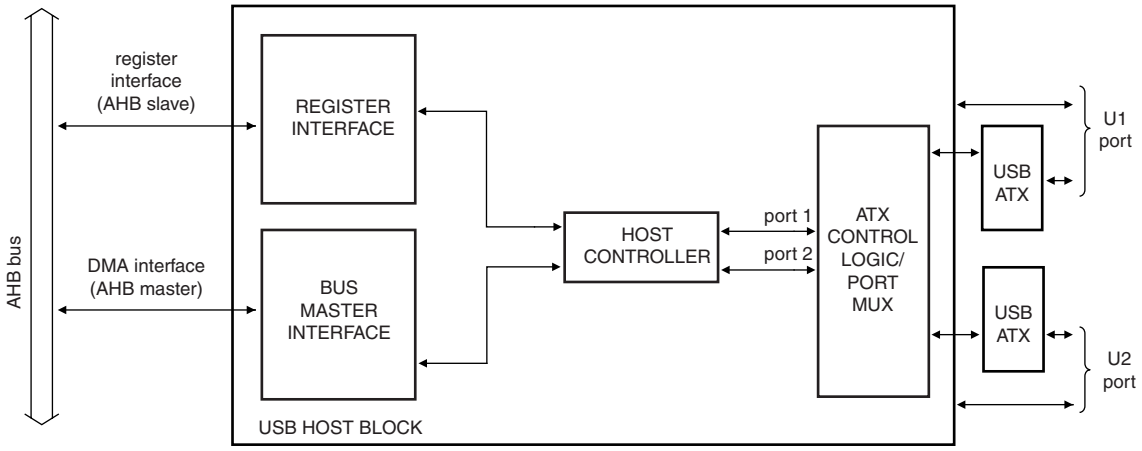


Fig 57. USB Host controller block diagram

2. Interfaces

The OTG controller has two USB ports indicated by suffixes 1 and 2 in the USB pin names and referred to as USB port 1 (U1) and USB port 2 (U2) in the following text.

2.1 Pin description

Table 318. USB OTG port pins

Pin name	Direction	Description	Pin category
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding PINSEL register, it is driven HIGH internally.	USB Connector
Port U1			
USB_D+1	I/O	Positive differential data	USB Connector
USB_D−1	I/O	Negative differential data	USB Connector
USB_CONNECT1	O	SoftConnect control signal	Control
USB_UP_LED1	O	GoodLink LED control signal	Control
$\overline{\text{USB\_INT1}}$	I	OTG ATX interrupt	External OTG transceiver
USB_SCL1	I/O	I^2C serial clock	External OTG transceiver
USB_SDA1	I/O	I^2C serial data	External OTG transceiver
$\overline{\text{USB\_TX\_E1}}$	O	Transmit enable	External OTG transceiver
USB_TX_DP1	O	D+ transmit data	External OTG transceiver
USB_TX_DM1	O	D− transmit data	External OTG transceiver
USB_RCV1	I	Differential receive data	External OTG transceiver
USB_RX_DP1	I	D+ receive data	External OTG transceiver
USB_RX_DM1	I	D− receive data	External OTG transceiver
$\overline{\text{USB\_LS1}}$	O	Low speed status (applies to host functionality only)	External OTG transceiver
$\overline{\text{USB\_SSPND1}}$	O	Bus suspend status	External OTG transceiver
$\overline{\text{USB\_PPWR1}}$	O	Port power enable	Host power switch
USB_PWRD1	I	Port power status	Host power switch

Table 318. USB OTG port pins

Pin name	Direction	Description	Pin category
$\overline{\text{USB\_OVRCR1}}$	I	Over-current status	Host power switch
$\overline{\text{USB\_HSTEN1}}$	O	Host enabled status	
Port U2			
USB_D+2	I/O	Positive differential data	USB Connector
USB_D−2	I/O	Negative differential data	USB Connector
USB_CONNECT2	O	SoftConnect control signal	Control
USB_UP_LED2	O	GoodLink LED control signal	Control
$\overline{\text{USB\_PPWR2}}$	O	Port power enable	Host power switch
U2PWRD2	I	Port power status	Host power switch
$\overline{\text{USB\_OVRCR2}}$	I	Over-current status	Host power switch
$\overline{\text{USB\_HSTEN2}}$	O	Host enabled status	Control

2.1.1 USB host usage note

Both ports can be configured as USB hosts. For details on how to connect the USB ports, see the USB OTG chapter, Section 15–5 “Pin configuration” .

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. For details on these two aspects see the OHCI specification. The register map is shown in the next subsection.

2.2.1 Register map

The following registers are located in the AHB clock ‘cclk’ domain. They can be accessed directly by the processor. All registers are 32 bit wide and aligned in the word address boundaries.

Table 319. 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

Table 319. USB Host register address definitions *…continued*

Name	Address	R/W[1]	Function	Reset value
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
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
Module_ID/Ver_Rev_ID	0x3102 00FC	R	IP number, where yy (0x00) is unique version number and zz (0x00) is a unique revision number.	0x3505yyzz

[1] The R/W column in Table 14–319 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.

2.2.2 USB Host Register Definitions

Refer to the OHCI specification document on Compaq’s website for register definitions.

UM10237

Chapter 15: USB OTG controller

Rev. 01 — 18 December 2006 **User manual**

1. Introduction

This chapter describes the OTG and I^2C portions of the USB 2.0 OTG dual role device controller which integrates the (OHCI) host controller, device controller, and I^2C. The I^2C interface (Master only) controls an external OTG transceiver.

USB OTG (On-The-Go) is a supplement to the USB 2.0 specification that augments the capability of existing mobile devices and USB peripherals by adding host functionality for connection to USB peripherals. The specification and more information on USB OTG can be found on the USB Implementers Forum web site.

2. Features

- Fully compliant with On-The-Go supplement to the *USB 2.0 Specification, Revision 1.0a*.
- Hardware support for Host Negotiation Protocol (HNP).
- Includes a programmable timer required for HNP and SRP.
- Supports any OTG transceiver compliant with the *OTG Transceiver Specification (CEA-2011), Rev. 1.0*.

3. Architecture

The architecture of the USB OTG controller is shown below in the block diagram.

The host, device, OTG, and I2C controllers can be programmed through the register interface. The OTG controller enables dynamic switching between host and device roles through the HNP protocol. One port may be connected to an external OTG transceiver to support an OTG connection. The communication between the register interface and an external OTG transceiver is handled through an I^2C interface and through the external OTG transceiver interrupt signal.

For USB connections that use the device or host controller only (not OTG), the ports use an embedded USB Analog Transceiver (ATX).

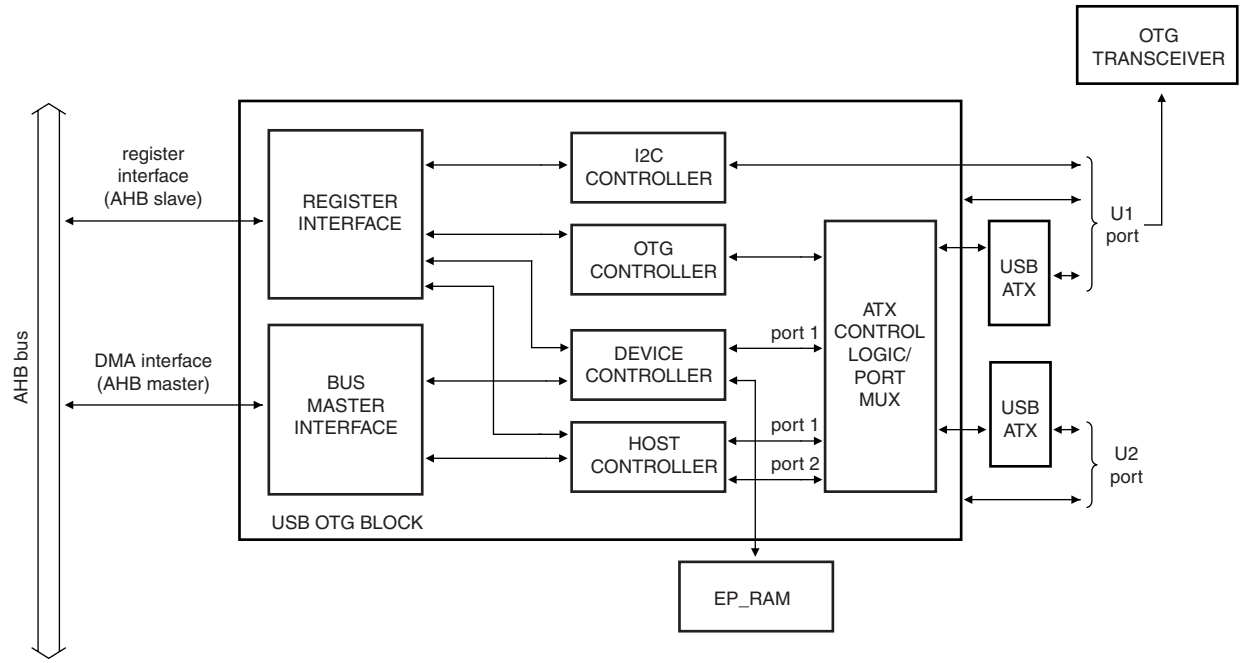


Fig 58. USB OTG controller block diagram

4. Modes of operation

The OTG controller is capable of operating in the following modes:

- One port host and one port dual-role device (see Figure 15–59)
- One port host and one port device (see Figure 15–61)
- Two port host (see Figure 15–62)

5. Pin configuration

The OTG controller has two USB ports indicated by suffixes 1 and 2 in the USB pin names and referred to as USB port 1 (U1) and USB port 2 (U2) in the following text.

Table 320. USB OTG port 1 pins

Pin name	Direction	Description	Pin category
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding PINSEL register, it is driven HIGH internally.	USB Connector
Port U1			
USB_D+1	I/O	Positive differential data	USB Connector
USB_D−1	I/O	Negative differential data	USB Connector
USB_CONNECT1	O	SoftConnect control signal	Control
USB_UP_LED1	O	GoodLink LED control signal	Control
$\overline{\text{USB\_INT1}}$	I	OTG ATX interrupt	External OTG transceiver
USB_SCL1	I/O	I^2C serial clock	External OTG transceiver

Table 320. USB OTG port 1 pins

Pin name	Direction	Description	Pin category
USB_SDA1	I/O	I^2C serial data	External OTG transceiver
USB_TX_E1	O	Transmit enable	External OTG transceiver
USB_TX_DP1	O	D+ transmit data	External OTG transceiver
USB_TX_DM1	O	D− transmit data	External OTG transceiver
USB_RCV1	I	Differential receive data	External OTG transceiver
USB_RX_DP1	I	D+ receive data	External OTG transceiver
USB_RX_DM1	I	D− receive data	External OTG transceiver
USB_LS1	O	Low speed status (applies to host functionality only)	External OTG transceiver
USB_SSPND1	O	Bus suspend status	External OTG transceiver
USB_PPWR1	O	Port power enable	Host power switch
USB_PWRD1	I	Port power status	Host power switch
USB_OVRCR1	I	Over-current status	Host power switch
USB_HSTEN1	O	Host enabled status	
Port U2			
USB_D+2	I/O	Positive differential data	USB Connector
USB_D−2	I/O	Negative differential data	USB Connector
USB_CONNECT2	O	SoftConnect control signal	Control
USB_UP_LED2	O	GoodLink LED control signal	Control
USB_PPWR2	O	Port power enable	Host power switch
USB_PWRD2	I	Port power status	Host power switch
USB_OVRCR2	I	Over-current status	Host power switch
USB_HSTEN2	O	Host enabled status	Control

The following figures show different ways to realize connections to an USB device using ports U1 and U2. The example described here uses an ISP1301 (NXP) for the external OTG transceiver and the USB Host power switch LM3526-L (National Semiconductors).

5.1 Connecting port U1 to an external OTG transceiver

For OTG functionality an external OTG transceiver must be connected to the LPC2400 device. There are two ways to connect the OTG transceiver (here ISP1301) to port U1:

1. Use the internal USB transceiver for USB signalling and use the external OTG transceiver for OTG functionality only (see Figure 15–59). This option uses the internal transceiver in VP/VM mode.
2. Use the external OTG transceiver in VP/VM mode for OTG functionality and USB signalling (see Figure 15–60).

In both cases port U2 is connected as a host. Solution one uses fewer pins.

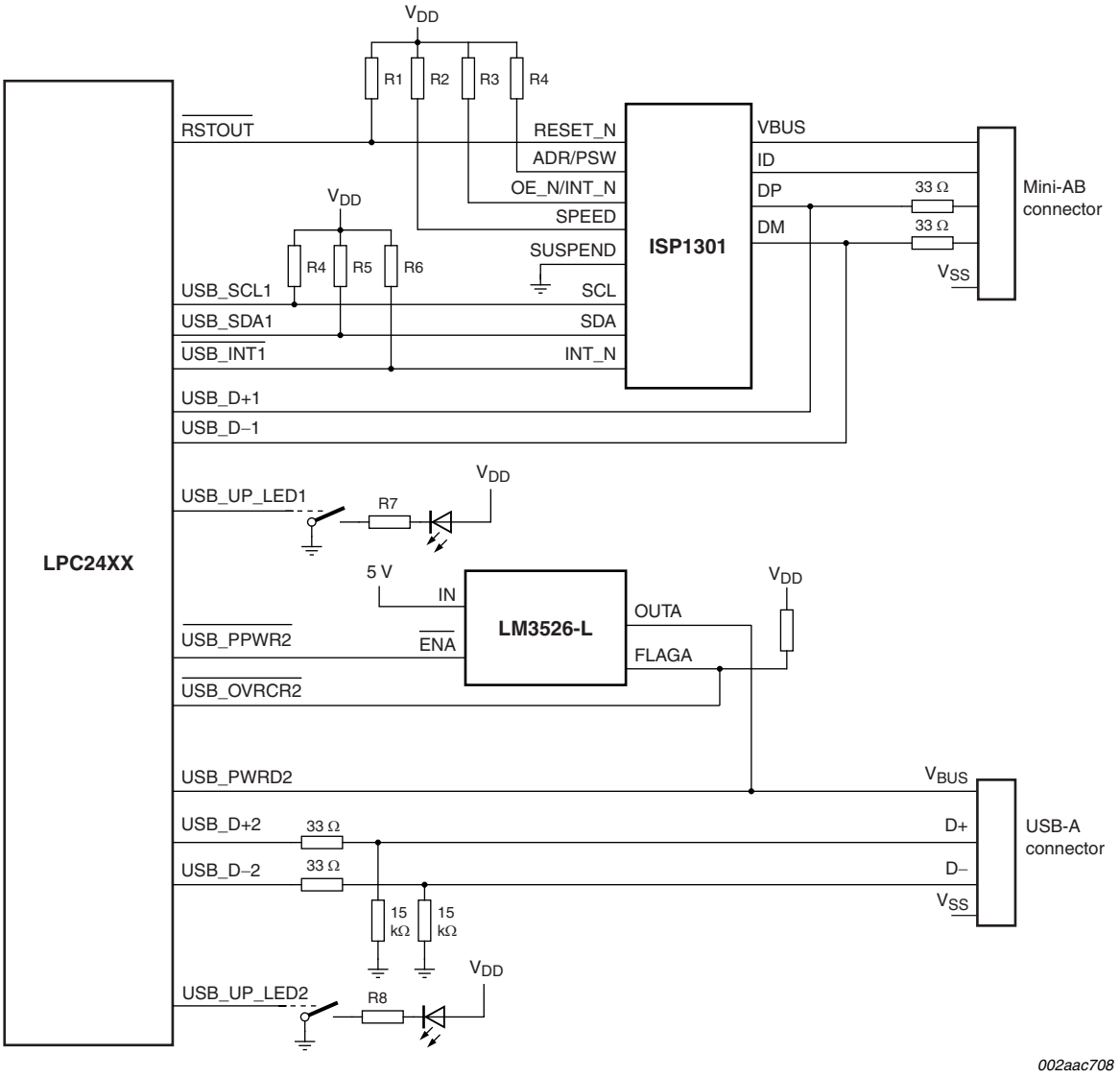


Fig 59. USB OTG port configuration: port U1 OTG Dual-Role device, port U2 host

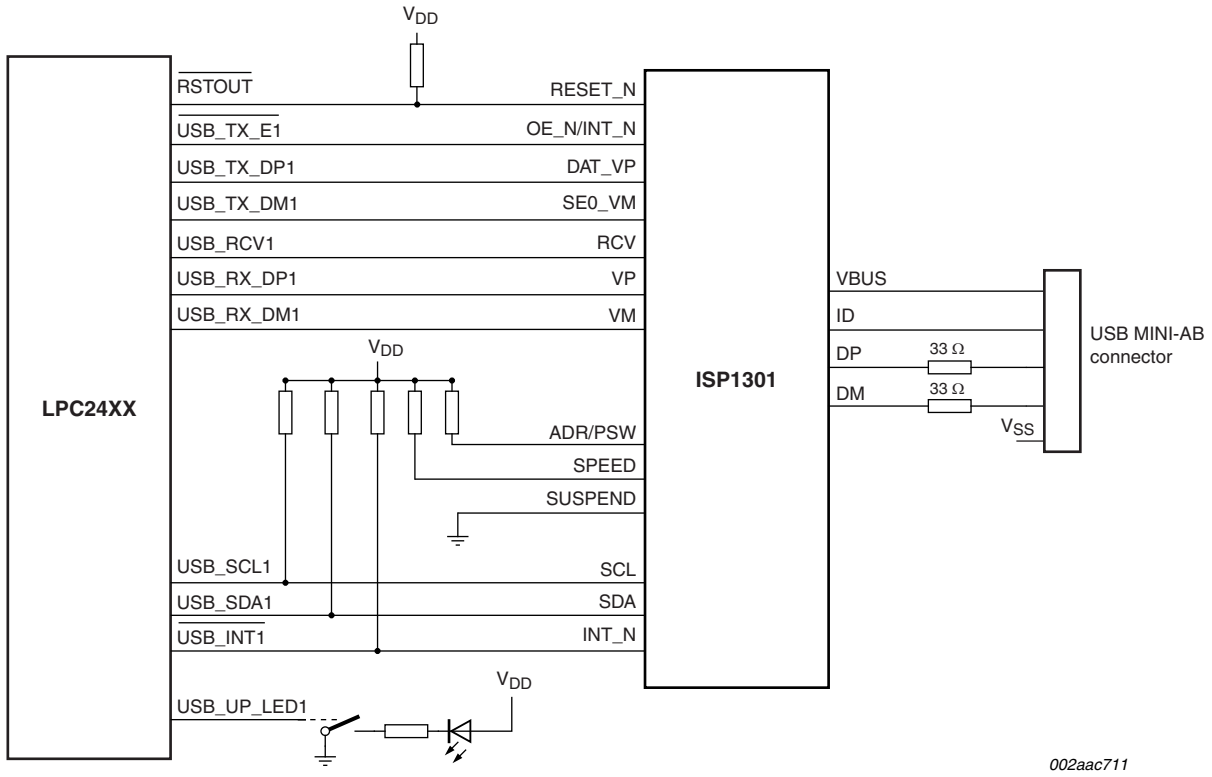


Fig 60. USB OTG port configuration: VP_VM mode

5.2 Connecting USB as a two port host

Both ports U1 and U2 are connected as hosts using an embedded USB transceiver. There is no OTG functionality on either port.

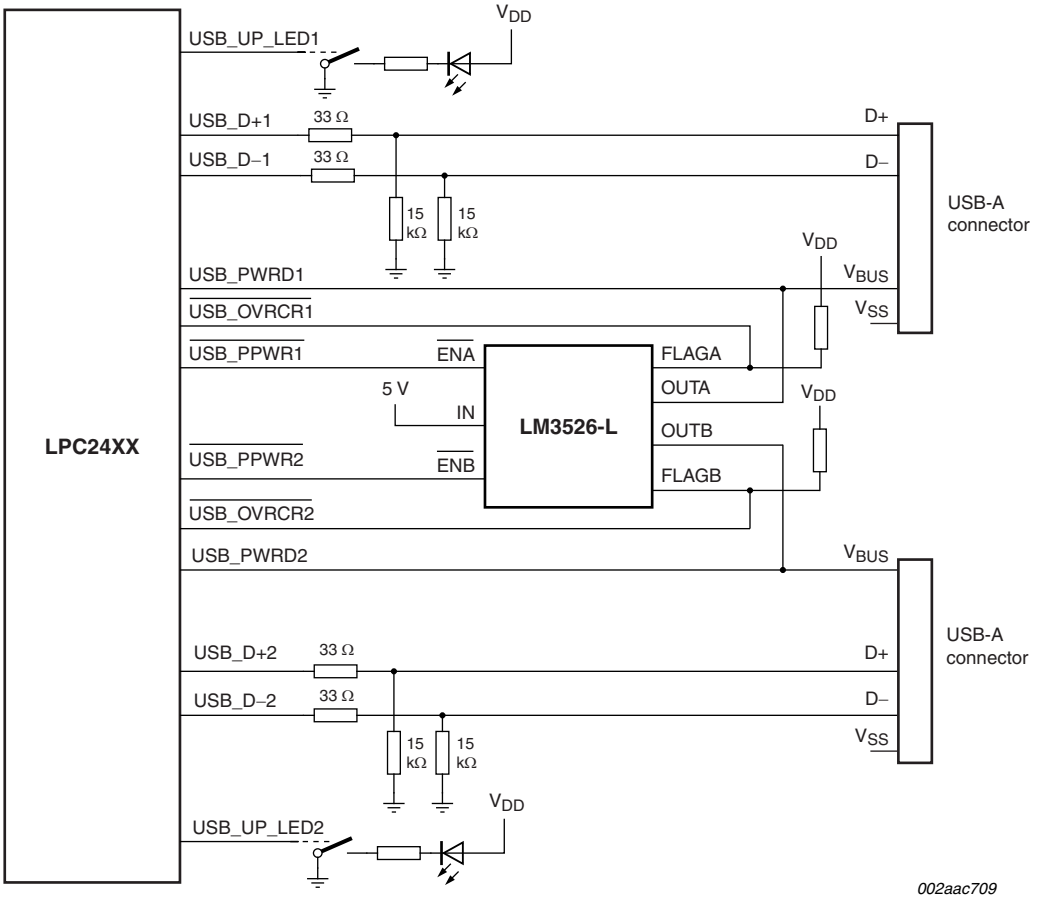


Fig 61. USB OTG port configuration: port U2 host, port U1 host

5.3 Connecting USB as one port host and one port device

Port U2 is connected as device, and port U1 is connected as host using an embedded USB transceiver. There is no OTG functionality on either port.

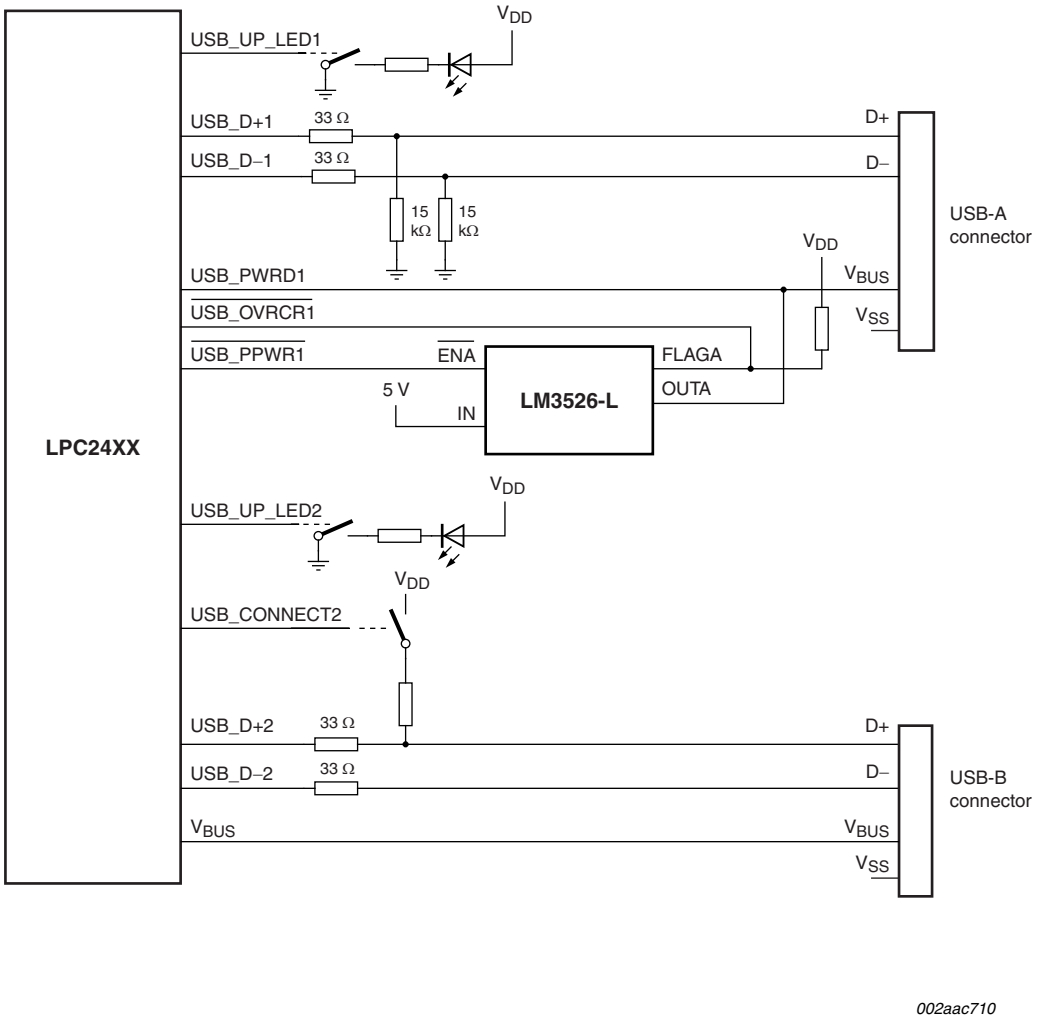


Fig 62. USB OTG port configuration: port U1 host, port U2 device

6. Register description

The OTG and I^2C registers are summarized in the following table.

The Device and Host registers are explained in Section 14–2.2.1 and Section 13–8 in the USB Device Controller and USB Host (OHCI) Controller chapters. All registers are 32 bits wide and aligned to word address boundaries.

Table 321. USB OTG and I^2C register address definitions

Name	Address	Access	Function
Interrupt register			
USBIntSt	0xE01F C1C0	R/W	USB Interrupt Status
OTG registers			
OTGIntSt	0xFFE0 C100	RO	OTG Interrupt Status
OTGIntEn	0xFFE0 C104	R/W	OTG Interrupt Enable

Table 321. USB OTG and I²C register address definitions

Name	Address	Access	Function
OTGIntSet	0xFFE0 C108	WO	OTG Interrupt Set
OTGIntClr	0xFFE0 C10C	WO	OTG Interrupt Clear
OTGStCtrl	0xFFE0 C110	R/W	OTG Status and Control
OTGTmr	0xFFE0 C114	R/W	OTG Timer
I²C registers			
I2C_RX	0xFFE0 C300	RO	I2C Receive
I2C_TX	0xFFE0 C300	WO	I2C Transmit
I2C_STS	0xFFE0 C304	RO	I2C Status
I2C_CTL	0xFFE0 C308	R/W	I2C Control
I2C_CLKHI	0xFFE0 C30C	R/W	I2C Clock High
I2C_CLKLO	0xFFE0 C310	WO	I2C Clock Low
Clock control registers			
OTGClkCtrl	0xFFE0 CFF4	R/W	OTG clock controller
OTGClkSt	0xFFE0 CFF8	RO	OTG clock status

6.1 USB Interrupt Status Register (USBIntSt - 0xE01F C1C0)

The USB OTG controller has seven interrupt lines. This register allows software to determine their status with a single read operation.

The interrupt lines are ORed together to a single channel of the vectored interrupt controller.

Table 322. USB Interrupt Status register - (USBIntSt - address 0xE01F C1) bit description

Bit	Symbol	Description	Reset Value
0	USB_INT_REQ_LP	Low priority interrupt line status. This bit is read only.	0
1	USB_INT_REQ_HP	High priority interrupt line status. This bit is read only.	0
2	USB_INT_REQ_DMA	DMA interrupt line status. This bit is read only.	0
3	USB_HOST_INT	USB host interrupt line status. This bit is read only.	0
4	USB_ATX_INT	External ATX interrupt line status. This bit is read only.	0
5	USB_OTG_INT	OTG interrupt line status. This bit is read only.	0
6	USB_I2C_INT	I²C module interrupt line status. This bit is read only.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	USB_NEED_CLK	USB need clock indicator. This bit is read only.	??
30:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31	EN_USB_INTS	Enable all USB interrupts. When this bit is cleared, the VIC does not see the ORed output of the USB interrupt lines.	1

6.2 OTG Interrupt Status Register (OTGIntSt - 0xE01F C100)

Bits is this register are set by hardware when the interrupt event occurs during the HNP handoff sequence. See Section 15–7 for more information on when these bits are set.

Table 323. OTG Interrupt Status register (OTGIntSt - address 0xE01F C100) bit description

Bit	Symbol	Description	Reset Value
0	TMR	Timer time-out.	0
1	REMOVE_PU	Remove pull-up. This bit is set by hardware to indicate that software needs to disable the D+ pull-up resistor.	0
2	HNP_FAILURE	HNP failed. This bit is set by hardware to indicate that the HNP switching has failed.	0
3	HNP_SUCCESS	HNP succeeded. This bit is set by hardware to indicate that the HNP switching has succeeded.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.3 OTG Interrupt Enable Register (OTGIntEn - 0xFFE0 C104)

Writing a one to a bit in this register enables the corresponding bit in OTGIntSt to generate an interrupt on one of the interrupt lines. The interrupt is routed to the USB_OTG_INT interrupt line in the USBIntSt register.

The bit allocation and reset value of OTGIntEn is the same as OTGIntSt.

6.4 OTG Interrupt Set Register (OTGIntSet - 0xFFE0 C20C)

Writing a one to a bit in this register will set the corresponding bit in the OTGIntSt register. Writing a zero has no effect. The bit allocation of OTGIntSet is the same as in OTGIntSt.

6.5 OTG Interrupt Clear Register (OTGIntClr - 0xFFE0 C10C)

Writing a one to a bit in this register will clear the corresponding bit in the OTGIntSt register. Writing a zero has no effect. The bit allocation of OTGIntClr is the same as in OTGIntSt.

6.6 OTG Status and Control Register (OTGStCtrl - 0xFFE0 C110)

The OTGStCtrl register allows enabling hardware tracking during the HNP hand over sequence, controlling the OTG timer, monitoring the timer count, and controlling the functions mapped to port U1 and U2.

Time critical events during the switching sequence are controlled by the OTG timer. The timer can operate in two modes:

1. Monoshot mode: an interrupt is generated at the end of TIMEOUT_CNT (see Section 15–6.7 “OTG Timer Register (OTGTmr - 0xFFE0 C114)”), the TMR bit is set in OTGIntSt, and the timer will be disabled.

2. Free running mode: an interrupt is generated at the end of TIMEOUT_CNT (see Section 15–6.7 “OTG Timer Register (OTGTmr - 0xFFE0 C114)”), the TMR bit is set, and the timer value is reloaded into the counter. The timer is not disabled in this mode.

Table 324. OTG Status Control register (OTGStCtrl - address 0xFFE0 C110) bit description

Bit	Symbol	Description	Reset Value
1:0	PORT_FUNC	Controls the function of ports U1 and U2. Bit 0 is set or cleared by hardware when B_HNP_TRACK or A_HNP_TRACK is set and HNP succeeds. See Section 15–7.	-
3:2	TMR_SCALE	Timer scale selection. This field determines the duration of each timer count. 00: 10 μs (100 KHz) 01: 100 μs (10 KHz) 10: 1000 μs (1 KHz) 11: Reserved	0x0
4	TMR_MODE	Timer mode selection. 0: monoshot 1: free running	0
5	TMR_EN	Timer enable. When set, TMR_CNT increments. When cleared, TMR_CNT is reset to 0.	0
6	TMR_RST	Timer reset. Writing one to this bit resets TMR_CNT to 0. This provides a single bit control for the software to restart the timer when the timer is enabled.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	B_HNP_TRACK	Enable HNP tracking for B-device (peripheral), see Section 15–7. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
9	A_HNP_TRACK	Enable HNP tracking for A-device (host), see Section 15–7. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
10	PU_REMOVED	When the B-device changes its role from peripheral to host, software sets this bit when it removes the D+ pull-up, see Section 15–7. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31:16	TMR_CNT	Current timer count value.	0x0

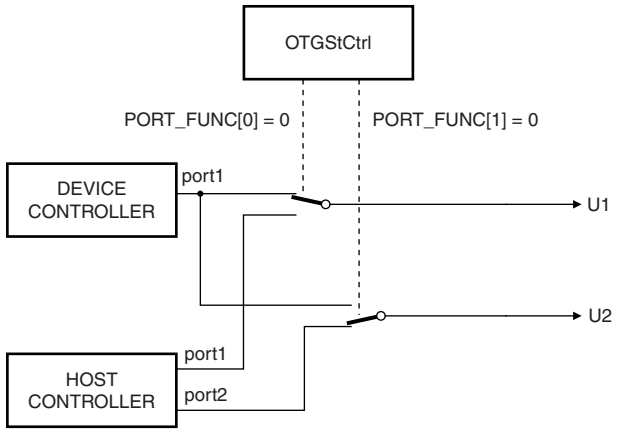


Fig 63. Port selection for PORT_FUNC bit 0 = 0 and PORT_FUNC bit 1 = 0.

Table 325. Port function truth table

	PORT_FUNC[0] = 0	PORT_FUNC[0] = 1
PORT_FUNC[1] = 0	U1 = device (OTG) U2 = host	U1 = host (OTG) U2 = host
PORT_FUNC[1] = 1	reserved	U1 = host U2 = device

6.7 OTG Timer Register (OTGTmr - 0xFFE0 C114)

Table 326. OTG Timer register (OTGTmr - address 0xFFE0 C114) bit description

Bit	Symbol	Description	Reset Value
15:0	TIMEOUT_CNT	The TMR interrupt is set when TMR_CNT reaches this value.	0xFFFF
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.8 OTG Clock Control Register (OTGClkCtrl - 0xFFE0 CFF4)

This register controls the clocking of the OTG controller. Whenever software wants to access the registers, the corresponding clock control bit needs to be set. The software does not have to repeat this exercise for every register access, provided that the corresponding OTGClkCtrl bits are already set.

Table 327. OTG_clock_control register (OTG_clock_control - address 0xFFE0 CFF4) bit description

Bit	Symbol	Value	Description	Reset Value
0	HOST_CLK_EN		Host clock enable	0
		0	Disable the Host clock.	
		1	Enable the Host clock.	

Table 327. OTG_clock_control register (OTG_clock_control - address 0xFFE0 CFF4) bit description

Bit	Symbol	Value	Description	Reset Value
1	DEV_CLK_EN		Device clock enable	0
		0	Disable the Device clock.	
		1	Enable the Device clock.	
2	I2C_CLK_EN		I2C clock enable	0
		0	Disable the I^2C clock.	
		1	Enable the I^2C clock.	
3	OTG_CLK_EN		OTG clock enable	0
		0	Disable the OTG clock.	
		1	Enable the OTG clock.	
4	AHB_CLK_EN		AHB master clock enable	0
		0	Disable the AHB clock.	
		1	Enable the AHB clock.	
31:5	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.9 OTG Clock Status Register (OTGClkSt - 0xFFE0 CFF8)

This register holds the clock availability status. When enabling a clock via OTGClkCtrl, software should poll the corresponding bit in this register. If it is set, then software can go ahead with the register access. Software does not have to repeat this exercise for every access, provided that the OTGClkCtrl bits are not disturbed.

Table 328. OTG_clock_status register (OTGClkSt - address 0xFFE0 CFF8) bit description

Bit	Symbol	Value	Description	Reset Value
0	HOST_CLK_ON		Host clock status.	0
		0	Host clock is not available.	
		1	Host clock is available.	
1	DEV_CLK_ON		Device clock status.	0
		0	Device clock is not available.	
		1	Device clock is available.	
2	I2C_CLK_ON		I2C clock status.	0
		0	I2C clock is not available.	
		1	I2C clock is available.	
3	OTG_CLK_ON		OTG clock status.	0
		0	OTG clock is not available.	
		1	OTG clock is available.	

Table 328. OTG_clock_status register (OTGClkSt - address 0xFFE0 CFF8) bit description

Bit	Symbol	Value	Description	Reset Value
4	AHB_CLK_ON		AHB master clock status.	0
		0	AHB clock is not available.	
		1	AHB clock is available.	
31:5	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.10 I2C Receive Register (I2C_RX - 0xFFE0 C300)

This register is the top byte of the receive FIFO. The receive FIFO is 4 bytes deep. The Rx FIFO is flushed by a hard reset or by a soft reset (I2C_CTL bit 7). Reading an empty FIFO gives unpredictable data results.

Table 329. I2C Receive register (I2C_RX - address 0xFFE0 C300) bit description

Bit	Symbol	Description	Reset Value
7:0	RX Data	Receive data.	-

6.11 I2C Transmit Register (I2C_TX - 0xFFE0 C300)

This register is the top byte of the transmit FIFO. The transmit FIFO is 4 bytes deep.

The Tx FIFO is flushed by a hard reset, soft reset (I2C_CTL bit 7) or if an arbitration failure occurs (I2C_STS bit 3). Data writes to a full FIFO are ignored.

I2C_TX must be written for both write and read operations to transfer each byte. Bits [7:0] are ignored for master-receive operations. The master-receiver must write a dummy byte to the TX FIFO for each byte it expects to receive in the RX FIFO. When the STOP bit is set or the START bit is set to cause a RESTART condition on a byte written to the TX FIFO (master-receiver), then the byte read from the slave is not acknowledged. That is, the last byte of a master-receive operation is not acknowledged.

Table 330. I2C Transmit register (I2C_TX - address 0xFFE0 C300) bit description

Bit	Symbol	Description	Reset Value
7:0	TX Data	Transmit data.	-
8	START	When 1, issue a START condition before transmitting this byte.	-
9	STOP	When 1, issue a STOP condition after transmitting this byte.	-
31:10	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

6.12 I2C Status Register (I2C_STS - 0xFFE0 C304)

The I2C_STS register provides status information on the TX and RX blocks as well as the current state of the external buses. Individual bits are enabled as interrupts by the I2C_CTL register and routed to the I2C_USB_INT bit in USBIntSt.

Table 331. I2C status register (I2C_STS - address 0xFFE0 C304) bit description

Bit	Symbol	Value	Description	Reset Value
0	TDI		Transaction Done Interrupt. This flag is set if a transaction completes successfully. It is cleared by writing a one to bit 0 of the status register. It is unaffected by slave transactions.	0
		0	Transaction has not completed.	
		1	Transaction completed.	
1	AFI		Arbitration Failure Interrupt. When transmitting, if the SDA is low when SDAOUT is high, then this 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 one to bit 1 of the status register.	0
		0	No arbitration failure on last transmission.	
		1	Arbitration failure occurred on last transmission.	
2	NAI		No Acknowledge Interrupt. After every byte of data is sent, the transmitter expects an acknowledge from the receiver. This bit is set if the acknowledge is not received. It is cleared when a byte is written to the master TX FIFO.	0
		0	Last transmission received an acknowledge.	
		1	Last transmission did not receive an acknowledge.	
3	DRMI		Master Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a stop condition or it will hold SCL low until more data is available. The Master Data Request bit is set when the master transmitter is data-starved. If the master TX FIFO is empty and the last byte did not have a STOP condition flag, then SCL is held low until the CPU writes another byte to transmit. This bit is cleared when a byte is written to the master TX FIFO.	0
		0	Master transmitter does not need data.	
		1	Master transmitter needs data.	
4	DRSI		Slave Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a STOP condition or it will hold SCL low until more data is available. The Slave Data Request bit is set when the slave transmitter is data-starved. If the slave TX FIFO is empty and the last byte transmitted was acknowledged, then SCL is held low until the CPU writes another byte to transmit. This bit is cleared when a byte is written to the slave Tx FIFO.	0
		0	Slave transmitter does not need data.	
		1	Slave transmitter needs data.	
5	Active		Indicates whether the bus is busy. This bit is set when a START condition has been seen. It is cleared when a STOP condition is seen..	0
6	SCL		The current value of the SCL signal.	-
7	SDA		The current value of the SDA signal.	-

Table 331. I2C status register (I2C_STS - address 0xFFE0 C304) bit description

Bit	Symbol	Value	Description	Reset Value
8	RFF		Receive FIFO Full (RFF). This bit is set when the RX FIFO is full and cannot accept any more data. It is cleared when the RX FIFO is not full. If a byte arrives when the Receive FIFO is full, the SCL is held low until the CPU reads the RX FIFO and makes room for it.	0
		0	RX FIFO is not full	
		1	RX FIFO is full	
9	RFE		Receive FIFO Empty. RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data.	1
		0	RX FIFO contains data.	
		1	RX FIFO is empty	
10	TFF		Transmit FIFO Full. TFF is set when the TX FIFO is full and is cleared when the TX FIFO is not full.	0
		0	TX FIFO is not full.	
		1	TX FIFO is full	
11	TFE		Transmit FIFO Empty. TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data.	1
		0	TX FIFO contains valid data.	
		1	TX FIFO is empty	
31:12	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.13 I2C Control Register (I2C_CTL - 0xFFE0 C308)

The I2C_CTL register is used to enable interrupts and reset the I^2C state machine. Enabled interrupts cause the USB_I2C_INT interrupt output line to be asserted when set.

Table 332. I2C Control register (I2C_CTL - address 0xFFE0 C308) bit description

Bit	Symbol	Value	Description	Reset Value
0	TDIE		Transmit Done Interrupt Enable. This enables the TDI interrupt signalling that this I^2C issued a STOP condition.	0
		0	Disable the TDI interrupt.	
		1	Enable the TDI interrupt.	
1	AFIE		Transmitter Arbitration Failure Interrupt Enable. This enables the AFI interrupt which is asserted during transmission when trying to set SDA high, but the bus is driven low by another device.	0
		0	Disable the AFI.	
		1	Enable the AFI.	
2	NAIE		Transmitter No Acknowledge Interrupt Enable. This enables the NAI interrupt signalling that transmitted byte was not acknowledged.	0
		0	Disable the NAI.	
		1	Enable the NAI.	

Table 332. I2C Control register (I2C_CTL - address 0xFFE0 C308) bit description

Bit	Symbol	Value	Description	Reset Value
3	DRMIE		Master Transmitter Data Request Interrupt Enable. This enables the DRMI interrupt which signals that the master transmitter has run out of data, has not issued a STOP, and is holding the SCL line low.	0
		0	Disable the DRMI interrupt.	
		1	Enable the DRMI interrupt.	
4	DRSIE		Slave Transmitter Data Request Interrupt Enable. This enables the DRSI interrupt which signals that the slave transmitter has run out of data and the last byte was acknowledged, so the SCL line is being held low.	0
		0	Disable the DRSI interrupt.	
		1	Enable the DRSI interrupt.	
5	REFIE		Receive FIFO Full Interrupt Enable. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data.	0
		0	Disable the RFFI.	
		1	Enable the RFFI.	
6	RFDAIE		Receive Data Available Interrupt Enable. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty).	0
		0	Disable the DAI.	
		1	Enable the DAI.	
7	TFFIE		Transmit FIFO Not Full Interrupt Enable. This enables the Transmit FIFO Not Full interrupt to indicate that the more data can be written to the transmit FIFO. Note that this is not full. It is intended help the CPU to write to the I^2C block only when there is room in the FIFO and do this without polling the status register.	0
		0	Disable the TFFI.	
		1	Enable the TFFI.	
8	SRST		Soft reset. This is only needed in unusual circumstances. If a device issues a start condition without issuing a stop condition. A system timer may be used to reset the 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
		0	See the text.	
		1	Reset the I^2C to idle state. Self clearing.	
31:9	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.14 I2C Clock High Register (I2C_CLKHI - 0xFFE0 C30C)

The CLK register holds a terminal count for counting 48 MHz clock cycles to create the high period of the slower I^2C serial clock, SCL.

Table 333. I2C_CLKHI register (I2C_CLKHI - address 0xFFE0 C30C) bit description

Bit	Symbol	Description	Reset Value
7:0	CDHI	Clock divisor high. This value is the number of 48 MHz clocks the serial clock (SCL) will be high.	0xB9

6.15 I2C Clock Low Register (I2C_CLKLO - 0xFFE0 C310)

The CLK register holds a terminal count for counting 48 MHz clock cycles to create the low period of the slower I^2C serial clock, SCL.

Table 334. I2C_CLKLO register (I2C_CLKLO - address 0xFFE0 C310) bit description

Bit	Symbol	Description	Reset Value
7:0	CDLO	Clock divisor low. This value is the number of 48 MHz clocks the serial clock (SCL) will be low.	0xB9

6.16 Interrupt handling

The interrupts set in the OTGIntSt register are set and cleared during HNP switching. All OTG related interrupts, if enabled, are routed to the USB_OTG_INT bit in the USBIntSt register.

I2C related interrupts are set in the I2C_STS register and routed, if enabled by I2C_CTL, to the USB_I2C_INT bit.

For more details on the interrupts created by device controller, see the USB device chapter. For interrupts created by the host controllers, see the OHCI specification.

The EN_USB_INTS bit in the USBIntSt register enables the routing of any of the USB related interrupts to the VIC controller (see Figure 15–64).

Remark: During the HNP switching between host and device with the OTG stack active, an action may raise several levels of interrupts. It is advised to let the OTG stack initiate any actions based on interrupts and ignore device and host level interrupts. This means that during HNP switching, the OTG stack provides the communication to the host and device controllers.

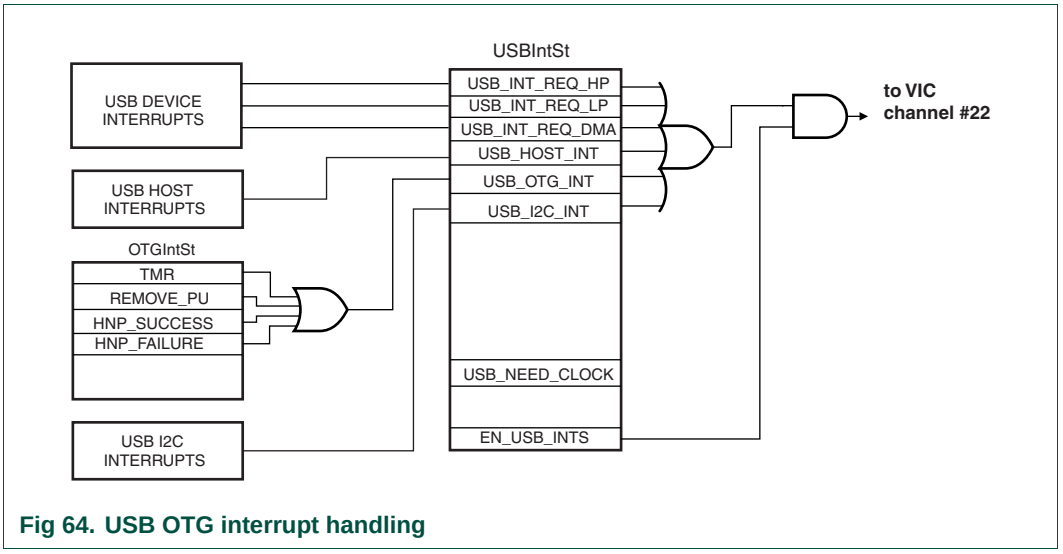


Fig 64. USB OTG interrupt handling

7. HNP support

This section describes the hardware support for the Host Negotiation Protocol (HNP) provided by the OTG controller.

When two dual-role OTG devices are connected to each other, the plug inserted into the mini-AB receptacle determines the default role of each device. The device with the mini-A plug inserted becomes the default Host (A-device), and the device with the mini-B plug inserted becomes the default Peripheral (B-device).

Once connected, the default Host (A-device) and the default Peripheral (B-device) can switch Host and Peripheral roles using HNP.

The context of the OTG controller operation is shown in Figure 15–65. Each controller (Host, Device, or OTG) communicates with its software stack through a set of status and control registers and interrupts. In addition, the OTG software stack communicates with the external OTG transceiver through the I2C interface and the external transceiver interrupt signal.

The OTG software stack is responsible for implementing the HNP state machines as described in the On-The-Go Supplement to the USB 2.0 Specification.

The OTG controller hardware provides support for some of the state transitions in the HNP state machines as described in the following subsections.

The USB state machines, the HNP switching, and the communications between the USB controllers are described in more detail in the following documentation:

- USB OHCI specification
- USB OTG supplement, version 1.2
- USB 2.0 specification
- ISP1301 datasheet and usermanual

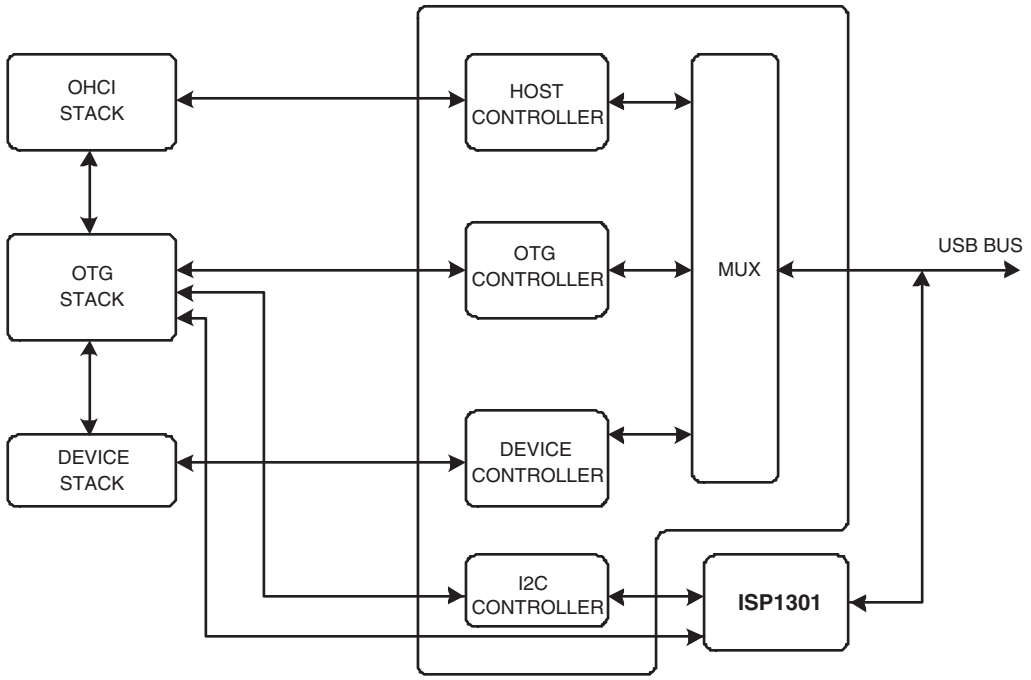


Fig 65. USB OTG controller with software stack

7.1 B-device: peripheral to host switching

In this case, the default role of the OTG controller is peripheral (B-device), and it switches roles from Peripheral to Host.

The On-The-Go Supplement defines the behavior of a dual-role B-device during HNP using a state machine diagram. The OTG software stack is responsible for implementing all of the states in the Dual-Role B-Device State Diagram.

The OTG controller hardware provides support for the state transitions between the states b_peripheral, b_wait_acon, and b_host in the Dual-Role B-Device state diagram. Setting B_HNP_TRACK in the OTGStCtrl register enables hardware support for the B-device switching from peripheral to host. The hardware actions after setting this bit are shown in Figure 15–66.

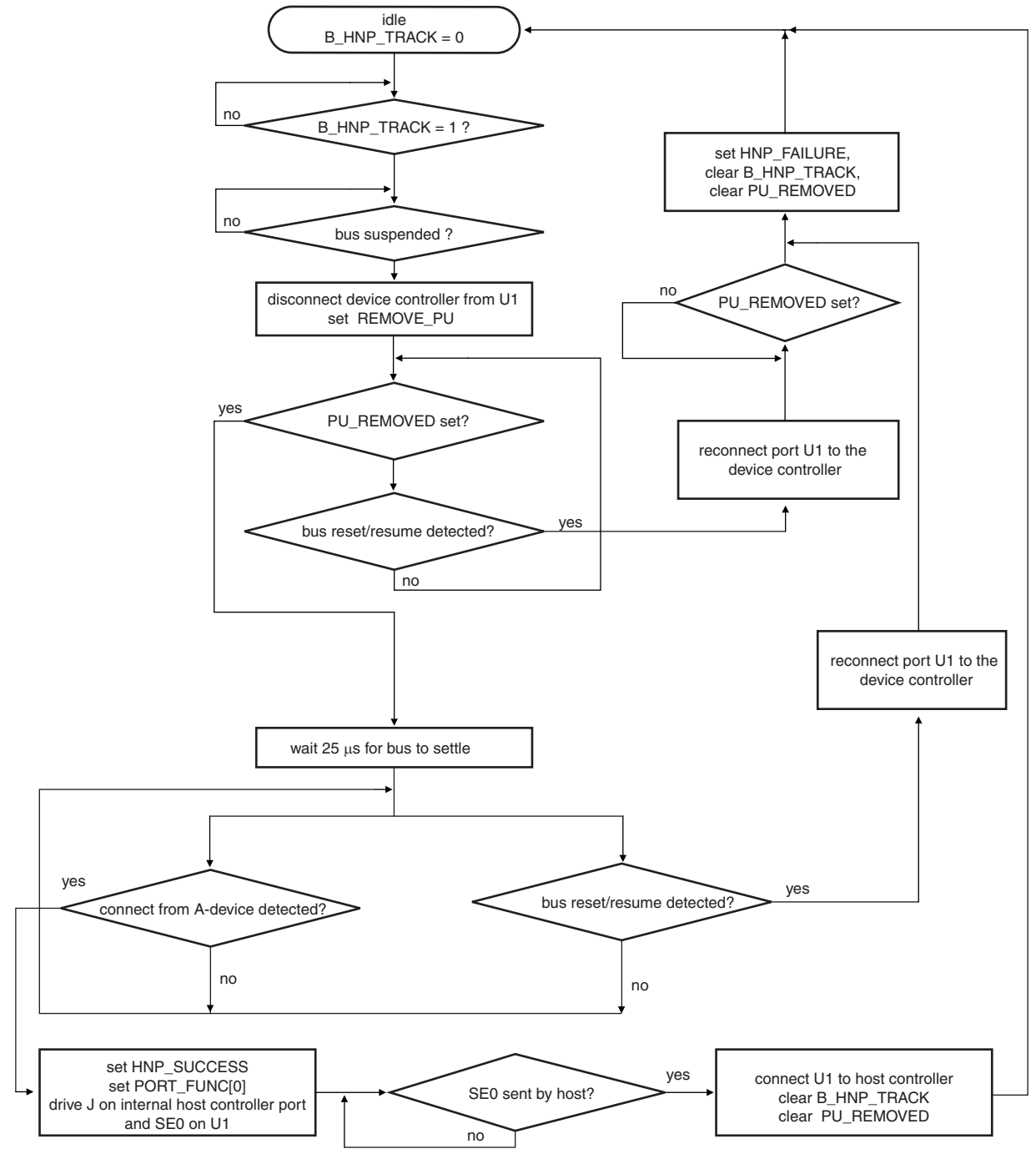


Fig 66. Hardware support for B-device switching from peripheral state to host state

Figure 15–67 shows the actions that the OTG software stack should take in response to the hardware actions setting REMOVE_PU, HNP_SUCCESS, AND HNP_FAILURE. The relationship of the software actions to the Dual-Role B-Device states is also shown. B-device states are in bold font with a circle around them.

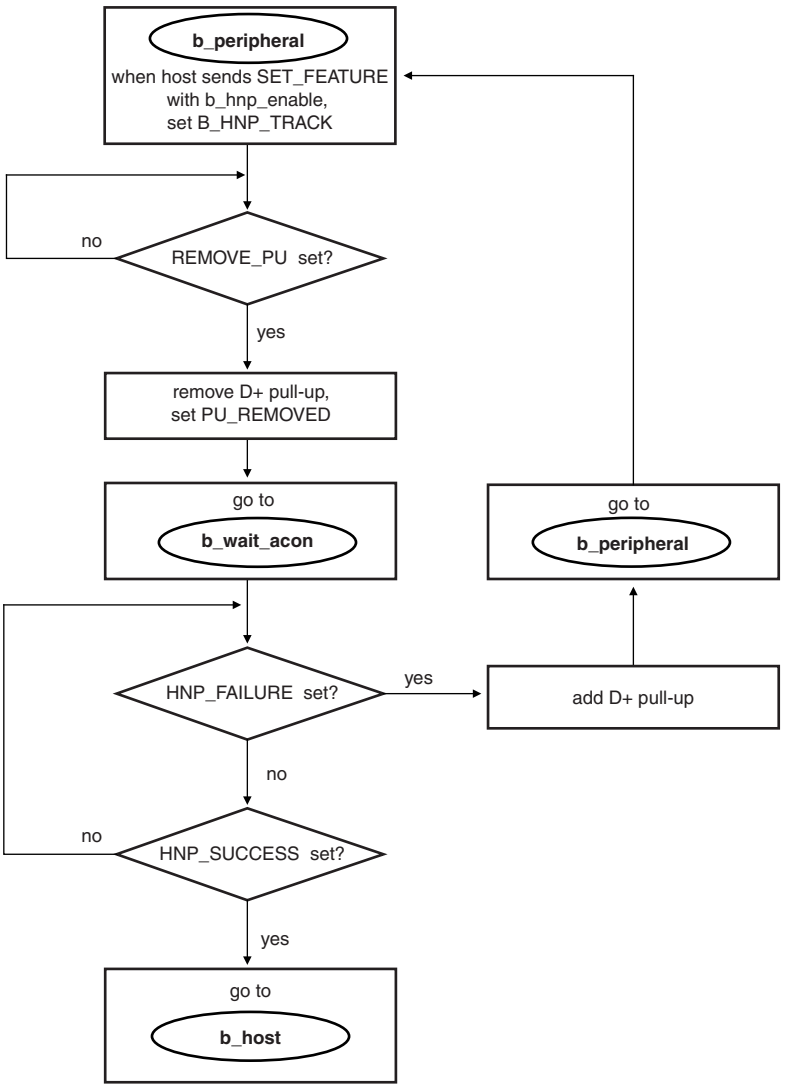


Fig 67. State transitions implemented in software during B-device switching from peripheral to host

Note that only the subset of B-device HNP states and state transitions supported by hardware are shown. Software is responsible for implementing all of the HNP states.

Figure 15–67 may appear to imply that the interrupt bits such as REMOVE_PU should be polled, but this is not necessary if the corresponding interrupt is enabled.

Following are code examples that show how the actions in Figure 15–67 are accomplished. The examples assume that ISP1301 is being used as the external OTG transceiver.

Remove D+ pull-up

```
/* Remove D+ pull-up through ISP1301 */
OTG_I2C_TX = 0x15A; // Send ISP1301 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x201; // Clear DP_PULLUP bit, send STOP condition
```

```
/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Add D+ pull-up

```
/* Add D+ pull-up through ISP1301 */
OTG_I2C_TX = 0x15A; // Send ISP1301 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x201; // Set DP_PULLUP bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

7.2 A-device: host to peripheral HNP switching

In this case, the role of the OTG controller is host (A-device), and the A-device switches roles from host to peripheral.

The On-The-Go Supplement defines the behavior of a dual-role A-device during HNP using a state machine diagram. The OTG software stack is responsible for implementing all of the states in the Dual-Role A-Device State Diagram.

The OTG controller hardware provides support for the state transitions between a_host, a_suspend, a_wait_vfall, and a_peripheral in the Dual-Role A-Device state diagram. Setting A_HNP_TRACK in the OTGStCtrl register enables hardware support for switching the A-device from the host state to the device state. The hardware actions after setting this bit are shown in Figure 15–68.

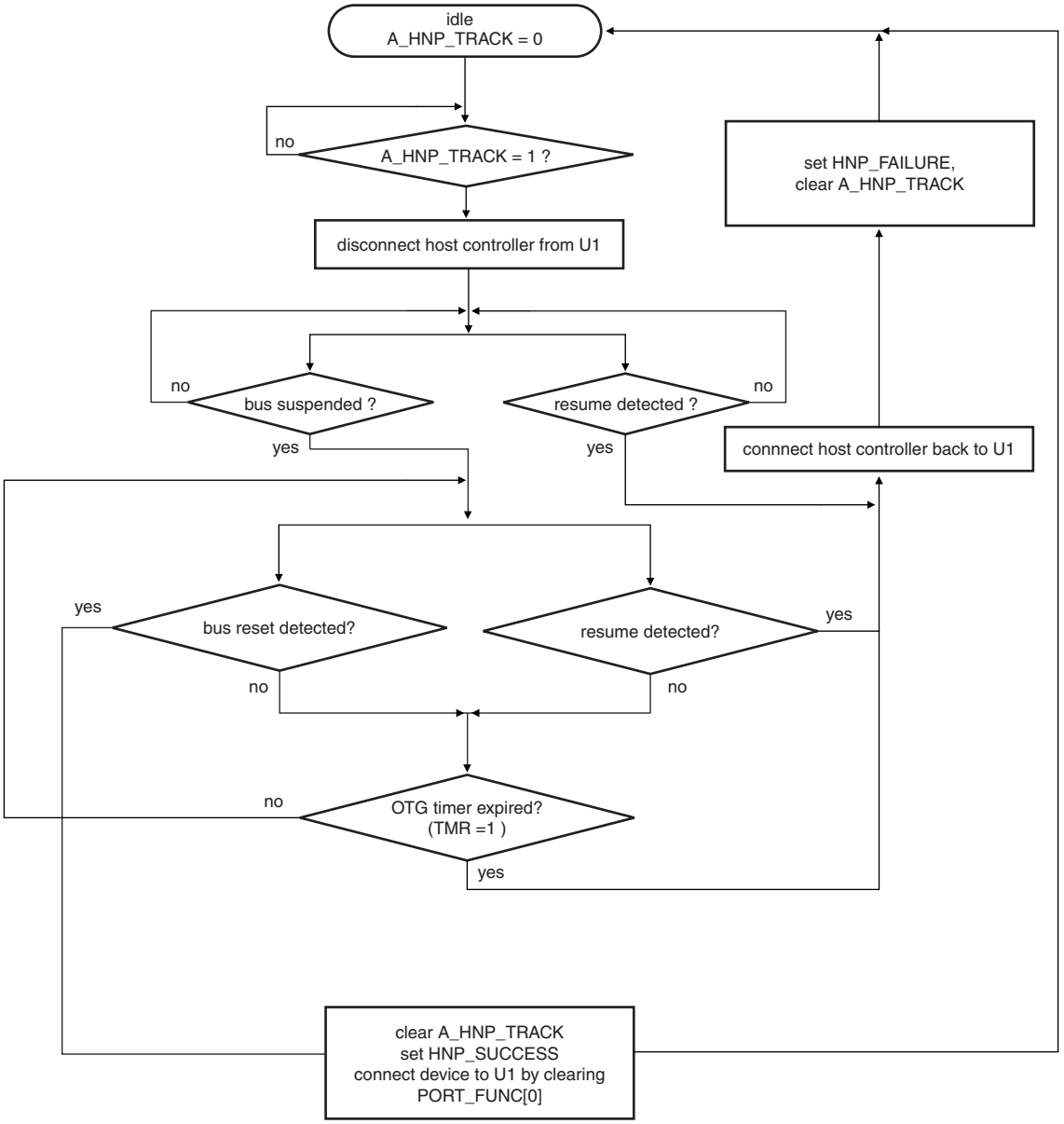


Fig 68. Hardware support for A-device switching from host state to peripheral state

Figure 15–69 shows the actions that the OTG software stack should take in response to the hardware actions setting TMR, HNP_SUCCESS, and HNP_FAILURE. The relationship of the software actions to the Dual-Role A-Device states is also shown. A-device states are shown in bold font with a circle around them.

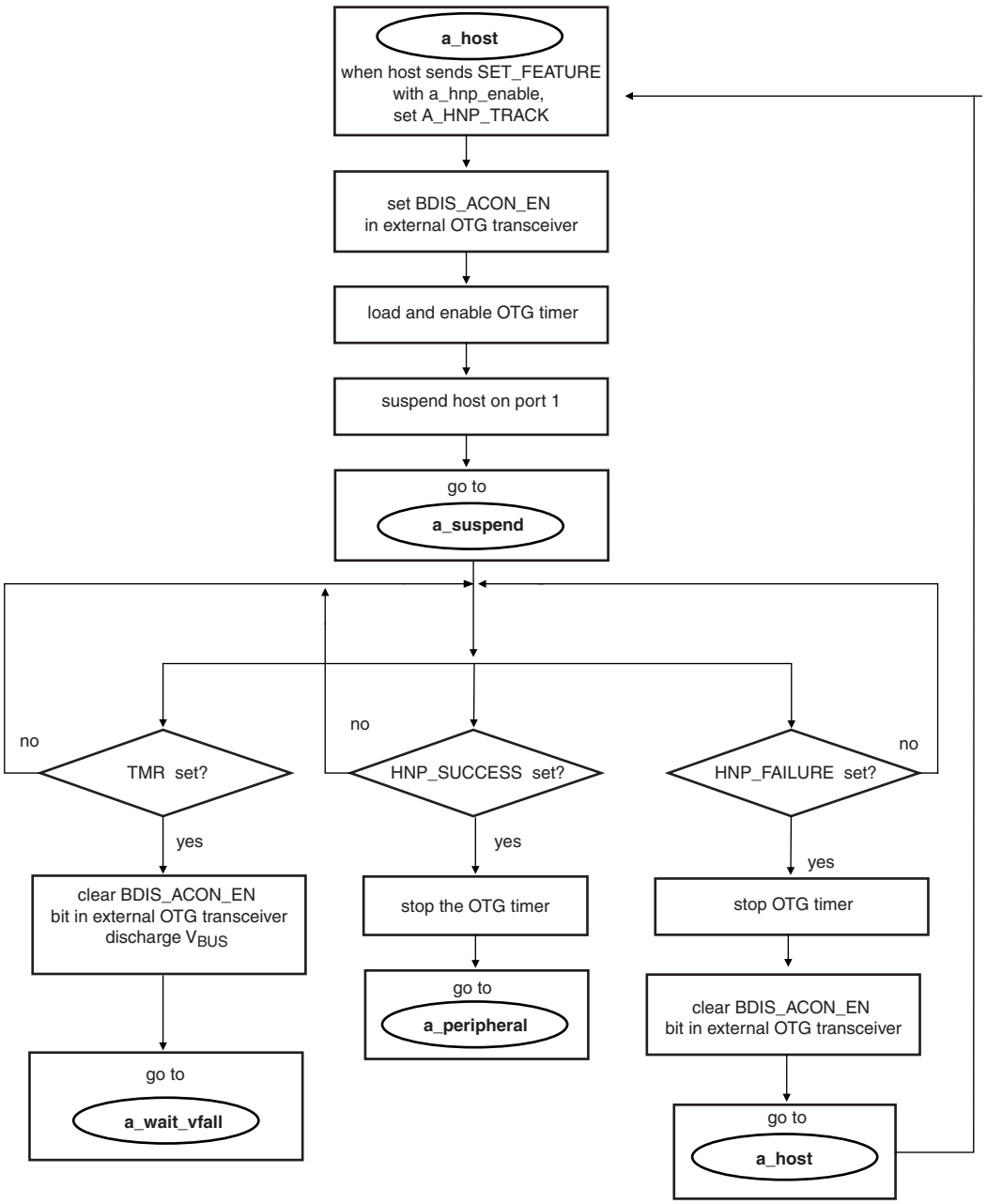


Fig 69. State transitions implemented in software during A-device switching from host to peripheral

Note that only the subset of A-device HNP states and state transitions supported by hardware are shown. Software is responsible for implementing all of the HNP states.

Figure 15–69 may appear to imply that the interrupt bits such as TMR should be polled, but this is not necessary if the corresponding interrupt is enabled.

Following are code examples that show how the actions in Figure 15–69 are accomplished. The examples assume that ISP1301 is being used as the external OTG transceiver.

Set BDIS_ACON_EN in external OTG transceiver

```
/* Set BDIS_ACON_EN in ISP1301 */
OTG_I2C_TX = 0x15A; // Send ISP1301 address, R/W=0
OTG_I2C_TX = 0x004; // Send Mode Control 1 (Set) register address
OTG_I2C_TX = 0x210; // Set BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Clear BDIS_ACON_EN in external OTG transceiver

```
/* Set BDIS_ACON_EN in ISP1301 */
OTG_I2C_TX = 0x15A; // Send ISP1301 address, R/W=0
OTG_I2C_TX = 0x005; // Send Mode Control 1 (Clear) register address
OTG_I2C_TX = 0x210; // Clear BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Discharge V_{BUS}

```
/* Clear the VBUS_DRV bit in ISP1301 */
OTG_I2C_TX = 0x15A; // Send ISP1301 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x220; // Clear VBUS_DRV bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;

/* Set the VBUS_DISCHRG bit in ISP1301 */
OTG_I2C_TX = 0x15A; // Send ISP1301 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x240; // Set VBUS_DISCHRG bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Load and enable OTG timer

```
/* The following assumes that the OTG timer has previously been  */

/* configured for a time scale of 1 ms (TMR_SCALE = “10”)        */
/* and monoshot mode (TMR_MODE = 0)                              */

/* Load the timeout value to implement the a_aidl_bdis_tmr timer */
/*     the minimum value is 200 ms                               */
OTG_TIMER = 200;

/* Enable the timer */
OTG_STAT_CTRL |= TMR_EN;
```

Stop OTG timer

```
/* Disable the timer - causes TMR_CNT to be reset to 0  */
OTG_STAT_CTRL &= ~TMR_EN;

/* Clear TMR interrupt */
OTG_INT_CLR = TMR;
```

Suspend host on port 1

```
/* Write to PortSuspendStatus bit to suspend host port 1 -               */
/* this example demonstrates the low-level action software needs to take. */
/* The host stack code where this is done will be somewhat more involved. */
HC_RH_PORT_STAT1 = PSS;
```

8. Clocking and power management

The OTG controller clocking is shown in Figure 15–70.

A clock switch controls each clock with the exception of ahb_slave_clk. When the enable of the clock switch is asserted, its clock output is turned on and its CLK_ON output is asserted. The CLK_ON signals are observable in the OTGClkSt register.

To conserve power, the clocks to the Device, Host, OTG, and I2C controllers can be disabled when not in use by clearing the respective CLK_EN bit in the OTGClkCtrl register. When the entire USB block is not in use, all of its clocks can be disabled by clearing the PCUSB bit in the PCONP register.

When software wishes to access registers in one of the controllers, it should first ensure that the respective controller’s 48 MHz clock is enabled by setting its CLK_EN bit in the OTGClkCtrl register and then poll the corresponding CLK_ON bit in OTGClkSt until set. Once set, the controller’s clock will remain enabled until CLK_EN is cleared by software. Accessing the register of a controller when its 48 MHz clock is not enabled will result in a data abort exception.

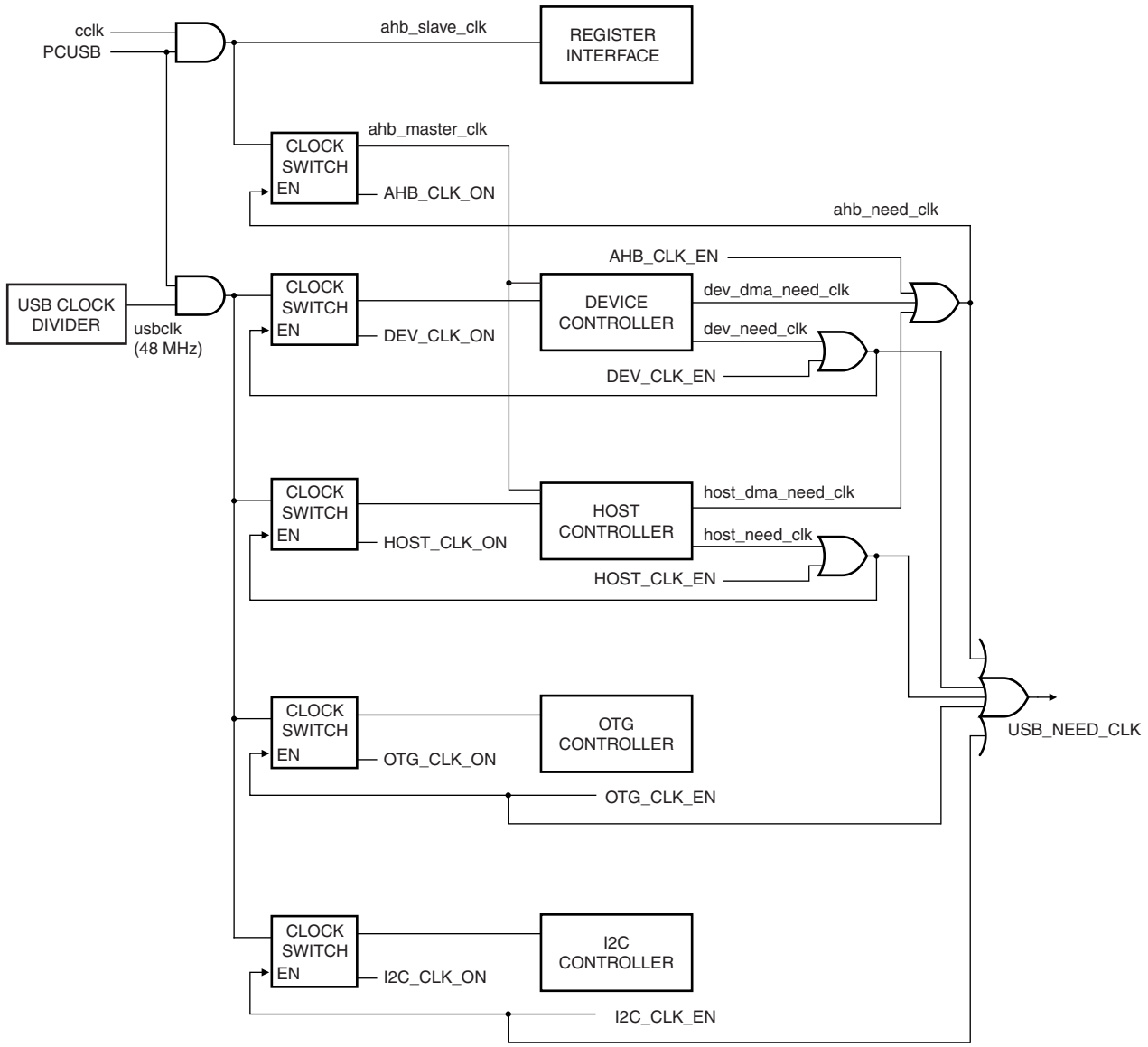


Fig 70. Clocking and power control

8.1 Device clock request signals

The Device controller has two clock request signals, dev_need_clk and dev_dma_need_clk. When asserted, these signals turn on the device's 48 MHz clock and ahb_master_clk respectively.

The dev_need_clk signal is asserted while the device is not in the suspend state, or if the device is in the suspend state and activity is detected on the USB bus. The dev_need_clk signal is de-asserted if a disconnect is detected (CON bit is cleared in the SIE Get Device Status register – Section 13–10). This signal allows DEV_CLK_EN to be cleared during normal operation when software does not need to access the Device controller registers – the Device will continue to function normally and automatically shut off its clock when it is suspended or disconnected.

The dev_dma_need_clk signal is asserted on any Device controller DMA access to memory. Once asserted, it remains active for 2 ms (2 frames), to help assure that DMA throughput is not affected by any latency associated with re-enabling ahb_master_clk. 2 ms after the last DMA access, dev_dma_need_clk is de-asserted to help conserve power. This signal allows AHB_CLK_EN to be cleared during normal operation.

8.1.1 Host clock request signals

The Host controller has two clock request signals, host_need_clk and host_dma_need_clk. When asserted, these signals turn on the host's 48 MHz clock and ahb_master_clk respectively.

The host_need_clk signal is asserted while the Host controller functional state is not UsbSuspend, or if the functional state is UsbSuspend and resume signaling or a disconnect is detected on the USB bus. This signal allows HOST_CLK_EN to be cleared during normal operation when software does not need to access the Host controller registers – the Host will continue to function normally and automatically shut off its clock when it goes into the UsbSuspend state.

The host_dma_need_clk signal is asserted on any Host controller DMA access to memory. Once asserted, it remains active for 2 ms (2 frames), to help assure that DMA throughput is not affected by any latency associated with re-enabling ahb_master_clk. 2 ms after the last DMA access, host_dma_need_clk is de-asserted to help conserve power. This signal allows AHB_CLK_EN to be cleared during normal operation.

8.2 Power-down mode support

The LPC2400 can be configured to wake up from Power Down mode on any USB bus activity. When the USBWAKE bit is set in the INTWAKE register, the assertion of the USB_NEED_CLK signal causes the chip to wake up from Power Down.

Before Power Down mode can be entered when USBWAKE is set, USB_NEED_CLK must be de-asserted. This is accomplished by clearing all of the CLK_EN bits in OTGClkCtrl and putting the Host controller into the UsbSuspend functional state. If it is necessary to wait for either of the dma_need_clk signals or the dev_need_clk to be de-asserted, the status of USB_NEED_CLK can be polled in the USBIntSt register to determine when they have all been de-asserted.

9. USB OTG controller initialization

The LPC2400 OTG device controller initialization includes the following steps:

1. Enable the device controller by setting the PCUSB bit of PCONP.
2. Configure and enable the PLL and Clock Dividers to provide 48 MHz for usbclk, and the desired frequency for cclk. For correct operation of synchronization logic in the device controller, the minimum cclk frequency is 18 MHz. For the procedure for determining the PLL setting and configuration, see Section 4–5.12 "Procedure for determining PLL settings".
3. Enable the desired controller clocks by setting their respective CLK_EN bits in the USBClkCtrl register. Poll the corresponding CLK_ON bits in the USBClkSt register until they are set.

4. Enable the desired USB pin functions by writing to the corresponding PINSEL registers.
5. Follow the appropriate steps in Section 13–11 “USB device controller initialization” to initialize the device controller.
6. Follow the guidelines given in the OpenHCI specification for initializing the host controller.

1. Features

- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- Fractional divider for baud rate control, autobaud capabilities and mechanism that enables software flow control implementation.
- In addition, UART3 includes an IrDA mode to support infrared communication.

2. Pin description

Table 335: UART0 Pin description

Pin	Type	Description
RXD0, RXD2, RXD3	Input	**Serial Input.** Serial receive data.
TXD0, TXD2, TXD3	Output	**Serial Output.** Serial transmit data.

3. Register description

Each UART contains registers as shown in Table 16–336. The Divisor Latch Access Bit (DLAB) is contained in UnLCR7 and enables access to the Divisor Latches.

Table 336. UART Register Map

Generic Name	Description	Bit functions and addresses								Access	Reset value[1]	UARTn Register Name & Address
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
RBR (DLAB= 0)	Receiver Buffer Register	8 bit Read Data								RO	NA	U0RBR - 0xE000 C000 U2RBR - 0xE007 8000 U3RBR - 0xE007 C000
THR (DLAB= 0)	Transmit Holding Register	8 bit Write Data								WO	NA	U0THR - 0xE000 C000 U2THR - 0xE007 8000 U3THR - 0xE007 C000
DLL (DLAB= 1)	Divisor Latch LSB	8 bit Data								R/W	0x01	U0DLL - 0xE000 C000 U2DLL - 0xE007 8000 U3DLL - 0xE007 C000
DLM (DLAB= 1)	Divisor Latch MSB	8 bit Data								R/W	0x00	U0DLM - 0xE000 C004 U2DLM - 0xE007 8004 U3DLM - 0xE007 C004
IER (DLAB= 0)	Interrupt Enable Register	Reserved						Enable Auto- Baud Time- Out Interrupt	Enable End of Auto- Baud Interrupt	R/W	0x00	U0IER - 0xE000 C004 U2IER - 0xE007 8004 U3IER - 0xE007 C004
		0					Enable RX Line Status Interrupt	Enable THRE Interrupt	Enable RX Data Available Interrupt			
IIR	Interrupt ID Register	Reserved						ABTOInt	ABEOint	RO	0x01	U0IIR - 0xE000 C008 U2IIR - 0xE007 8008 U3IIR - 0xE007 C008
		FIFOs Enabled		0		IIR3	IIR2	IIR1	IIR0			
FCR	FIFO Control Register	RX Trigger		Reserved			TX FIFO Reset	RX FIFO Reset	FIFO Enable	WO	0x00	U0FCR - 0xE000 C008 U2FCR - 0xE007 8008 U3FCR - 0xE007 C008
LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0x00	U0LCR - 0xE000 C00C U2LCR - 0xE007 800C U3LCR - 0xE007 C00C

Table 336. UART Register Map

Generic Name	Description	Bit functions and addresses								Access	Reset value[1]	UARTn Register Name & Address
		MSB							LSB			
LSR	Line Status Register	RX FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	U0LSR - 0xE000 C014 U2LSR - 0xE007 8014 U3LSR - 0xE007 C014
SCR	Scratch Pad Register	8 bit Data								R/W	0x00	U0SCR - 0xE000 C01C U2SCR - 0xE007 801C U3SCR - 0xE007 C01C
ACR	Auto-baud Control Register	Reserved [31:10]						ABTO IntClr	ABEO IntClr	R/W	0x00	U0ACR - 0xE000 C020 U2ACR - 0xE007 8020 U3ACR - 0xE007 C020
		Reserved [7:3]					Auto Reset	Mode	Start			
ICR	IrDA Control Register	Reserved		PulseDiv			FixPulse En	IrDAInv	IrDAEn	R/W	0	U3ICR - 0xE000 C024 (UART3 only)
FDR	Fractional Divider Register	MulVal				DivAddVal				R/W	0x10	U0FDR - 0xE000 C028 U2FDR - 0xE007 8028 U3FDR - 0xE007 C028
TER	Transmit Enable Register	TXEN	Reserved							R/W	0x80	U0TER - 0xE000 C030 U2TER - 0xE007 8030 U3TER - 0xE007 C030

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

16.3.1 UARTn Receiver Buffer Register (U0RBR - 0xE000 C000, U2RBR - 0xE007 8000, U3RBR - 0xE007 C000 when DLAB = 0, Read Only)

The UnRBR is the top byte of the UARTn Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the UnRBR. The UnRBR is always Read Only.

Since PE, FE and BI bits correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the U0LSR register, and then to read a byte from the UnRBR.

Table 337: UARTn Receiver Buffer Register (U0RBR - address 0xE000 C000, U2RBR - 0xE007 8000, U3RBR - 0E007 C000 when DLAB = 0, Read Only) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UARTn Receiver Buffer Register contains the oldest received byte in the UARTn Rx FIFO.	Undefined

3.2 UARTn Transmit Holding Register (U0THR - 0xE000 C000, U2THR - 0xE007 8000, U3THR - 0xE007 C000 when DLAB = 0, Write Only)

The UnTHR is the top byte of the UARTn TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in UnLCR must be zero in order to access the UnTHR. The UnTHR is always Write Only.

Table 338: UART0 Transmit Holding Register (U0THR - address 0xE000 C000, U2THR - 0xE007 8000, U3THR - 0xE007 C000 when DLAB = 0, Write Only) bit description

Bit	Symbol	Description	Reset Value
7:0	THR	Writing to the UARTn Transmit Holding Register causes the data to be stored in the UARTn transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA

3.3 UARTn Divisor Latch LSB Register (U0DLL - 0xE000 C000, U2DLL - 0xE007 8000, U3DLL - 0xE007 C000 when DLAB = 1) and UARTn Divisor Latch MSB Register (U0DLM - 0xE000 C004, U2DLL - 0xE007 8004, U3DLL - 0xE007 C004 when DLAB = 1)

The UARTn Divisor Latch is part of the UARTn Baud Rate Generator and holds the value used to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16× the desired baud rate. The UnDLL and UnDLM registers together form a 16 bit divisor where UnDLL contains the lower 8 bits of the divisor and UnDLM contains the

higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed. The Divisor Latch Access Bit (DLAB) in UnLCR must be one in order to access the UARTn Divisor Latches.

Table 339: UARTn Divisor Latch LSB Register (U0DLL - address 0xE000 C000, U2DLL - 0xE007 8000, U3DLL - 0xE007 C000 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.	0x01

Table 340: UARTn Divisor Latch MSB Register (U0DLM - address 0xE000 C004, U2DLM - 0xE007 8004, U3DLM - 0xE007 C004 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UARTn Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UARTn.	0x00

3.4 UARTn Interrupt Enable Register (U0IER - 0xE000 C004, U2IER - 0xE007 8004, U3IER - 0xE007 C004 when DLAB = 0)

The UnIER is used to enable the three UARTn interrupt sources.

Table 341: UARTn Interrupt Enable Register (U0IER - address 0xE000 C004, U2IER - 0xE007 8004, U3IER - 0xE007 C004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBR Interrupt Enable		UnIER[0] enables the Receive Data Available interrupt for UARTn. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THRE Interrupt Enable	0	UnIER[1] enables the THRE interrupt for UARTn. The status of this can be read from UnLSR[5]. Disable the THRE interrupts.	0
		1	Enable the THRE interrupts.	
2	RX Line Status Interrupt Enable	0	UnIER[2] enables the UARTn RX line status interrupts. The status of this interrupt can be read from UnLSR[4:1]. Disable the RX line status interrupts.	0
		1	Enable the RX line status interrupts.	
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	ABTOIntEn		U1IER8 enables the auto-baud time-out interrupt.	0
		0	Disable Auto-baud Time-out Interrupt.	
		1	Enable Auto-baud Time-out Interrupt.	
9	ABEOIntEn		U1IER9 enables the end of auto-baud interrupt.	0
		0	Disable End of Auto-baud Interrupt.	
		1	Enable End of Auto-baud Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

3.5 UARTn Interrupt Identification Register (U0IIR - 0xE000 C008, U2IIR - 0xE007 8008, U3IIR - 0x7008 C008, Read Only)

The UnIIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an UnIIR access. If an interrupt occurs during an UnIIR access, the interrupt is recorded for the next UnIIR access.

Table 342: UARTn Interrupt Identification Register (U0IIR - address 0xE000 C008, U2IIR - 0x7008 8008, U3IIR - 0x7008 C008, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	IntStatus		Interrupt status. Note that U1IIR[0] is active low. The pending interrupt can be determined by evaluating UnIIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	IntId		Interrupt identification. UnIER[3:1] identifies an interrupt corresponding to the UARTn Rx FIFO. All other combinations of UnIER[3:1] not listed above are reserved (000,100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		These bits are equivalent to UnFCR[0].	0
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Bit UnIIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 16–343. Given the status of UnIIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The UnIIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UARTn RLS interrupt (UnIIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UARTn Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UARTn Rx error condition that set the interrupt can be observed via U0LSR[4:1]. The interrupt is cleared upon an UnLSR read.

The UARTn RDA interrupt (UnIIR[3:1] = 010) shares the second level priority with the CTI interrupt (UnIIR[3:1] = 110). The RDA is activated when the UARTn Rx FIFO reaches the trigger level defined in UnFCR[7:6] and is reset when the UARTn Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (UnIIR[3:1] = 110) is a second level interrupt and is set when the UARTn Rx FIFO contains at least one character and no UARTn Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UARTn Rx FIFO activity (read or write of UARTn RSR) will clear the interrupt. This interrupt is intended to flush the UARTn RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 343: UARTn Interrupt Handling

U0IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	UnLSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (UnFCR0=1)	UnRBR Read[3] or UARTn FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	UnRBR Read[3]
0010	Third	THRE	THRE[2]	UnIIR Read (if source of interrupt) or THR write[4]

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 16–3.8 “UARTn Line Status Register (U0LSR - 0xE000 C014, U2LSR - 0xE007 8014, U3LSR - 0xE007 C014, Read Only)”

[3] For details see Section 16–16.3.1 “UARTn Receiver Buffer Register (U0RBR - 0xE000 C000, U2RBR - 0xE007 8000, U3RBR - 0xE007 C000 when DLAB = 0, Read Only)”

[4] For details see Section 16–3.5 “UARTn Interrupt Identification Register (U0IIR - 0xE000 C008, U2IIR - 0xE007 8008, U3IIR - 0x7008 C008, Read Only)” and Section 16–3.2 “UARTn Transmit Holding Register (U0THR - 0xE000 C000, U2THR - 0xE007 8000, U3THR - 0xE007 C000 when DLAB = 0, Write Only)”

The UARTn THRE interrupt (UnIIR[3:1] = 001) is a third level interrupt and is activated when the UARTn THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UARTn THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever

THRE = 1 and there have not been at least two characters in the UnTHR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to UnTHR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UARTn THR FIFO has held two or more characters at one time and currently, the UnTHR is empty. The THRE interrupt is reset when a UnTHR write occurs or a read of the UnIIR occurs and the THRE is the highest interrupt (UnIIR[3:1] = 001).

3.6 UARTn FIFO Control Register (U0FCR - 0xE000 C008, U2FCR - 0xE007 8008, U3FCR - 0xE007 C008, Write Only)

The UnFCR controls the operation of the UARTn Rx and TX FIFOs.

Table 344: UARTn FIFO Control Register (U0FCR - address 0xE000 C008, U2FCR - 0xE007 8008, U3FCR - 0xE007 C008, Write Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFO Enable	0	UARTn FIFOs are disabled. Must not be used in the application.	0
		1	Active high enable for both UARTn Rx and TX FIFOs and UnFCR[7:1] access. This bit must be set for proper UARTn operation. Any transition on this bit will automatically clear the UARTn FIFOs.	
1	RX FIFO Reset	0	No impact on either of UARTn FIFOs.	0
		1	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.	
2	TX FIFO Reset	0	No impact on either of UARTn FIFOs.	0
		1	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.	
5:3	-	0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RX Trigger Level		These two bits determine how many receiver UARTn FIFO characters must be written before an interrupt is activated.	0
		00	Trigger level 0 (1 character or 0x01)	
		01	Trigger level 1 (4 characters or 0x04)	
		10	Trigger level 2 (8 characters or 0x08)	
		11	Trigger level 3 (14 characters or 0x0E)	

3.7 UARTn Line Control Register (U0LCR - 0xE000 C00C, U2LCR - 0xE007 800C, U3LCR - 0xE007 C00C)

The UnLCR determines the format of the data character that is to be transmitted or received.

Table 345: UARTn Line Control Register (U0LCR - address 0xE000 C00C, U2LCR - 0xE007 800C, U3LCR - 0xE007 C00C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Word Length Select	00	5 bit character length	0
		01	6 bit character length	
		10	7 bit character length	
		11	8 bit character length	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if UnLCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UART0 TXD is forced to logic 0 when UnLCR[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

3.8 UARTn Line Status Register (U0LSR - 0xE000 C014, U2LSR - 0xE007 8014, U3LSR - 0xE007 C014, Read Only)

The UnLSR is a read-only register that provides status information on the UARTn TX and RX blocks.

Table 346: UARTn Line Status Register (U0LSR - address 0xE000 C014, U2LSR - 0xE007 8014, U3LSR - 0xE007 C014, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	Receiver Data Ready (RDR)		UnLSR0 is set when the UnRBR holds an unread character and is cleared when the UARTn RBR FIFO is empty.	0
		0	UnRBR is empty.	
		1	UnRBR contains valid data.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An UnLSR read clears UnLSR1. UnLSR1 is set when UARTn RSR has a new character assembled and the UARTn RBR FIFO is full. In this case, the UARTn RBR FIFO will not be overwritten and the character in the UARTn RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	

Table 346: UARTn Line Status Register (U0LSR - address 0xE000 C014, U2LSR - 0xE007 8014, U3LSR - 0xE007 C014, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An UnLSR read clears UnLSR[2]. Time of parity error detection is dependent on UnFCR[0]. **Note:** A parity error is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An UnLSR read clears UnLSR[3]. The time of the framing error detection is dependent on UnFCR0. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXDn is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXDn goes to marking state (all 1's). An UnLSR read clears this status bit. The time of break detection is dependent on UnFCR[0]. **Note:** The break interrupt is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE))		THRE is set immediately upon detection of an empty UARTn THR and is cleared on a UnTHR write.	1
		0	UnTHR contains valid data.	
		1	UnTHR is empty.	
6	Transmitter Empty (TEMT)		TEMT is set when both UnTHR and UnTSR are empty; TEMT is cleared when either the UnTSR or the UnTHR contain valid data.	1
		0	UnTHR and/or the UnTSR contains valid data.	
		1	UnTHR and the UnTSR are empty.	
7	Error in RX FIFO (RXFE)		UnLSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the UnRBR. This bit is cleared when the UnLSR register is read and there are no subsequent errors in the UARTn FIFO.	0
		0	UnRBR contains no UARTn RX errors or UnFCR[0]=0.	
		1	UARTn RBR contains at least one UARTn RX error.	

3.9 UARTn Scratch Pad Register (U0SCR - 0xE000 C01C, U2SCR - 0xE007 801C U3SCR - 0xE007 C01C)

The UnSCR has no effect on the UARTn operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the UnSCR has occurred.

Table 347: UARTn Scratch Pad Register (U0SCR - address 0xE000 C01C, U2SCR - 0xE007 801C, U3SCR - 0xE007 C01C) bit description

Bit	Symbol	Description	Reset Value
7:0	Pad	A readable, writable byte.	0x00

3.10 UARTn Auto-baud Control Register (U0ACR - 0xE000 C020, U2ACR - 0xE007 8020, U3ACR - 0xE007 C020)

The UARTn Auto-baud Control Register (UnACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 348: UARTn Auto-baud Control Register (U0ACR - 0xE000 C020, U2ACR - 0xE007 8020, U3ACR - 0xE007 C020) bit description

Bit	Symbol	Value	Description	Reset value
0	Start		This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running).Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	Mode		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AutoRestart	0	No restart.	0
		1	Restart in case of time-out (counter restarts at next UART0 Rx falling edge)	0
7:3	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOIntClr		End of auto-baud interrupt clear bit (write only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
9	ABTOIntClr		Auto-baud time-out interrupt clear bit (write only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
31:10	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

16.3.10.1 Auto-baud

The UARTn auto-baud function can be used to measure the incoming baud-rate based on the ”AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers UnDLM and UnDLL accordingly.

Auto-baud is started by setting the UnACR Start bit. Auto-baud can be stopped by clearing the UnACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the UnACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UARTn Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UARTn Rx pin (the length of the start bit).

The UnACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UARTn Rx pin.

The auto-baud function can generate two interrupts.

- The UnIIR ABTOInt interrupt will get set if the interrupt is enabled (UnIER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The UnIIR ABEOInt interrupt will get set if the interrupt is enabled (UnIER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding UnACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UARTn Rx pin baud-rate, but the value of the UnFDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to UnDLM and UnDLL registers should be done before UnACR register write. The minimum and the maximum baudrates supported by UARTn are function of pclk, number of data bits, stop bits and parity bits.

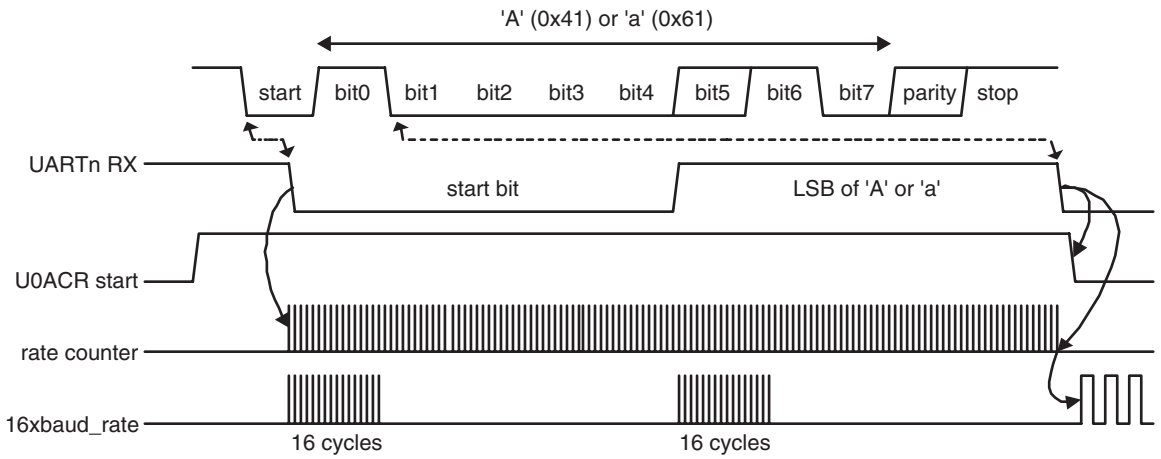
(7)

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UARTn_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

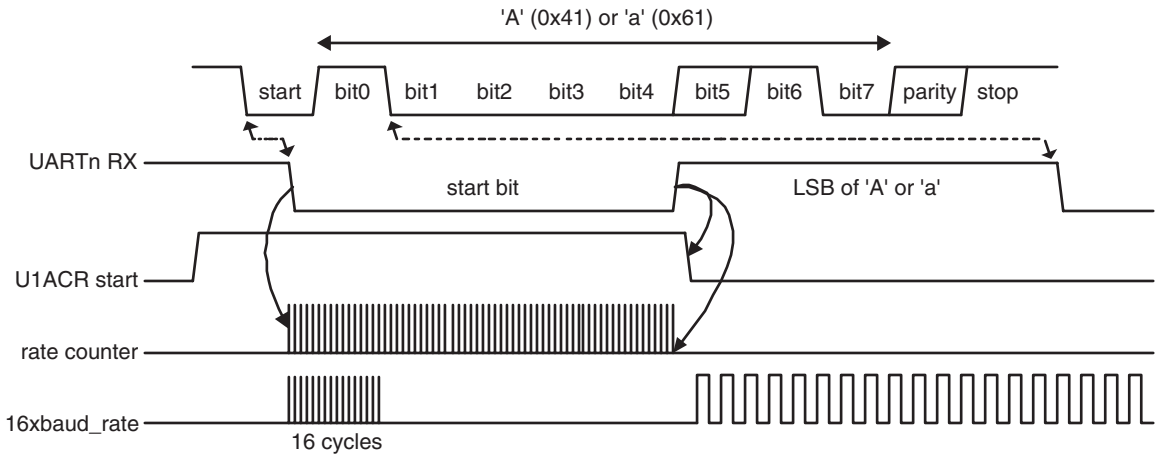
16.3.10.2 Auto-baud modes

When the software is expecting an ”AT" command, it configures the UARTn with the expected character format and sets the UnACR Start bit. The initial values in the divisor latches UnDLM and UnDLM don‘t care. Because of the ”A" or ”a" ASCII coding (”A" = 0x41, ”a" = 0x61), the UARTn Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the UnACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On UnACR Start bit setting, the baud-rate measurement counter is reset and the UARTn UnRSR is reset. The UnRSR baud rate is switch to the highest rate.
2. A falling edge on UARTn Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UARTn input clock, guaranteeing the start bit is stored in the UnRSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UARTn input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UARTn Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UARTn Rx pin.
6. The rate counter is loaded into UnDLM/UnDLL and the baud-rate will be switched to normal operation. After setting the UnDLM/UnDLL the end of auto-baud interrupt UnIIR ABEOInt will be set, if enabled. The UnRSR will now continue receiving the remaining bits of the ”A/a" character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 71. Autobaud a) mode 0 and b) mode 1 waveform

3.11 IrDA Control Register for UART3 Only (U3ICR - 0xE007 C024)

The IrDA Control Register enables and configures the IrDA mode for UART3 only. The value of U3ICR should not be changed while transmitting or receiving data, or data loss or corruption may occur.

Table 349: IrDA Control Register for UART3 only (U3ICR - address 0xE007 C024) bit description

Bit	Symbol	Value	Description	Reset value
0	IrDAEn	0	IrDA mode on UART3 is disabled, UART3 acts as a standard UART.	0
		1	IrDA mode on UART3 is enabled.	
1	IrDAInv		When 1, the serial input is inverted. This has no effect on the serial output. When 0, the serial input is not inverted.	0
2	FixPulseEn		When 1, enabled IrDA fixed pulse width mode.	0
5:3	PulseDiv		Configures the pulse when FixPulseEn = 1. See text below for details.	0
31:6	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

The PulseDiv bits in U3ICR are used to select the pulse width when the fixed pulse width mode is used in IrDA mode (IrDAEn = 1 and FixPulseEn = 1). The value of these bits should be set so that the resulting pulse width is at least 1.63 µs. Table 16–350 shows the possible pulse widths.

Table 350: IrDA Pulse Width

FixPulseEn	PulseDiv	IrDA Transmitter Pulse width (µs)
0	x	3 / (16 × baud rate)
1	0	$2 \times T_{PCLK}$
1	1	$4 \times T_{PCLK}$

Table 350: IrDA Pulse Width

FixPulseEn	PulseDiv	IrDA Transmitter Pulse width (µs)
1	2	$8 \times T_{PCLK}$
1	3	$16 \times T_{PCLK}$
1	4	$32 \times T_{PCLK}$
1	5	$64 \times T_{PCLK}$
1	6	$128 \times T_{PCLK}$
1	7	$256 \times T_{PCLK}$

3.12 UARTn Fractional Divider Register (U0FDR - 0xE000 C028, U2FDR - 0xE007 8028, U3FDR - 0xE007 C028)

The UARTn Fractional Divider Register (UnFDR) controls the clock pre-scaler for the baud rate generation and can be read and written at user's discretion.

Table 351: UARTn Fractional Divider Register (U0FDR - address 0xE000 C028, U2FDR - 0xE007 8028, U3FDR - 0xE007 C028) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UARTn baudrate.	0
7:4	MULVAL	1	Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud-rate generator is used or not.	1
31:8	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

This register controls the clock pre-scaler for the baud rate generation. The clock can be pre-scaled by a value of:

(8)

$$\frac{MULVAL}{(MULVAL + DIVADDVAL)}$$

UARTn baud-rate can be calculated as:

(9)

$$UARTn_{baudrate} = \frac{PCLK}{16 \times UnDL \times \left(1 + \frac{DIVADDVAL}{MULVAL}\right)}$$

Where PCLK is the peripheral clock, UnDL is value determined by the UnDLM and UnDLL registers (UnDL = 256 × UnDLM + UnDLL), and DIVADDVAL and MULVAL are UARTn fractional baud-rate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. 0 < MULVAL ≤ 15
2. 0 ≤ DIVADDVAL ≤ 15

If the UnFDR register value does not comply to these two requests then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled and the clock will not be divided.

Remark: f DIVADDVAL>0, UnDL must be UnDL ≥ 0x0002 or the UART will not operate at the desired baud-rate!

The value of the UnFDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

Usage Note: For practical purposes, UARTnbaudrate formula can be written in a way that identifies the part of a UART baudrate generated without the fractional baud-rate generator, and the correction factor that this module adds:

(10)

$$UARTn_{baudrate} = \frac{PCLK}{16 \times UnDL} \times \frac{MULVAL}{(MULVAL + DIVADDVAL)}$$

Based on this representation, fractional baud-rate generator contribution can also be described as a prescaling with a factor of MULVAL/(MULVAL+DIVADDVAL).

3.13 UARTn Baudrate Calculation

Example 1: Using UARTnbaudrate formula from above, it can be determined that system with pclk = 20 MHz, UnDL = 130 (UnDLM = 0x00 and UnDLL = 0x82), DIVADDVAL = 0 and MULVAL = 1 will enable UARTn with UARTnbaudrate = 9615 bauds.

Example 2: Using UARTnbaudrate formula from above, it can be determined that system with PCLK = 20 MHz, UnDL = 93 (UnDLM = 0x00 and UnDLL = 0x5D), DIVADDVAL = 2 and MULVAL = 5 will enable UARTn with UARTnbaudrate = 9600 bauds.

Additonal examples of Baud Rate Vales: Table 16–352 shows additional examples of baud rate for PCLK = 20 MHz

Table 352: Baud-rates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

Desired baud-rate	MULVAL = 0 DIVADDVAL = 0			Optimal MULVAL & DIVADDVAL		
	UnDLM:UnDLL hex[2]	UnDLM:UnDLL dec[1]	% error[3]	UnDLM:UnDLL dec[1]	Fractional pre-scaler value MULDIV / (MULDIV + DIVADDVAL)	% error[3]
50	61A8	25000	0.0000	25000	1/(1+0)	0.0000
75	411B	16667	0.0020	12500	3/(3+1)	0.0000
110	2C64	11364	0.0032	6250	11/(11+9)	0.0000
134.5	244E	9294	0.0034	3983	3/(3+4)	0.0001

Table 352: Baud-rates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

Desired baud-rate	MULVAL = 0 DIVADDVAL = 0			Optimal MULVAL & DIVADDVAL		
	UnDLM:UnDLL hex[2]	UnDLM:UnDLL dec[1]	% error[3]	UnDLM:UnDLL dec[1]	Fractional pre-scaler value MULDIV / (MULDIV + DIVADDVAL)	% error[3]
150	208D	8333	0.0040	6250	3/(3+1)	0.0000
300	1047	4167	0.0080	3125	3/(3+1)	0.0000
600	0823	2083	0.0160	1250	3/(3+2)	0.0000
1200	0412	1042	0.0320	625	3/(3+2)	0.0000
1800	02B6	694	0.0640	625	9/(9+1)	0.0000
2000	0271	625	0.0000	625	1/(1+0)	0.0000
2400	0209	521	0.0320	250	12/(12+13)	0.0000
3600	015B	347	0.0640	248	5/(5+2)	0.0064
4800	0104	260	0.1600	125	12/(12+13)	0.0000
7200	00AE	174	0.2240	124	5/(5+2)	0.0064
9600	0082	130	0.1600	93	5/(5+2)	0.0064
19200	0041	65	0.1600	31	10/(10+11)	0.0064
38400	0021	33	1.3760	12	7/(7+12)	0.0594
56000	0021	22	1.4400	13	7/(7+5)	0.0160
57600	0016	22	1.3760	19	7/(7+1)	0.0594
112000	000B	11	1.4400	6	7/(7+6)	0.1600
115200	000B	11	1.3760	4	7/(7+12)	0.0594
224000	0006	6	7.5200	3	7/(7+6)	0.1600
448000	0003	3	7.5200	2	5/(5+2)	0.3520

[1] Values in the row represent decimal equivalent of a 16 bit long content (DLM:DLL).

[2] Values in the row represent hex equivalent of a 16 bit long content (DLM:DLL).

[3] Refers to the percent error between desired and actual baud-rate.

3.14 UARTn Transmit Enable Register (U0TER - 0xE000 C030, U2TER - 0xE007 8030, U3TER - 0xE007 C030)

LPC2400's UnTER enables implementation of software flow control. When TXEn=1, UARTn transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UARTn transmission will stop.

Table 16–353 describes how to use TXEn bit in order to achieve software flow control.

Table 353: UARTn Transmit Enable Register (U0TER - address 0xE000 C030, U2TER - 0xE007 8030, U3TER - 0xE007 C030) bit description

Bit	Symbol	Description	Reset Value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit is cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software implementing software-handshaking can clear this bit when it receives an XOFF character (DC3). Software can set this bit again when it receives an XON (DC1) character.	1

4. Architecture

The architecture of the UARTs 0, 2 and 3 are shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART.

The UARTn receiver block, UnRX, monitors the serial input line, RXDn, for valid input. The UARTn RX Shift Register (UnRSR) accepts valid characters via RXDn. After a valid character is assembled in the UnRSR, it is passed to the UARTn RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UARTn transmitter block, UnTX, accepts data written by the CPU or host and buffers the data in the UARTn TX Holding Register FIFO (UnTHR). The UARTn TX Shift Register (UnTSR) reads the data stored in the UnTHR and assembles the data to transmit via the serial output pin, TXDn.

The UARTn Baud Rate Generator block, UnBRG, generates the timing enables used by the UARTn TX block. The UnBRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the UnDLL and UnDLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The interrupt interface contains registers UnIER and UnIIR. The interrupt interface receives several one clock wide enables from the UnTX and UnRX blocks.

Status information from the UnTX and UnRX is stored in the UnLSR. Control information for the UnTX and UnRX is stored in the UnLCR.

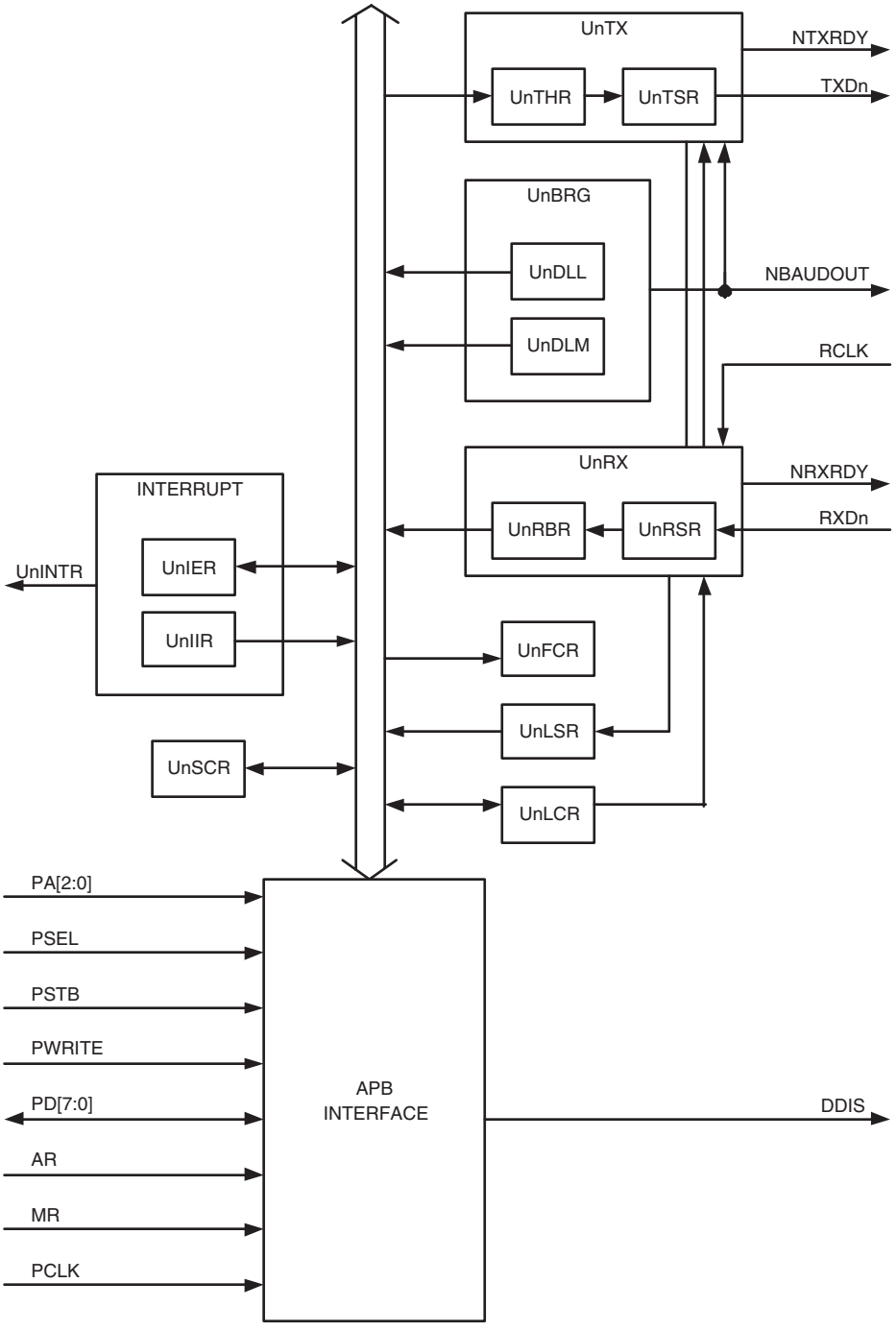


Fig 72. UART0, 2 and 3 block diagram

1. Features

- UART1 is identical to UART0/2/3, with the addition of a modem interface.
- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- Standard modem interface signals included (CTS, DCD, DTS, DTR, RI, RTS).
- LPC2468 UART1 allows for implementation of either software or hardware flow control.

2. Pin description

Table 354: UART1 Pin Description

Pin	Type	Description
RXD1	Input	**Serial Input.** Serial receive data.
TXD1	Output	**Serial Output.** Serial transmit data.
CTS1	Input	**Clear To Send.** Active low signal indicates if the external modem is ready to accept transmitted data via TXD1 from the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[4]. State change information is stored in U1MSR[0] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1). Only CTS1 is also used in auto-cts mode to control the UART1 transmitter. Clear to send. CTS1 is an asynchronous, active low modem status signal. Its condition can be checked by reading bit 4 (CTS) of the modem status register. Bit 0 (DCTS) of the Modem Status Register (MSR) indicates that CTS1 has changed states since the last read from the MSR. If the modem status interrupt is enabled when CTS1 changes levels and the auto-cts mode is not enabled, an interrupt is generated. CTS1 is also used in the auto-cts mode to control the transmitter. (IP_3106)
DCD1	Input	**Data Carrier Detect.** Active low signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged. In normal operation of the modem interface (U1MCR[4]=0), the complement value of this signal is stored in U1MSR[7]. State change information is stored in U1MSR3 and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
DSR1	Input	**Data Set Ready.** Active low signal indicates if the external modem is ready to establish a communications link with the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[5]. State change information is stored in U1MSR[1] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).

Table 354: UART1 Pin Description

Pin	Type	Description
DTR1	Output	Data Terminal Ready. Active low signal indicates that the UART1 is ready to establish connection with external modem. The complement value of this signal is stored in U1MCR[0].
RI1	Input	**Ring Indicator.** Active low signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[6]. State change information is stored in U1MSR[2] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
RTS1	Output	**Request To Send.** Active low signal indicates that the UART1 would like to transmit data to the external modem. The complement value of this signal is stored in U1MCR[1]. Only in the auto-rts mode uses RTS1 to control the transmitter FIFO threshold logic. Request to send. RTS1 is an active low signal informing the modem or data set that the UART is ready to receive data. RTS1 is set to the active (low) level by setting the RTS modem control register bit and is set to the inactive (high) level either as a result of a system reset or during loop-back mode operations or by clearing bit 1 (RTS) of the MCR. In the auto-rts mode, RTS1 is controlled by the transmitter FIFO threshold logic. (IP_3106)

3. Register description

UART1 contains registers organized as shown in Table 17–355. The Divisor Latch Access Bit (DLAB) is contained in U1LCR[7] and enables access to the Divisor Latches.

Table 355: UART1 register map

Name	Description	Bit functions and addresses								Access	Reset Value[1]	Address
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U1RBR	Receiver Buffer Register	8 bit Read Data								RO	NA	0xE001 0000 (DLAB=0)
U1THR	Transmit Holding Register	8 bit Write Data								WO	NA	0xE001 0000 (DLAB=0)
U1DLL	Divisor Latch LSB	8 bit Data								R/W	0x01	0xE001 0000 (DLAB=1)
U1DLM	Divisor Latch MSB	8 bit Data								R/W	0x00	0xE001 0004 (DLAB=1)
U1IER	Interrupt Enable Register	Reserved						Enable Autobaud Time-Out Interrupt	Enable End of Autobaud Interrupt	R/W	0x00	0xE001 0004 (DLAB=0)
		Enable CTS Interrupt		0		Enable Modem Status interrupt	Enable RX Line Status Interrupt	Enable THRE Interrupt	Enable RX Data Available Interrupt			
U1IIR	Interrupt ID Register	Reserved						ABTO Itn	ABEO int	RO	0x01	0xE001 0008
		FIFOs Enabled		0		IIR3	IIR2	IIR1	IIR0			
U1FCR	FIFO Control Register	RX Trigger		Reserved			TX FIFO Reset	RX FIFO Reset	FIFO Enable	WO	0x00	0xE001 0008
U1LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0x00	0xE001 000C
U1MCR	Modem Control Register	CTSen	RTSen	0	Loop Back	0		RTS	DTR	R/W	0x00	0xE001 0010
U1LSR	Line Status Register	RX FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE001 0014
U1MSR	Modem Status Register	DCD	RI	DSR	CTS	Delta DCD	Trailing Edge RI	Delta DSR	Delta CTS	RO	0x00	0xE001 0018

Table 355: UART1 register map *…continued*

Name	Description	Bit functions and addresses								Access	Reset Value[1]	Address
		MSB							LSB			
U1SCR	Scratch Pad Register	8 bit Data								R/W	0x00	0xE001 001C
U1ACR	Autobaud Control Register	Reserved [31:10]						ABTO IntClr	ABEO IntClr	R/W	0x00	0xE001 0020
		Reserved [7:3]					Auto Reset	Mode	Start			
U1FDR	Fractional Divider Register	Reserved [31:8]								R/W	0x10	0xE001 0028
		Mulval				DivAddVal						
U1TER	Transmit Enable Register	TXEN	Reserved							R/W	0x80	0xE001 0030

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

3.1 UART1 Receiver Buffer Register (U1RBR - 0xE001 0000, when DLAB = 0 Read Only)

The U1RBR is the top byte of the UART1 RX FIFO. The top byte of the RX FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1RBR. The U1RBR is always Read Only.

Since PE, FE and BI bits correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the U1LSR register, and then to read a byte from the U1RBR.

Table 356: UART1 Receiver Buffer Register (U1RBR - address 0xE001 0000 when DLAB = 0, Read Only) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UART1 Receiver Buffer Register contains the oldest received byte in the UART1 RX FIFO.	undefined

3.2 UART1 Transmitter Holding Register (U1THR - 0xE001 0000 when DLAB = 0, Write Only)

The U1THR is the top byte of the UART1 TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1THR. The U1THR is always Write Only.

Table 357: UART1 Transmitter Holding Register (U1THR - address 0xE001 0000 when DLAB = 0, Write Only) bit description

Bit	Symbol	Description	Reset Value
7:0	THR	Writing to the UART1 Transmit Holding Register causes the data to be stored in the UART1 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA

3.3 UART1 Divisor Latch LSB and MSB Registers (U1DLL - 0xE001 0000 and U1DLM - 0xE001 0004, when DLAB = 1)

The UART1 Divisor Latch is part of the UART1 Baud Rate Generator and holds the value used to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16x the desired baud rate (Equation 17–11). The U1DLL and U1DLM registers together form a 16 bit divisor where U1DLL contains the lower 8 bits of the divisor and U1DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U1LCR must be one in order to access the UART1 Divisor Latches. Details on how to select the right value for U1DLL and U1DLM can be found later on in this chapter.

(11)

$$UART1_{baudrate} = \frac{pclk}{16 \times (256 \times U1DLM + U1DLL)}$$

Table 358: UART1 Divisor Latch LSB Register (U1DLL - address 0xE001 0000 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UART1 Divisor Latch LSB Register, along with the U1DLM register, determines the baud rate of the UART1.	0x01

Table 359: UART1 Divisor Latch MSB Register (U1DLM - address 0xE001 0004 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UART1 Divisor Latch MSB Register, along with the U1DLL register, determines the baud rate of the UART1.	0x00

3.4 UART1 Interrupt Enable Register (U1IER - 0xE001 0004, when DLAB = 0)

The U1IER is used to enable the four UART1 interrupt sources.

Table 360: UART1 Interrupt Enable Register (U1IER - address 0xE001 0004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBR Interrupt Enable		U1IER[0] enables the Receive Data Available interrupt for UART1. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THRE Interrupt Enable		U1IER[1] enables the THRE interrupt for UART1. The status of this interrupt can be read from U1LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RX Line Interrupt Enable		U1IER[2] enables the UART1 RX line status interrupts. The status of this interrupt can be read from U1LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
3	Modem Status Interrupt Enable		U1IER[3] enables the modem interrupt. The status of this interrupt can be read from U1MSR[3:0].	0
		0	Disable the modem interrupt.	
		1	Enable the modem interrupt.	
6:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 360: UART1 Interrupt Enable Register (U1IER - address 0xE001 0004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
7	CTS Interrupt Enable		If auto-cts mode is enabled this bit enables/disables the modem status interrupt generation on a CTS1 signal transition. If auto-cts mode is disabled a CTS1 transition will generate an interrupt if Modem Status Interrupt Enable (U1IER[3]) is set. In normal operation a CTS1 signal transition will generate a Modem Status Interrupt unless the interrupt has been disabled by clearing the U1IER[3] bit in the U1IER register. In auto-cts mode a transition on the CTS1 bit will trigger an interrupt only if both the U1IER[3] and U1IER[7] bits are set.	0
		0	Disable the CTS interrupt.	
		1	Enable the CTS interrupt.	
8	ABTOIntEn		U1IER8 enables the auto-baud time-out interrupt.	0
		0	Disable Auto-baud Time-out Interrupt.	
		1	Enable Auto-baud Time-out Interrupt.	
9	ABEOIntEn		U1IER9 enables the end of auto-baud interrupt.	0
		0	Disable End of Auto-baud Interrupt.	
		1	Enable End of Auto-baud Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

3.5 UART1 Interrupt Identification Register (U1IIR - 0xE001 0008, Read Only)

The U1IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U1IIR access. If an interrupt occurs during an U1IIR access, the interrupt is recorded for the next U1IIR access.

Table 361: UART1 Interrupt Identification Register (U1IIR - address 0xE001 0008, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	IntStatus		Interrupt status. Note that U1IIR[0] is active low. The pending interrupt can be determined by evaluating U1IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	

Table 361: UART1 Interrupt Identification Register (U1IIR - address 0xE001 0008, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
3:1	IntId		Interrupt identification. U1IER[3:1] identifies an interrupt corresponding to the UART1 Rx FIFO. All other combinations of U1IER[3:1] not listed above are reserved (100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt.	
		000	4 - Modem Interrupt.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		These bits are equivalent to U1FCR[0].	0
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Bit U1IIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 17–362. Given the status of U1IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U1IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART1 RLS interrupt (U1IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART1RX input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART1 Rx error condition that set the interrupt can be observed via U1LSR[4:1]. The interrupt is cleared upon an U1LSR read.

The UART1 RDA interrupt (U1IIR[3:1] = 010) shares the second level priority with the CTI interrupt (U1IIR[3:1] = 110). The RDA is activated when the UART1 Rx FIFO reaches the trigger level defined in U1FCR7:6 and is reset when the UART1 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (U1IIR[3:1] = 110) is a second level interrupt and is set when the UART1 Rx FIFO contains at least one character and no UART1 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART1 Rx FIFO activity (read or write of UART1 RSR) will clear the interrupt. This interrupt is intended to flush the UART1 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral

wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 362: UART1 Interrupt Handling

U1IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	U1LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U1FCR0=1)	U1RBR Read[3] or UART1 FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the RX FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	U1RBR Read[3]
0010	Third	THRE	THRE[2]	U1IIR Read[4] (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 17–3.10 “UART1 Line Status Register (U1LSR - 0xE001 0014, Read Only)”

[3] For details see Section 17–3.1 “UART1 Receiver Buffer Register (U1RBR - 0xE001 0000, when DLAB = 0 Read Only)”

[4] For details see Section 17–3.5 “UART1 Interrupt Identification Register (U1IIR - 0xE001 0008, Read Only)” and Section 17–3.2 “UART1 Transmitter Holding Register (U1THR - 0xE001 0000 when DLAB = 0, Write Only)”

The UART1 THRE interrupt (U1IIR[3:1] = 001) is a third level interrupt and is activated when the UART1 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART1 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the U1THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U1THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART1 THR FIFO has held two or more characters at one time and currently, the U1THR is empty. The THRE interrupt is reset when a U1THR write occurs or a read of the U1IIR occurs and the THRE is the highest interrupt (U1IIR[3:1] = 001).

It is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a low to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining U1MSR[3:0]. A U1MSR read will clear the modem interrupt.

3.6 UART1 FIFO Control Register (U1FCR - 0xE001 0008, Write Only)

The U1FCR controls the operation of the UART1 RX and TX FIFOs.

Table 363: UART1 FIFO Control Register (U1FCR - address 0xE001 0008, Write Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFO Enable	0	UART1 FIFOs are disabled. Must not be used in the application.	0
		1	Active high enable for both UART1 Rx and TX FIFOs and U1FCR[7:1] access. This bit must be set for proper UART1 operation. Any transition on this bit will automatically clear the UART1 FIFOs.	
1	RX FIFO Reset	0	No impact on either of UART1 FIFOs.	0
		1	Writing a logic 1 to U1FCR[1] will clear all bytes in UART1 Rx FIFO and reset the pointer logic. This bit is self-clearing.	
2	TX FIFO Reset	0	No impact on either of UART1 FIFOs.	0
		1	Writing a logic 1 to U1FCR[2] will clear all bytes in UART1 TX FIFO and reset the pointer logic. This bit is self-clearing.	
5:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RX Trigger Level		These two bits determine how many receiver UART1 FIFO characters must be written before an interrupt is activated.	0
		00	Trigger level 0 (1 character or 0x01).	
		01	Trigger level 1 (4 characters or 0x04).	
		10	Trigger level 2 (8 characters or 0x08).	
		11	Trigger level 3 (14 characters or 0x0E).	

3.7 UART1 Line Control Register (U1LCR - 0xE001 000C)

The U1LCR determines the format of the data character that is to be transmitted or received.

Table 364: UART1 Line Control Register (U1LCR - address 0xE001 000C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Word Length Select	00	5 bit character length.	0
		01	6 bit character length.	
		10	7 bit character length.	
		11	8 bit character length.	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if U1LCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	

Table 364: UART1 Line Control Register (U1LCR - address 0xE001 000C) bit description

Bit	Symbol	Value	Description	Reset Value
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UART1 TXD is forced to logic 0 when U1LCR[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

3.8 UART1 Modem Control Register (U1MCR - 0xE001 0010)

The U1MCR enables the modem loopback mode and controls the modem output signals.

Table 365: UART1 Modem Control Register (U1MCR - address 0xE001 0010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTR Control		Source for modem output pin, DTR. This bit reads as 0 when modem loopback mode is active.	0
1	RTS Control		Source for modem output pin RTS. This bit reads as 0 when modem loopback mode is active.	0
3-2	-	NA	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		The modem loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RXD1, has no effect on loopback and output pin, TXD1 is held in marking state. The 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 U1MSR will be driven by the lower four bits of the U1MCR 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 U1MCR.	0
		0	Disable modem loopback mode.	
		1	Enable modem loopback mode.	
5	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

Table 365: UART1 Modem Control Register (U1MCR - address 0xE001 0010) bit description

Bit	Symbol	Value	Description	Reset value
6	RTSen	0	Disable auto-rts flow control.	0
		1	Enable auto-rts flow control.	
7	CTSen	0	Disable auto-cts flow control.	0
		1	Enable auto-cts flow control.	

3.9 Auto-Flow control

If auto-RTS mode is enabled the UART1‘s receiver FIFO hardware controls the RTS1 output of the UART1. If the auto-cts mode is enabled the UART1‘s U1TSR hardware will only start transmitting if the CTS1 input signal is asserted.

17.3.9.1 Auto-RTS

The Auto-RTS function is enabled by setting the CTSen bit. Auto-RTS data flow control originates in the U1RBR module and is linked to the programmed receiver FIFO trigger level. If auto-rts is enabled, when the receiver FIFO level reaches the programmed trigger level RTS1 is deasserted (to a high value). The sending UART may send an additional byte after the trigger level is reached (assuming the sending UART has another byte to send) because it may not recognize the deassertion of RTS1 until after it has begun sending the additional byte. RTS1 is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The reassertion of RTS1 signals the sending UART to continue transmitting data.

If Auto-RTS mode is disabled the RTSen bit controls the RTS1 output of the UART1. If Auto-RTS mode is enabled hardware controls the RTS1 output and the actual value of RTS1 will be copied in the RTSen bit of the UART1. As long as Auto-RTS is enabled the value if the RTSen bit is read-only for software.

Example: Suppose the UART1 operating in type 550 has trigger level in U1FCR set to 0x2 then if Auto-RTS is enabled the UART1 will deassert the RTS1 output as soon as the receive FIFO contains 8 bytes (Table 17–363 on page 397). The RTS1 output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

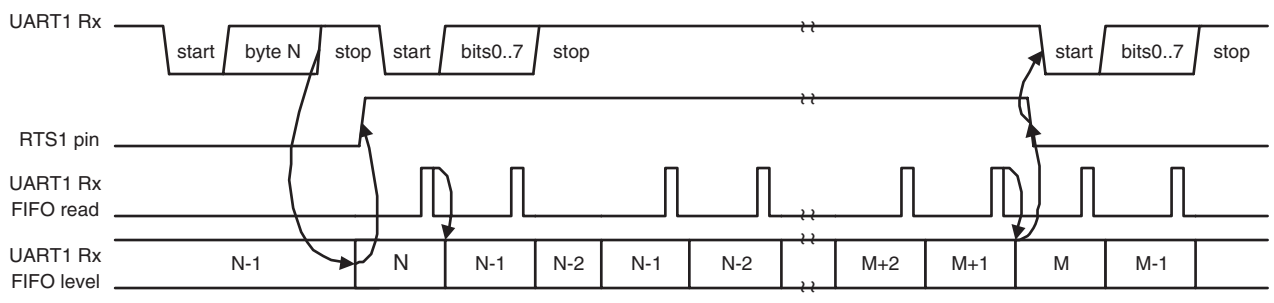


Fig 73. Auto-RTS Functional Timing

17.3.9.2 Auto-CTS

The Auto-CTS function is enabled by setting the CTSen bit. If Auto-CTS is enabled the transmitter circuitry in the U1TSR module checks CTS1 input before sending the next data byte. When CTS1 is active (low), the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS1 must be released before the middle of the last stop bit that is currently being sent. In Auto-CTS mode a change of the CTS1 signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, Delta CTS bit in the U1MSR will be set though. Table 17–366 lists the conditions for generating a Modem Status interrupt.

Table 366: Modem status interrupt generation

Enable Modem Status Interrupt (U1ER[3])	CTSen (U1MCR[7])	CTS Interrupt Enable (U1IER[7])	Delta CTS (U1MSR[0])	Delta DCD or Trailing Edge RI or Delta DSR (U1MSR[3] or U1MSR[2] or U1MSR[1])	Modem Status Interrupt
0	x	x	x	x	No
1	0	x	0	0	No
1	0	x	1	x	Yes
1	0	x	x	1	Yes
1	1	0	x	0	No
1	1	0	x	1	Yes
1	1	1	0	0	No
1	1	1	1	x	Yes
1	1	1	x	1	Yes

The auto-CTS function reduces interrupts to the host system. When flow control is enabled, a CTS1 state change does not trigger host interrupts because the device automatically controls its own transmitter. Without Auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. Figure 17–74 illustrates the Auto-CTS functional timing.

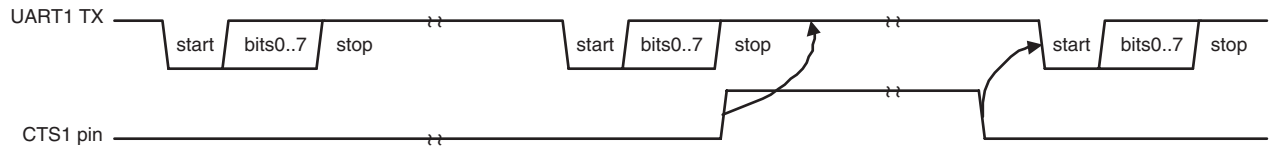


Fig 74. Auto-CTS Functional Timing

While starting transmission of the initial character the CTS1 signal is asserted. Transmission will stall as soon as the pending transmission has completed. The UART will continue transmitting a 1 bit as long as CTS1 is deasserted (high). As soon as CTS1 gets deasserted transmission resumes and a start bit is sent followed by the data bits of the next character.

3.10 UART1 Line Status Register (U1LSR - 0xE001 0014, Read Only)

The U1LSR is a read-only register that provides status information on the UART1 TX and RX blocks.

Table 367: UART1 Line Status Register (U1LSR - address 0xE001 0014, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	Receiver Data Ready (RDR)		U1LSR[0] is set when the U1RBR holds an unread character and is cleared when the UART1 RBR FIFO is empty.	0
		0	U1RBR is empty.	
		1	U1RBR contains valid data.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An U1LSR read clears U1LSR[1]. U1LSR[1] is set when UART1 RSR has a new character assembled and the UART1 RBR FIFO is full. In this case, the UART1 RBR FIFO will not be overwritten and the character in the UART1 RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An U1LSR read clears U1LSR[2]. Time of parity error detection is dependent on U1FCR[0]. **Note:** A parity error is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An U1LSR read clears U1LSR[3]. The time of the framing error detection is dependent on U1FCR0. Upon detection of a framing error, the RX will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXD1 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all 1's). An U1LSR read clears this status bit. The time of break detection is dependent on U1FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	

Table 367: UART1 Line Status Register (U1LSR - address 0xE001 0014, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
5	Transmitter Holding Register Empty (THRE)		THRE is set immediately upon detection of an empty UART1 THR and is cleared on a U1THR write.	1
		0	U1THR contains valid data.	
		1	U1THR is empty.	
6	Transmitter Empty (TEMT)		TEMT is set when both U1THR and U1TSR are empty; TEMT is cleared when either the U1TSR or the U1THR contain valid data.	1
		0	U1THR and/or the U1TSR contains valid data.	
		1	U1THR and the U1TSR are empty.	
7	Error in RX FIFO (RXFE)		U1LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the U1RBR. This bit is cleared when the U1LSR register is read and there are no subsequent errors in the UART1 FIFO.	0
		0	U1RBR contains no UART1 RX errors or U1FCR[0]=0.	
		1	UART1 RBR contains at least one UART1 RX error.	

3.11 UART1 Modem Status Register (U1MSR - 0xE001 0018)

The U1MSR is a read-only register that provides status information on the modem input signals. U1MSR[3:0] is cleared on U1MSR read. Note that modem signals have no direct affect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 368: UART1 Modem Status Register (U1MSR - address 0xE001 0018) bit description

Bit	Symbol	Value	Description	Reset Value
0	Delta CTS		Set upon state change of input CTS. Cleared on an U1MSR read.	0
		0	No change detected on modem input, CTS.	
		1	State change detected on modem input, CTS.	
1	Delta DSR		Set upon state change of input DSR. Cleared on an U1MSR read.	0
		0	No change detected on modem input, DSR.	
		1	State change detected on modem input, DSR.	
2	Trailing Edge RI		Set upon low to high transition of input RI. Cleared on an U1MSR read.	0
		0	No change detected on modem input, RI.	
		1	Low-to-high transition detected on RI.	
3	Delta DCD		Set upon state change of input DCD. Cleared on an U1MSR read.	0
		0	No change detected on modem input, DCD.	
		1	State change detected on modem input, DCD.	
4	CTS		Clear To Send State. Complement of input signal CTS. This bit is connected to U1MCR[1] in modem loopback mode.	0

Table 368: UART1 Modem Status Register (U1MSR - address 0xE001 0018) bit description

Bit	Symbol	Value	Description	Reset Value
5	DSR		Data Set Ready State. Complement of input signal DSR. This bit is connected to U1MCR[0] in modem loopback mode.	0
6	RI		Ring Indicator State. Complement of input RI. This bit is connected to U1MCR[2] in modem loopback mode.	0
7	DCD		Data Carrier Detect State. Complement of input DCD. This bit is connected to U1MCR[3] in modem loopback mode.	0

3.12 UART1 Scratch Pad Register (U1SCR - 0xE001 001C)

The U1SCR has no effect on the UART1 operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the U1SCR has occurred.

Table 369: UART1 Scratch Pad Register (U1SCR - address 0xE001 0014) bit description

Bit	Symbol	Description	Reset Value
7:0	Pad	A readable, writable byte.	0x00

3.13 UART1 Auto-baud Control Register (U1ACR - 0xE001 0020)

The UART1 Auto-baud Control Register (U1ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 370: Auto-baud Control Register (U1ACR - address 0xE001 0020) bit description

Bit	Symbol	Value	Description	Reset value
0	Start		This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running).Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	Mode		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AutoRestart	0	No restart	0
		1	Restart in case of time-out (counter restarts at next UART1 Rx falling edge)	0
7:3	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOIntClr		End of auto-baud interrupt clear bit (write only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the U1IIR.	

Table 370: Auto-baud Control Register (U1ACR - address 0xE001 0020) bit description

Bit	Symbol	Value	Description	Reset value
9	ABTOIntClr		Auto-baud time-out interrupt clear bit (write only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the U1IIR.	
31:10	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

3.14 Auto-baud

The UART1 auto-baud function can be used to measure the incoming baud-rate based on the "AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers U1DLM and U1DLL accordingly.

Auto-baud is started by setting the U1ACR Start bit. Auto-baud can be stopped by clearing the U1ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the U1ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART1 Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UART1 Rx pin (the length of the start bit).

The U1ACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART1 Rx pin.

The auto-baud function can generate two interrupts.

- The U1IIR ABTOInt interrupt will get set if the interrupt is enabled (U1IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U1IIR ABEOInt interrupt will get set if the interrupt is enabled (U1IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U1ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART1 Rx pin baud-rate, but the value of the U1FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U1DLM and U1DLL registers should be done before U1ACR register write. The minimum and the maximum baudrates supported by UART1 are function of pclk, number of data bits, stop bits and parity bits.

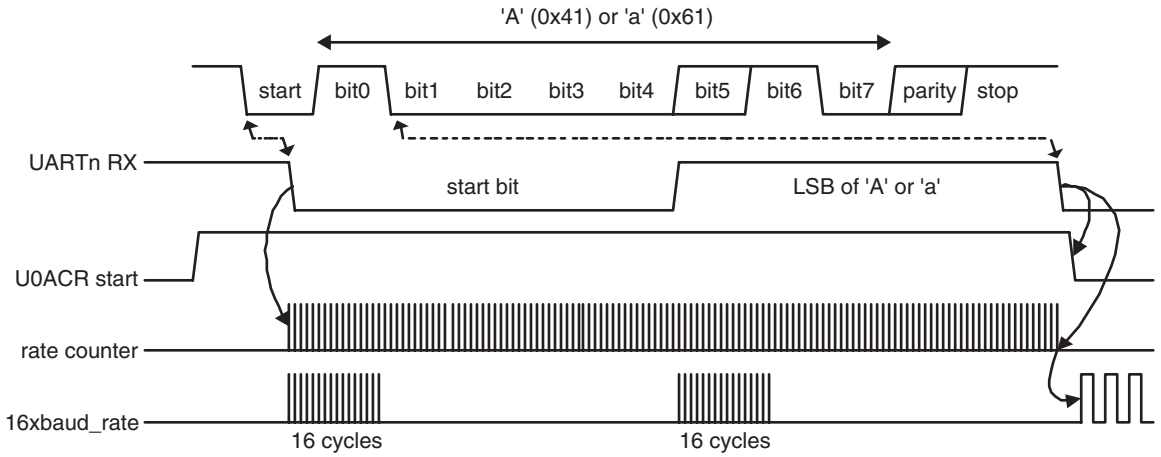
(12)

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART1_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

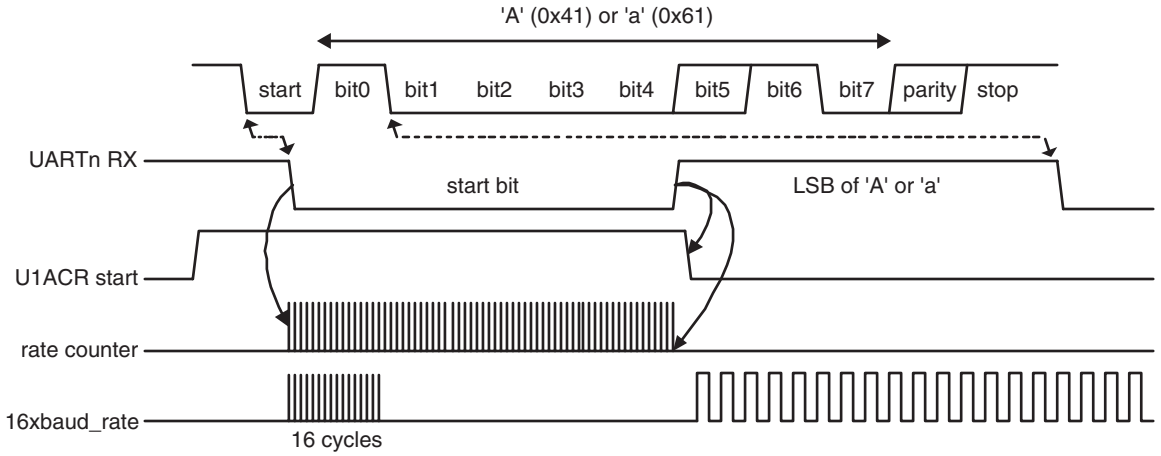
3.15 Auto-baud modes

When the software is expecting an ”AT" command, it configures the UART1 with the expected character format and sets the U1ACR Start bit. The initial values in the divisor latches U1DLM and U1DLM don‘t care. Because of the ”A" or ”a" ASCII coding (”A" = 0x41, ”a" = 0x61), the UART1 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U1ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U1ACR Start bit setting, the baud-rate measurement counter is reset and the UART1 U1RSR is reset. The U1RSR baud rate is switch to the highest rate.
2. A falling edge on UART1 Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UART1 input clock, guaranteeing the start bit is stored in the U1RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART1 input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART1 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART1 Rx pin.
6. The rate counter is loaded into U1DLM/U1DLL and the baud-rate will be switched to normal operation. After setting the U1DLM/U1DLL the end of auto-baud interrupt U1IIR ABEOInt will be set, if enabled. The U1RSR will now continue receiving the remaining bits of the ”A/a" character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 75. Auto-baud a) mode 0 and b) mode 1 waveform

3.16 UART1 Fractional Divider Register (U1FDR - 0xE001 0028)

The UART1 Fractional Divider Register (U1FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at user's discretion.

Table 371: UART1 Fractional Divider Register (U1FDR - address 0xE001 C028) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UART1 baudrate.	0
7:4	MULVAL	1	Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UART1 to operate properly, regardless of whether the fractional baud-rate generator is used or not.	1
31:8	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

This register controls the clock pre-scaler for the baud rate generation.

UART1 baud-rate can be calculated as:

(13)

$$UART1_{baudrate} = \frac{PCLK}{16 \times U1DL \times \left(1 + \frac{DIVADDVAL}{MULVAL}\right)}$$

Where PCLK is the peripheral clock, U1DL is value determined by the U1DLM and U1DLL registers, and DIVADDVAL and MULVAL are UART1 fractional baud-rate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $0 < MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 15$

If the U1FDR register value does not comply to these two requests then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled and the clock will not be divided.

Remark: if DIVADDVAL>0, UnDL must be UnDL ≥ 0x0002, or the UART will not operate at the desired baud-rate!

The value of the U1FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

Usage Note: For practical purposes, the UART0 baudrate formula can be written in a way that identifies the part of a UART baudrate generated without the fractional baud-rate generator, and the correction factor that this module adds:

(14)

$$UART1_{baudrate} = \frac{PCLK}{16 \times U1DL} \times \frac{MULVAL}{(MULVAL + DIVADDVAL)}$$

Based on this representation, fractional baud-rate generator contribution can also be described as a prescaling with a factor of MULVAL/(MULVAL+DIVADDVAL).

3.17 UART1 Baudrate Calculation

Example 1: Using UART1baudrate formula from above, it can be determined that system with PCLK = 20 MHz, UnDL = 130 (U1DLM = 0x00 and U1DLL = 0x82), DIVADDVAL = 0 and MULVAL = 1 will enable UART1 with UART1baudrate = 9615 bauds.

Example 2: Using UART1baudrate formula from above, it can be determined that system with PCLK = 20 MHz, U1DL = 93 (U1DLM = 0x00 and U1DLL = 0x5D), DIVADDVAL = 2 and MULVAL = 5 will enable UART1 with UART1baudrate = 9600 bauds.

Table 372: Baud-rates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

Desired baud-rate	MULVAL = 0 DIVADDVAL = 0			Optimal MULVAL & DIVADDVAL		
	U1DLM:U1DLL hex[2]	U1DLM:U1DLL dec[1]	% error[3]	U1DLM:U1DLL dec[1]	Fractional pre-scaler value MULDIV / (MULDIV + DIVADDVAL)	% error[3]
50	61A8	25000	0.0000	25000	1/(1+0)	0.0000
75	411B	16667	0.0020	12500	3/(3+1)	0.0000
110	2C64	11364	0.0032	6250	11/(11+9)	0.0000
134.5	244E	9294	0.0034	3983	3/(3+4)	0.0001
150	208D	8333	0.0040	6250	3/(3+1)	0.0000
300	1047	4167	0.0080	3125	3/(3+1)	0.0000
600	0823	2083	0.0160	1250	3/(3+2)	0.0000
1200	0412	1042	0.0320	625	3/(3+2)	0.0000
1800	02B6	694	0.0640	625	9/(9+1)	0.0000
2000	0271	625	0.0000	625	1/(1+0)	0.0000
2400	0209	521	0.0320	250	12/(12+13)	0.0000
3600	015B	347	0.0640	248	5/(5+2)	0.0064
4800	0104	260	0.1600	125	12/(12+13)	0.0000
7200	00AE	174	0.2240	124	5/(5+2)	0.0064
9600	0082	130	0.1600	93	5/(5+2)	0.0064
19200	0041	65	0.1600	31	10/(10+11)	0.0064
38400	0021	33	1.3760	12	7/(7+12)	0.0594
56000	0021	22	1.4400	13	7/(7+5)	0.0160
57600	0016	22	1.3760	19	7/(7+1)	0.0594
112000	000B	11	1.4400	6	7/(7+6)	0.1600
115200	000B	11	1.3760	4	7/(7+12)	0.0594
224000	0006	6	7.5200	3	7/(7+6)	0.1600
448000	0003	3	7.5200	2	5/(5+2)	0.3520

[1] Values in the row represent decimal equivalent of a 16 bit long content (DLM:DLL).

[2] Values in the row represent hex equivalent of a 16 bit long content (DLM:DLL).

[3] Refers to the percent error between desired and actual baud-rate.

3.18 UART1 Transmit Enable Register (U1TER - 0xE001 0030)

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), U1TER enables implementation of software flow control, too. When TxEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TxEn becomes 0, UART1 transmission will stop.

Although Table 17–373 describes how to use TxEn bit in order to achieve hardware flow control, it is strongly suggested to let UART1 hardware implemented auto flow control features take care of this, and limit the scope of TxEn to software flow control.

LPC2400's U1TER enables implementation of software and hardware flow control. When TXEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART1 transmission will stop.

Table 17–373 describes how to use TXEn bit in order to achieve software flow control.

Table 373: UART1 Transmit Enable Register (U1TER - address 0xE001 0030) bit description

Bit	Symbol	Description	Reset Value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software can clear this bit when it detects that the a hardware-handshaking TX-permit signal (CTS) has gone false, or with software handshaking, when it receives an XOFF character (DC3). Software can set this bit again when it detects that the TX-permit signal has gone true, or when it receives an XON (DC1) character.	1

4. Architecture

The architecture of the UART1 is shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART1.

The UART1 receiver block, U1RX, monitors the serial input line, RXD1, for valid input. The UART1 RX Shift Register (U1RSR) accepts valid characters via RXD1. After a valid character is assembled in the U1RSR, it is passed to the UART1 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART1 transmitter block, U1TX, accepts data written by the CPU or host and buffers the data in the UART1 TX Holding Register FIFO (U1THR). The UART1 TX Shift Register (U1TSR) reads the data stored in the U1THR and assembles the data to transmit via the serial output pin, TXD1.

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 TX block. The U1BRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the U1DLL and U1DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The modem interface contains registers U1MCR and U1MSR. This interface is responsible for handshaking between a modem peripheral and the UART1.

The interrupt interface contains registers U1IER and U1IIR. The interrupt interface receives several one clock wide enables from the U1TX and U1RX blocks.

Status information from the U1TX and U1RX is stored in the U1LSR. Control information for the U1TX and U1RX is stored in the U1LCR.

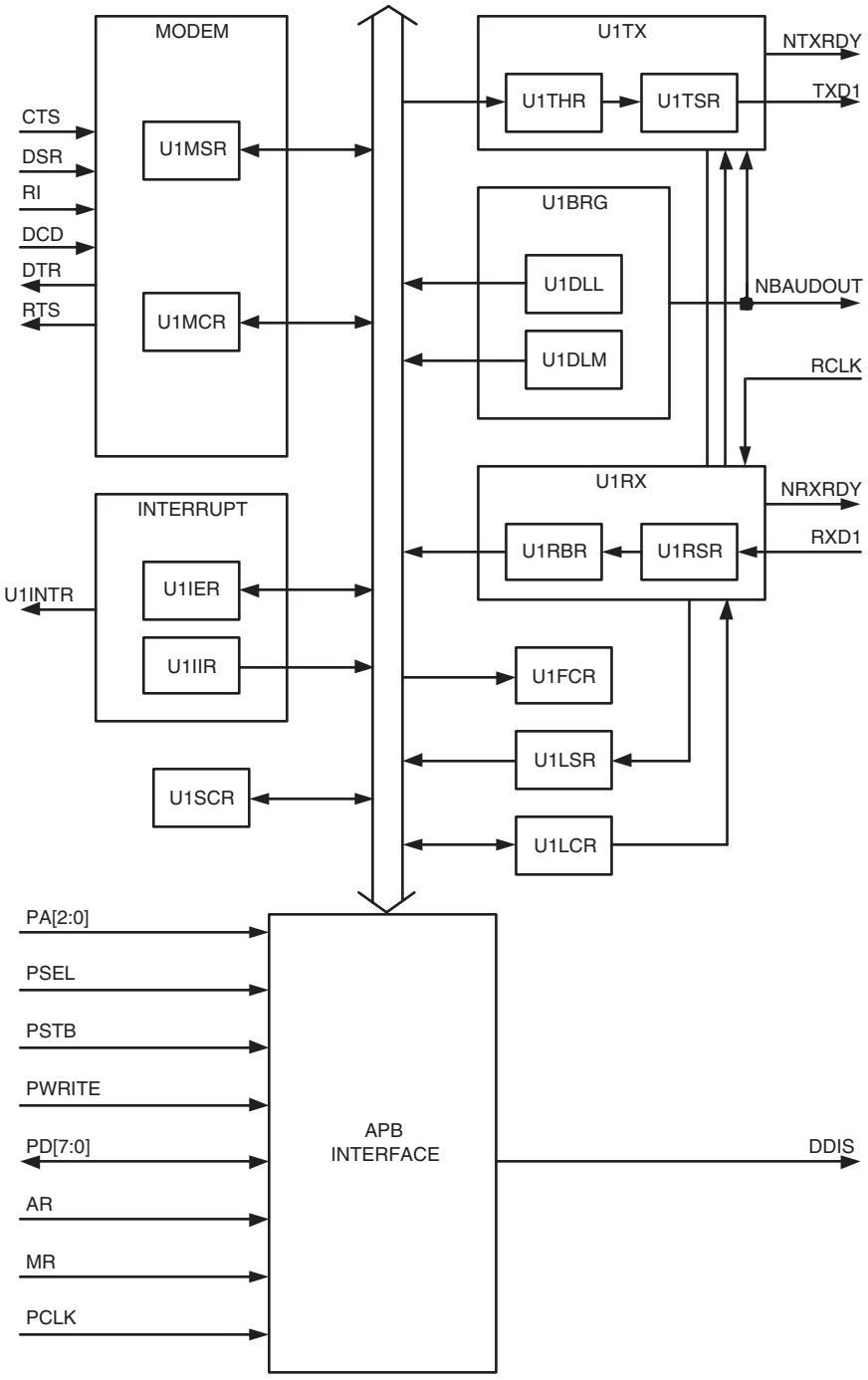


Fig 76. UART1 block diagram

1. Features

- Compliant with Serial Peripheral Interface (SPI) specification.
- Synchronous, Serial, Full Duplex Communication.
- SPI master or slave.
- Maximum data bit rate of one eighth of the input clock rate.
- 8 to 16 bits per transfer

2. SPI overview

SPI is a full duplex serial interfaces. It can handle multiple masters and slaves being connected to a given bus. 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 8 to 16 bits of data to the slave, and the slave always sends a byte of data to the master.

3. SPI data transfers

Figure 18–77 is a timing diagram that illustrates the four different data transfer formats that are available with the SPI. This timing diagram illustrates a single 8 bit data transfer. The first thing you should notice in this timing diagram is that it is divided into three horizontal parts. The first part describes the SCK and SSEL signals. The second part describes the MOSI and MISO signals when the CPHA variable is 0. The third part describes the MOSI and MISO signals when the CPHA variable is 1.

In the first part of the timing diagram, note two points. First, the SPI is illustrated with CPOL set to both 0 and 1. The second point to note is the activation and de-activation of the SSEL signal. When CPHA = 0, the SSEL signal will always go inactive between data transfers. This is not guaranteed when CPHA = 1 (the signal can remain active).

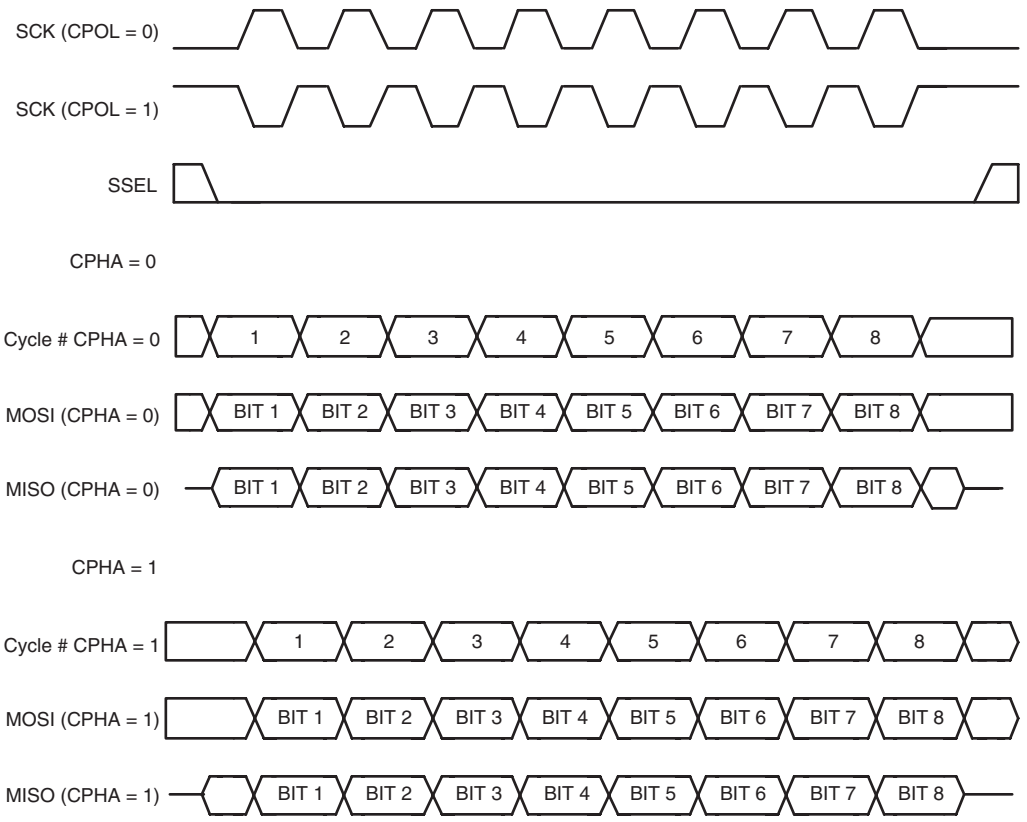


Fig 77. SPI data transfer format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in Table 18–374. This table summarizes the following for each setting of CPOL and CPHA.

- When the first data bit is driven.
- When all other data bits are driven.
- When data is sampled.

Table 374. SPI Data To Clock Phase Relationship

CPOL and CPHA settings	First data driven	Other data driven	Data sampled
CPOL = 0, CPHA = 0	Prior to first SCK rising edge	SCK falling edge	SCK rising edge
CPOL = 0, CPHA = 1	First SCK rising edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 0	Prior to first SCK falling edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 1	First SCK falling edge	SCK falling edge	SCK rising edge

The definition of when an 8 bit transfer starts and stops is dependent on whether a device is a master or a slave, and the setting of the CPHA variable.

When a device is a master, the start of a transfer is indicated by the master having a byte of data that is ready to be transmitted. At this point, the master can activate the clock, and begin the transfer. The transfer ends when the last clock cycle of the transfer is complete.

When a device is a slave and CPHA is set to 0, the transfer starts when the SSEL signal goes active, and ends when SSEL goes inactive. When a device is a slave, and CPHA is set to 1, the transfer starts on the first clock edge when the slave is selected, and ends on the last clock edge where data is sampled.

4. SPI peripheral details

4.1 General information

There are four registers that control the SPI peripheral. They are described in detail in Section 18–6 “Register description” on page 417.

The SPI control register contains a number of programmable bits used to control the function of the SPI block. The settings for this register must be set up prior to a given data transfer taking place.

The SPI status register contains read only bits that are used to monitor the status of the SPI interface, including normal functions, and exception conditions. The primary purpose of this register is to detect completion of a data transfer. This is indicated by the SPIF bit. The remaining bits in the register are exception condition indicators. These exceptions will be described later in this section.

The SPI data register is used to provide the transmit and receive data bytes. An internal shift register in the SPI block logic is used for the actual transmission and reception of the serial data. Data is written to the SPI data register for the transmit case. There is no buffer between the data register and the internal shift register. A write to the data register goes directly into the internal shift register. Therefore, data should only be written to this register when a transmit is not currently in progress. Read data is buffered. When a transfer is complete, the receive data is transferred to a single byte data buffer, where it is later read. A read of the SPI data register returns the value of the read data buffer.

The SPI clock counter register controls the clock rate when the SPI block is in master mode. This needs to be set prior to a transfer taking place, when the SPI block is a master. This register has no function when the SPI block is a slave.

The SPI controller has an enable bit (SPEN), which controls the following functions:

- Activation of SPI I/Os.
- Enables the SPI internal state machine. If a transfer is in progress, and the enable signal is deactivated, the transfer will complete first, then the SPI controller will be disabled.
- Disables any other logic that can be disabled, to conserve power.
- The enable does not disable register accesses to the SPI controller.

The I/Os for this implementation of SPI are standard CMOS I/Os. The open drain SPI option is not implemented in this design. When a device is set up to be a slave, its I/Os are only active when it is selected by the SSEL signal being active.

4.2 Master operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be the master. This process assumes that any prior data transfer has already completed.

1. Set the SPI clock counter register to the desired clock rate.
2. Set the SPI control register to the desired settings.
3. Write the data to transmitted to the SPI data register. This write starts the SPI data transfer.
4. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
5. Read the SPI status register.
6. Read the received data from the SPI data register (optional).
7. Go to step 3 if more data is required to transmit.

Note: A read or write of the SPI data register is required in order to clear the SPIF status bit. Therefore, if the optional read of the SPI data register does not take place, a write to this register is required in order to clear the SPIF status bit.

4.3 Slave operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be a slave. This process assumes that any prior data transfer has already completed. It is required that the system clock driving the SPI logic be at least 8X faster than the SPI.

1. Set the SPI control register to the desired settings.
2. Write the data to transmitted to the SPI data register (optional). Note that this can only be done when a slave SPI transfer is not in progress.
3. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last sampling clock edge of the SPI data transfer.
4. Read the SPI status register.
5. Read the received data from the SPI data register (optional).
6. Go to step 2 if more data is required to transmit.

Note: A read or write of the SPI data register is required in order to clear the SPIF status bit. Therefore, at least one of the optional reads or writes of the SPI data register must take place, in order to clear the SPIF status bit.

4.4 Exception conditions

Read Overrun

A read overrun occurs when the SPI block internal read buffer contains data that has not been read by the processor, and a new transfer has completed. The read buffer containing valid data is indicated by the SPIF bit in the status register being active. When a transfer completes, the SPI block needs to move the received data to the read buffer. If the SPIF bit is active (the read buffer is full), the new receive data will be lost, and the read overrun (ROVR) bit in the status register will be activated.

Write Collision

As stated previously, there is no write buffer between the SPI block bus interface, and the internal shift register. As a result, data must not be written to the SPI data register when a SPI data transfer is currently in progress. The time frame where data cannot be written to the SPI data register is from when the transfer starts, until after the status register has been read when the SPIF status is active. If the SPI data register is written in this time frame, the write data will be lost, and the write collision (WCOL) bit in the status register will be activated.

Mode Fault

If the SSEL signal goes active, when the SPI block is a master, this indicates another master has selected the device to be a slave. This condition is known as a mode fault. When a mode fault is detected, the mode fault (MODF) bit in the status register will be activated, the SPI signal drivers will be de-activated, and the SPI mode will be changed to be a slave.

If the Px.y/SSEL/... pin is assigned the SSEL function in Pin Function Select Register 0, the SSEL signal must always be inactive when the SPI controller is a master.

Slave Abort

A slave transfer is considered to be aborted, if the SSEL signal goes inactive before the transfer is complete. In the event of a slave abort, the transmit and receive data for the transfer that was in progress are lost, and the slave abort (ABRT) bit in the status register will be activated.

5. Pin description

Table 375. SPI pin description

Pin Name	Type	Pin Description
SCK	Input/ Output	**Serial Clock.** The SPI is a clock signal used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL	Input	**Slave Select.** The SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. Each slave has its own unique slave select signal input. The SSEL must be low before data transactions begin and normally stays low for the duration of the transaction. If the SSEL signal goes high any time during a data transfer, the transfer is considered to be aborted. In this event, the slave returns to idle, and any data that was received is thrown away. There are no other indications of this exception. This signal is not directly driven by the master. It could be driven by a simple general purpose I/O under software control. On the LPC2400 (unlike earlier NXP ARM devices) the SSEL pin can be used for a different function when the SPI interface is only used in Master mode. For example, pin hosting the SSEL function can be configured as an output digital GPIO pin and used to select one of the SPI slaves.
MISO	Input/ Output	**Master In Slave Out.** The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. When a device is a slave, serial data is output on this signal. When a device is a master, serial data is input on this signal. When a slave device is not selected, the slave drives the signal high impedance.
MOSI	Input/ Output	**Master Out Slave In.** The MOSI signal is a unidirectional signal used to transfer serial data from the master to the slave. When a device is a master, serial data is output on this signal. When a device is a slave, serial data is input on this signal.

6. Register description

The SPI contains 5 registers as shown in Table 18–376. All registers are byte, half word and word accessible.

Table 376. SPI register map

Name	Description	Access	Reset Value[1]	Address
S0SPCR	SPI Control Register. This register controls the operation of the SPI.	R/W	0x00	0xE002 0000
S0SPSR	SPI Status Register. This register shows the status of the SPI.	RO	0x00	0xE002 0004

Table 376. SPI register map

Name	Description	Access	Reset Value[1]	Address
S0SPDR	SPI Data Register. This bi-directional register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI0 by writing to this register. Data received by the SPI0 can be read from this register.	R/W	0x00	0xE002 0008
S0SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK0.	R/W	0x00	0xE002 000C
S0SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	R/W	0x00	0xE002 001C

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

6.1 SPI Control Register (S0SPCR - 0xE002 0000)

The S0SPCR register controls the operation of the SPI0 as per the configuration bits setting.

Table 377: SPI Control Register (S0SPCR - address 0xE002 0000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	BitEnable	0	The SPI controller sends and receives 8 bits of data per transfer.	0
		1	The SPI controller sends and receives the number of bits selected by bits 11:8.	
3	CPHA		Clock phase control determines the relationship between the data and the clock on SPI transfers, and controls when a slave transfer is defined as starting and ending.	0
		0	Data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	
		1	Data is sampled on the second clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active.	
4	CPOL		Clock polarity control.	0
		0	SCK is active high.	
		1	SCK is active low.	
5	MSTR		Master mode select.	0
		0	The SPI operates in Slave mode.	
		1	The SPI operates in Master mode.	
6	LSBF		LSB First controls which direction each byte is shifted when transferred.	0
		0	SPI data is transferred MSB (bit 7) first.	
		1	SPI data is transferred LSB (bit 0) first.	

Table 377: SPI Control Register (S0SPCR - address 0xE002 0000) bit description

Bit	Symbol	Value	Description	Reset Value
7	SPIE		Serial peripheral interrupt enable.	0
		0	SPI interrupts are inhibited.	
		1	A hardware interrupt is generated each time the SPIF or MODF bits are activated.	
11:8	BITS		When bit 2 of this register is 1, this field controls the number of bits per transfer:	0000
		1000	8 bits per transfer	
		1001	9 bits per transfer	
		1010	10 bits per transfer	
		1011	11 bits per transfer	
		1100	12 bits per transfer	
		1101	13 bits per transfer	
		1110	14 bits per transfer	
		1111	15 bits per transfer	
		0000	16 bits per transfer	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2 SPI Status Register (S0SPSR - 0xE002 0004)

The S0SPSR register controls the operation of the SPI0 as per the configuration bits setting.

Table 378: SPI Status Register (S0SPSR - address 0xE002 0004) bit description

Bit	Symbol	Description	Reset Value
2:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort. When 1, this bit indicates that a slave abort has occurred. This bit is cleared by reading this register.	0
4	MODF	Mode fault. when 1, this bit indicates that a Mode fault error has occurred. This bit is cleared by reading this register, then writing the SPI0 control register.	0
5	ROVR	Read overrun. When 1, this bit indicates that a read overrun has occurred. This bit is cleared by reading this register.	0
6	WCOL	Write collision. When 1, this bit indicates that a write collision has occurred. This bit is cleared by reading this register, then accessing the SPI data register.	0
7	SPIF	SPI transfer complete flag. When 1, this bit indicates when a SPI data transfer is complete. When a master, this bit is set at the end of the last cycle of the transfer. When a slave, this bit is set on the last data sampling edge of the SCK. This bit is cleared by first reading this register, then accessing the SPI data register. **Note:** this is not the SPI interrupt flag. This flag is found in the SPINT register.	0

6.3 SPI Data Register (S0SPDR - 0xE002 0008)

This bi-directional data register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register. When a master, a write to this register will start a SPI data transfer. Writes to this register will be blocked from when a data transfer starts to when the SPIF status bit is set, and the status register has not been read.

Table 379: SPI Data Register (S0SPDR - address 0xE002 0008) bit description

Bit	Symbol	Description	Reset Value
7:0	DataLow	SPI Bi-directional data port.	0x00
15:8	DataHigh	If bit 2 of the SPCR is 1 and bits 11:8 are other than 1000, some or all of these bits contain the additional transmit and receive bits. When less than 16 bits are selected, the more significant among these bits read as zeroes.	0x00

6.4 SPI Clock Counter Register (S0SPCCR - 0xE002 000C)

This register controls the frequency of a master's SCK. The register indicates the number of PCLK cycles that make up an SPI clock. The value of this register must always be an even number. As a result, bit 0 must always be 0. The value of the register must also always be greater than or equal to 8. Violations of this can result in unpredictable behavior.

Table 380: SPI Clock Counter Register (S0SPCCR - address 0xE002 000C) bit description

Bit	Symbol	Description	Reset Value
7:0	Counter	SPI0 Clock counter setting.	0x00

The SPI0 rate may be calculated as: PCLK / SPCCR0 value. The PCLK rate is CCLK /APB divider rate as determined by the PCLKSEL0 register contents for PCLK_SPI.

6.5 SPI Test Control Register (SPTCR - 0xE002 0010)

Note that the bits in this register are intended for functional verification only. This register should not be used for normal operation.

Table 381: SPI Test Control Register (SPTCR - address 0xE002 0010) bit description

Bit	Symbol	Description	Reset Value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:1	Test	SPI test mode. When 0, the SPI operates normally. When 1, SCK will always be on, independent of master mode select, and data availability setting.	0

6.6 SPI Test Status Register (SPTSR - 0xE002 0014)

Note: The bits in this register are intended for functional verification only. This register should not be used for normal operation.

This register is a replication of the SPI status register. The difference between the registers is that a read of this register will not start the sequence of events required to clear these status bits. A write to this register will set an interrupt if the write data for the respective bit is a 1.

Table 382: SPI Test Status Register (SPTSR - address 0xE002 0014) bit description

Bit	Symbol	Description	Reset Value
2:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort.	0
4	MODF	Mode fault.	0
5	ROVR	Read overrun.	0
6	WCOL	Write collision.	0
7	SPIF	SPI transfer complete flag.	0

6.7 SPI Interrupt Register (S0SPINT - 0xE002 001C)

This register contains the interrupt flag for the SPI0 interface.

Table 383: SPI Interrupt Register (S0SPINT - address 0xE002 001C) bit description

Bit	Symbol	Description	Reset Value
0	SPI Interrupt Flag	SPI interrupt flag. Set by the SPI interface to generate an interrupt. Cleared by writing a 1 to this bit. **Note:** this bit will be set once when SPIE = 1 and at least one of SPIF and WCOL bits is 1. However, only when the SPI Interrupt bit is set and SPI0 Interrupt is enabled in the VIC, SPI based interrupt can be processed by interrupt handling software.	0
7:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7. Architecture

The block diagram of the SPI solution implemented in SPI0 interface is shown in the Figure 18–78.

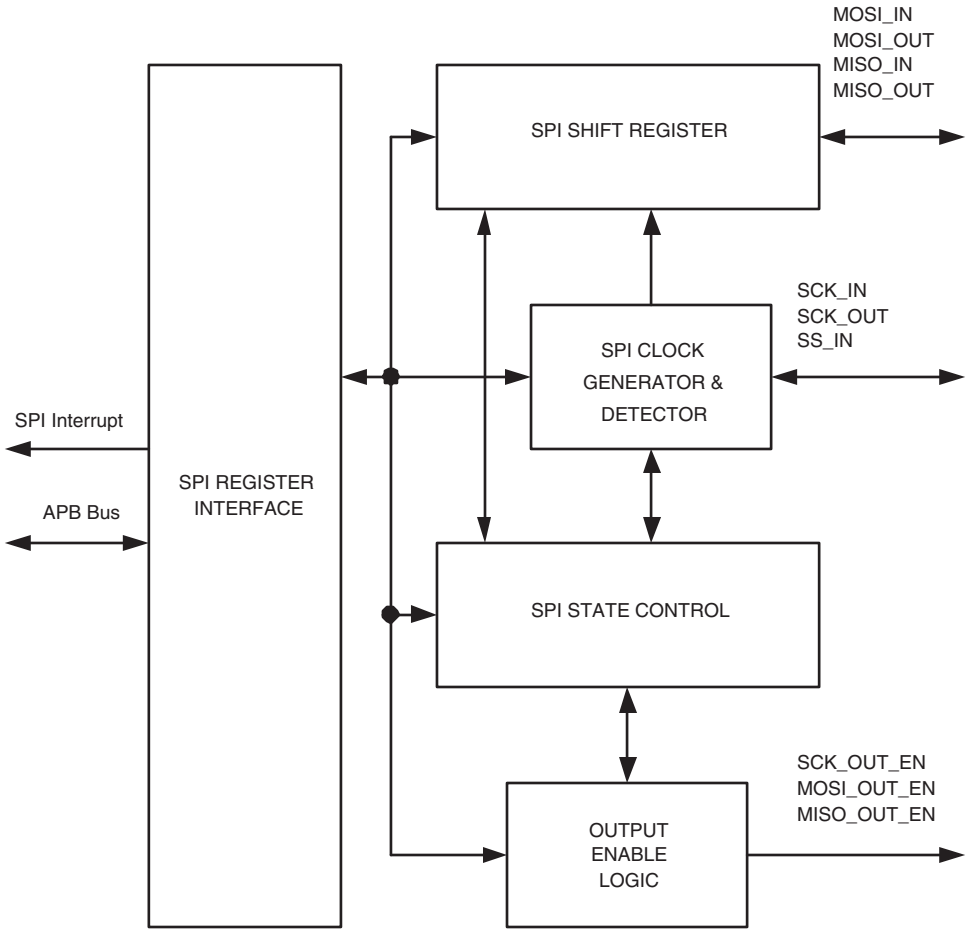


Fig 78. SPI block diagram

1. 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.

2. 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.

LPC2468 has two Synchronous Serial Port controllers -- SSP0 and SSP1.

3. Pin descriptions

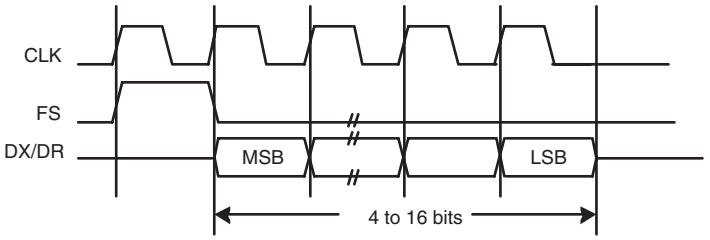
Table 384. SSP pin descriptions

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.

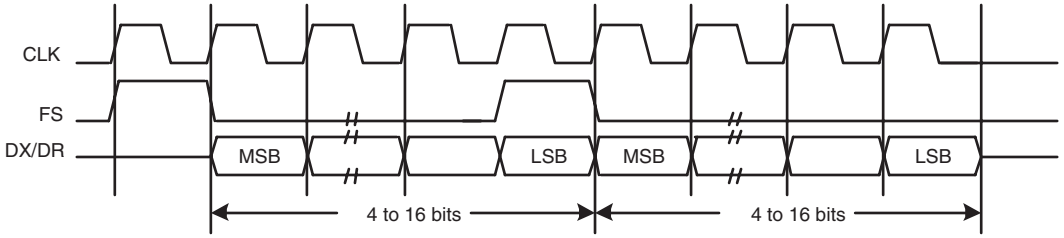
4. Bus description

4.1 Texas Instruments synchronous serial frame format

Figure 19–79 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 79. 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 tristated 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.

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.

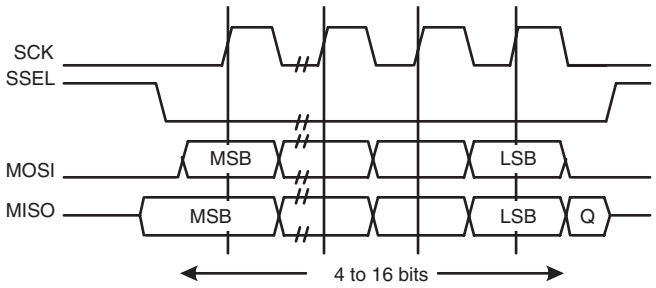
4.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

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.

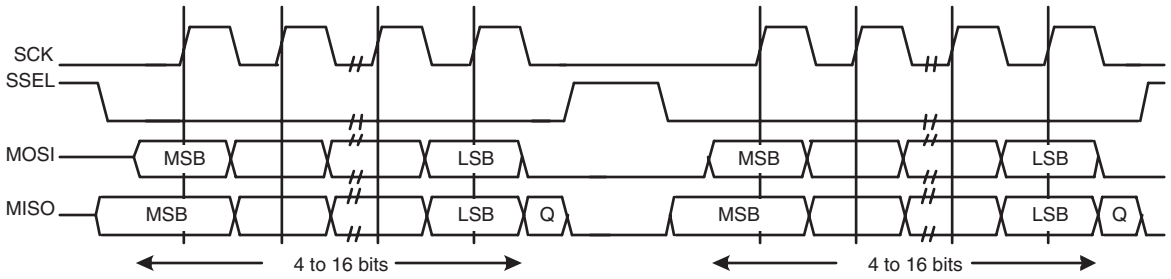
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is LOW, data is captured on the first clock edge transition. If the CPHA clock phase control bit is HIGH, data is captured on the second clock edge transition.

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 19–80.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 80. 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.

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 19–81, which covers both single and continuous transfers.

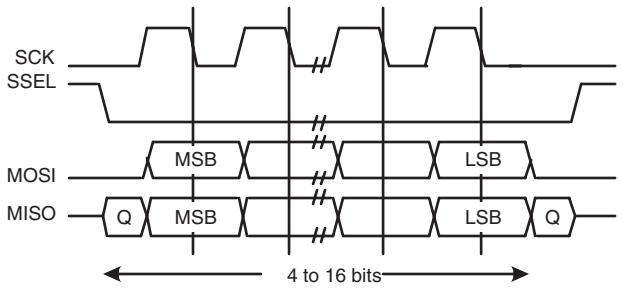


Fig 81. SPI frame format with CPOL=0 and CPHA=1

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.

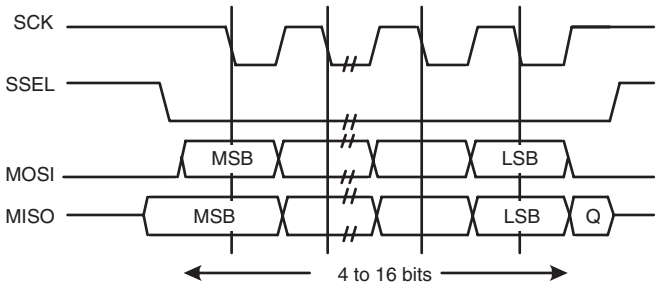
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

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.

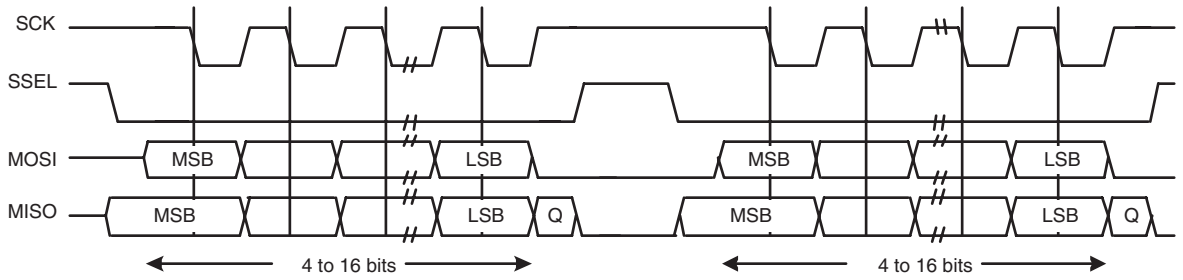
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

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 19–82.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 82. 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.

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 19–83, which covers both single and continuous transfers.

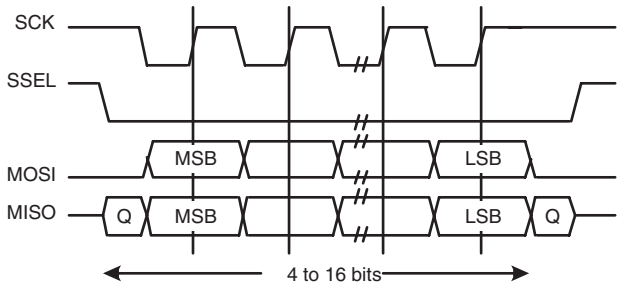


Fig 83. SPI Frame Format with CPOL = 1 and CPHA = 1

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.

4.3 Semiconductor Microwire frame format

Figure 19–84 shows the Microwire frame format for a single frame. Figure 19–85 shows the same format when back-to-back frames are transmitted.

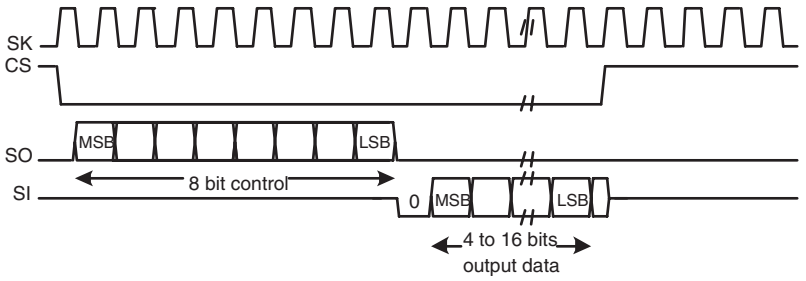


Fig 84. Microwire frame format (single transfer)

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 tristated 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 tristate 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.

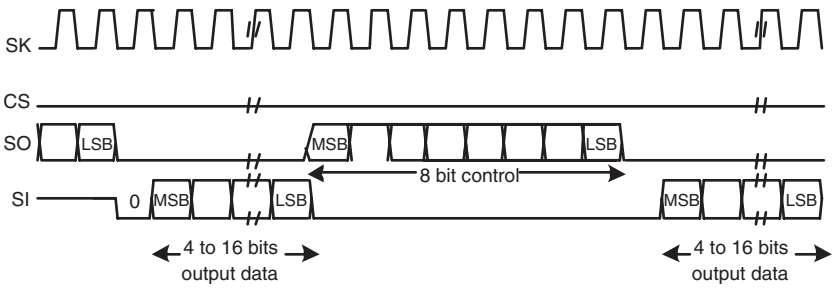


Fig 85. Microwire frame format (continuos transfers)

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 19–86 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.

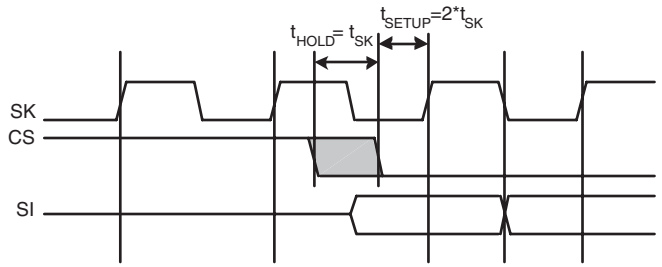


Fig 86. Microwire frame format setup and hold details

5. Register description

The register offsets from the SSP controller base addresses are shown in the Table 19–385.

Table 385. SSP Register Map

Generic Name	Description	Access	Reset Value[1]	SSPn Register Name & Address
CR0	Control Register 0. Selects the serial clock rate, bus type, and data size.	R/W	0	SSP0CR0 - 0xE006 8000 SSP1CR0 - 0xE003 0000
CR1	Control Register 1. Selects master/slave and other modes.	R/W	0	SSP0CR1 - 0xE006 8004 SSP1CR1 - 0xE003 0004
DR	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	R/W	0	SSP0DR - 0xE006 8008 SSP1DR - 0xE003 0008
SR	Status Register	RO		SSP0SR - 0xE006 800C SSP1SR - 0xE003 000C
CPSR	Clock Prescale Register	R/W	0	SSP0CPSR - 0xE006 8010 SSP1CPSR - 0xE003 0010
IMSC	Interrupt Mask Set and Clear Register	R/W	0	SSP0IMSC - 0xE006 8014 SSP1IMSC - 0xE003 0014
RIS	Raw Interrupt Status Register	R/W		SSP0RIS - 0xE006 8018 SSP1RIS - 0xE003 0018
MIS	Masked Interrupt Status Register	R/W	0	SSP0MIS - 0xE006 801C SSP1MIS - 0xE003 001C
ICR	SSPICR Interrupt Clear Register	R/W	NA	SSP0ICR - 0xE006 8020 SSP1ICR - 0xE003 0020
DMACR	DMA Control Register	R/W	0	SSP0DMACR - 0xE006 8024 SSP1DMACR - 0xE003 0024

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

5.1 SSPn Control Register 0 (SSP0CR0 - 0xE006 8000, SSP1CR0 - 0xE003 0000)

This register controls the basic operation of the SSP controller.

Table 386: SSPn Control Register 0 (SSP0CR0 - address 0xE006 8000, SSP1CR0 - 0xE003 0000) 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	SPO		Clock Out Polarity. 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.	
7	SPH		Clock Out Phase. 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.	
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 APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR $\times$ [SCR+1]).	0x00

5.2 SSPn Control Register 1 (SSP0CR1 - 0xE006 8004, SSP1CR1 - 0xE003 0004)

This register controls certain aspects of the operation of the SSP controller.

Table 387: SSPn Control Register 1 (SSP0CR1 - address 0xE006 8004, SSP1CR1 - 0xE003 0004) 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

5.3 SSPn Data Register (SSP0DR - 0xE006 8008, SSP1DR - 0xE003 0008)

Software can write data to be transmitted to this register, and read data that has been received.

Table 388: SSPn Data Register (SSP0DR - address 0xE006 8008, SSP1DR - 0xE003 0008) 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

5.4 SSPn Status Register (SSP0SR - 0xE006 800C, SSP1SR - 0xE003 000C)

This read-only register reflects the current status of the SSP controller.

Table 389: SSPn Status Register (SSP0SR - address 0xE006 800C, SSP1SR - 0xE003 000C) 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

5.5 SSPn Clock Prescale Register (SSP0CPSR - 0xE006 8010, SSP1CPSR - 0xE003 0010)

This register controls the factor by which the Prescaler divides the APB clock PCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in SSPnCR0, to determine the bit clock.

Table 390: SSPn Clock Prescale Register (SSP0CPSR - address 0xE006 8010, SSP1CPSR - 0xE003 8010) bit description

Bit	Symbol	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which PCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0

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 $CPSDVSR_{min} = 12$.

5.6 SSPn Interrupt Mask Set/Clear Register (SSP0IMSC - 0xE006 8014, SSP1IMSC - 0xE003 0014)

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 391: SSPn Interrupt Mask Set/Clear register (SSP0IMSC - address 0xE006 8014, SSP1IMSC - 0xE003 0014) 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

5.7 SSPn Raw Interrupt Status Register (SSP0RIS - 0xE006 8018, SSP1RIS - 0xE003 0018)

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 392: SSPn Raw Interrupt Status register (SSP0RIS - address 0xE006 8018, SSP1RIS - 0xE003 0018) 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

5.8 SSPn Masked Interrupt Status Register (SSP0MIS - 0xE006 801C, SSP1MIS - 0xE003 001C)

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 393: SSPn Masked Interrupt Status register (SSPnMIS -address 0xE006 801C, SSP1MIS - 0xE003 001C) 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

5.9 SSPn Interrupt Clear Register (SSP0ICR - 0xE006 8020, SSP1ICR - 0xE003 0020)

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 394: SSPn interrupt Clear Register (SSP0ICR - address 0xE006 8020, SSP1ICR - 0xE003 0020) 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

5.10 SSPn DMA Control Register (SSP0DMACR - 0xE006 8024, SSP1DMACR - 0xE003 0024)

The SSPnDMACR register is the DMA control register.It is a read/write register. Table 19–395 shows the bit assignments of the SSPnDMACR register.

Table 395: SSPn DMA Control Register (SSP0DMACR - address 0xE006 8024, SSP1DMACR - 0xE003 0024) 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

1. Introduction

The Secure Digital and Multimedia Card Interface (MCI) is an interface between the Advanced Peripheral Bus (APB) system bus and multimedia and/or secure digital memory cards. It consists of two parts:

- The MCI adapter block provides all functions specific to the Secure Digital/MultiMedia memory card, such as the clock generation unit, power management control, command and data transfer.
- The APB interface accesses the MCI adapter registers, and generates interrupt and DMA request signals.

2. Features of the MCI

The following features are provided by the MCI:

- Conformance to *Multimedia Card Specification v2.11*.
- Conformance to *Secure Digital Memory Card Physical Layer Specification, v0.96*.
- Use as a multimedia card bus or a secure digital memory card bus host. It can be connected to several multimedia cards, or a single secure digital memory card.
- DMA supported through the General Purpose DMA Controller.

3. SD/MMC card interface pin description

Table 396. SD/MMC card interface pin description

Pin Name	Type	Description
MCICLK	Output	Clock output
MCICMD	Input	Command input/output.
MCIDAT[3:0]	Output	Data lines. Only MCIDAT[0] is used for Multimedia cards.
MCIPWR	Output	Power Supply Enable for external SD/MMC power supply.

There is one additional signal needed in the interface, a power control line MCIPWR, but it can be sourced from any GPIO signal.

4. Functional overview

The MCI may be used as a multimedia card bus host (see Section 20–4.1 “Mutimedia card”) or a secure digital memory card bus host (see Section 20–4.2 “Secure digital memory card”). Up to approximately 4 multimedia cards (limited by I/O pin specifications and board loading) may be connected, or a single secure digital memory card.

4.1 Mutimedia card

Figure 20–87 shows the multimedia card system.

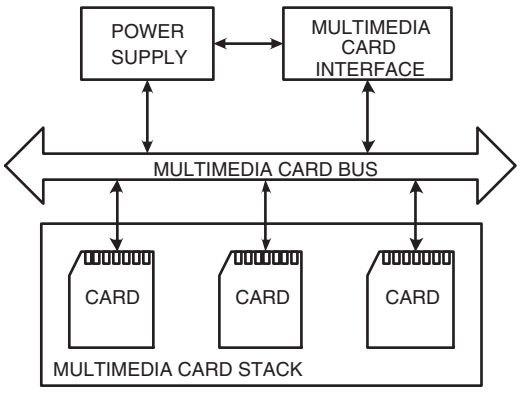


Fig 87. Multimedia card system

Multimedia cards are grouped into three types according to their function:

- Read Only Memory (ROM) cards, containing pre-programmed data
- Read/Write (R/W) cards, used for mass storage
- Input/Output (I/O) cards, used for communication

The multimedia card system transfers commands and data using three signal lines:

- CLK: One bit is transferred on both command and data lines with each clock cycle. The clock frequency varies between 0 MHz and 20 MHz (for a multimedia card) or 0 MHz and 25 MHz (for a secure digital memory card).
- CMD: Bidirectional command channel that initializes a card and transfers commands. CMD has two operational modes:
  - Open-drain for initialization
  - Push-pull for command transfer
- DAT: Bidirectional data channel, operating in push-pull mode

4.2 Secure digital memory card

Figure 20–88 shows the secure digital memory card connection.

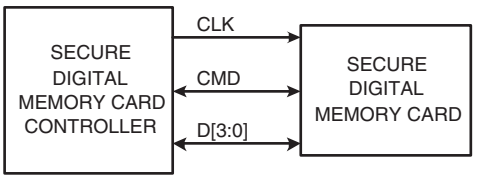


Fig 88. Secure digital memory card connection

4.2.1 Secure digital memory card bus signals

The following signals are used on the secure digital memory card bus:

- CLK Host to card clock signal

- CMD Bidirectional command/response signal
- DAT[3:0] Bidirectional data signals

4.3 MCI adapter

Figure 20–89 shows a simplified block diagram of the MCI adapter.

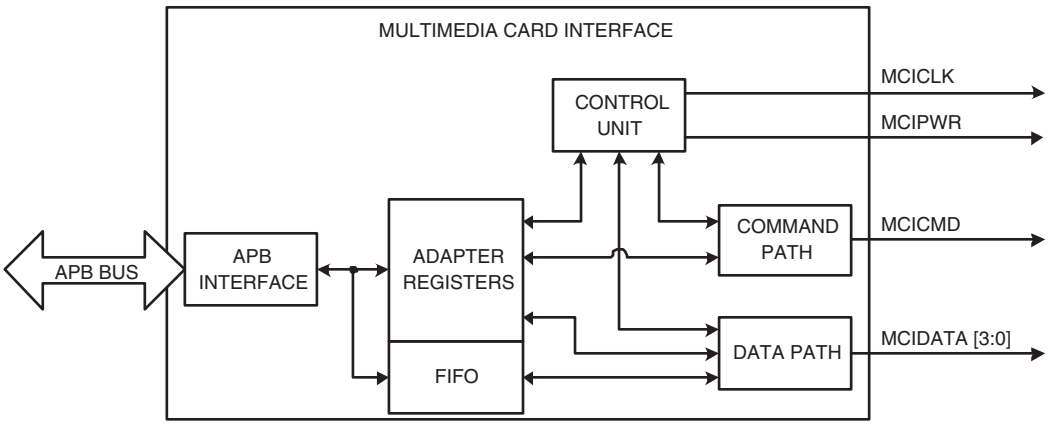


Fig 89. MCI adapter

The MCI adapter is a multimedia/secure digital memory card bus master that provides an interface to a multimedia card stack or to a secure digital memory card. It consists of five subunits:

- Adapter register block
- Control unit
- Command path
- Data path
- Data FIFO

4.3.1 Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when 1 is written into the corresponding bit location of the MCIClear register.

4.3.2 Control unit

The control unit contains the power management functions and the clock divider for the memory card clock.

There are three power phases:

- Power-off
- Power-up
- Power-on

The power management logic controls an external power supply unit, and disables the card bus output signals during the power-off or power-up phases. The power-up phase is a transition phase between the power-off and power-on phases, and allows an external power supply to reach the card bus operating voltage. A device driver is used to ensure that the PrimeCell MCI remains in the power-up phase until the external power supply reaches the operating voltage.

The clock management logic generates and controls the MCICLK signal. The MCICLK output can use either a clock divide or clock bypass mode. The clock output is inactive:

- after reset
- during the power-off or power-up phases
- if the power saving mode is enabled and the card bus is in the IDLE state (eight clock periods after both the command and data path subunits enter the IDLE phase)

4.3.3 Command path

The command path subunit sends commands to and receives responses from the cards.

4.3.4 Command path state machine

When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the Command Path State Machine (CPSM) sets the status flags and enters the IDLE state if a response is not required. If a response is required, it waits for the response (see Figure 20–90). When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set.

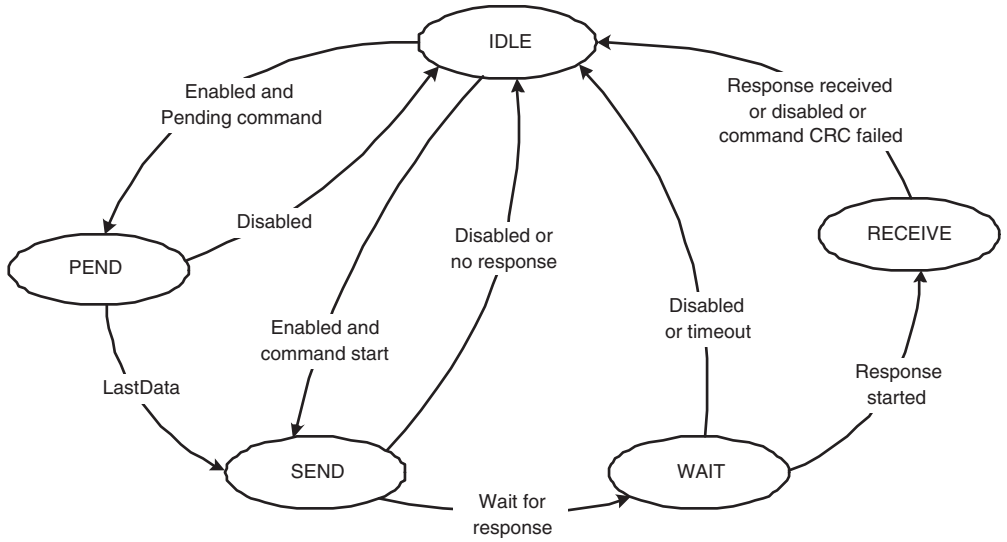


Fig 90. Command path state machine

When the WAIT state is entered, the command timer starts running. If the timeout[1] is reached before the CPSM moves to the RECEIVE state, the timeout flag is set and the IDLE[2] state is entered.

1. The timeout period has a fixed value of 64 MCICLK clocks period.

If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, the CPSM enters the PEND state, and waits for a CmdPend signal from the data path subunit. When CmdPend is detected, the CPSM moves to the SEND state. This enables the data counter to trigger the stop command transmission.

Figure 20–91 shows the MCI command transfer.

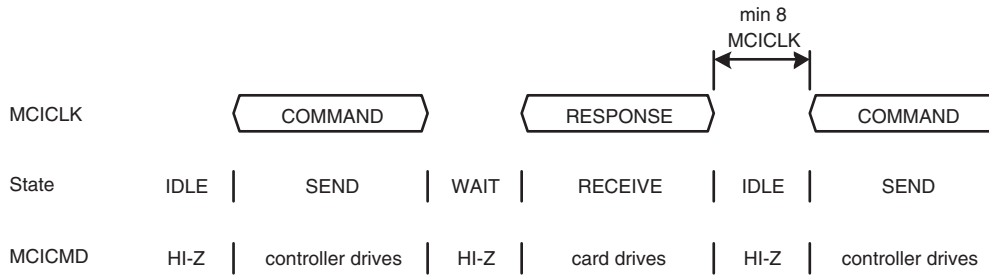


Fig 91. MCI command transfer

4.3.5 Command format

The command path operates in a half-duplex mode, so that commands and responses can either be sent or received. If the CPSM is not in the SEND state, the MCICMD output is in HI-Z state, as shown in Figure 20–91. Data on MCICMD is synchronous to the rising MCICLK edge. All commands have a fixed length of 48 bits. Table 20–397 shows the command format.

Table 397. Command format

Bit Position	Width	Value	Description
0	1	1	End bit.
7:1	7	-	CRC7
39:8	32	-	Argument.
45:40	6	-	Command index.
46	1	1	Transmission bit.
47	1	0	Stat bit.

The MCI adapter supports two response types. Both use CRC error checking:

- 48 bit short response (see Table 20–398)
- 136 bit long response (see Table 20–399)

Note: If the response does not contain CRC (CMD1 response), the device driver must ignore the CRC failed status.

2. The CPSM remains in the IDLE state for at least eight MCICLK periods to meet Ncc and Nrc timing constraints.

Table 398. Simple response format

Bit Position	Width	Value	Description
0	1	1	End bit.
7:1	7	-	CRC7 (or 1111111).
39:8	32	-	Argument.
45:40	6	-	Command index.
46	1	0	Transmission bit.
47	1	0	Start bit.

Table 399. Long response format

Bit Position	Width	Value	Description
0	1	1	End bit.
127:1	127	-	CID or CSD (including internal CRC7).
133:128	6	111111	Reserved.
134	1	1	Transmission bit.
135	1	0	Start bit.

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response, and whether the response is 48 or 136 bits long (see Section 20–5.5 “Command Register (MCICommand - 0xE008 C00C)” for more information). The command path implements the status flags shown in Table 20–400 (see Section 20–5.12 “Status Register (MCIStatus - 0xE008 C034)” for more information).

Table 400. 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:

$$CRC[6:0] = \text{Remainder } [(M(x) \times x_7) / G(x)]$$

$$G(x) = x_7 + x_3 + 1$$

$$M(x) = (\text{start bit}) \times x_{39} + \ldots + (\text{last bit before CRC}) \times x_0 \text{ , or}$$

$$M(x) = (\text{start bit}) \times x_{119} + \ldots + (\text{last bit before CRC}) \times x_0$$

4.3.6 Data path

The card data bus width can be programmed using the clock control register. If the wide bus mode is enabled, data is transferred at four bits per clock cycle over all four data signals (MCIDAT[3:0]). If the wide bus mode is not enabled, only one bit per clock cycle is transferred over MCIDAT0.

Depending on the transfer direction (send or receive), the Data Path State Machine (DPSM) moves to the WAIT_S or WAIT_R state when it is enabled:

- Send: The DPSM moves to the WAIT_S state. If there is data in the send FIFO, the DPSM moves to the SEND state, and the data path subunit starts sending data to a card.
- Receive: The DPSM moves to the WAIT_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the RECEIVE state, and the data path subunit starts receiving data from a card.

4.3.7 Data path state machine

The DPSM operates at MCICLK frequency. Data on the card bus signals is synchronous to the rising edge of MCICLK. The DPSM has six states, as shown in Figure 20–92.

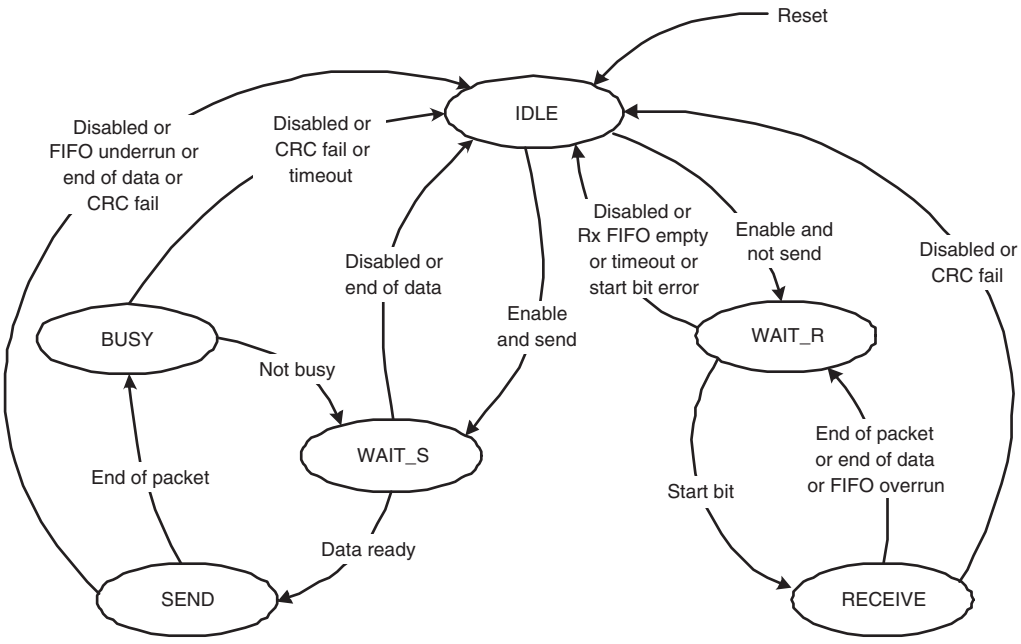


Fig 92. Data path state machine

- IDLE: The data path is inactive, and the MCIDAT[3:0] outputs are in HI-Z. When the data control register is written and the enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the WAIT_S or WAIT_R state.

  WAIT_R: If the data counter equals zero, the DPSM moves to the IDLE state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on MCIDAT.

The DPSM moves to the RECEIVE state if it receives a start bit before a timeout, and loads the data block counter. If it reaches a timeout before it detects a start bit, or a start bit error occurs, it moves to the IDLE state and sets the timeout status flag.

- RECEIVE: Serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - In block mode, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to the WAIT_R state. If not, the CRC fail status flag is set and the DPSM moves to the IDLE state.
  - In stream mode, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the WAIT_R state.

If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to the WAIT_R state.

- WAIT_S: The DPSM moves to the IDLE state if the data counter is zero. If not, it waits until the data FIFO empty flag is deasserted, and moves to the SEND state.

Note: The DPSM remains in the WAIT_S state for at least two clock periods to meet Nwr timing constraints.

- SEND: The DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - In block mode, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit, and moves to the BUSY state.
  - In stream mode, the DPSM sends data to a card while the enable bit is HIGH and the data counter is not zero. It then moves to the IDLE state.

If a FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the IDLE state.

- BUSY: The DPSM waits for the CRC status flag:
  - If it does not receive a positive CRC status, it moves to the IDLE state and sets the CRC fail status flag.
  - If it receives a positive CRC status, it moves to the WAIT_S state if MCIDAT0 is not LOW (the card is not busy).

If a timeout occurs while the DPSM is in the BUSY state, it sets the data timeout flag and moves to the IDLE state.

The data timer is enabled when the DPSM is in the WAIT_R or BUSY state, and generates the data timeout error:

- When transmitting data, the timeout occurs if the DPSM stays in the BUSY state for longer than the programmed timeout period
- When receiving data, the timeout occurs if the end of the data is not true, and if the DPSM stays in the WAIT_R state for longer than the programmed timeout period.

4.3.8 Data counter

The data counter has two functions:

- To stop a data transfer when it reaches zero. This is the end of the data condition.
- To start transferring a pending command (see Figure 20–93). This is used to send the stop command for a stream data transfer.

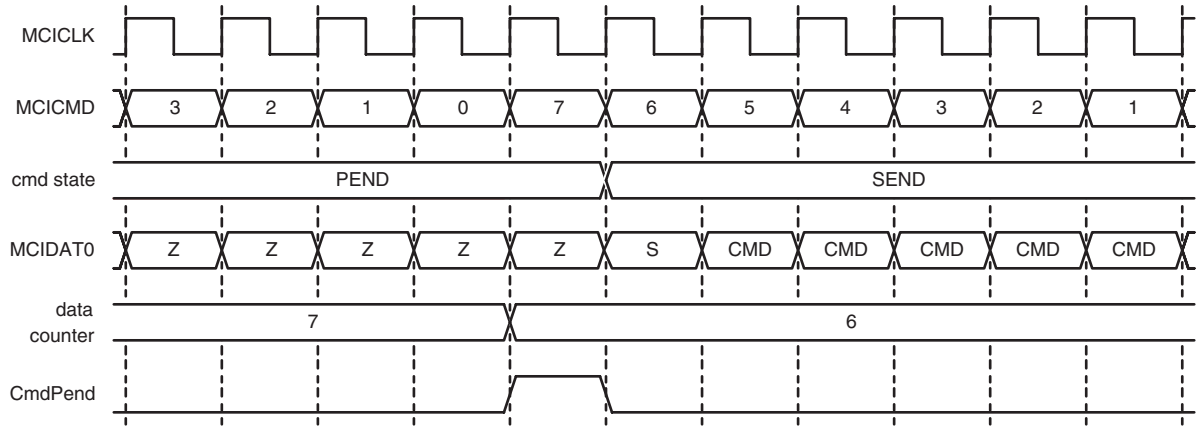


Fig 93. Pending command start

The data block counter determines the end of a data block. If the counter is zero, the end-of-data condition is TRUE (see Section 20–5.10 “Data Control Register (MCIDataCtrl - 0xE008 C02C)” for more information).

4.3.9 Bus mode

In wide bus mode, all four data signals (MCIDAT[3:0]) are used to transfer data, and the CRC code is calculated separately for each data signal. While transmitting data blocks to a card, only MCIDAT0 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, MCIDAT[3:0] are in the HI-Z state.

Data on these signals is synchronous to the rising edge of the clock period.

If standard bus mode is selected the MCIDAT[3:1] outputs are always in HI-Z state and only the MCIDAT0 output is driven LOW when data is transmitted.

Design note: If wide mode is selected, both nMCIDAT0EN and nMCIDATEN outputs are driven low at the same time. If not, the MCIDAT[3:1] outputs are always in HI-Z state (nMCIDATEN) is driven HIGH), and only the MCIDAT0 output is driven LOW when data is transmitted.

4.3.10 CRC Token status

The CRC token status follows each write data block, and determines whether a card has received the data block correctly. When the token has been received, the card asserts a busy signal by driving MCIDAT0 LOW. Table 20–401 shows the CRC token status values.

Table 401. CRC token status

Token	Description
010	Card has received error-free data block.
101	Card has detected a CRC error.

4.3.11 Status flags

Table 20–402 lists the data path status flags (see Section 20–5.12 “Status Register (MCIStatus - 0xE008 C034)” on page 455 for more information).

Table 402. 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.

4.3.12 CRC generator

The CRC generator calculates the CRC checksum only for the data bits in a single block, and is bypassed in data stream mode. The checksum is a 16 bit value:

$$CRC[15:0] = \text{Remainder } [(M(x) \times x^{15}) / G(x)]$$

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

$$M(x) - (\text{first data bit}) \times x^n + \ldots + (\text{last data bit}) ¥ X^0$$

4.3.13 Data FIFO

The data FIFO (first-in-first-out) subunit is a data buffer with transmit and receive logic.

The FIFO contains a 32 bit wide, 16-word deep data buffer, and transmit and receive logic. Because the data FIFO operates in the APB clock domain (PCLK), all signals from the subunits in the MCI clock domain (MCLK) are resynchronized.

Depending on TxActive and RxActive, the FIFO can be disabled, transmit enabled, or receive enabled. TxActive and RxActive are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when TxActive is asserted (see Section 20–4.3.14 “Transmit FIFO”)
- The receive FIFO refers to the receive logic and data buffer when RxActive is asserted (see Section 20–4.3.15 “Receive FIFO”).

4.3.14 Transmit FIFO

Data can be written to the transmit FIFO through the APB interface once the MCI is enabled for transmission.

The transmit FIFO is accessible via 16 sequential addresses (see Section 20–5.16 “Data FIFO Register (MCIFIFO - 0xE008 C080 to 0xE008 C0BC)”). The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out.

If the transmit FIFO is disabled, all status flags are deasserted. The data path subunit asserts TxActive when it transmits data. Table 20–403 lists the transmit FIFO status flags.

Table 403. Transmit FIFO status flags

Flag	Description
TxFifoFull	Set to HIGH when all 16 transmit FIFO words contain valid data.
TxFifoEmpty	Set to HIGH when the transmit FIFO does not contain valid data.
TxHalfEmpty	Set to HIGH when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TxDataAvlbl	Set to HIGH when the transmit FIFO contains valid data. This flag is the inverse of the TxFifoEmpty flag.
TxUnderrun	Set to HIGH when an underrun error occurs. This flag is cleared by writing to the MCIClear register.

4.3.15 Receive FIFO

When the data path subunit receives a word of data, it drives data on the write data bus and asserts the write enable signal. This signal is synchronized to the PCLK domain. The write pointer is incremented after the write is completed, and the receive FIFO control logic asserts RxWrDone, that then deasserts the write enable signal.

On the read side, the content of the FIFO word pointed to by the current value of the read pointer is driven on the read data bus. The read pointer is incremented when the APB bus interface asserts RxRdPrtInc.

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RxActive when it receives data. Table 353 lists the receive FIFO status flags.

The receive FIFO is accessible via 16 sequential addresses (see Section 20–5.16 “Data FIFO Register (MCIFIFO - 0xE008 C080 to 0xE008 C0BC)”).

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RxActive when it receives data. Table 20–404 lists the receive FIFO status flags.

Table 404. Receive FIFO status flags

Symbol	Description
RxFifoFull	Set to HIGH when all 16 receive FIFO words contain valid data.
RxFifoEmpty	Set to HIGH when the receive FIFO does not contain valid data.
RxHalfFull	Set to HIGH when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request.
RxDataAvlbl	Set to HIGH when the receive FIFO is not empty. This flag is the inverse of the RxFifoEmpty flag.
RxOverrun	Set to HIGH when an overrun error occurs. This flag is cleared by writing to the MCIClear register.

4.3.16 APB interfaces

The APB interface generates the interrupt and DMA requests, and accesses the MCI adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt/DMA logic. DMA is controlled by the General Purpose DMA controller, see that chapter for details.

4.3.17 Interrupt logic

The interrupt logic generates an interrupt request signal that is asserted when at least one of the selected status flags is HIGH. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set.

5. Register description

This section describes the MCI registers and provides programming details.

5.1 Summary of MCI Registers

The MCI registers are shown in Table 20–405.

Table 405. SPI register map

Name	Description	Access	Width	Reset Value[1]	Address
MCIPower	Power control register.	R/W	8	0x00	0xE008 C000
MCIClock	Clock control register.	R/W	12	0x000	0xE008 C004
MCIArgument	Argument register.	R/W	32	0x00000000	0xE008 C008
MMCCommand	Command register.	R/W	11	0x000	0xE008 C00C
MCIRespCmd	Response command register.	RO	6	0x00	0xE008 C010
MCIResponse0	Response register.	RO	32	0x00000000	0xE008 C014
MCIResponse1	Response register.	RO	32	0x00000000	0xE008 C018
MCIResponse2	Response register.	RO	32	0x00000000	0xE008 C01C

Table 405. SPI register map

Name	Description	Access	Width	Reset Value[1]	Address
MCIResponse3	Response register.	RO	31	0x00000000	0xE008 C020
MCIDataTimer	Data Timer.	R/W	32	0x00000000	0xE008 C024
MCIDataLength	Data control register.	R/W	16	0x0000	0xE008 C028
MCIDataCtrl	Data control register.	R/W	8	0x00	0xE008 C02C
MCIDataCnt	Data counter.	RO	16	0x0000	0xE008 C030
MCIStatus	Status register.	RO	22	0x000000	0xE008 C034
MCIClear	Clear register.	WO	11	-	0xE008 C038
MCIMask0	Interrupt 0 mask register.	R/W	22	0x000000	0xE008 C03C
MCIMask1	Interrupt 1 mask register.	R/W	22	0x000000	0xE008 C040
MCIFifoCnt	FIFO Counter.	RO	15	0x0000	0xE008 C048
MCIFIFO	Data FIFO Register.	R/W	32	0x00000000	0xE008 C080 to 0xE008 C0BC

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

5.2 Power Control Register (MCI Power - 0xE008 C000)

The MCIPower register controls an external power supply. Power can be switched on and off, and adjust the output voltage. Table 20–406 shows the bit assignment of the MCIPower register.

The active level of the MCIPWR (Power Supply Enable) pin can be selected by bit 3 of the SCS register (see Section 3–7.1 “System Controls and Status register (SCS - 0xE01F C1A0)” on page 26 for details).

Table 406: Power Control register (MCIPower - address 0xE008 C000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Ctrl	00	Power-off	00
		01	Reserved	
		10	Power-up	
		11	Power-on	
5:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
6	OpenDrain		MCICMD output control.	0
7	Rod		Rod control.	0
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

When the external power supply is switched on, the software first enters the power-up phase, and waits until the supply output is stable before moving to the power-on phase. During the power-up phase, MCIPWR is set HIGH. The card bus outlets are disabled during both phases.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

5.3 Clock Control Register (MCIClock - 0xE008 C004)

The MCIClock register controls the MCICLK output. Table 20–407 shows the bit assignment of the clock control register.

Table 407: Clock Control register (MCIClock - address 0xE008 C004) bit description

Bit	Symbol	Value	Description	Reset Value
7:0	ClkDiv		MCI bus clock period: MCLCLK frequency = MCLK / [2×(ClkDiv+1)].	0x00
8	Enable		Enable MCI bus clock:	0
		0	Clock disabled.	
		1	Clock enabled.	
9	PwrSave		Disable MCI clock output when bus is idle:	0
		0	Always enabled.	
		1	Clock enabled when bus is active.	
10	Bypass		Enable bypass of clock divide logic:	0
		0	Disable bypass.	
		1	Enable bypass. MCLK driven to card bus output (MCICLK).	
11	WideBus		Enable wide bus mode:	0
		0	Standard bus mode (only MCIDAT0 used).	
		1	Wide bus mode (MCIDAT3:0 used)	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

While the MCI is in identification mode, the MCICLK frequency must be less than 400 kHz. The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

5.4 Argument Register (MCIArgument - 0xE008 C008)

The MCIArgument register contains a 32 bit command argument, which is sent to a card as part of a command message. Table 20–408 shows the bit assignment of the MCIArgument register.

Table 408: Argument register (MCIArgument - address 0xE008 C008) bit description

Bit	Symbol	Description	Reset Value
31:0	CmdArg	Command argument	0x0000 0000

If a command contains an argument, it must be loaded into the argument register before writing a command to the command register.

5.5 Command Register (MCICommand - 0xE008 C00C)

The MCICommand register contains the command index and command type bits:

- The command index is sent to a card as part of a command message.
- The command type bits control the Command Path State Machine (CPSM). Writing 1 to the enable bit starts the command send operation, while clearing the bit disables the CPSM.

Table 20–409 shows the bit assignment of the MCICommand register.

Table 409: Command register (MCICommand - address 0xE008 C00C) bit description

Bit	Symbol	Description	Reset Value
5:0	CmdIndex	Command index.	0
6	Response	If set, CPSM waits for a response.	0
7	LongRsp	If set, CPSM receives a 136 bit long response.	0
8	Interrupt	If set, CPSM disables command timer and waits for interrupt request.	0
9	Pending	If set, CPSM waits for CmdPend before it starts sending a command.	0
10	Enable	If set, CPSM is enabled.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Table 20–410 shows the response types.

Table 410: 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.

5.6 Command Response Register (MCIRespCommand - 0xE008 C010)

The MCIRespCommand register contains the command index field of the last command response received. Table 20–409 shows the bit assignment of the MCIRespCommand register.

Table 411: Command Response register (MCIRespCommand - address 0xE008 C010) bit description

Bit	Symbol	Description	Reset Value
5:0	RespCmd	Response command index	0x00
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

If the command response transmission does not contain the command index field (long response), the RespCmd field is unknown, although it must contain 111111 (the value of the reserved field from the response).

5.7 Response Registers (MCIResponse0-3 - 0xE008 C014, E008 C018, E008 C01C and E008 C020)

The MCIResponse0-3 registers contain the status of a card, which is part of the received response. Table 20–412 shows the bit assignment of the MCIResponse0-3 registers.

Table 412: Response registers (MCIResponse0-3 - addresses 0xE008 0014, 0xE008 C018, 0xE008 001C and 0xE008 C020) bit description

Bit	Symbol	Description	Reset Value
31:0	Status	Card status	0x0000 0000

The card status size can be 32 or 127 bits, depending on the response type (see Table 20–413).

Table 413: Response Register Type

Description	Short Response	Long Response
MCIResponse0	Card status [31:0]	Card status [127:96]
MCIResponse1	Unused	Card status [95:64]
MCIResponse2	Unused	Card status [63:32]
MCIResponse3	Unused	Card status [31:1]

The most significant bit of the card status is received first. The MCIResponse3 register LSBit is always 0.

5.8 Data Timer Register (MCIDataTimer - 0xE008 C024)

The MCIDataTimer register contains the data timeout period, in card bus clock periods. Table 20–414 shows the bit assignment of the MCIDataTimer register.

Table 414: Data Timer register (MCIDataTimer - address 0xE008 C024) bit description

Bit	Symbol	Description	Reset Value
31:0	DataTime	Data timeout period.	0x0000 0000

A counter loads the value from the data timer register, and starts decrementing when the Data Path State Machine (DPSM) enters the WAIT_R or BUSY state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

A data transfer must be written to the data timer register and the data length register before being written to the data control register.

5.9 Data Length Register (MCIDataLength - 0xE008 C028)

The MCIDataLength register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts. Table 20–415 shows the bit assignment of the MCIDataLength register.

Table 415: Data Length register (MCIDataLength - address 0xE008 C028) bit description

Bit	Symbol	Description	Reset Value
15:0	DataLength	Data length value	0x0000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

For a block data transfer, the value in the data length register must be a multiple of the block size (see Section 20–5.10 “Data Control Register (MCIDataCtrl - 0xE008 C02C)”).

To initiate a data transfer, write to the data timer register and the data length register before writing to the data control register.

5.10 Data Control Register (MCIDataCtrl - 0xE008 C02C)

The MCIDataCtrl register controls the DPSM. Table 20–416 shows the bit assignment of the MCIDataCtrl register.

Table 416: Data Control register (MCIDataCtrl - address 0xE008 C02C) bit description

Bit	Symbol	Value	Description	Reset Value
0	Enable		Data transfer enable.	0
1	Direction		Data transfer direction:	0
		0	From controller to card.	
		1	From card to controller.	
2	Mode		Data transfer mode:	0
		0	Block data transfer.	
		1	Stream data transfer.	
3	DMAEnable		Enable DMA:	0
		0	DMA disabled.	
		1	DMA enabled.	
7:4	BlockSize		Data block length	0
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Data transfer starts if 1 is written to the enable bit. Depending on the direction bit, the DPSM moves to the WAIT_S or WAIT_R state. It is not necessary to clear the enable bit after the data transfer. BlockSize controls the data block length if Mode is 0, as shown in Table 20–417.

Table 417: Data Block Length

Block Size	Block Length
0	2^0= 1 byte.
1	2^1 = 2 bytes.
...	-
11	2^{11} = 2048 bytes.
12:15	Reserved.

5.11 Data Counter Register (MCIDataCnt - 0xE008 C030)

The MCIDataCnt register loads the value from the data length register (see Section 20–5.9 “Data Length Register (MCIDataLength - 0xE008 C028)”) when the DPSM moves from the IDLE state to the WAIT_R or WAIT_S state. As data is transferred, the counter

decrements the value until it reaches 0. The DPSM then moves to the IDLE state and the data status end flag is set. Table 20–418 shows the bit assignment of the MCIDataCnt register.

Table 418: Data Counter register (MCIDataCnt - address 0xE008 C030) bit description

Bit	Symbol	Description	Reset Value
15:0	DataCount	Remaining data	0x0000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Note: This register should be read only when the data transfer is complete.

5.12 Status Register (MCIStatus - 0xE008 C034)

The MCIStatus register is a read-only register. It contains two types of flag:

- Static [10:0]: These remain asserted until they are cleared by writing to the Clear register (see Section 20–5.13 “Clear Register (MCIClear - 0xE008 C038)”).
- Dynamic [21:11]: These change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO).

Table 20–419 shows the bit assignment of the MCIStatus register.

Table 419: Status register (MCIStatus - address 0xE008 C034) bit description

Bit	Symbol	Description	Reset Value
0	CmdCrcFail	Command response received (CRC check failed).	0
1	DataCrcFail	Data block sent/received (CRC check failed).	0
2	CmdTimeOut	Command response timeout.	0
3	DataTimeOut	Data timeout.	0
4	TxUnderrun	Transmit FIFO underrun error.	0
5	RxOverrun	Receive FIFO overrun error.	0
6	CmdRespEnd	Command response received (CRC check passed).	0
7	CmdSent	Command sent (no response required).	0
8	DataEnd	Data end (data counter is zero).	0
9	StartBitErr	Start bit not detected on all data signals in wide bus mode.	0
10	DataBlockEnd	Data block sent/received (CRC check passed).	0
11	CmdActive	Command transfer in progress.	0
12	TxActive	Data transmit in progress.	0
13	RxActive	Data receive in progress.	0
14	TxFifoHalfEmpty	Transmit FIFO half empty.	0
15	RxFifoHalfFull	Receive FIFO half full.	0
16	TxFifoFull	Transmit FIFO full.	0
17	RxFifoFull	Receive FIFO full.	0
18	TxFifoEmpty	Transmit FIFO empty.	0
19	RxFifoEmpty	Receive FIFO empty.	0

Table 419: Status register (MCIStatus - address 0xE008 C034) bit description

Bit	Symbol	Description	Reset Value
20	TxDataAvlbl	Data available in transmit FIFO.	0
21	RxDataAvlbl	Data available in receive FIFO.	0
31:22	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.13 Clear Register (MCIClear - 0xE008 C038)

The MCIClear 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 20–420 shows the bit assignment of the MCIClear register.

Table 420: Clear register (MCIClear - address 0xE008 C038) bit description

Bit	Symbol	Description	Reset Value
0	CmdCrcFailClr	Clears CmdCrcFail flag.	-
1	DataCrcFailClr	Clears DataCrcFail flag.	-
2	CmdTimeOutClr	Clears CmdTimeOut flag.	-
3	DataTimeOutClr	Clears DataTimeOut flag.	-
4	TxUnderrunClr	Clears TxUnderrun flag.	-
5	RxOverrunClr	Clears RxOverrun flag.	-
6	CmdRespEndClr	Clears CmdRespEnd flag.	-
7	CmdSentClr	Clears CmdSent flag.	-
8	DataEndClr	Clears DataEnd flag.	-
9	StartBitErrClr	Clears StartBitErr flag.	-
10	DataBlockEndClr	Clears DataBlockEnd flag.	-
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.14 Interrupt Mask Registers (MCIMask0 - 0xE008 C03C and MCIMask1 - 0xE008 C0040)

The interrupt mask registers determine which status flags generate an interrupt request by setting the corresponding bit to 1. Table 20–421 shows the bit assignment of the MCIMaskx registers.

Table 421: Interrupt Mask registers (MCIMask0 - address 0xE008 C03C and MCIMask1 - address 0xE008 C040) bit description

Bit	Symbol	Description	Reset Value
0	Mask0	Mask CmdCrcFail flag.	0
1	Mask1	Mask DataCrcFail flag.	0
2	Mask2	Mask CmdTimeOut flag.	0
3	Mask3	Mask DataTimeOut flag.	0
4	Mask4	Mask TxUnderrun flag.	0
5	Mask5	Mask RxOverrun flag.	0

Table 421: Interrupt Mask registers (MCIMask0 - address 0xE008 C03C and MCIMask1 - address 0xE008 C040) bit description

Bit	Symbol	Description	Reset Value
6	Mask6	Mask CmdRespEnd flag.	0
7	Mask7	Mask CmdSent flag.	0
8	Mask8	Mask DataEnd flag.	0
9	Mask9	Mask StartBitErr flag.	0
10	Mask10	Mask DataBlockEnd flag.	0
11	Mask11	Mask CmdActive flag.	0
12	Mask12	Mask TxActive flag.	0
13	Mask13	Mask RxActive flag.	0
14	Mask14	Mask TxFifoHalfEmpty flag.	0
15	Mask15	Mask RxFifoHalfFull flag.	0
16	Mask16	Mask TxFifoFull flag.	0
17	Mask17	Mask RxFifoFull flag.	0
18	Mask18	Mask TxFifoEmpty flag.	0
19	Mask19	Mask RxFifoEmpty flag.	0
20	Mask20	Mask TxDataAvlbl flag.	0
21	Mask21	Mask RxDataAvlbl flag.	0
31:22	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.15 FIFO Counter Register (MCIFifoCnt - 0xE008 C048)

The MCIFifoCnt register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register (see Section 20–5.9 “Data Length Register (MCIDataLength - 0xE008 C028)”) when the Enable bit is set in the data control register. If the data length is not word aligned (multiple of 4), the remaining 1 to 3 bytes are regarded as a word. Table 20–422 shows the bit assignment of the MCIFifoCnt register.

Table 422: FIFO Counter register (MCIFifoCnt - address 0xE008 C048) bit description

Bit	Symbol	Description	Reset Value
14:0	DataCount	Remaining data	0x0000
31:15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.16 Data FIFO Register (MCIFIFO - 0xE008 C080 to 0xE008 C0BC)

The receive and transmit FIFOs can be read or written as 32 bit wide registers. The FIFOs contain 16 entries on 16 sequential addresses. This allows the microprocessor to use its load and store multiple operands to read/write to the FIFO. Table 20–423 shows the bit assignment of the MCIFIFO register.

Table 423: Data FIFO register (MCIFIFO - address 0xE008 C080 to 0xE008 C0BC) bit description

Bit	Symbol	Description	Reset Value
31:0	Data	FIFO data.	0x0000 0000

1. Features

- Standard I2C compliant bus interfaces that may be configured as Master, Slave, or Master/Slave.
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock to allow adjustment of I2C transfer rates.
- Bidirectional data transfer between masters and slaves.
- 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.
- The I2C bus may be used for test and diagnostic purposes.

2. Applications

Interfaces to external I2C standard parts, such as serial RAMs, LCDs, tone generators, etc.

3. Description

A typical I2C bus configuration is shown in Figure 21–94. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I2C 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 three I2C interfaces on the LPC2400 is byte oriented, and has four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The three I2C interfaces are identical except for the pin I/O characteristics. I2C0 complies with entire I2C specification, supporting the ability to turn power off to the LPC2400 without causing a problem with other devices on the same I2C bus (see "The I2C-bus specification" description under the heading "Fast-Mode", and notes for the table titled

"Characteristics of the SDA and SCL I/O stages for F/S-mode I2C-bus devices"). This is sometimes a useful capability, but intrinsically limits alternate uses for the same pins if the I2C interface is not used. Seldom is this capability needed on multiple I2C interfaces within the same microcontroller. Therefore, I2C1 and I2C2 are implemented using standard port pins, and do not support the ability to turn power off to the LPC2400 while leaving the I2C bus functioning between other devices. This difference should be considered during system design while assigning uses for the I2C interfaces.

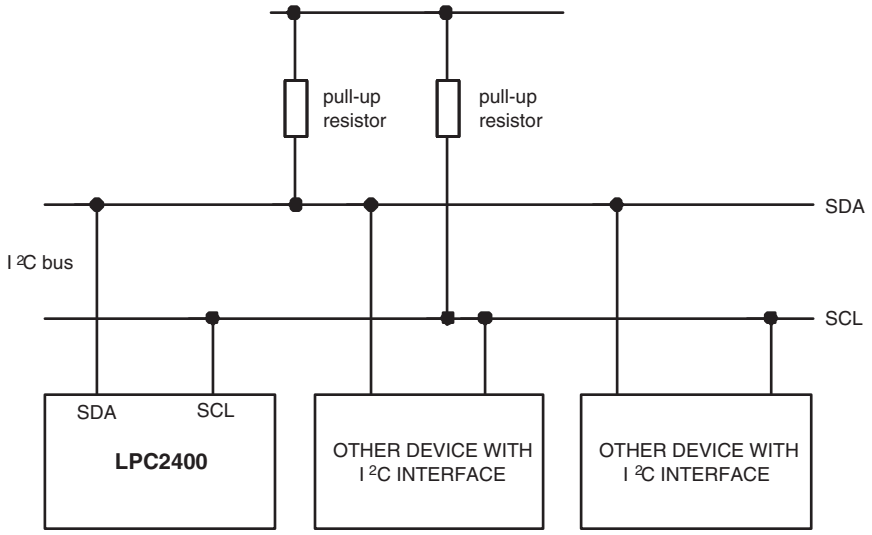


Fig 94. I2C bus configuration

4. Pin description

Table 424. I2C Pin Description

Pin	Type	Description
SDA0,1, 2	Input/Output	I2C Serial Data
SCL0,1, 2	Input/Output	I2C Serial Clock

5. I2C operating modes

In a given application, the I2C block may operate as a master, a slave, or both. In the slave mode, the I2C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I2C block switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

5.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 21–425. I2EN must be set to 1 to enable the I2C function. If the AA bit is 0, the I2C interface will not

acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI Bit is cleared by writing 1 to the SIC bit in the I2CONCLR register.

Table 425. I2CnCONSET used to configure Master mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	0	-	-

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I^2C interface will enter master transmitter mode when software sets the STA bit. The I^2C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the I2STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 0x18, 0x20, or 0x38 for the master mode, or 0x68, 0x78, or 0xB0 if the slave mode was enabled (by setting AA to 1). The appropriate actions to be taken for each of these status codes are shown in Table 21–440 to Table 21–443.

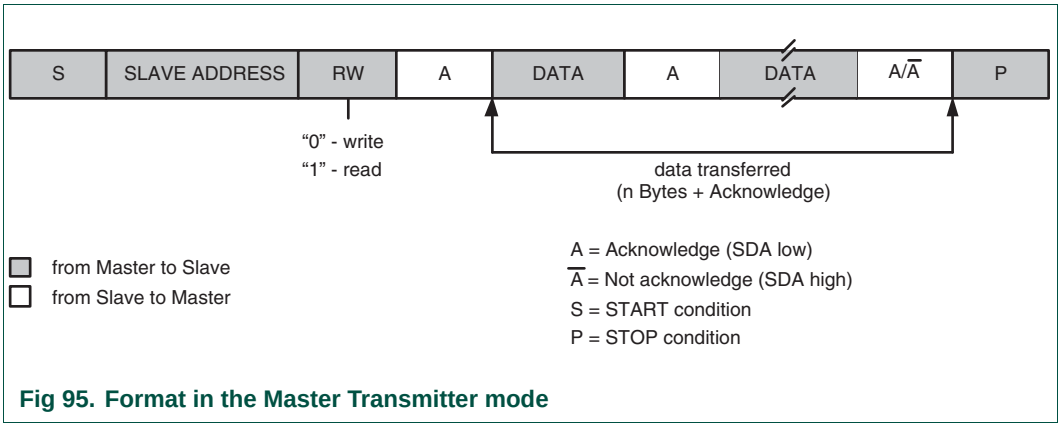


Fig 95. Format in the Master Transmitter mode

5.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^2C Data Register (I2DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 21–441.

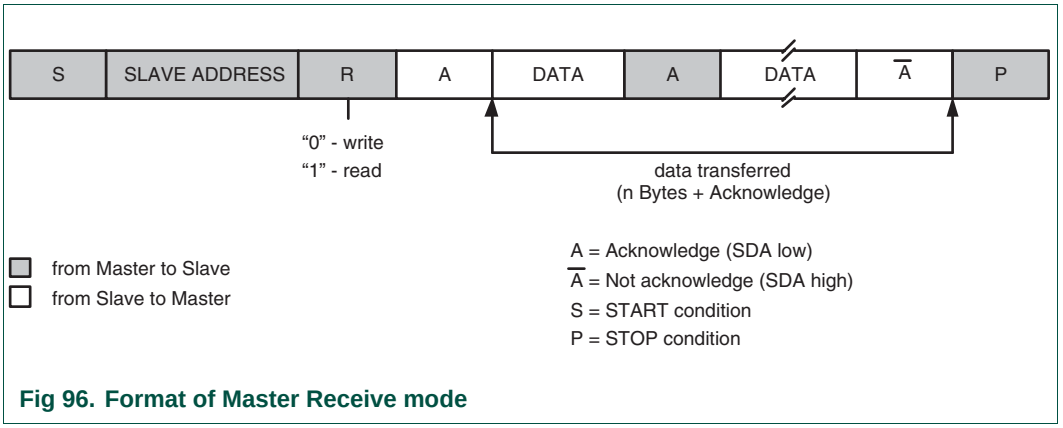


Fig 96. Format of Master Receive mode

After a repeated START condition, I2C may switch to the master transmitter mode.

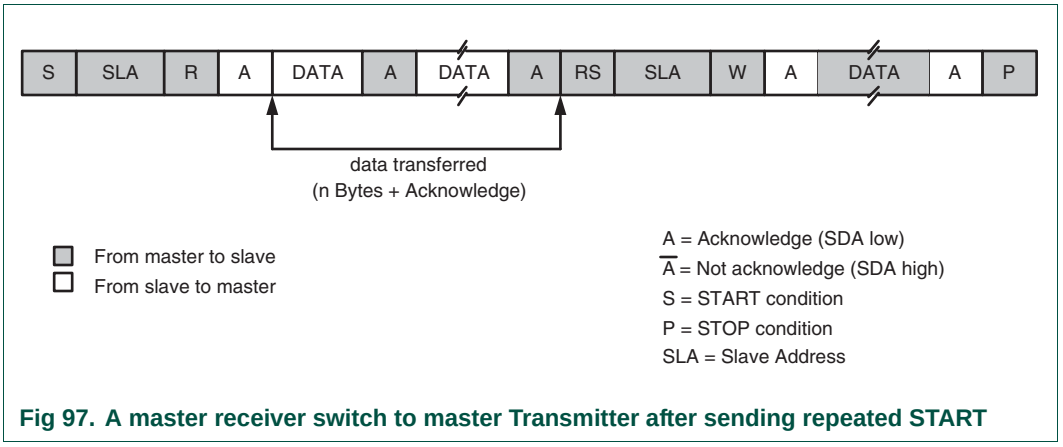


Fig 97. A master receiver switch to master Transmitter after sending repeated START

5.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, user write the Slave Address Register (I2ADR) and write the I2C Control Set Register (I2CONSET) as shown in Table 21–426.

Table 426. I2CnCONSET used to configure Slave mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

I2EN must be set to 1 to enable the I2C function. AA bit must be set to 1 to acknowledge its own slave address or the general call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I2C interface waits until it is addressed by its own address or general address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter

mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status Register (I2STAT). Refer to Table 21–442 for the status codes and actions.

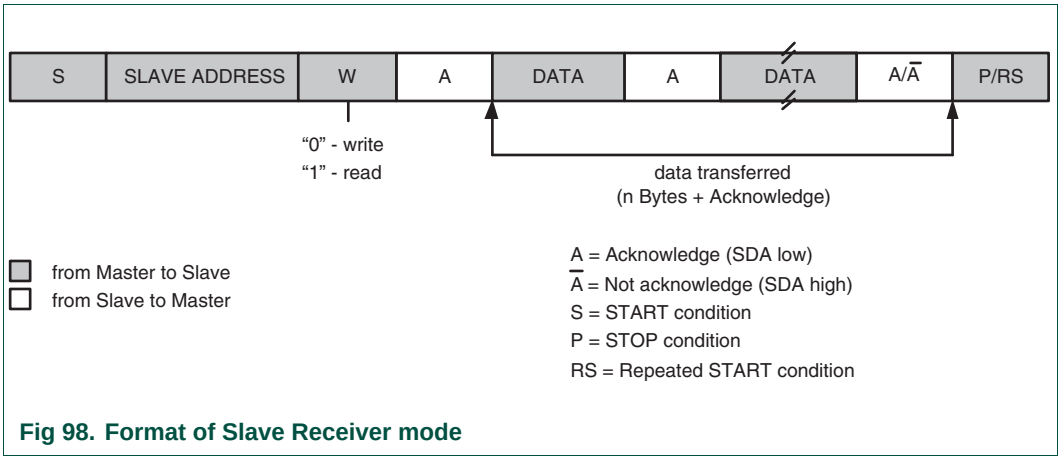


Fig 98. Format of Slave Receiver mode

5.4 Slave Transmitter mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^2C interface switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

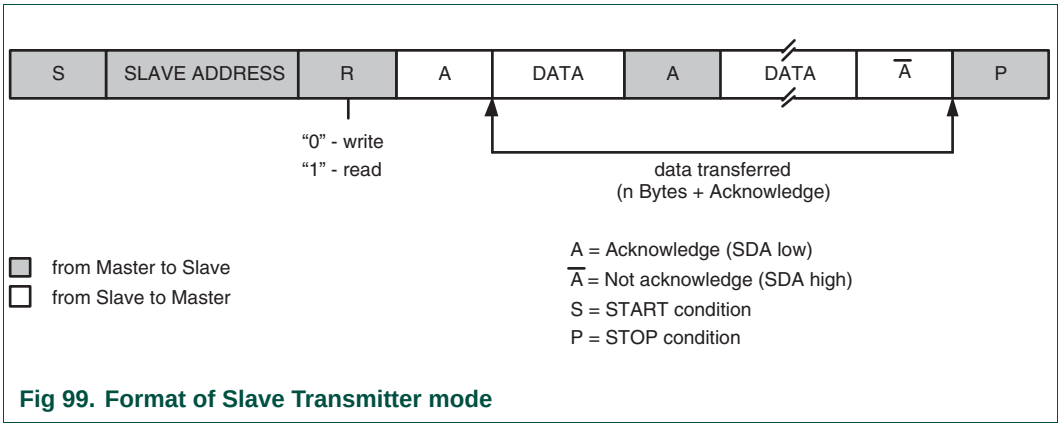


Fig 99. Format of Slave Transmitter mode

6. I2C implementation and operation

6.1 Input filters and output stages

Input signals are synchronized with the internal clock , and spikes shorter than three clocks are filtered out.

The output for I2C is a special pad designed to conform to the I2C specification. The outputs for I2C1 and I2C2 are standard port I/Os that support a subset of the full I2C specification.

Figure 21–100 shows how the on-chip I2C bus interface is implemented, and the following text describes the individual blocks.

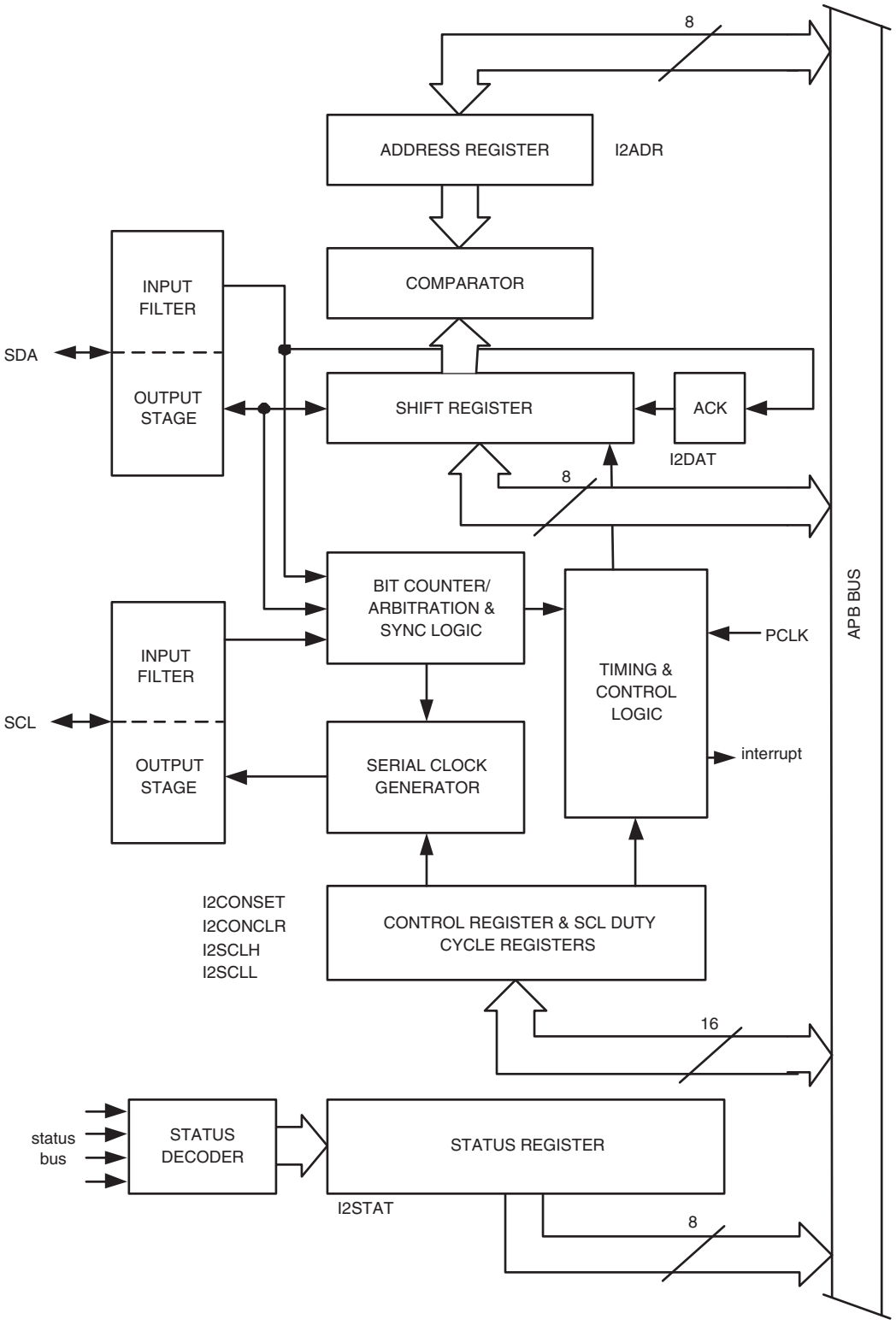


Fig 100.I²C Bus serial interface block diagram

6.2 Address Register I2ADDR

This register may be loaded with the 7 bit slave address (7 most significant bits) to which the I2C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable general call address (0x00) recognition.

6.3 Comparator

The comparator compares the received 7 bit slave address with its own slave address (7 most significant bits in I2ADR). It also compares the first received 8 bit byte with the general call address (0x00). If an equality is found, the appropriate status bits are set and an interrupt is requested.

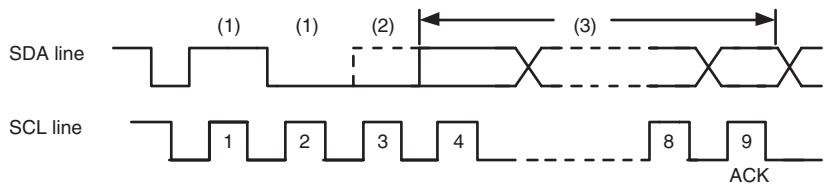
6.4 Shift register I2DAT

This 8 bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in I2DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in I2DAT.

6.5 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I2C bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I2C block immediately changes from master transmitter to slave receiver. The I2C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I2C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, the I2C block generates no further clock pulses. Figure 21–101 shows the arbitration procedure.



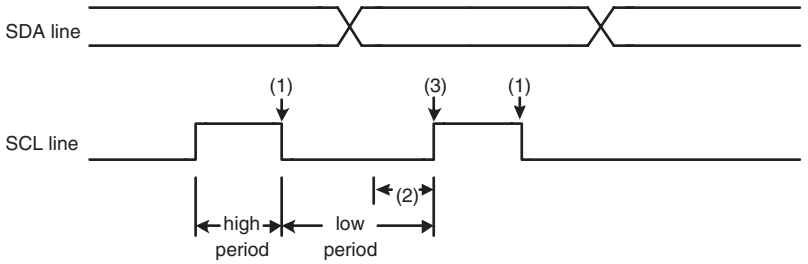
(1) A device transmits serial data.

(2) Another device overrules a logic 1 (dotted line), transmitted by this I2C master, by pulling the SDA line low. Arbitration is lost, and this I2C enters Slave Receiver mode.

(3) This I2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 101.Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 21–102 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I2C has timed a complete low time and released SCL. The I2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 102.Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

6.6 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I2C block is in the master transmitter or master receiver mode. It is switched off when the I2C block is in a slave mode. The I2C output clock frequency and duty cycle is programmable via the I2C Clock Control Registers. See the description of the I2CSCLL and I2CSCLH registers for details. The output clock pulses have a duty cycle as programmed unless the bus is synchronizing with other SCL clock sources as described above.

6.7 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for I2DAT, enables the comparator, generates and detects start and stop conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I2C bus status.

6.8 Control register I2CONSET and I2CONCLR

The I2C control register contains bits used to control the following I2C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I2C control register may be read as I2CONSET. Writing to I2CONSET will set bits in the I2C control register that correspond to ones in the value written. Conversely, writing to I2CONCLR will clear bits in the I2C control register that correspond to ones in the value written.

6.9 Status decoder and status register

The status decoder takes all of the internal status bits and compresses them into a 5 bit code. This code is unique for each I2C bus status. The 5 bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I2C block are used. The 5 bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

7. Register description

Each I2C interface contains 7 registers as shown in Table 21–427 below.

Table 427. I2C register map

Generic Name	Description	Access	Reset value[1]	I2Cn Register Name & Address
I2CONSET	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I2C control register is set. Writing a zero has no effect on the corresponding bit in the I2C control register.	R/W	0x00	I2C0CONSET - 0xE001 C000 I2C1CONSET - 0xE005 C000 I2C2CONSET - 0xE008 0000
I2STAT	**I2C Status Register.** During I2C operation, this register provides detailed status codes that allow software to determine the next action needed.	RO	0xF8	I2C0STAT - 0xE001 C004 I2C1STAT - 0xE005 C004 I2C2STAT - 0xE008 0004
I2DAT	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	R/W	0x00	I2C0DAT - 0xE001 C008 I2C1DAT - 0xE005 C008 I2C2DAT - 0xE008 0008
I2ADR	**I2C Slave Address Register.** Contains the 7 bit slave address for operation of the I2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the general call address.	R/W	0x00	I2C0ADR - 0xE001 C00C I2C1ADR - 0xE005 C00C I2C2ADR - 0xE008 000C

Table 427. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Register Name & Address
I2SCLH	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I²C clock.	R/W	0x04	I2C0SCLH - 0xE001 C010 I2C1SCLH - 0xE005 C010 I2C2SCLH - 0xE008 0010
I2SCLL	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I²C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I²C master and certain times used in slave mode.	R/W	0x04	I2C0SCLL - 0xE001 C014 I2C1SCLL - 0xE005 C014 I2C2SCLL - 0xE008 0014
I2CONCLR	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is cleared. Writing a zero has no effect on the corresponding bit in the I²C control register.	WO	NA	I2C0CONCLR - 0xE001 C018 I2C1CONCLR - 0xE005 C018 I2C2CONCLR - 0xE008 0018

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

7.1 I²C Control Set Register (I2C[0/1/2]CONSET: 0xE001 C000, 0xE005 C000, 0xE008 0000)

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I²C interface. Writing a one to a bit of this register causes the corresponding bit in the I²C control register to be set. Writing a zero has no effect.

Table 428. I²C Control Set Register (I2C[0/1/2]CONSET - addresses: 0xE001 C000, 0xE005 C000, 0xE008 0000) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag. See the text below.	
3	SI	I²C interrupt flag.	0
4	STO	STOP flag. See the text below.	0
5	STA	START flag. See the text below.	0
6	I2EN	I²C interface enable. See the text below.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I2EN I²C Interface Enable. When I2EN is 1, the I²C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I²C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I²C block is in the “not addressed” slave state, and the STO bit is forced to “0”.

I2EN should not be used to temporarily release the I²C bus since, when I2EN is reset, the I²C bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I²C interface to enter master mode and transmit a START condition or transmit a repeated START condition if it is already in master mode.

When STA is 1 and the I2C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I2C interface is already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I2C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the I2CONCLR register. When STA is 0, no START condition or repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I2C bus if it the interface is in master mode, and transmits a START condition thereafter. If the I2C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

STO is the STOP flag. Setting this bit causes the I2C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I2C bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I2C Interrupt Flag. This bit is set when the I2C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is high, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in I2CONCLR register.

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. The address in the Slave Address Register has been received.
2. The general call address has been received while the general call bit (GC) in I2ADR is set.
3. A data byte has been received while the I2C is in the master receiver mode.
4. A data byte has been received while the I2C is in the addressed slave receiver mode.

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I2C is in the master receiver mode.
2. A data byte has been received while the I2C is in the addressed slave receiver mode.

7.2 I2C Control Clear Register (I2C[0/1/2]CONCLR: 0xE001 C018, 0xE005 C018, 0xE008 0018)

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I2C interface. Writing a one to a bit of this register causes the corresponding bit in the I2C control register to be cleared. Writing a zero has no effect.

Table 429. I2C Control Set Register (I2C[0/1/2]CONCLR - addresses 0xE001 C018, 0xE005 C018, 0xE008 0018) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert acknowledge Clear bit.	
3	SIC	I2C interrupt Clear bit.	0
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	START flag Clear bit.	0
6	I2ENC	I2C interface Disable bit.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

AAC is the Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.

SIC is the I2C Interrupt Clear bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.

STAC is the Start flag Clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.

I**2ENC** is the I2C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.

7.3 I2C Status Register (I2C[0/1/2]STAT - 0xE001 C004, 0xE005 C004, 0xE008 0004)

Each I2C Status register reflects the condition of the corresponding I2C interface. The I2C Status register is Read-Only.

Table 430. I2C Status Register (I2C[0/1/2]STAT - addresses 0xE001 C004, 0xE005 C004, 0xE008 0004) bit description

Bit	Symbol	Description	Reset Value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I2C interface.	0x1F

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I2C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 21–440 to Table 21–443.

7.4 I2C Data Register (I2C[0/1/2]DAT - 0xE001 C008, 0xE005 C008, 0xE008 0008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

Table 431. I2C Data Register (I2C[0/1/2]DAT - addresses 0xE001 C008, 0xE005 C008, 0xE008 0008) bit description

Bit	Symbol	Description	Reset Value
7:0	Data	This register holds data values that have been received, or are to be transmitted.	0

7.5 I2C Slave Address Register (I2C[0/1/2]ADR - 0xE001 C00C, 0xE005 C00C, 0xE008 000C)

These registers are readable and writable, and is only used when an I2C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the general call bit. When this bit is set, the general call address (0x00) is recognized.

Table 432. I2C Slave Address register (I2C[0/1/2]ADR - addresses 0xE001 C00C, 0xE005 C00C, 0xE008 000C) bit description

Bit	Symbol	Description	Reset Value
0	GC	General Call enable bit.	0
7:1	Address	The I2C device address for slave mode.	0x00

7.6 I2C SCL High Duty Cycle Register (I2C[0/1/2]SCLH - 0xE001 C010, 0xE0015 C010, 0xE008 0010)

Table 433. I2C SCL High Duty Cycle register (I2C[0/1/2]SCLH - addresses 0xE001 C010, 0xE005 C010, 0xE008 0010) bit description

Bit	Symbol	Description	Reset Value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004

7.7 I2C SCL Low Duty Cycle Register (I2C[0/1/2]SCLL - 0xE001 C014, 0xE0015 C014, 0xE008 0014)

Table 434. I2C SCL Low Duty Cycle register (I2C[0/1/2]SCLL - addresses 0xE001 C014, 0xE005 C014, 0xE008 0014) bit description

Bit	Symbol	Description	Reset Value
15:0	SCLL	Count for SCL LOW time period selection.	0x0004

7.8 Selecting the appropriate I2C data rate and duty cycle

Software must set values for the registers I2SCLH and I2SCLL to select the appropriate data rate and duty cycle. I2SCLH defines the number of PCLK cycles for the SCL high time, I2SCLL defines the number of PCLK cycles for the SCL low time. The frequency is determined by the following formula (f_{PCLK} being the frequency of PCLK):

(15)

$$I^2C_{bitfrequency} = \frac{f_{PCLK}}{I2CSCLH + I2CSCLL}$$

The values for I2SCLL and I2SCLH should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I²C bus specification defines the SCL low time and high time at different values for a 400 kHz I²C rate. The value of the register must ensure that the data rate is in the I²C data rate range of 0 through 400 kHz. Each register value must be greater than or equal to 4. Table 21–435 gives some examples of I²C bus rates based on PCLK frequency and I2SCLL and I2SCLH values.

Table 435. Example I²C Clock Rates

I2SCLL + I2SCLH	I²C Bit Frequency (kHz) at PCLK (MHz)						
	1	5	10	16	20	40	60
8	125						
10	100						
25	40	200	400				
50	20	100	200	320	400		
100	10	50	100	160	200	400	
160	6.25	31.25	62.5	100	125	250	375
200	5	25	50	80	100	200	300
400	2.5	12.5	25	40	50	100	150
800	1.25	6.25	12.5	20	25	50	75

8. Details of I²C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figures 103 to 107. Table 21–436 lists abbreviations used in these figures when describing the I²C operating modes.

Table 436. Abbreviations used to describe an I²C operation

Abbreviation	Explanation
S	Start Condition
SLA	7 bit slave address
R	Read bit (high level at SDA)
W	Write bit (low level at SDA)
A	Acknowledge bit (low level at SDA)

Table 436. Abbreviations used to describe an I²C operation

Abbreviation	Explanation
$\overline{A}$	Not acknowledge bit (high level at SDA)
Data	8 bit data byte
P	Stop condition

In Figures 103 to 107, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the I2STAT register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in I2STAT is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in tables from Table 21–440 to Table 21–444.

8.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 21–103). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 437. I2CONSET used to initialize Master Transmitter mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	x	-	-

The I²C rate must also be configured in the I2SCLL and I2SCLH registers. I2EN must be set to logic 1 to enable the I²C block. If the AA bit is reset, the I²C block will not acknowledge its own slave address or the general call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I²C interface cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I²C logic will now test the I²C bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (I2STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads I2DAT with the slave address and the data direction bit (SLA+W). The SI bit in I2CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 21–440. After a repeated start condition (state 0x10). The I²C block may switch to the master receiver mode by loading I2DAT with SLA+R).

8.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 21–104). The transfer is initialized as in the master transmitter mode. When the start condition has been transmitted, the interrupt service routine must load I2DAT with the 7 bit slave address and the data direction bit (SLA+R). The SI bit in I2CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. These are 0x40, 0x48, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 21–441. After a repeated start condition (state 0x10), the I^2C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

8.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 21–105). To initiate the slave receiver mode, I2ADR and I2CON must be loaded as follows:

Table 438. I2C0ADR and I2C1ADR usage in Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	own slave 7 bit address							GC

The upper 7 bits are the address to which the I^2C block will respond when addressed by a master. If the LSB (GC) is set, the I^2C block will respond to the general call address (0x00); otherwise it ignores the general call address.

Table 439. I2C0CONSET and I2C1CONSET used to initialize Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

The I^2C bus rate settings do not affect the I^2C block in the slave mode. I2EN must be set to logic 1 to enable the I^2C block. The AA bit must be set to enable the I^2C block to acknowledge its own slave address or the general call address. STA, STO, and SI must be reset.

When I2ADR and I2CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^2C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 104. The slave receiver mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^2C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^2C block does not respond to its own slave address or a general call address. However, the I^2C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C bus.

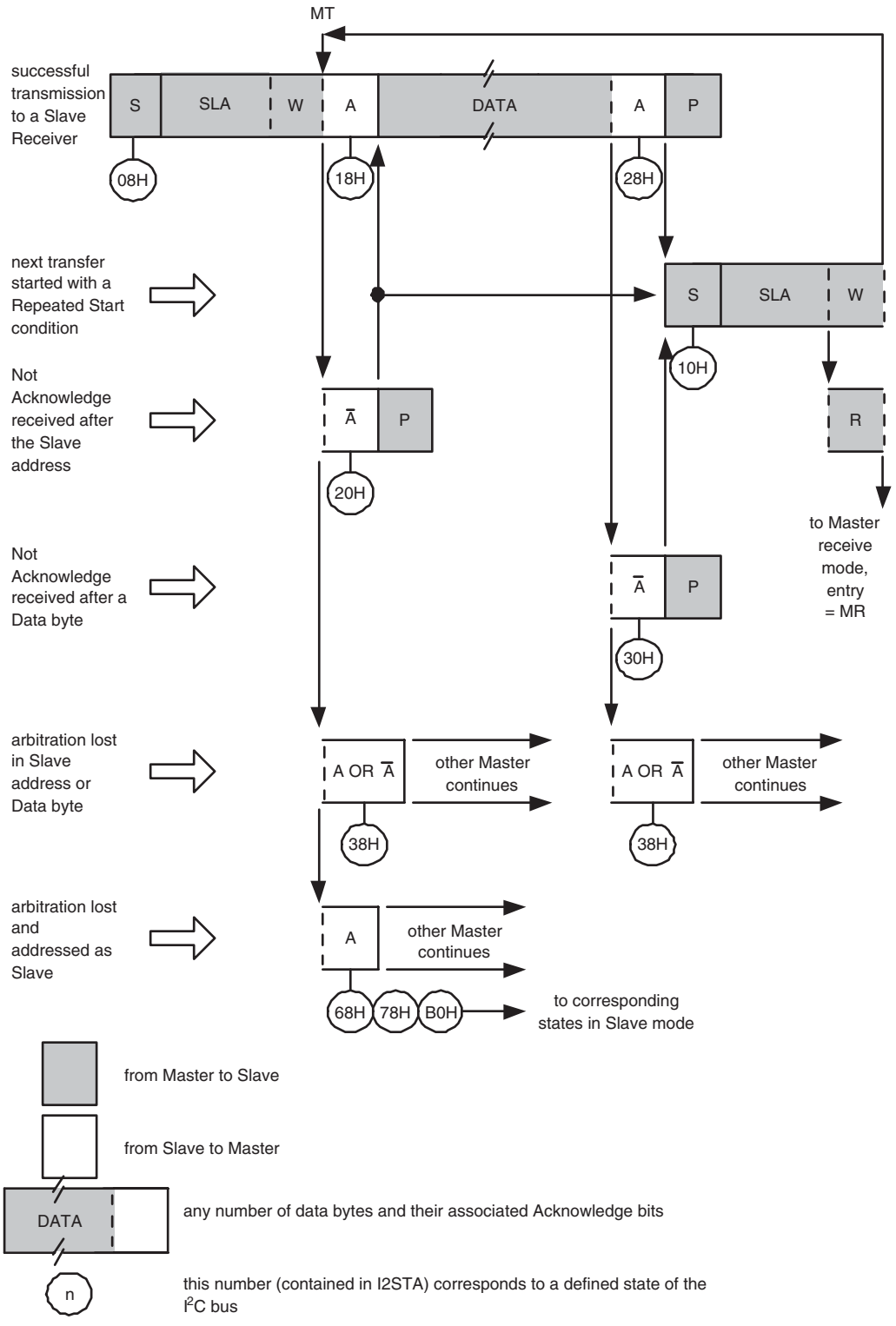


Fig 103.Format and States in the Master Transmitter mode

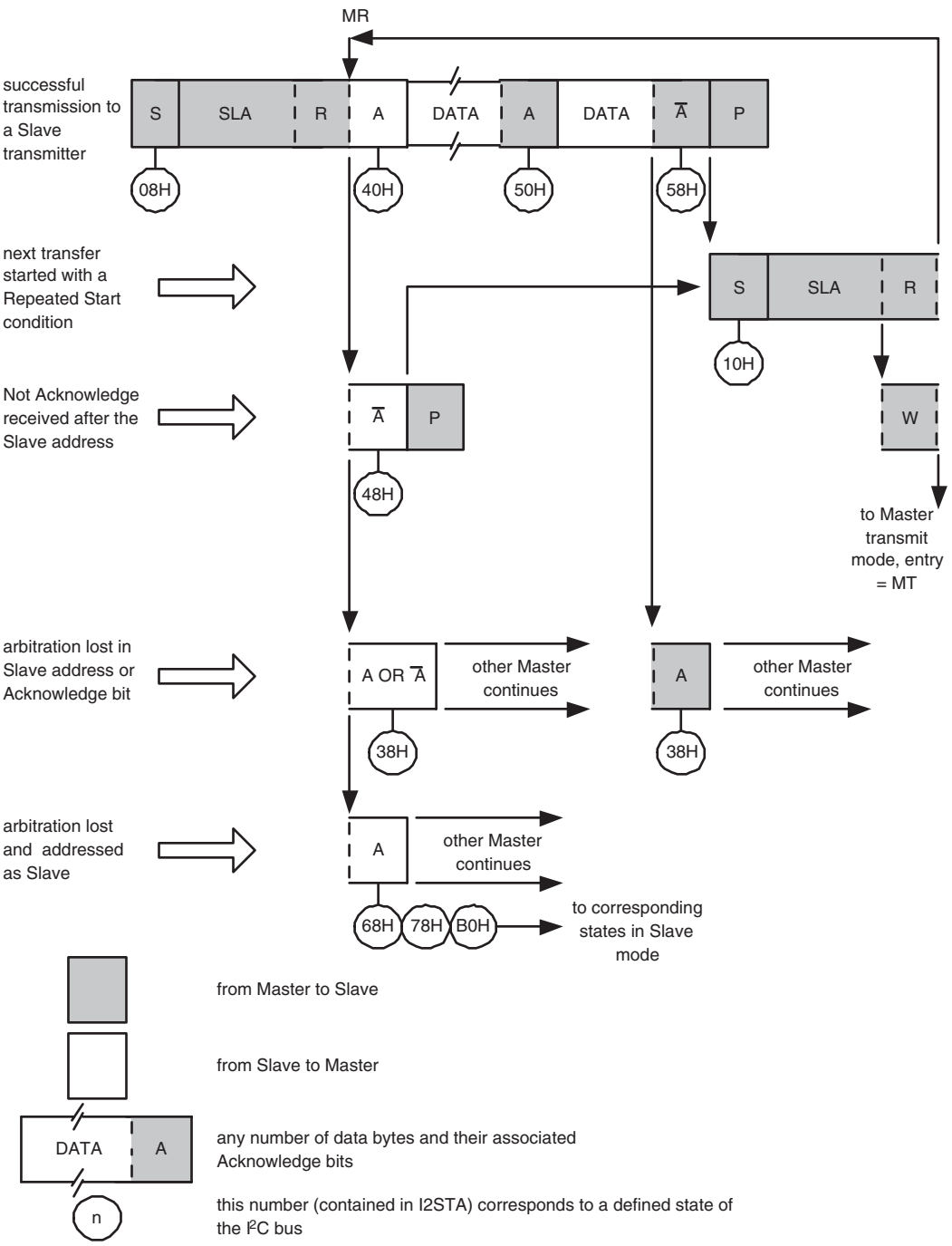


Fig 104.Format and States in the Master Receiver mode

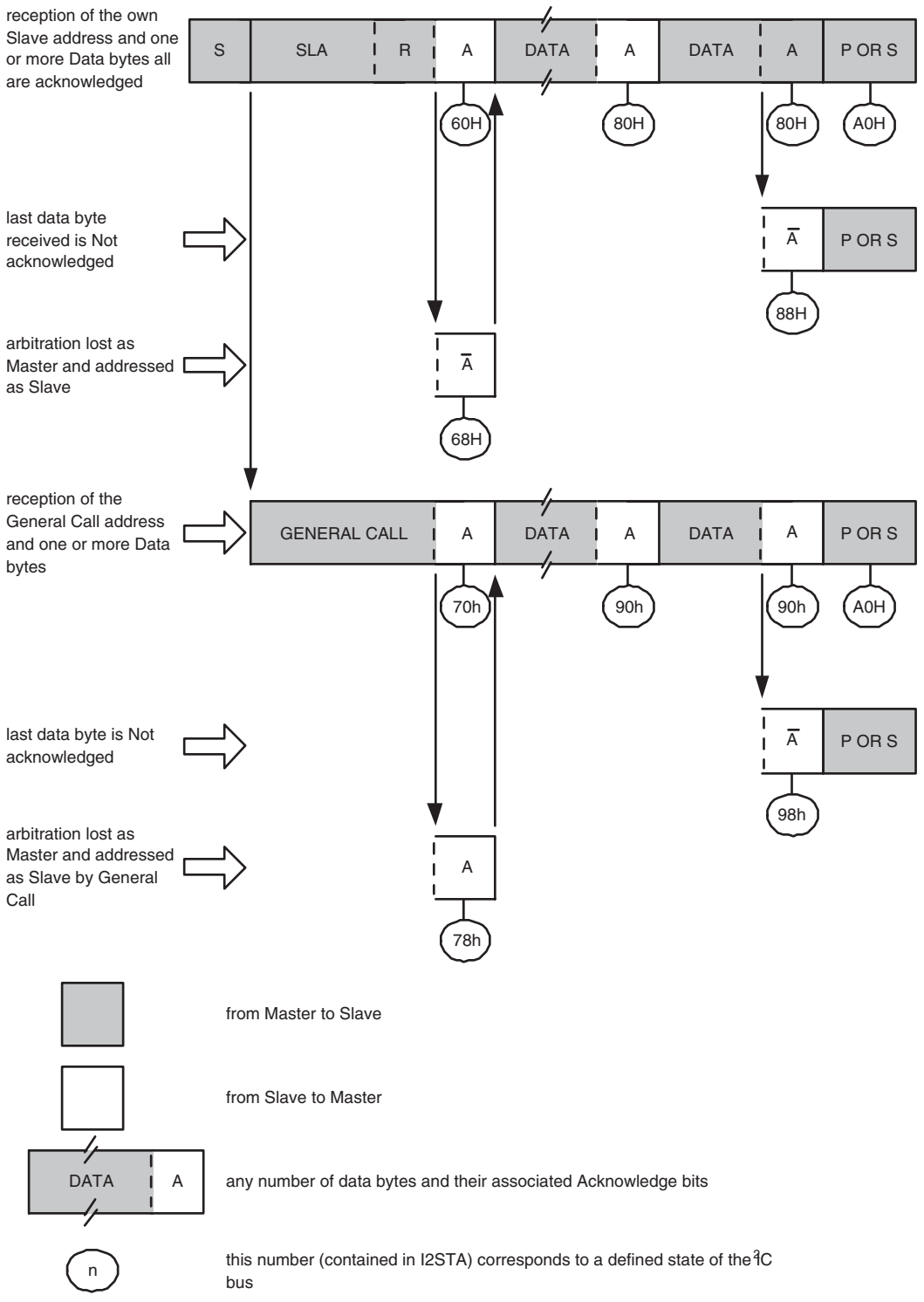


Fig 105.Format and States in the Slave Receiver mode

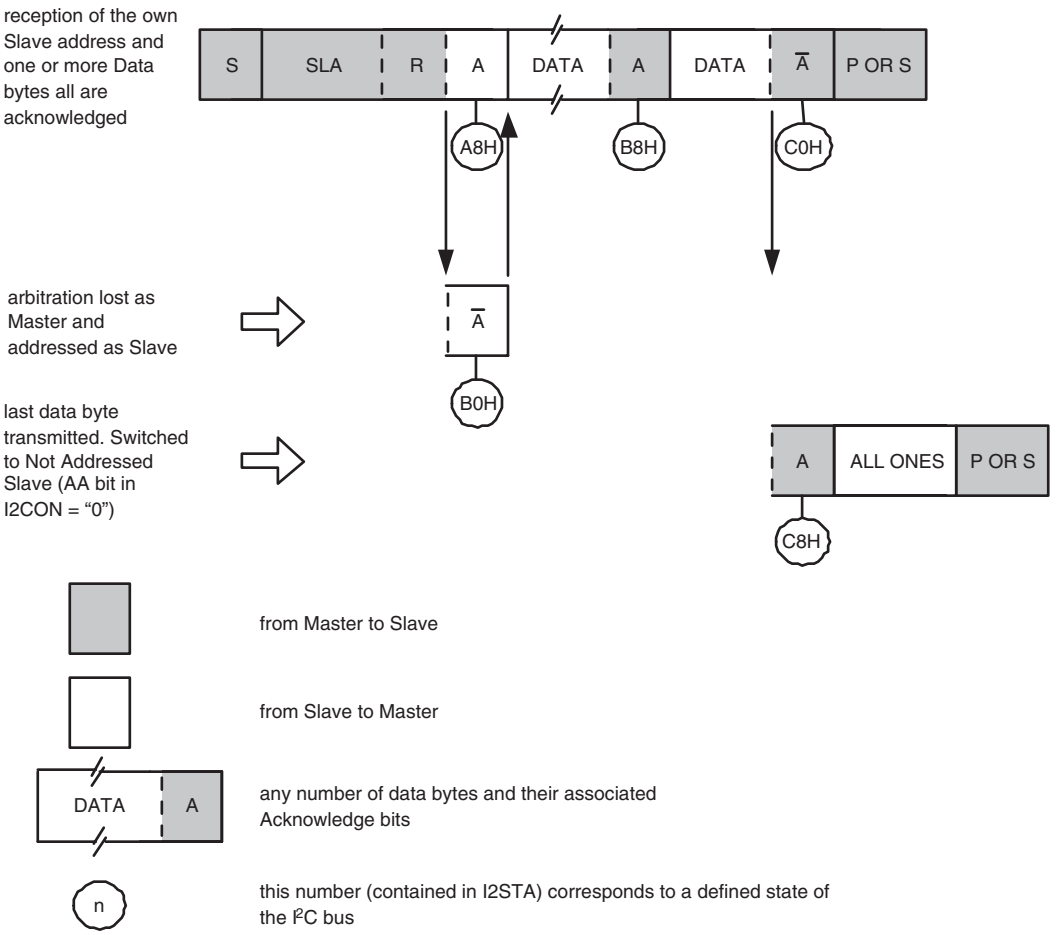


Fig 106.Format and States in the Slave Transmitter mode

8.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 21–106). Data transfer is initialized as in the slave receiver mode. When I2ADR and I2CON have been initialized, the I²C block waits until it is addressed by its own slave address followed by the data direction bit which must be "1" (R) for the I²C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 21–443. The slave transmitter mode may also be entered if arbitration is lost while the I²C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I²C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I²C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I²C block does not respond to its own slave address or a general call address. However, the I²C bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I²C block from the I²C bus.

Table 440. Master Transmitter mode

Status Code (I2CSTAT)	Status of the I²C bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R	X	0	0	X	SLA+W will be transmitted; the I²C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in I2DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No I2DAT action or	0	0	0	X	I²C bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

Table 441. Master Receiver mode

Status Code (I2CSTAT)	Status of the I²C bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I²C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No I2DAT action or	0	0	0	X	I²C bus will be released; the I²C block will enter a slave mode.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No I2DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No I2DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No I2DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 442. Slave Receiver Mode

Status Code (I2CSTAT)	Status of the I²C bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General call address (0x00) has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General call address has been received, ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLV address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Table 442. Slave Receiver Mode

Status Code (I2CSTAT)	Status of the I²C bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

Table 443. Tad_105: Slave Transmitter mode

Status Code (I2CSTAT)	Status of the I²C bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received, ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xC0	Data byte in I2DAT has been transmitted; NOT ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No I2DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in I2DAT has been transmitted (AA = 0); ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No I2DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

8.5 Miscellaneous states

There are two I2STAT codes that do not correspond to a defined I2C hardware state (see Table 21–444). These are discussed below.

21.8.5.1 I2STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I2C block is not involved in a serial transfer.

21.8.5.2 I2STAT = 0x00

This status code indicates that a bus error has occurred during an I2C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I2C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I2C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in I2CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 444. Miscellaneous states

Status Code (I2CSTAT)	Status of the I²C bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xF8	No relevant state information available; SI = 0.	No I2DAT action	No I2CON action				Wait or proceed current transfer.
0x00	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 0x00 can also occur when interference causes the I²C block to enter an undefined state.	No I2DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the I²C block is switched to the not addressed SLV mode. STO is reset.

8.6 Some special cases

The I²C hardware has facilities to handle the following special cases that may occur during a serial transfer:

8.7 Simultaneous repeated START conditions from two masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 21–107). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I²C hardware detects a repeated START condition on the I²C bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, the I²C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

8.8 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 21–101). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 21–103 and Figure 21–104).

If the STA flag in I2CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 0x08) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

8.9 Forced access to the I²C bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I^2C bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I^2C bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I^2C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 34).

8.10 I^2C Bus obstructed by a Low level on SCL or SDA

An I^2C bus hang-up occurs if SDA or SCL is pulled LOW by an uncontrolled source. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the I^2C hardware cannot resolve this type of problem. When this occurs, the problem must be resolved by the device that is pulling the SCL bus line LOW.

If the SDA line is obstructed by another device on the bus (e.g., a slave device out of bit synchronization), the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 21–109). The I^2C hardware transmits additional clock pulses when the STA flag is set, but no START condition can be generated because the SDA line is pulled LOW while the I^2C bus is considered free. The I^2C hardware attempts to generate a START condition after every two additional clock pulses on the SCL line. When the SDA line is eventually released, a normal START condition is transmitted, state 0x08 is entered, and the serial transfer continues.

If a forced bus access occurs or a repeated START condition is transmitted while SDA is obstructed (pulled LOW), the I^2C hardware performs the same action as described above. In each case, state 0x08 is entered after a successful START condition is transmitted and normal serial transfer continues. Note that the CPU is not involved in solving these bus hang-up problems.

8.11 Bus error

A bus error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I^2C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^2C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 21–444.

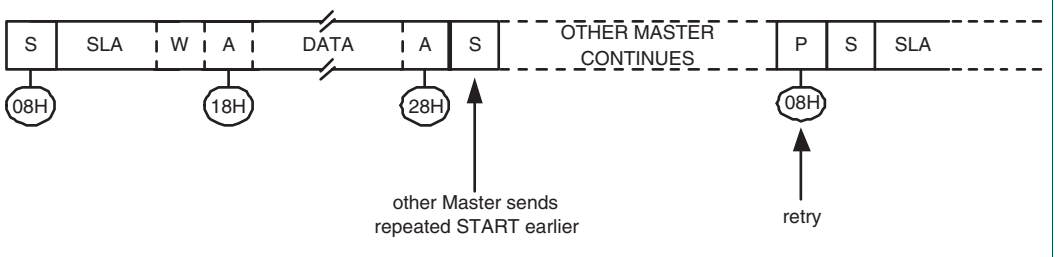


Fig 107.Simultaneous repeated START conditions from 2 masters

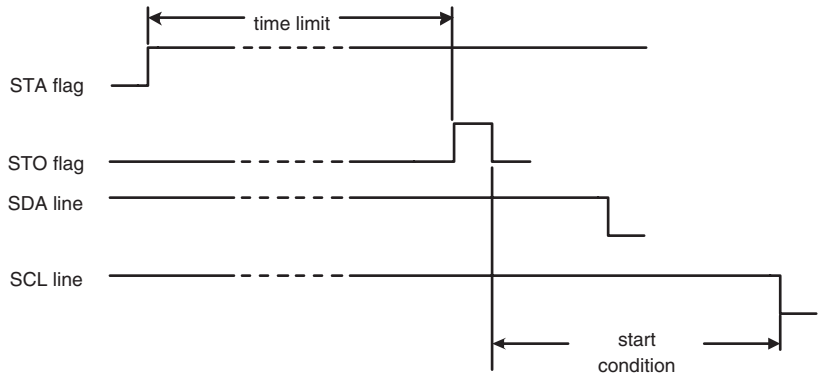
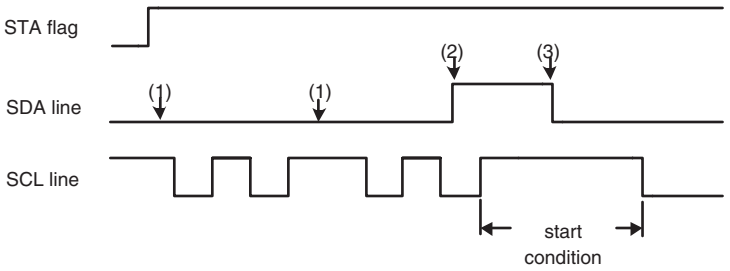


Fig 108.Forced access to a busy I2C bus



(1) Unsuccessful attempt to send a start condition.

(2) SDA line is released.

(3) Successful attempt to send a start condition. State 08H is entered.

Fig 109.Recovering from a bus obstruction caused by a low level on SDA

8.12 I2C State service routines

This section provides examples of operations that must be performed by various I2C state service routines. This includes:

- Initialization of the I2C block after a Reset.
- I2C Interrupt Service.
- The 26 state service routines providing support for all four I2C operating modes.

8.12.1 Initialization

In the initialization example, the I2C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- I2ADR is loaded with the part's own slave address and the general call bit (GC).
- The I2C interrupt enable and interrupt priority bits are set.
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by loading CR0 and CR1 in I2CON. The master routines must be started in the main program.

The I2C hardware now begins checking the I2C bus for its own slave address and general call. If the general call or the own slave address is detected, an interrupt is requested and I2STAT is loaded with the appropriate state information.

8.12.2 I2C interrupt service

When the I2C interrupt is entered, I2STAT contains a status code which identifies one of the 26 state services to be executed.

8.12.3 The state service routines

Each state routine is part of the I2C interrupt routine and handles one of the 26 states.

8.12.4 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I2C state codes. If one or more of the four I2C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of timeout during I2C operations, in order to trap an inoperative bus or a lost service routine.

9. Software example

9.1 Initialization routine

Example to initialize I2C Interface as a Slave and/or Master.

1. Load I2ADR with own Slave Address, enable general call recognition if needed.
2. Enable I2C interrupt.
3. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to I2CONSET.

9.2 Start master transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a Start.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

9.3 Start master receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a Start.

1. Initialize Master data counter.

2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

9.4 I2C interrupt routine

Determine the I2C state and which state routine will be used to handle it.

1. Read the I2C status from I2STA.
2. Use the status value to branch to one of 26 possible state routines.

9.5 Non mode specific states

9.5.1 State : 0x00

Bus Error. Enter not addressed Slave mode and release bus.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.6 Master states

State 08 and State 10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

9.6.1 State : 0x08

A Start condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

9.6.2 State : 0x10

A repeated Start condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.

4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

9.7 Master Transmitter states

9.7.1 State : 0x18

Previous state was State 8 or State 10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted, an ACK bit will be received.

1. Load I2DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

9.7.2 State : 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A Stop condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.7.3 State : 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a Stop condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit
5. Load I2DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to I2CONSET to set the AA bit.
7. Write 0x08 to I2CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

9.7.4 State : 0x30

Data has been transmitted, NOT ACK received. A Stop condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.7.5 State : 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new Start condition will be transmitted when the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.8 Master Receive states

9.8.1 State : 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be

received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.8.2 State : 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A Stop condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.8.3 State : 0x50

Data has been received, ACK has been returned. Data will be read from I2DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from I2DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

9.8.4 State : 0x58

Data has been received, NOT ACK has been returned. Data will be read from I2DAT. A Stop condition will be transmitted.

1. Read data byte from I2DAT into Master Receive buffer.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit

9.9 Slave Receiver states

9.9.1 State : 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

9.9.2 State : 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

9.9.3 State : 0x70

General call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

9.9.4 State : 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.

3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

9.9.5 State : 0x80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

9.9.6 State : 0x88

Previously addressed with own Slave Address . Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.9.7 State : 0x90

Previously addressed with general call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
3. Exit

9.9.8 State : 0x98

Previously addressed with general call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.9.9 State : 0xA0

A Stop condition or repeated Start has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

9.10 Slave Transmitter States

9.10.1 State : 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

9.10.2 State : 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to I2CONSET to set the STA and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

9.10.3 State : 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

9.10.4 State : 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.

3. Exit

9.10.5 State : 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

UM10237

Chapter 22: I²S Interface

Rev. 01 — 18 December 2006 **User manual**

1. Features

The I²S bus provides a standard communication interface for digital audio applications.

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 LPC2400 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 both master and slave mode.
- The I²S output can operate in both master and slave mode, independent of the I²S 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 - 48 kHz. (16, 22.05, 32, 44.1, 48 kHz).
- 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.

2. Description

The I²S performs serial data out via the transmit channel and serial data in via the receive channel. These support the NXP Inter IC Audio format for 8, 16 and 32 bits audio data both for stereo and mono modes. Configuration, data access and control is performed by a APB register set. Data streams are buffered by FIFOs with a depth of 8 bytes.

The I²S receive and transmit stage can operate independently in either slave or master mode. Within the I²S module the difference between these modes lies in the word select (WS) signal which determines the timing of data transmissions. Data words start on the next falling edge of the transmitting clock after a WS change. In stereo mode when WS is low left data is transmitted and right data when WS is high. In mono mode the same data is transmitted twice, once when WS is low and again when WS is high.

- In master mode (ws_sel = 0), word select is generated internally with a 9 bit counter. The half period count value of this counter can be set in the control register.
- In slave mode (ws_sel = 1) word select is input from the relevant bus pin.
- When an I²S bus is active, the word select, receive clock and transmit clock signals are sent continuously by the bus master, while data is sent continuously by the transmitter.

- Disabling the I2S can be done with the stop or mute control bits separately for the transmit and receive.
- The stop bit will disable accesses by the transmit channel or the receive channel to the FIFOs and will place the transmit channel in mute mode.
- The mute control bit will place the transmit channel in mute mode. In mute mode, the transmit channel FIFO operates normally, but the output is discarded and replaced by zeroes. This bit does not affect the receive channel, data reception can occur normally.

3. Pin descriptions

Table 445. Pin descriptions

Pin Name	Type	Description
I2SRX_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.
I2SRX_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).
I2SRX_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.
I2STX_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.
I2STX_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).
I2STX_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.

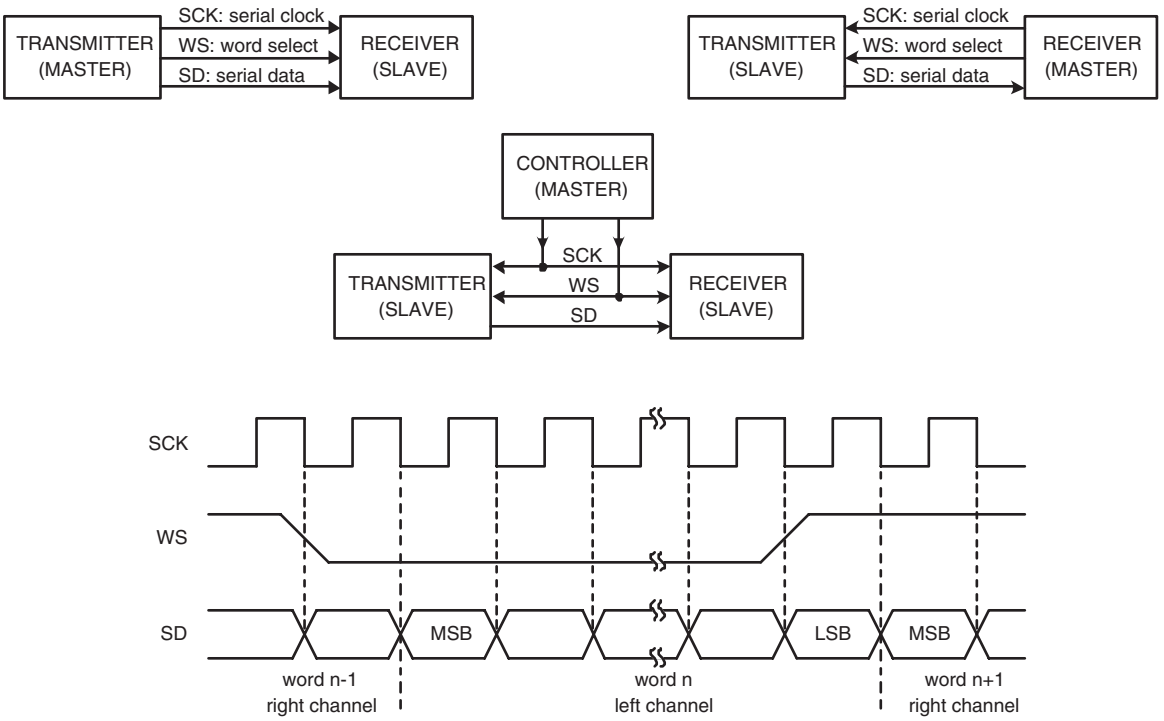


Fig 110.Simple I²S configurations and bus timing

4. Register description

Table 22–446 shows the registers associated with the I²S interface and a summary of their functions. Following the table are details for each register.

Table 446. I²S register map

Name	Description	Access	Reset Value[1]	Address
I2SDAO	Digital Audio Output Register. Contains control bits for the I²S transmit channel.	R/W		0xE008 8000
I2SDAI	Digital Audio Input Register. Contains control bits for the I²S receive channel.	R/W		0xE008 8004
I2STXFIFO	Transmit FIFO. Access register for the 8 × 32 bit transmitter FIFO.	WO		0xE008 8008
I2SRXFIFO	Receive FIFO. Access register for the 8 × 32 bit receiver FIFO.	RO		0xE008 800C
I2SSTATE	Status Feedback Register. Contains status information about the I²S interface.	RO		0xE008 8010
I2SDMA1	DMA Configuration Register 1. Contains control information for DMA request 1.	R/W		0xE008 8014
I2SDMA2	DMA Configuration Register 2. Contains control information for DMA request 2.	R/W		0xE008 8018

Table 446. I2S register map

Name	Description	Access	Reset Value[1]	Address
I2SIRQ	Interrupt Request Control Register. Contains bits that control how the I2S interrupt request is generated.	R/W		0xE008 801C
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		0xE008 8020
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		0xE008 8024

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

4.1 Digital Audio Output Register (I2SDAO - 0xE008 8000)

The I2SDAO register controls the operation of the I2S transmit channel. The function of bits in DAO are shown in Table 22–447.

Table 447: Digital Audio Output register (I2SDAO - address 0xE008 8000) 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		Disables accesses on FIFOs, 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

4.2 Digital Audio Input Register (I2SDAI - 0xE008 8004)

The I2SDAI register controls the operation of the I2S receive channel. The function of bits in DAI are shown in Table 22–448.

Table 448: Digital Audio Input register (I2SDAI - address 0xE008 8004) 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		Disables accesses on FIFOs, 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	Unused		Unused.	1

4.3 Transmit FIFO Register (I2STXFIFO - 0xE008 8008)

The I2STXFIFO register provides access to the transmit FIFO. The function of bits in I2STXFIFO are shown in Table 22–449.

Table 449: Transmit FIFO register (I2STXFIFO - address 0xE008 8008) bit description

Bit	Symbol	Description	Reset Value
31:0	I2STXFIFO	8 × 32 bits transmit FIFO.	Level = 0

4.4 Receive FIFO Register (I2SRXFIFO - 0xE008 800C)

The I2SRXFIFO register provides access to the receive FIFO. The function of bits in I2SRXFIFO are shown in Table 22–450.

Table 450: Receive FIFO register (I2RXFIFO - address 0xE008 800C) bit description

Bit	Symbol	Description	Reset Value
31:0	I2SRXFIFO	8 × 32 bits transmit FIFO.	level = 0

4.5 Status Feedback Register (I2SSTATE - 0xE008 8010)

The I2SSTATE register provides status information about the I2S interface. The meaning of bits in I2SSTATE are shown in Table 22–451.

Table 451: Status Feedback register (I2SSTATE - address 0xE008 8010) bit description

Bit	Symbol	Description	Reset Value
0	irq	This bit reflects the presence of Receive Interrupt or Transmit Interrupt.	0
1	dmareq1	This bit reflects the presence of Receive or Transmit DMA Request 1.	0
2	dmareq2	This bit reflects the presence of Receive or Transmit DMA Request 2.	0
7:3	Unused	Unused.	0

Table 451: Status Feedback register (I2SSTATE - address 0xE008 8010) bit description

Bit	Symbol	Description	Reset Value
15:8	rx_level	Reflects the current level of the Receive FIFO.	0
23:16	tx_level	Reflects the current level of the Transmit FIFO.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.6 DMA Configuration Register 1 (I2SDMA1 - 0xE008 8014)

The I2SDMA1 register controls the operation of DMA request 1. The function of bits in I2SDMA1 are shown in Table 22–452. Refer to the General Purpose DMA Controller chapter for details of DMA operation.

Table 452: DMA Configuration register 1 (I2SDMA1 - address 0xE008 8014) bit description

Bit	Symbol	Description	Reset Value
0	rx_dma1_enable	When 1, enables DMA1 for I2S receive.	0
1	tx_dma1_enable	When 1, enables DMA1 for I2S transmit.	0
7:2	Unused	Unused.	0
15:8	rx_depth_dma1	Set the FIFO level that triggers a receive DMA request on DMA1.	0
23:16	tx_depth_dma1	Set the FIFO level that triggers a transmit DMA request on DMA1.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.7 DMA Configuration Register 2 (I2SDMA2 - 0xE008 8018)

The I2SDMA2 register controls the operation of DMA request 2. The function of bits in I2SDMA2 are shown in Table 22–447.

Table 453: DMA Configuration register 2 (I2SDMA2 - address 0xE008 8018) bit description

Bit	Symbol	Description	Reset Value
0	rx_dma2_enable	When 1, enables DMA1 for I2S receive.	0
1	tx_dma2_enable	When 1, enables DMA1 for I2S transmit.	0
7:2	Unused	Unused.	0
15:8	rx_depth_dma2	Set the FIFO level that triggers a receive DMA request on DMA2.	0
23:16	tx_depth_dma2	Set the FIFO level that triggers a transmit DMA request on DMA2.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.8 Interrupt Request Control Register (I2SIRQ - 0xE008 801C)

The I2SIRQ register controls the operation of the I2S interrupt request. The function of bits in I2SIRQ are shown in Table 22–447.

Table 454: Interrupt Request Control register (I2SIRQ - address 0xE008 801C) bit description

Bit	Symbol	Description	Reset Value
0	rx_Irq_enable	When 1, enables I2S receive interrupt.	0
1	tx_Irq_enable	When 1, enables I2S transmit interrupt.	0
7:2	Unused	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

4.9 Transmit Clock Rate Register (I2STXRATE - 0xE008 8020)

The bit rate for the I2S transmitter is determined by the value of the I2STXRATE register. The value depends on the audio sample rate desired, and the data size and format (stereo/mono) used. For example, a 48 kHz sample rate for 16 bit stereo data requires a bit rate of 48,000×16×2 = 1.536 MHz.

Table 455: Transmit Clock Rate register (I2TXRATE - address 0xE008 8020) bit description

Bit	Symbol	Description	Reset Value
9:0	tx_rate	I2S transmit bit rate. This value plus one is used to divide PCLK by to produce the transmit bit clock. Ten bits of divide supports a wide range of I2S rates over a wide range of pclk rates.	0
15:10	Unused	Unused.	0

4.10 Receive Clock Rate Register (I2SRXRATE - 0xE008 8024)

The bit rate for the I2S receiver is determined by the value of the I2SRXRATE register. The value depends on the audio sample rate, as well as the data size and format used. The calculation is the same as for I2STXRATE.

Table 456: Receive Clock Rate register (I2SRXRATE - address 0xE008 8024) bit description

Bit	Symbol	Description	Reset Value
9:0	rx_rate	I2S receive bit rate. This value plus one is used to divide PCLK by to produce the receive bit clock. Ten bits of divide supports a wide range of I2S rates over a wide range of pclk rates.	0
15:10	Unused	Unused.	0

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 22–111 shows the possible data sequences.

A data sample in the FIFO consists of:

- 1×32 bits in 8 or 16 bit stereo modes.
- 1×32 bits in mono modes.
- 2×32 bits, first left data, second right data, in 32 bit stereo modes.

Data is read from the transmit FIFO after the falling edge of WS, it will be transferred to the transmit clock domain after the rising edge of WS. On the next falling edge of WS the left data will be loaded in the shift register and transmitted and on the following rising edge of WS the right data is loaded and transmitted.

The receive channel will start receiving data after a change of WS. When word select becomes low it expects this data to be left data, when WS is high received data is expected to be right data. Reception will stop when the bit counter has reached the limit set by wordwidth. On the next change of WS the received data will be stored in the appropriate hold register. When complete data is available it will be written into the receive FIFO.

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 APB status register.

Table 457. Conditions for FIFO level comparison

Level Comparison	Condition
dmareq_tx_1	tx_depth_dma1 >= tx_level
dmareq_rx_1	rx_depth_dma1 <= rx_level
dmareq_tx_2	tx_depth_dma2 >= tx_level
dmareq_rx_2	rx_depth_dma2 <= rx_level
irq_tx	tx_depth_irq >= tx_level
irq_rx	rx_depth_irq <= rx_level

System signaling occurs when a level detection is true and enabled.

Table 458. DMA and interrupt request generation

System Signaling	Condition
irq	(irq_rx & rx_irq_enable) | (irq_tx & tx_irq_enable
dmareq[0]	(dmareq_tx_1 & tx_dma1_enable) | (dmareq_rx_1 & rx_dma1_enable)
dmareq[1]	(dmareq_tx_2 & tx_dma2_enable) | (dmareq_rx_2 & rx_dma2_enable)

Table 459. Status feedback in the I2SSTATE register

Status Feedback	Status
irq	irq_rx | irq_tx
dmareq1	(dmareq_tx_1 | dmareq_rx_1)
dmareq2	(dmareq_rx_2 | dmareq_tx_2)

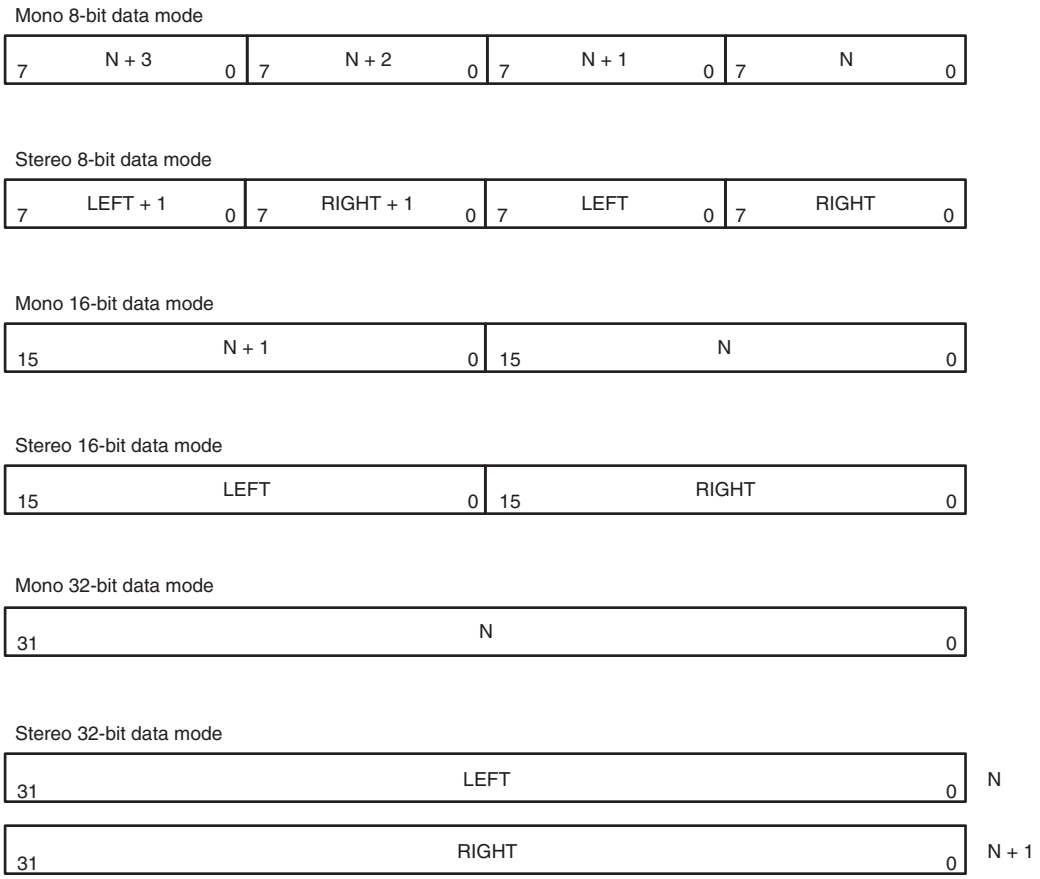


Fig 111.FIFO contents for various I²S modes

1. Features

Remark: The four Timer/Counters are identical except for the peripheral base address. A minimum of two Capture inputs and two Match outputs are pinned out for all four timers, with a choice of several pins for each. Timer 1 brings out a third Match output, while Timers 2 and 3 bring out all four Match outputs.

- 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.

2. Applications

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

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.

4. Pin description

Table 23–460 gives a brief summary of each of the Timer/Counter related pins.

Table 460. Timer/Counter pin description

Pin	Type	Description
CAP0[1:0] CAP1[1:0] CAP2[1:0] CAP3[1:0]	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. When more than one pin is selected for a Capture input on a single TIMER0/1 channel, the pin with the lowest Port number is used Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 23–5.3 “Count Control Register (T[0/1/2/3]CTCR - 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070)” on page 512.
MAT0[1:0] MAT1[2:0] MAT2[3:0] MAT3[3:0]	Output	External Match Output 0/1- When a match register 0/1 (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.

4.1 Multiple CAP and MAT pins

Software can select multiple pins for most of the CAP or MAT functions in the Pin Select registers, which are described in the Packaging, Pinning, and Multiplexing chapter. When more than one pin is selected for a MAT output, all such pins are driven identically. When more than one pin is selected for a CAP input, the pin with the lowest Port number is used.

5. Register description

Each Timer/Counter contains the registers shown in Table 23–461 ("Reset Value" refers to the data stored in used bits only; it does not include reserved bits content). More detailed descriptions follow.

Table 461. 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 - 0xE000 4000 T1IR - 0xE000 8000 T2IR - 0xE007 0000 T3IR - 0xE007 4000
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 - 0xE000 4004 T1TCR - 0xE000 8004 T2TCR - 0xE007 0004 T3TCR - 0xE007 4004
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 - 0xE000 4008 T1TC - 0xE000 8008 T2TC - 0xE007 0008 T3TC - 0xE007 4008
PR	Prescale Register. The Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0xE000 400C T1PR - 0xE000 800C T2PR - 0xE007 000C T3PR - 0xE007 400C

Table 461. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
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 - 0xE000 4010 T1PC - 0xE000 8010 T2PC - 0xE007 0010 T3PC - 0xE007 4010
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 - 0xE000 4014 T1MCR - 0xE000 8014 T2MCR - 0xE007 0014 T3MCR - 0xE007 4014
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 - 0xE000 4018 T1MR0 - 0xE000 8018 T2MR0 - 0xE007 0018 T3MR0 - 0xE007 4018
MR1	Match Register 1. See MR0 description.	R/W	0	T0MR1 - 0xE000 401C T1MR1 - 0xE000 801C T2MR1 - 0xE007 001C T3MR1 - 0xE007 401C
MR2	Match Register 2. See MR0 description.	R/W	0	T0MR2 - 0xE000 4020 T1MR2 - 0xE000 8020 T2MR2 - 0xE007 0020 T3MR2 - 0xE007 4020
MR3	Match Register 3. See MR0 description.	R/W	0	T0MR3 - 0xE000 4024 T1MR3 - 0xE000 8024 T2MR3 - 0xE007 0024 T3MR3 - 0xE007 4024
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 - 0xE000 4028 T1CCR - 0xE000 8028 T2CCR - 0xE007 0028 T3CCR - 0xE007 4028
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0(CAP0.0 or CAP1.0 respectively) input.	RO	0	T0CR0 - 0xE000 402C T1CR0 - 0xE000 802C T2CR0 - 0xE007 002C T3CR0 - 0xE007 402C
CR1	Capture Register 1. See CR0 description.	RO	0	T0CR1 - 0xE000 4030 T1CR1 - 0xE000 8030 T2CR1 - 0xE007 0030 T3CR1 - 0xE007 4030
CR2	Capture Register 2. See CR0 description.	RO	0	T0CR2 - 0xE000 4034 T1CR2 - 0xE000 8034 T2CR2 - 0xE007 0034 T3CR2 - 0xE007 4034

Table 461. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
CR3	Capture Register 3. See CR0 description.	RO	0	T0CR3 - 0xE000 4038 T1CR3 - 0xE000 8038 T2CR3 - 0xE007 0038 T3CR3 - 0xE007 4038
EMR	External Match Register. The EMR controls the external match pins MATn.0-3 (MAT0.0-3 and MAT1.0-3 respectively).	R/W	0	T0EMR - 0xE000 403C T1EMR - 0xE000 803C T2EMR - 0xE007 003C T3EMR - 0xE007 403C
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 - 0xE000 4070 T1CTCR - 0xE000 8070 T2CTCR - 0xE007 0070 T3CTCR - 0xE007 4070

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

5.1 Interrupt Register (T[0/1/2/3]IR - 0xE000 4000, 0xE000 8000, 0xE007 0000, 0xE007 4000)

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 462: Interrupt Register (T[0/1/2/3]IR - addresses 0xE000 4000, 0xE000 8000, 0xE007 0000, 0xE007 4000) 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

5.2 Timer Control Register (T[0/1/2/3]CR - 0xE000 4004, 0xE000 8004, 0xE007 0004, 0xE007 4004)

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 463: Timer Control Register (TCR, TIMERn: TnTCR - addresses 0xE000 4004, 0xE000 8004, 0xE007 0004, 0xE007 4004) 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

5.3 Count Control Register (T[0/1/2/3]CTCR - 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070)

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 464: Count Control Register (T[0/1/2/3]CTCR - addresses 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070) bit description

Bit	Symbol	Value	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).	00
			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 CAP input selected by bits 3:2.	
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	

Table 464: Count Control Register (T[0/1/2/3]CTCR - addresses 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070) bit description

Bit	Symbol	Value	Description	Reset Value
3:2	Count Input Select		When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking:	00
		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.	
7:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.4 Match Registers (MR0 - MR3)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

5.5 Match Control Register (T[0/1/2/3]MCR - 0xE000 4014, 0xE000 8014, 0xE007 0014, 0xE007 4014)

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 23–465.

Table 465: Match Control Register (T[0/1/2/3]MCR - addresses 0xE000 4014, 0xE000 8014, 0xE007 0014, 0xE007 4014) bit description

Bit	Symbol	Value	Description	Reset Value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	

Table 465: Match Control Register (T[0/1/2/3]MCR - addresses 0xE000 4014, 0xE000 8014, 0xE007 0014, 0xE007 4014) bit description

Bit	Symbol	Value	Description	Reset Value
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		0	Feature disabled.	
6	MR2I	1	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		0	This interrupt is disabled	
7	MR2R	1	Reset on MR2: the TC will be reset if MR2 matches it.	0
		0	Feature disabled.	
8	MR2S	1	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		0	Feature disabled.	
9	MR3I	1	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		0	This interrupt is disabled	
10	MR3R	1	Reset on MR3: the TC will be reset if MR3 matches it.	0
		0	Feature disabled.	
11	MR3S	1	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		0	Feature disabled.	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.6 Capture Registers (CR0 - CR3)

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.

5.7 Capture Control Register (T[0/1/2/3]CCR - 0xE000 4028, 0xE000 8028, 0xE007 0028, 0xE007 4028)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the Timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Table 466: Capture Control Register (T[0/1/2/3]CCR - addresses 0xE000 4028, 0xE000 8020, 0xE007 0028, 0xE007 4028) bit description

Bit	Symbol	Value	Description	Reset Value
0	CAP0RE	1	Capture on CAPn.0 rising edge: a sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
1	CAP0FE	1	Capture on CAPn.0 falling edge: a sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
2	CAP0I	1	Interrupt on CAPn.0 event: a CR0 load due to a CAPn.0 event will generate an interrupt.	0
		0	This feature is disabled.	
3	CAP1RE	1	Capture on CAPn.1 rising edge: a sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
4	CAP1FE	1	Capture on CAPn.1 falling edge: a sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
5	CAP1I	1	Interrupt on CAPn.1 event: a CR1 load due to a CAPn.1 event will generate an interrupt.	0
		0	This feature is disabled.	
6	CAP2RE	1	Capture on CAPn.2 rising edge: A sequence of 0 then 1 on CAPn.2 will cause CR2 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
7	CAP2FE	1	Capture on CAPn.2 falling edge: a sequence of 1 then 0 on CAPn.2 will cause CR2 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
8	CAP2I	1	Interrupt on CAPn.2 event: a CR2 load due to a CAPn.2 event will generate an interrupt.	0
		0	This feature is disabled.	
9	CAP3RE	1	Capture on CAPn.3 rising edge: a sequence of 0 then 1 on CAPn.3 will cause CR3 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
10	CAP3FE	1	Capture on CAPn.3 falling edge: a sequence of 1 then 0 on CAPn.3 will cause CR3 to be loaded with the contents of TC	0
		0	This feature is disabled.	
11	CAP3I	1	Interrupt on CAPn.3 event: a CR3 load due to a CAPn.3 event will generate an interrupt.	0
		0	This feature is disabled.	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.8 External Match Register (T[0/1/2/3]EMR - 0xE000 403C, 0xE000 803C, 0xE007 003C, 0xE007 403C)

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the Timer number, 0 or 1, and “m” represent a Match number, 0 through 3.

Table 467: External Match Register (T[0/1/2/3]EMR - addresses 0xE000 403C, 0xE000 803C, 0xE007 003C, 0xE007 403C) bit description

Bit	Symbol	Description	Reset Value
0	EM0	External Match 0. When a match occurs between the TC and MR0, this bit can either toggle, go low, go high, or do nothing, depending on bits 5:4 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
1	EM1	External Match 1. When a match occurs between the TC and MR1, this bit can either toggle, go low, go high, or do nothing, depending on bits 7:6 of this register. This bit can be driven onto a MATn.1 pin, in a positive-logic manner (0 = low, 1 = high).	0
2	EM2	External Match 2. When a match occurs between the TC and MR2, this bit can either toggle, go low, go high, or do nothing, depending on bits 9:8 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
3	EM3	External Match 3. When a match occurs between the TC and MR3, this bit can either toggle, go low, go high, or do nothing, depending on bits 11:10 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
5:4	EMC0	External Match Control 0. Determines the functionality of External Match 0. Table 23–468 shows the encoding of these bits.	00
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 23–468 shows the encoding of these bits.	00
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 23–468 shows the encoding of these bits.	00
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 23–468 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 468. 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.

6. Example Timer Operation

Figure 23–112 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 23–113 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.

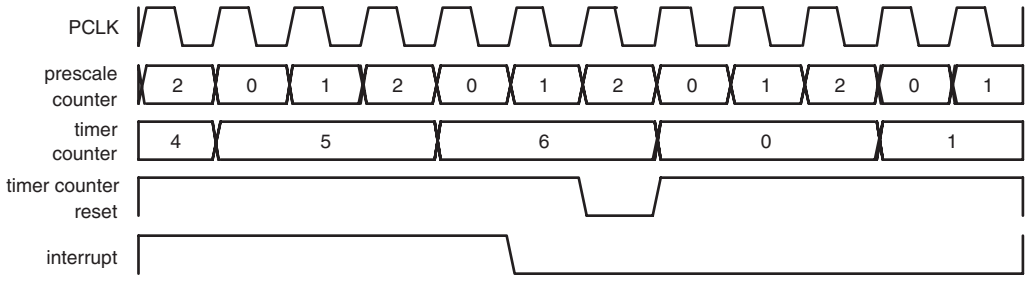


Fig 112.A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

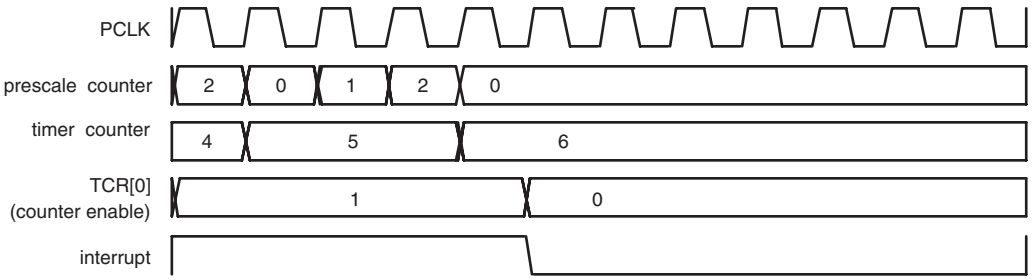


Fig 113.A timer Cycle in Which PR=2, MRx=6, and both interrupt and stop on match are enabled

7. Architecture

The block diagram for TIMER/COUNTER0 and TIMER/COUNTER1 is shown in Figure 23–114.

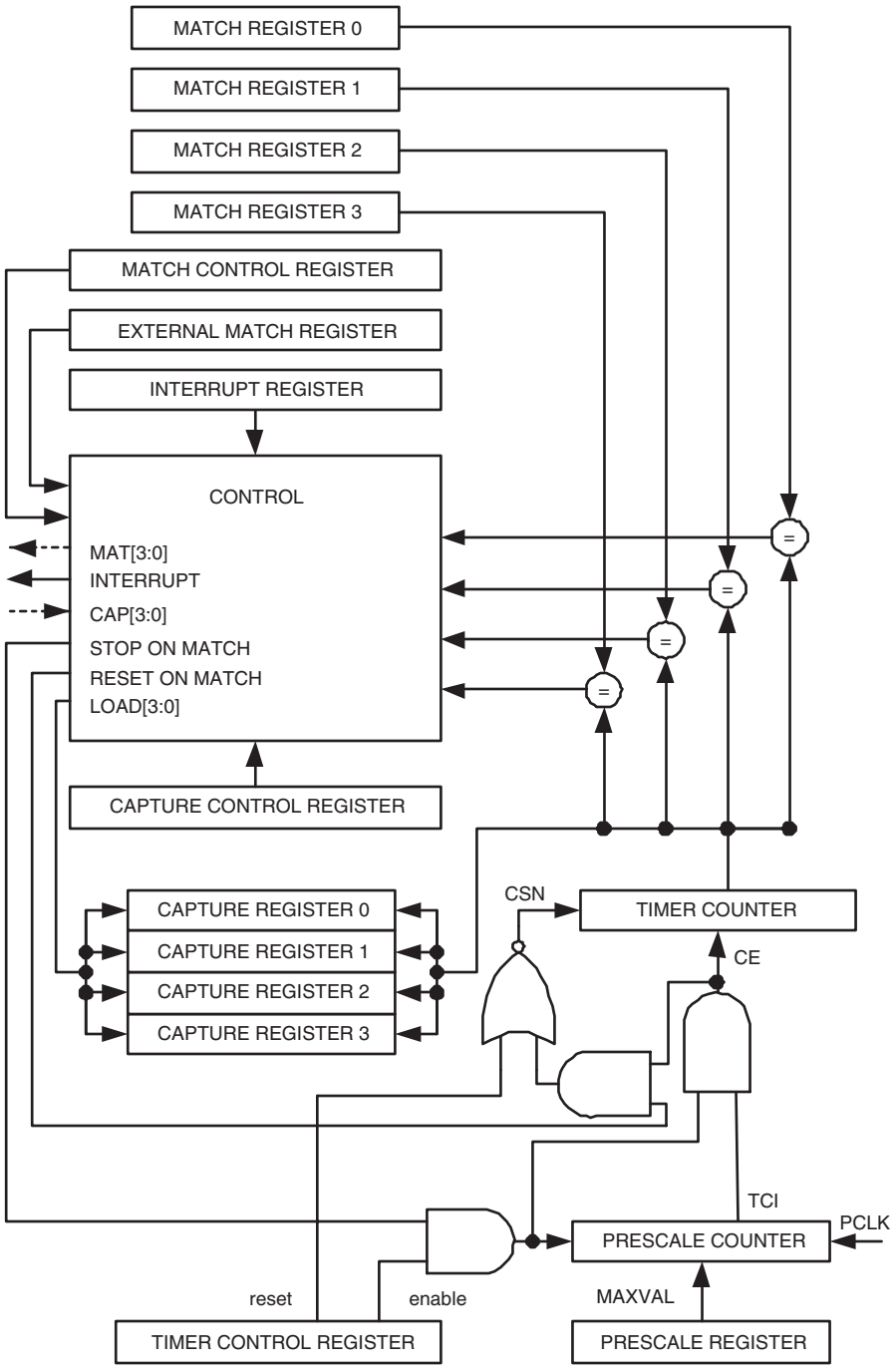


Fig 114.Timer block diagram

1. Features

- Internally resets chip if not periodically reloaded.
- Debug mode.
- Enabled by software but requires a hardware reset or a Watchdog reset/interrupt to be disabled.
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled.
- Flag to indicate Watchdog reset.
- Programmable 32 bit timer with internal pre-scaler.
- Selectable time period from ($T_{WDCLK} \times 256 \times 4$) to ($T_{WDCLK} \times 2^{32} \times 4$) in multiples of $T_{WDCLK} \times 4$.
- The Watchdog clock (WDCLK) source can be selected from the RTC clock, the Internal RC oscillator (IRC), or the APB peripheral clock (PCLK). This gives a wide range of potential timing choices for Watchdog operation under different power reduction conditions. It also provides the ability to run the Watchdog timer from an entirely internal source that is not dependent on an external crystal and its associated components and wiring, for increased reliability.

2. Applications

The purpose of the Watchdog is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state. When enabled, the Watchdog will generate a system reset if the user program fails to "feed" (or reload) the Watchdog within a predetermined amount of time.

For interaction of the on-chip watchdog and other peripherals, especially the reset and boot-up procedures, please read Section 3–4 “Reset” on page 20 of this document.

3. Description

The Watchdog consists of a divide by 4 fixed pre-scaler and a 32 bit counter. The clock is fed to the timer via a pre-scaler. The timer decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{32} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup mode in WDMOD register.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register.
- Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.

When the Watchdog counter underflows, the program counter will start from 0x0000 0000 as in the case of external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

The watchdog timer block uses two clocks: PCLK and WDCLK. PCLK is used for the APB accesses to the watchdog registers. The WDCLK is used for the watchdog timer counting.

There is some synchronization logic between these two clock domains. When the WDMOD and WDTC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain. When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the PCLK for reading as the WDTV register by the CPU.

4. Register description

The Watchdog contains 4 registers as shown in Table 24–469 below.

Table 469. Watchdog register map

Name	Description	Access	Reset Value[1]	Address
WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	R/W	0	0xE000 0000
WDTC	Watchdog timer constant register. This register determines the time-out value.	R/W	0xFF	0xE000 0004
WDFEED	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	WO	NA	0xE000 0008
WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	RO	0xFF	0xE000 000C
WDCLKSEL	Watchdog clock source selection register.	R/W	0	0xE000 0010

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

4.1 Watchdog Mode Register (WDMOD - 0xE000 0000)

The WDMOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits.

Table 470. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: debug with the Watchdog interrupt but no WDRESET enabled. When this mode is selected, a watchdog counter underflow will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: operate with the Watchdog interrupt and WDRESET enabled. When this mode is selected, a watchdog counter underflow will reset the microcontroller. Although the Watchdog interrupt is also enabled in this case (WDEN = 1) it will not be recognized since the watchdog reset will clear the WDINT flag.

Once the **WDEN** and/or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer underflow.

WDTOF The Watchdog time-out flag is set when the Watchdog times out. This flag is cleared by software.

WDINT The Watchdog interrupt flag is set when the Watchdog times out. This flag is cleared when any reset occurs. Once the watchdog interrupt is serviced, it can be disabled in the VIC or the watchdog interrupt request will be generated indefinitely.

Table 471: Watchdog Mode register (WDMOD - address 0xE000 0000) bit description

Bit	Symbol	Description	Reset Value
0	WDEN	WDEN Watchdog interrupt enable bit (Set Only).	0
1	WDRESET	WDRESET Watchdog reset enable bit (Set Only).	0
2	WDTOF	WDTOF Watchdog time-out flag.	0 (Only after external reset)
3	WDINT	WDINT Watchdog interrupt flag (Read Only).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.2 Watchdog Timer Constant Register (WDTC - 0xE000 0004)

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. It's a 32 bit register with 8 LSB set to 1 on reset. Writing values below 0xFF will cause 0x0000 00FF to be loaded to the WDTC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

Table 472: Watchdog Constant register (WDTC - address 0xE000 0004) bit description

Bit	Symbol	Description	Reset Value
31:0	Count	Watchdog time-out interval.	0x0000 00FF

4.3 Watchdog Feed Register (WDFEED - 0xE000 0008)

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the WDTC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed

errors. After writing 0xAA to WDFEED, any APB register access other than writing 0x55 to WDFEED causes an immediate reset/interrupt when the Watchdog is enabled. The reset will be generated during the second PCLK following an incorrect access to a Watchdog register during a feed sequence.

This implies that interrupts should be disabled during the feed sequence.

Table 473: Watchdog Feed Register (WDFEED - address 0xE000 0008) bit description

Bit	Symbol	Description	Reset Value
7:0	Feed	Feed value should be 0xAA followed by 0x55.	NA

4.4 Watchdog Timer Value Register (WDTV - 0xE000 000C)

The WDTV register is used to read the current value of Watchdog timer.

When reading the value of the 32 bit timer, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 PCLK cycles, so the value of WDTV is older than the actual value of the timer when it's being read by the CPU.

Table 474: Watchdog Timer Value register (WDTV - address 0xE000 000C) bit description

Bit	Symbol	Description	Reset Value
31:0	Count	Counter timer value.	0x0000 00FF

4.5 Watchdog Timer Clock Source Selection Register (WDCLKSEL - 0xE000 0010)

This register allows selecting the clock source for the Watchdog timer. The possibilities are: the Internal RC oscillator (IRC), the RTC oscillator, and the APB peripheral clock (pclk). The function of bits in WDCLKSEL are shown in Table 24–475.

Table 475: Watchdog Timer Clock Source Selection register (WDCLKSEL - address 0xE000 0010) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WDSEL		These bits select the clock source for the Watchdog timer as described below. **Warning:** Improper setting of this value may result in incorrect operation of the Watchdog timer, which could adversely affect system operation.	0
		00	Selects the Internal RC oscillator as the Watchdog clock source (default).	
		01	Selects the APB peripheral clock (PCLK) as the Watchdog clock source.	
		10	Selects the RTC oscillator as the Watchdog clock source.	
		11	Reserved	
31:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5. Block diagram

The block diagram of the Watchdog is shown below in the Figure 24–115. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

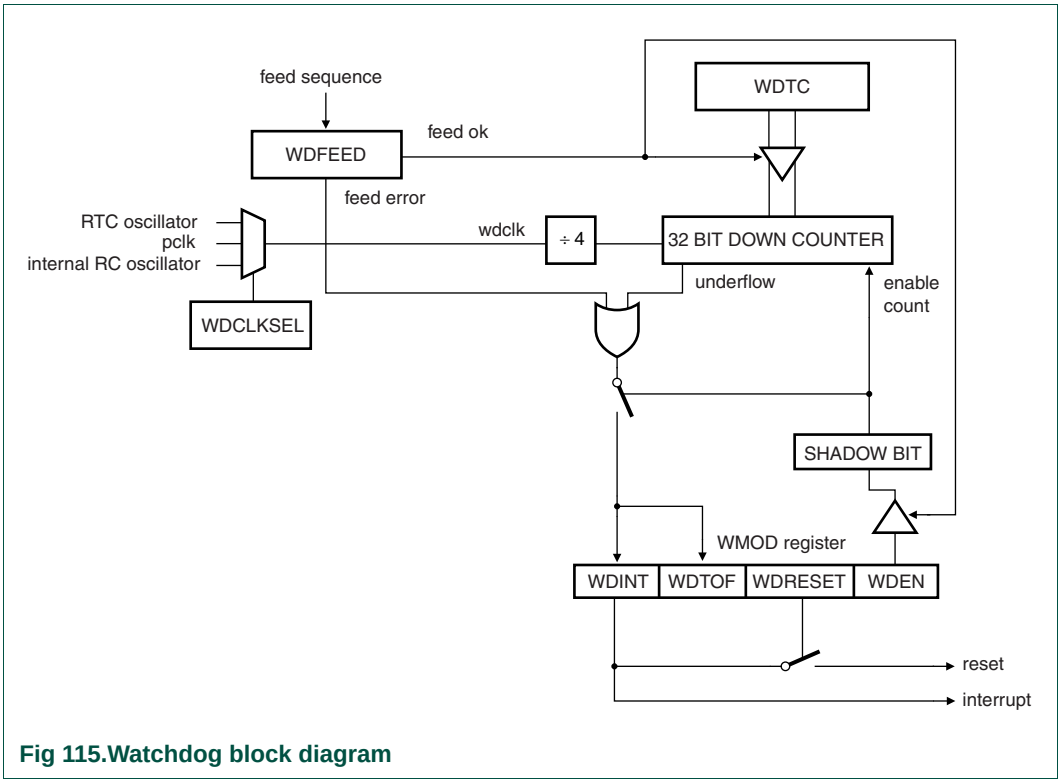


Fig 115.Watchdog block diagram

1. Features

- Two PWMs with the same operational features. These may be operated in a synchronized fashion by setting them both up to run at the same rate, then enabling both simultaneously. PWM0 acts as the Master and PWM1 as the slave for this use.
- Counter or Timer operation (may use the peripheral clock or one of the capture inputs as the clock source).
- Seven match registers allow up to 6 single edge controlled or 3 double edge controlled PWM outputs, or a mix of both types. The match registers also allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Supports single edge controlled and/or double edge controlled PWM outputs. Single edge controlled PWM outputs all go high at the beginning of each cycle unless the output is a constant low. Double edge controlled PWM outputs can have either edge occur at any position within a cycle. This allows for both positive going and negative going pulses.
- Pulse period and width can be any number of timer counts. This allows complete flexibility in the trade-off between resolution and repetition rate. All PWM outputs will occur at the same repetition rate.
- Double edge controlled PWM outputs can be programmed to be either positive going or negative going pulses.
- Match register updates are synchronized with pulse outputs to prevent generation of erroneous pulses. Software must "release" new match values before they can become effective.
- May be used as a standard timer if the PWM mode is not enabled.
- A 32 bit Timer/Counter with a programmable 32 bit Prescaler.
- Three 32 bit capture channels take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.

2. Description

The PWM is based on the standard Timer block and inherits all of its features, although only the PWM function is pinned out on the microcontroller. The Timer is designed to count cycles of the peripheral clock (PCLK) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. It also includes four capture inputs to save the timer value when an input signal transitions, and optionally generate an interrupt when those events occur. The PWM function is in addition to these features, and is based on match register events.

The ability to separately control rising and falling edge locations allows the PWM to be used for more applications. For instance, multi-phase motor control typically requires three non-overlapping PWM outputs with individual control of all three pulse widths and positions.

Two match registers can be used to provide a single edge controlled PWM output. One match register (MR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position. Additional single edge controlled PWM outputs require only one match register each, since the repetition rate is the same for all PWM outputs. Multiple single edge controlled PWM outputs will all have a rising edge at the beginning of each PWM cycle, when an MR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the MR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions. Additional double edge controlled PWM outputs require only two match registers each, since the repetition rate is the same for all PWM outputs.

With double edge controlled PWM outputs, specific match registers control the rising and falling edge of the output. This allows both positive going PWM pulses (when the rising edge occurs prior to the falling edge), and negative going PWM pulses (when the falling edge occurs prior to the rising edge).

Figure 25–116 shows the block diagram of the PWM. The portions that have been added to the standard timer block are on the right hand side and at the top of the diagram. At the lower left of the diagram may be found the Master Enable output from the Timer Control register that allows the Master PWM (PWM0) to enable both itself and the Salve PWM (PWM1) at the same time, if desired. The Master Enable output from PWM0 is connected to the external enable input of both PWM blocks.

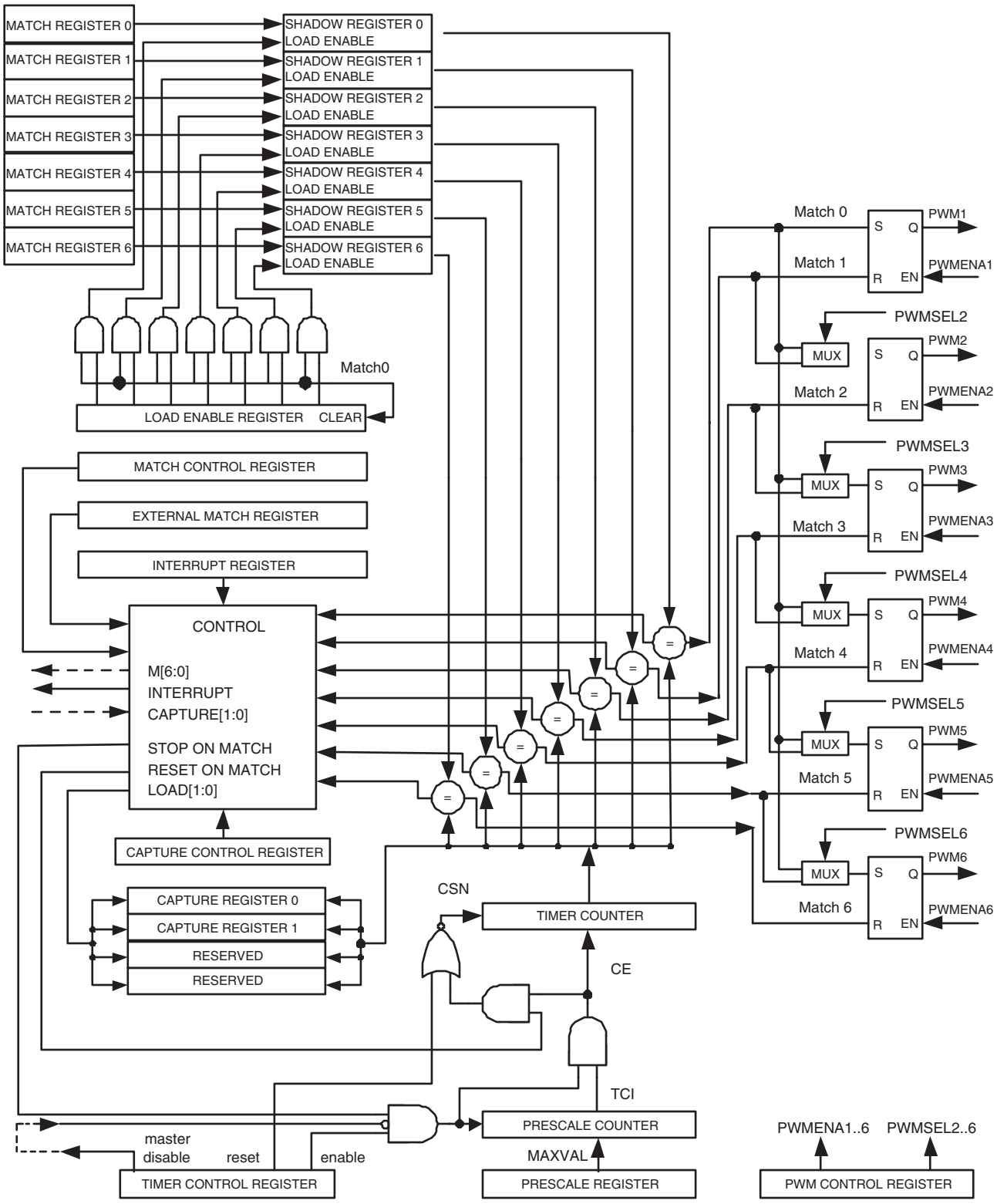


Fig 116.PWM block diagram

2.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go high at the beginning of a PWM cycle unless their match value is equal to 0.
2. Each PWM output will go low when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM rate), the PWM output remains continuously high.

2.2 Rules for double edge controlled PWM outputs

Five rules are used to determine the next value of a PWM output when a new cycle is about to begin:

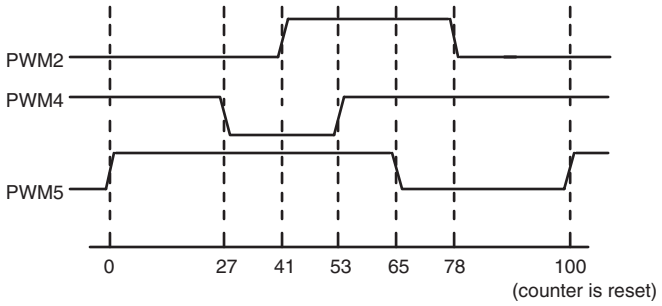
1. The match values for the **next** PWM cycle are used at the end of a PWM cycle (a time point which is coincident with the beginning of the next PWM cycle), except as noted in rule 3.
2. A match value equal to 0 or the current PWM rate (the same as the Match channel 0 value) have the same effect, except as noted in rule 3. For example, a request for a falling edge at the beginning of the PWM cycle has the same effect as a request for a falling edge at the end of a PWM cycle.
3. When match values are changing, if one of the "old" match values is equal to the PWM rate, it is used again once if the neither of the new match values are equal to 0 or the PWM rate, and there was no old match value equal to 0.
4. If both a set and a clear of a PWM output are requested at the same time, clear takes precedence. This can occur when the set and clear match values are the same as in, or when the set or clear value equals 0 and the other value equals the PWM rate.
5. If a match value is out of range (i.e. greater than the PWM rate value), no match event occurs and that match channel has no effect on the output. This means that the PWM output will remain always in one state, allowing always low, always high, or "no change" outputs.

2.3 Summary of differences from the standard timer block

1. A synchronizing register (shadow register) is added to each match register to allow changes to take effect only when requested by software, and only at the transition between PWM cycles.
2. A new Load Enable Register (LER) is added to allow software to control Match register updates. The LER contains one bit for each Match register. When a bit in the LER is written with a one, the shadow register contents for the corresponding Match channel are loaded into the actual Match register when the counter is reset (when Match 0 occurs). LER bits are reset automatically when the counter is reset.
3. A single PWM mode bit is added to the TCR register. The PWM mode enables loading the actual match registers from the shadow registers under software/hardware control as described above. When PWM mode is not enabled, the match value shadow registers are either transparent or bypassed.
4. A Master Enable bit is added to the TCR register, the value of which is brought out of the PWM block. An external enable input is added to the PWM block, that is connected to the Master Enable output of the Master PWM block.

5. The maximum number of match registers is increased to 7 in order to allow support for up to 3 double edge PWM channels. This includes the necessary match outputs, control bits, etc. for each match register:
   - Three new Match registers are added, creating Match channels 4 through 6.
   - Three additional sets of stop (S), reset (R), and interrupt (I) bits are added to the MCR register (3 per additional match register).
6. Add PWM outputs to the timer that connect a functional equivalent of an RS Flip-Flop to two match outputs. A 2-to-1 mux on each PWM output allows selection of either a single or a double edged PWM. A new register (PCR) is added to hold the control bits for the muxes (PWMSEL bits).
7. Three interrupt bits are added to the IR register.

A sample of how PWM values relate to waveform outputs is shown in Figure 25–117. PWM output logic is shown in Figure 25–116 that allows selection of either single or double edge controlled PWM outputs via the muxes controlled by the PWMSELn bits. The match register selections for various PWM outputs is shown in Table 25–476. This implementation of the PWM module supports up to N-1 single edge PWM outputs or (N-1)/2 double edge PWM outputs, where N is the number of match registers that are implemented. PWM types can be mixed if desired. In case of LPC2468 devices N = 7 which gives up to 6 single edge PWM outputs or up to 3 double edge PWM outputs available at the same time



The waveforms below show a single PWM cycle and demonstrate PWM outputs under the following conditions:

The timer is configured for PWM mode.

Match 0 is configured to reset the timer/counter when a match event occurs.

Control bits PWMSEL2 and PWMSEL4 are set.

The Match register values are as follows:

MR0 = 100 (PWM rate)

MR1 = 41, MR2 = 78 (PWM2 output)

MR3 = 53, MR4 = 27 (PWM4 output)

MR5 = 65 (PWM5 output)

Fig 117.Sample PWM waveforms

Table 476. Set and reset inputs for PWM flip-flops

PWM Channel	Single Edge PWM (PWMSELn = 0)		Double Edge PWM (PWMSELn = 1)	
	Set by	Reset by	Set by	Reset by
1	Match 0	Match 1	Match 0[1]	Match 1[1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2[2]	Match 3[2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4[2]	Match 5[2]
6	Match 0	Match 6	Match 5	Match 6

[1] Identical to single edge mode in this case since Match 0 is the neighboring match register. Essentially, PWM1 cannot be a double edged output.

[2] It is generally not advantageous to use PWM channels 3 and 5 for double edge PWM outputs because it would reduce the number of double edge PWM outputs that are possible. Using PWM 2, PWM4, and PWM6 for double edge PWM outputs provides the most pairings.

3. Pin description

Table 25–477 gives a brief summary of each of PWM related pins.

Table 477. Pin summary

Pin	Type	Description
PWM1	Output	Output from PWM channel 1.
PWM2	Output	Output from PWM channel 2.
PWM3	Output	Output from PWM channel 3.
PWM4	Output	Output from PWM channel 4.
PWM5	Output	Output from PWM channel 5.
PWM6	Output	Output from PWM channel 6.
PCAP0.0 PCAP1[1:0]	Input	Capture Inputs. A transition on a capture pin can be configured to load the corresponding Capture register with the value of the Timer Counter and optionally generate an interrupt. PWM0 brings out one capture input, PWM1 brings out 2 capture inputs.

4. PWM base addresses

Table 478: Addresses for PWM 0 and 1

PWM	Base addresses
0	0xE001 4000
1	0xE001 8000

5. Register description

The PWM0 and PWM1 function adds new registers and registers bits as shown in Table 25–479 below.

Table 479. PWM0 and PWM1 register map

Generic Name	Description	Access	Reset Value[1]	PWM0 Address & Name	PWM1 Address & Name
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	0xE001 4000 PWM0IR	0xE001 8000 PWM1IR
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	0xE001 4004 PWM0TCR	0xE001 8004 PWM1TCR
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	0xE001 4008 PWM0TC	0xE001 8008 PWM1TC
PR	Prescale Register. The TC is incremented every PR+1 cycles of PCLK.	R/W	0	0xE001 400C PWM0PR	0xE001 800C PWM1PR
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. The PC is observable and controllable through the bus interface.	R/W	0	0xE000 4010 PWM0PC	0xE001 8010 PWM1PC
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	0xE001 4014 PWM0MCR	0xE001 8014 PWM0MCR
MR0	Match Register 0. MR0 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC sets any PWM output that is in single-edge mode, and sets PWM1 if it's in double-edge mode.	R/W	0	0xE001 4018 PWM0MR0	0xE001 8018 PWM1MR0
MR1	Match Register 1. MR1 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM1 in either edge mode, and sets PWM2 if it's in double-edge mode.	R/W	0	0xE001 401C PWM0MR1	0xE001 801C PWM1MR1
MR2	Match Register 2. MR2 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM2 in either edge mode, and sets PWM3 if it's in double-edge mode.	R/W	0	0xE001 4020 PWM0MR2	0xE001 8020 PWM1MR2
MR3	Match Register 3. MR3 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM3 in either edge mode, and sets PWM4 if it's in double-edge mode.	R/W	0	0xE001 4024 PWM0MR3	0xE001 8024 PWM1MR3
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	0xE001 4028 PWM0CCR	0xE001 8028 PWM1CCR
CR0	Capture Register 0. PWMn CR0 is loaded with the value of the TC when there is an event on the CAPn.0 input.	RO	0	0xE001 402C PWM0CR0	-
CR1	Capture Register 1. See CR0 description.	RO	0	0xE001 4030 PWM0CR1	0xE001 8030 PWM1CR1

Table 479. PWM0 and PWM1 register map

Generic Name	Description	Access	Reset Value[1]	PWM0 Address & Name	PWM1 Address & Name
MR4	Match Register 4. MR4 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM4 in either edge mode, and sets PWM5 if it's in double-edge mode.	R/W	0	0xE001 4040 PWM0MR	0xE001 8040 PWM1MR
MR5	Match Register 5. MR5 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM5 in either edge mode, and sets PWM6 if it's in double-edge mode.	R/W	0	0xE001 4044 PWM0MR	0xE001 8044 PWM1MR
MR6	Match Register 6. MR6 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM6 in either edge mode.	R/W	0	0xE001 4048 PWM0MR	0xE001 8048 PWM1MR
PCR	PWM Control Register. Enables PWM outputs and selects PWM channel types as either single edge or double edge controlled.	R/W	0	0xE001 404C PWM0PCR	0xE001 804C PWM1PCR
LER	Load Enable Register. Enables use of new PWM match values.	R/W	0	0xE001 4050 PWM0LER	0xE001 8050 PWM1LER
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	0xE001 4070 PWM0CTCR	0xE001 8070 PWM1CTCR

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

5.1 PWM Interrupt Register (PWM0IR - 0xE001 4000 and PWM1IR 0xE001 8000)

The PWM Interrupt register consists of eleven bits (Table 25–480), seven for the match interrupts and four reserved. If an interrupt is generated then the corresponding bit in the PWMIR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 480: PWM Interrupt Register (PWM0IR - address 0xE001 4000 and PWM1IR address 0xE001 8000) bit description

Bit	Symbol	Description	Reset Value
0	PWMMR0 Interrupt	Interrupt flag for PWM match channel 0.	0
1	PWMMR1 Interrupt	Interrupt flag for PWM match channel 1.	0
2	PWMMR2 Interrupt	Interrupt flag for PWM match channel 2.	0
3	PWMMR3 Interrupt	Interrupt flag for PWM match channel 3.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0000
8	PWMMR4 Interrupt	Interrupt flag for PWM match channel 4.	0

Table 480: PWM Interrupt Register (PWM0IR - address 0xE001 4000 and PWM1IR address 0xE001 8000) bit description

Bit	Symbol	Description	Reset Value
9	PWMMR5 Interrupt	Interrupt flag for PWM match channel 5.	0
10	PWMMR6 Interrupt	Interrupt flag for PWM match channel 6.	0
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.2 PWM Timer Control Register (PWM0TCR - 0xE001 4004 and PWM1TCR 0xE001 8004)

The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter. The function of each of the bits is shown in Table 25–481.

Table 481: PWM Timer Control Register (PWM0TCR - address 0xE001 4004 PWM1TCR address 0xE001 8004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the PWM Timer Counter and PWM Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until this bit is returned to zero.	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PWM Enable	When one, PWM mode is enabled. PWM mode causes shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match Register 0 - MR0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	0
4	Master Disable (PWM0 only)	The two PWMs may be synchronized using the Master Disable control bit. The Master disable bit of the Master PWM (PWM0 module) controls a secondary enable input to both PWMs, as shown in Figure 25–116. If the PWMs are used independently, the Master Disable bit should always be 0, and the individual Counter Enable bits used to control them. This bit performs no function in the Slave PWM (PWM1).	0
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.3 PWM Count Control Register (PWM0CTCR - 0xE001 4070 and PWM1CTCR 0xE001 8070)

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. The function of each of the bits is shown in Table 25–482.

Table 482: PWM Count control Register (PWM0TCR - address 0xE001 4004 and PWM1CTCR address 0xE001 8004) bit description

Bit	Symbol	Description	Reset Value
1:0	Counter/ Timer Mode	00: Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale register. 01: Counter Mode: the TC is incremented on rising edges of the PCAP input selected by bits 3:2. 10: Counter Mode: the TC is incremented on falling edges of the PCAP input selected by bits 3:2. 11: Counter Mode: the TC is incremented on both edges of the PCAP input selected by bits 3:2.	00
3:2	Count Input Select	When bits 1:0 are not 00, these bits select which PCAP pin carries the signal used to increment the TC. For PWM0: 00 = PCAP0.0 (Other combinations are reserved) For PWM1: 00 = PCAP1.0, 01 = PCAP1.1(Other combinations are reserved)	00
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] PCAP input signal frequency must not exceed PCLK/4. When the PWM clock is supplied via the PCAP pin, at no time high(low) level of the signal on this pin can last less than 1/(2×PCLK).

5.4 PWM Match Control Register (PWM0MCR - 0xE001 4014 and PWM1MCR 0xE001 8014)

The PWM Match Control registers are used to control what operations are performed when one of the PWM Match registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 25–483.

Table 483: Match Control Register (PWM0MCR - address 0xE000 4014 and PWM1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
0	PWMMR0I	1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
1	PWMMR0R	1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.	0
		0	This feature is disabled.	
2	PWMMR0S	1	Stop on PWMMR0: the PWMTC and PWMPC will be stopped and PWMTCR bt 0 will be set to 0 if PWMMR0 matches the PWMTC.	0
		0	This feature is disabled	

Table 483: Match Control Register (PWM0MCR - address 0xE000 4014 and PWM1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
3	PWMMR1I	1	Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
4	PWMMR1R	1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.	0
		0	This feature is disabled.	
5	PWMMR1S	1	Stop on PWMMR1: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR1 matches the PWMTC.	0
		0	This feature is disabled.	
6	PWMMR2I	1	Interrupt on PWMMR2: an interrupt is generated when PWMMR2 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
7	PWMMR2R	1	Reset on PWMMR2: the PWxMTC will be reset if PWMMR2 matches it.	0
		0	This feature is disabled.	
8	PWMMR2S	1	Stop on PWMMR2: the PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR2 matches the PWxMTC.	0
		0	This feature is disabled	
9	PWMMR3I	1	Interrupt on PWMMR3: an interrupt is generated when PWMMR3 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
10	PWMMR3R	1	Reset on PWMMR3: the PWMTC will be reset if PWMMR3 matches it.	0
		0	This feature is disabled	
11	PWMMR3S	1	Stop on PWMMR3: The PWMTC and PWMPC will be stopped and PWMTCR bit 0 will be set to 0 if PWMMR3 matches the PWMTC.	0
		0	This feature is disabled	
12	PWMMR4I	1	Interrupt on PWMMR4: An interrupt is generated when PWMMR4 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
13	PWMMR4R	1	Reset on PWMMR4: the PWMTC will be reset if PWMMR4 matches it.	0
		0	This feature is disabled.	
14	PWMMR4S	1	Stop on PWMMR4: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR4 matches the PWMTC.	0
		0	This feature is disabled	
15	PWMMR5I	1	Interrupt on PWMMR5: An interrupt is generated when PWMMR5 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	

Table 483: Match Control Register (PWM0MCR - address 0xE000 4014 and PWM1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
16	PWMMR5R	1	Reset on PWMMR5: the PWMTC will be reset if PWMMR5 matches it.	0
		0	This feature is disabled.	
17	PWMMR5S	1	Stop on PWMMR5: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR5 matches the PWMTC.	0
		0	This feature is disabled	
18	PWMMR6I	1	Interrupt on PWMMR6: an interrupt is generated when PWMMR6 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
19	PWMMR6R	1	Reset on PWMMR6: the PWMTC will be reset if PWMMR6 matches it.	0
		0	This feature is disabled.	
20	PWMMR6S	1	Stop on PWMMR6: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR6 matches the PWMTC.	0
		0	This feature is disabled	
31:21	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.5 PWM Capture Control Register (PWM0CCR - 0xE001 4028 and PWM1CCR 0xE001 8028)

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 a 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 descriptions below, “n” represents the Timer number, 0 or 1.

Note: If Counter mode is selected for a particular PCAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other two PCAP inputs.

Table 484: PWM Capture Control Register (PWM0CCR - address 0xE001 4028 and PWM1CCR address 0xE001 8028) bit description

Bit	Symbol	Value	Description	Reset Value
0	Capture on PCAPn.0 rising edge	0	This feature is disabled.	0
		1	A synchronously sampled rising edge on the PCAPn.0 input will cause CR0 to be loaded with the contents of the TC.	
1	Capture on PCAPn.0 falling edge	0	This feature is disabled.	0
		1	A synchronously sampled falling edge on PCAPn.0 will cause CR0 to be loaded with the contents of TC.	
2	Interrupt on PCAPn.0 event	0	This feature is disabled.	0
		1	A CR0 load due to a PCAPn.0 event will generate an interrupt.	

Table 484: PWM Capture Control Register (PWM0CCR - address 0xE001 4028 and PWM1CCR address 0xE001 8028) bit description

Bit	Symbol	Value	Description	Reset Value
3	Capture on PCAPn.1 rising edge[1]	0	This feature is disabled.	0
		1	A synchronously sampled rising edge on the PCAPn.1 input will cause CR1 to be loaded with the contents of the TC.	
4	Capture on PCAPn.1 falling edge[1]	0	This feature is disabled.	0
		1	A synchronously sampled falling edge on PCAPn.1 will cause CR1 to be loaded with the contents of TC.	
5	Interrupt on PCAPn.1 event[1]	0	This feature is disabled.	0
		1	A CR1 load due to a PCAPn.1 event will generate an interrupt.	
31:6	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] This is a reserved pin for PWM0.

5.6 PWM Control Registers (PWM0PCR - 0xE001 404C and PWM1PCR 0xE001 804C)

The PWM Control registers are used to enable and select the type of each PWM channel. The function of each of the bits are shown in Table 25–485.

Table 485: PWM Control Registers (PWMPCR - address 0xE001 404C and PWM1PCR address 0xE001 804C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Unused		Unused, always zero.	NA
2	PWMSEL2		PWM2 output single/double edge mode control.	0
		1	Double edge controlled mode is selected.	
		0	Single edge controlled mode is selected.	
3	PWMSEL3	1	PWM3 output edge control. See PWMSEL2 for details.	0
4	PWMSEL4	1	PWM4 output edge control. See PWMSEL2 for details.	0
5	PWMSEL5	1	PWM5 output edge control. See PWMSEL2 for details.	0
6	PWMSEL6	1	PWM6 output edge control. See PWMSEL2 for details.	0
8:7	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9	PWMENA1		The PWM1 output enable control.	0
		0	The PWM output is disabled.	
		1	The PWM output is enabled.	
10	PWMENA2		The PWM2 output enable control. See PWMENA1 for details.	0
11	PWMENA3		The PWM3 output enable control. See PWMENA1 for details.	0
12	PWMENA4		The PWM4 output enable control. See PWMENA1 for details.	0
13	PWMENA5		The PWM5 output enable control. See PWMENA1 for details.	0
14	PWMENA6		The PWM6 output enable control. See PWMENA1 for details.	0
31:15	Unused		Unused, always zero.	NA

5.7 PWM Latch Enable Register (PWM0LER - 0xE001 4050 and PWM1LER 0xE001 8050)

The PWM Latch Enable registers are used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to the location of a PWM Match register while the Timer is in PWM mode, the value is actually held in a shadow register and not used immediately.

When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the actual Match registers if the corresponding bit in the Latch Enable register has been set. At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared. Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.

For example, if PWM2 is configured for double edge operation and is currently running, a typical sequence of events for changing the timing would be:

- Write a new value to the PWM Match1 register.
- Write a new value to the PWM Match2 register.
- Write to the PWMLER, setting bits 1 and 2 at the same time.
- The altered values will become effective at the next reset of the timer (when a PWM Match 0 event occurs).

The order of writing the two PWM Match registers is not important, since neither value will be used until after the write to PWMLER. This insures that both values go into effect at the same time, if that is required. A single value may be altered in the same way if needed.

The function of each of the bits in the PWMLER is shown in Table 25–486.

Table 486: PWM Latch Enable Register (PWM0LER - address 0xE001 4050 and PWM1LER address 0xE001 8050) bit description

Bit	Symbol	Description	Reset Value
0	Enable PWM Match 0 Latch	PWM MR0 register update control. Writing a one to this bit allows the last value written to the PWM Match Register 0 to be become effective when the timer is next reset by a PWM Match event. See Section 25–5.4 “PWM Match Control Register (PWM0MCR - 0xE001 4014 and PWM1MCR 0xE001 8014)”.	0
1	Enable PWM Match 1 Latch	PWM MR1 register update control. See bit 0 for details.	0
2	Enable PWM Match 2 Latch	PWM MR2 register update control. See bit 0 for details.	0
3	Enable PWM Match 3 Latch	PWM MR3 register update control. See bit 0 for details.	0
4	Enable PWM Match 4 Latch	PWM MR4 register update control. See bit 0 for details.	0

Table 486: PWM Latch Enable Register (PWM0LER - address 0xE001 4050 and PWM1LER address 0xE001 8050) bit description

Bit	Symbol	Description	Reset Value
5	Enable PWM Match 5 Latch	PWM MR5 register update control. See bit 0 for details.	0
6	Enable PWM Match 6 Latch	PWM MR6 register update control. See bit 0 for details.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

1. Features

- 10 bit successive approximation analog to digital converter.
- Input multiplexing among 8 pins.
- Power down mode.
- Measurement range 0 to 3 V.
- 10 bit conversion time ≥ 2.44 μs.
- Burst conversion mode for single or multiple inputs.
- Optional conversion on transition on input pin or Timer Match signal.
- Individual result registers for each A/D channel to reduce interrupt overhead.

2. Description

Basic clocking for the A/D converters is provided by the APB clock (PCLK). A programmable divider is included in each converter, to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks.

3. Pin description

Table 26–487 gives a brief summary of each of ADC related pins.

Table 487. A/D pin description

Pin	Type	Description
AD0[7:0]	Input	**Analog Inputs.** The A/D converter cell can measure the voltage on any of these input signals. Note that these analog inputs are always connected to their pins, even if the Pin Multiplexing Register assigns them to port pins. A simple self-test of the A/D Converter can be done by driving these pins as port outputs. **Note:** while the ADC pins are specified as 5 V tolerant (see Section 8–2 “Pin configuration” on page 102 and Section 8–2 “Pin configuration” on page 102), the analog multiplexing in the ADC block is not. More than $V_{DD(3V3)}$/VREF/3.3 V (V_{DDA}) +10 % should not be applied to any pin that is selected as an ADC input, or the ADC reading will be incorrect. If for example AD0.0 and AD0.1 are used as the ADC0 inputs and voltage on AD0.0 = 4.5 V while AD0.1 = 2.5 V, an excessive voltage on the AD0.0 can cause an incorrect reading of the AD0.1, although the AD0.1 input voltage is within the right range. If the A/D converter is not used in an application then the pins associated with A/D inputs can be used as 5V tolerant digital IO pins
VREF	Reference	**Voltage Reference.** This pin provides a voltage reference level for the A/D converter.
V_{DDA} , V_{SSA}	Power	**Analog Power and Ground.** These should be nominally the same voltages as $V_{DD(3V3)}$ and V_{SS} respectively but should be isolated to minimize noise and error.

4. Register description

The base address of the ADC is 0xE003 4000. The A/D Converter includes registers as shown in Table 26–488.

Table 488. A/D registers

Name	Description	Access	Reset Value[1]	Address
AD0CR	A/D Control Register. The AD0CR register must be written to select the operating mode before A/D conversion can occur.	R/W	0x0000 0001	0xE003 4000
AD0GDR	A/D Global Data Register. Contains the result of the most recent A/D conversion.	R/W	NA	0xE003 4004
AD0STAT	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt flag.	RO	0	0xE003 4030
AD0INTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x0000 0100	0xE003 400C
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0	R/W	NA	0xE003 4010
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	R/W	NA	0xE003 4014
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	R/W	NA	0xE003 4018
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	R/W	NA	0xE003 401C
ADDR4	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	R/W	NA	0xE003 4020
ADDR5	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	R/W	NA	0xE003 4024
ADDR6	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	R/W	NA	0xE003 4028
ADDR7	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	R/W	NA	0xE003 402C

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

4.1 A/D Control Register (AD0CR - 0xE003 4000)

The A/D Control Register provides bits to select A/D channels to be converted, A/D timing, A/D modes, and the A/D start trigger.

Table 489: A/D Control Register (AD0CR - address 0xE003 4000) bit description

Bit	Symbol	Value	Description	Reset Value
7:0	SEL		Selects which of the AD0.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The APB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST	0	Conversions are software controlled and require 11 clocks.	0
		1	The AD converter does repeated conversions at the rate selected by the CLKS field, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1 bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. **Important:** START bits must be 000 when BURST = 1 or conversions will not start.	
19:17	CLKS		This field selects the number of clocks used for each conversion in Burst mode, and the number of bits of accuracy of the result in the LS bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits).	000
		000	11 clocks / 10 bits	
		001	10 clocks / 9 bits	
		010	9 clocks / 8 bits	
		011	8 clocks / 7 bits	
		100	7 clocks / 6 bits	
		101	6 clocks / 5 bits	
		110	5 clocks / 4 bits	
		111	4 clocks / 3 bits	
20			Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PDN	1	The A/D converter is operational.	0
		0	The A/D converter is in power-down mode.	
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	
		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on P2.10/EINT0.	
		011	Start conversion when the edge selected by bit 27 occurs on P1.27/CAP0.1.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1.	

Table 489: A/D Control Register (AD0CR - address 0xE003 4000) bit description

Bit	Symbol	Value	Description	Reset Value
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.2 A/D Global Data Register (AD0GDR - 0xE003 4004)

The A/D Global Data Register contains the result of the most recent A/D conversion. This includes the data, DONE, and Overrun flags, and the number of the A/D channel to which the data relates.

Table 490: A/D Global Data Register (AD0GDR - address 0xE003 4004) bit description

Bit	Symbol	Description	Reset Value
5:0	Unused	These bits always read as zeroes. They provide compatible expansion room for future, higher-resolution A/D converters.	0
15:6	V/V_{REF}	When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin selected by the SEL field, divided by the voltage on the V_{DDA} pin. Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on V_{SSA}, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on V_{REF}.	X
23:16	Unused	These bits always read as zeroes. They allow accumulation of successive A/D values without AND-masking, for at least 256 values without overflow into the CHN field.	0
26:24	CHN	These bits contain the channel from which the LS bits were converted.	X
29:27	Unused	These bits always read as zeroes. They could be used for expansion of the CHN field in future compatible A/D converters that can convert more channels.	0
30	OVERUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the LS bits. In non-FIFO operation, this bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

4.3 A/D Status Register (ADSTAT - 0xE003 4030)

The A/D Status register allows checking the status of all A/D channels simultaneously. The DONE and OVERRUN flags appearing in the ADDRn register for each A/D channel are mirrored in ADSTAT. The interrupt flag (the logical OR of all DONE flags) is also found in ADSTAT.

Table 491: A/D Status Register (ADSTAT - address 0xE003 4030) bit description

Bit	Symbol	Description	Reset Value
7:0	Done7:0	These bits mirror the DONE status flags that appear in the result register for each A/D channel.	0
15:8	Overrun7:0	These bits mirror the OVERRRUN status flags that appear in the result register for each A/D channel. Reading ADSTAT allows checking the status of all A/D channels simultaneously.	0
16	ADINT	This bit is the A/D interrupt flag. It is one when any of the individual A/D channel Done flags is asserted and enabled to contribute to the A/D interrupt via the ADINTEN register.	0
31:17	Unused	Unused, always 0.	0

4.4 A/D Interrupt Enable Register (ADINTEN - 0xE003 400C)

This register allows control over which A/D channels generate an interrupt when a conversion is complete. For example, it may be desirable to use some A/D channels to monitor sensors by continuously performing conversions on them. The most recent results are read by the application program whenever they are needed. In this case, an interrupt is not desirable at the end of each conversion for some A/D channels.

Table 492: A/D Interrupt Enable Register (ADINTEN - address 0xE003 400C) bit description

Bit	Symbol	Description	Reset Value
7:0	ADINTEN 7:0	These bits allow control over which A/D channels generate interrupts for conversion completion. When bit 0 is one, completion of a conversion on A/D channel 0 will generate an interrupt, when bit 1 is one, completion of a conversion on A/D channel 1 will generate an interrupt, etc.	0x00
8	ADGINTEN	When 1, enables the global DONE flag in ADDR to generate an interrupt. When 0, only the individual A/D channels enabled by ADINTEN 7:0 will generate interrupts.	1
31:9	Unused	Unused, always 0.	0

4.5 A/D Data Registers (ADDR0 to ADDR7 - 0xE003 4010 to 0xE003 402C)

The A/D Data Register hold the result when an A/D conversion is complete, and also include the flags that indicate when a conversion has been completed and when a conversion overrun has occurred.

Table 493: A/D Data Registers (ADDR0 to ADDR7 - addresses 0xE003 4010 to 0xE003 402C) bit description

Bit	Symbol	Description	Reset Value
5:0	Unused	Unused, always 0. These bits always read as zeroes. They provide compatible expansion room for future, higher-resolution ADCs.	0
15:6	V/V_{REF}	When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin, divided by the voltage on the Vref pin. Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on V_{REF}, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on Vref.	NA
29:16	Unused	These bits always read as zeroes. They allow accumulation of successive A/D values without AND-masking, for at least 256 values without overflow into the CHN field.	0
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the LS bits.This bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.	0

5. Operation

5.1 Hardware-triggered conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the A/D converter will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

5.2 Interrupts

An interrupt is requested to the Vectored Interrupt Controller (VIC) when the ADINT bit in the ADSTAT register is 1. The ADINT bit is one when any of the DONE bits of A/D channels that are enabled for interrupts (via the ADINTEN register) are one. Software can use the Interrupt Enable bit in the VIC that corresponds to the ADC to control whether this results in an interrupt. The result register for an A/D channel that is generating an interrupt must be read in order to clear the corresponding DONE flag.

5.3 Accuracy vs. digital receiver

While the A/D converter can be used to measure the voltage on any AD0 pin, regardless of the pin's setting in the Pin Select register (Table 9–97 "Pin Connect Block Register Map" on page 120), selecting the AD0 function improves the conversion accuracy by disabling the pin's digital receiver.

1. Features

- 10 bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power down mode
- Selectable speed vs. power

2. Pin description

Table 27–494 gives a brief summary of each of DAC related pins.

Table 494. D/A Pin Description

Pin	Type	Description
AOUT	Output	**Analog Output.** After the selected settling time after the DACR is written with a new value, the voltage on this pin (with respect to V_{SSA}) is VALUE/1024 × VREF.
VREF	Reference	**Voltage Reference.** This pin provides a voltage reference level for the D/A converter.
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should be nominally the same voltages as $V_{DD(3V3)}$ and V_{SS}, but should be isolated to minimize noise and error.

3. Register description (DACR - 0xE006 C000)

This read/write register includes the digital value to be converted to analog, and a bit that trades off performance vs. power. Bits 5:0 are reserved for future, higher-resolution D/A converters.

Table 495: D/A Converter Register (DACR - address 0xE006 C000) bit description

Bit	Symbol	Value	Description	Reset Value
5:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	VALUE		After the selected settling time after this field is written with a new VALUE, the voltage on the A_{OUT} pin (with respect to V_{SSA}) is VALUE/1024 × VREF.	0
16	BIAS	0	The settling time of the DAC is 1 μs max, and the maximum current is 700 υA.	0
		1	The settling time of the DAC is 2.5 μs and the maximum current is 350 μA.	
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4. Operation

Bits 21:20 of the PINSEL1 register (Section 9–6.2 “Pin Function Select Register 1 (PINSEL1 - 0xE002 C004)” on page 121) control whether the DAC is enabled and controlling the state of pin P0.26/AD0.3/AOUT/RXD3. When these bits are 10, the DAC is powered on and active.

The settling times noted in the description of the BIAS bit are valid for a capacitance load on the AOUT pin not exceeding 100 pF. A load impedance value greater than that value will cause settling time longer than the specified time. One or more graph(s) of load impedance vs. settling time will be included in the final data sheet.

1. Features

- Measures the passage of time to maintain a calendar and clock.
- Ultra Low Power design to support battery powered systems.
- Provides Seconds, Minutes, Hours, Day of Month, Month, Year, Day of Week, and Day of Year.
- Dedicated 32 kHz oscillator or programmable prescaler from APB clock.
- Dedicated power supply pin can be connected to a battery or to the main 3.3 V.
- An alarm output pin is included to assist in waking up from Deep Power Down mode, or when the chip has had power removed to all functions except the RTC and Battery RAM.
- Periodic interrupts can be generated from increments of any field of the time registers, and selected fractional second values.
- 2 kilobyte static RAM powered by VBAT.
- RTC and Battery RAM power supply is isolated from the rest of the chip.

2. Description

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off. It uses little power in power down mode. On the LPC2400, the RTC can be clocked by a separate 32.768 KHz oscillator or by a programmable prescale divider based on the APB clock. Also, the RTC is powered by its own power supply pin, VBAT, which can be connected to a battery or to the same 3.3 V supply used by the rest of the device.

The VBAT pin supplies power only to the RTC and the Battery RAM. These two functions require a minimum of power to operate, which can be supplied by an external battery. When the CPU and the rest of chip functions are stopped and power removed, the RTC can supply an alarm output that can be used by external hardware to restore chip power and resume operation. The alarm output has a nominal voltage swing of 1.8 V.

3. Architecture

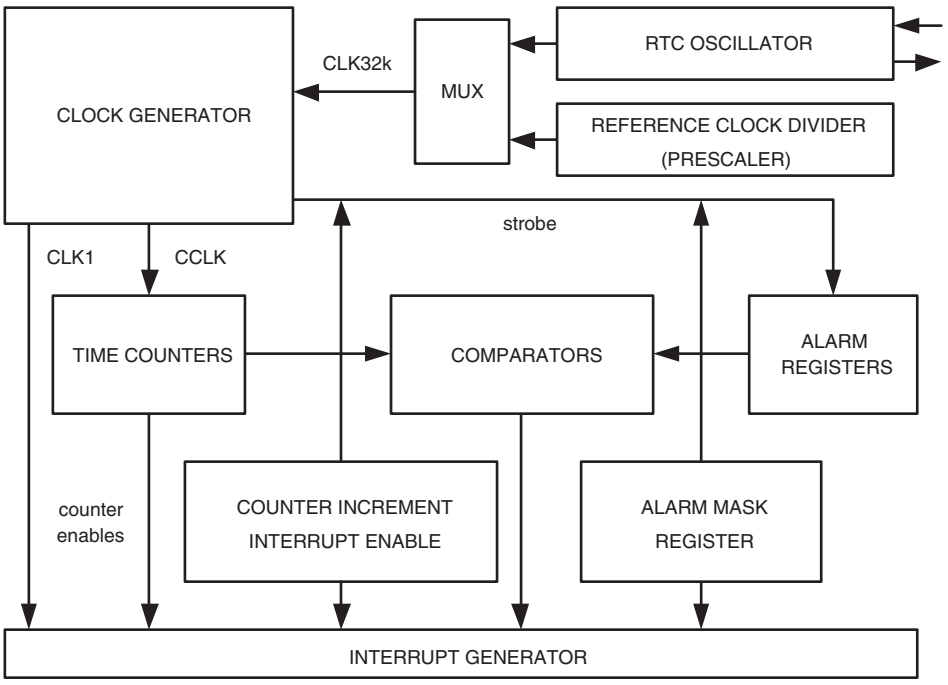


Fig 118.RTC block diagram

4. Register description

The RTC includes a number of registers. The address space is split into four sections by functionality. The first eight addresses are the Miscellaneous Register Group (Section 28–4.2). The second set of eight locations are the Time Counter Group (Section 28–4.4). The third set of eight locations contain the Alarm Register Group (Section 28–5). The remaining registers control the Reference Clock Divider.

The Real Time Clock includes the register shown in Table 28–496. Detailed descriptions of the registers follow. In these descriptions, for most of the registers the Reset Value column shows "NC", meaning that these registers are not changed by a Reset. Software must initialize these registers between power-on and setting the RTC into operation.

Table 496. Real Time Clock register map

Name	Size	Description	Access	Reset Value[1]	Address
ILR	2	Interrupt Location Register	R/W	*	0xE002 4000
CTC	15	Clock Tick Counter	RO	*	0xE002 4004
CCR	4	Clock Control Register	R/W	*	0xE002 4008
CIIR	8	Counter Increment Interrupt Register	R/W	*	0xE002 400C
AMR	8	Alarm Mask Register	R/W	*	0xE002 4010
CTIME0	32	Consolidated Time Register 0	RO	*	0xE002 4014
CTIME1	32	Consolidated Time Register 1	RO	*	0xE002 4018

Table 496. Real Time Clock register map

Name	Size	Description	Access	Reset Value[1]	Address
CTIME2	32	Consolidated Time Register 2	RO	*	0xE002 401C
SEC	6	Seconds Counter	R/W	*	0xE002 4020
MIN	6	Minutes Register	R/W	*	0xE002 4024
HOUR	5	Hours Register	R/W	*	0xE002 4028
DOM	5	Day of Month Register	R/W	*	0xE002 402C
DOW	3	Day of Week Register	R/W	*	0xE002 4030
DOY	9	Day of Year Register	R/W	*	0xE002 4034
MONTH	4	Months Register	R/W	*	0xE002 4038
YEAR	12	Years Register	R/W	*	0xE002 403C
CISS	8	Counter Increment select mask for Sub-Second interrupt	R/W	*	0xE002 4040
ALSEC	6	Alarm value for Seconds	R/W	*	0xE002 4060
ALMIN	6	Alarm value for Minutes	R/W	*	0xE002 4064
ALHOUR	5	Alarm value for Seconds	R/W	*	0xE002 4068
ALDOM	5	Alarm value for Day of Month	R/W	*	0xE002 406C
ALDOW	3	Alarm value for Day of Week	R/W	*	0xE002 4070
ALDOY	9	Alarm value for Day of Year	R/W	*	0xE002 4074
ALMON	4	Alarm value for Months	R/W	*	0xE002 4078
ALYEAR	12	Alarm value for Year	R/W	*	0xE002 407C
PREINT	13	Prescaler value, integer portion	R/W	0	0xE002 4080
PREFRAC	15	Prescaler value, integer portion	R/W	0	0xE002 4084

[1] Registers in the RTC other than those that are part of the Prescaler are not affected by chip Reset. These registers must be initialized by software if the RTC is enabled. Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

4.1 RTC interrupts

Interrupt generation is controlled through the Interrupt Location Register (ILR), Counter Increment Interrupt Register (CIIR), the alarm registers, and the Alarm Mask Register (AMR). Interrupts are generated only by the transition into the interrupt state. The ILR separately enables CIIR and AMR interrupts. Each bit in CIIR corresponds to one of the time counters. If CIIR is enabled for a particular counter, then every time the counter is incremented an interrupt is generated. The alarm registers allow the user to specify a date and time for an interrupt to be generated. The AMR provides a mechanism to mask alarm compares. If all nonmasked alarm registers match the value in their corresponding time counter, then an interrupt is generated.

The RTC interrupt can bring the microcontroller out of power-down mode if the RTC is operating from its own oscillator on the RTCX1-2 pins. When the RTC interrupt is enabled for wakeup and its selected event occurs, the oscillator wakeup cycle associated with the XTAL1/2 pins is started. For details on the RTC based wakeup process see Section 4–7.8 “Interrupt Wakeup Register (INTWAKE - 0xE01F C144)” on page 46 and Section 4–8 “Wakeup timer” on page 49.

4.2 Miscellaneous register group

Table 28–497 summarizes the registers located from 0 to 7 of A[6:2]. More detailed descriptions follow.

Table 497. Miscellaneous registers

Name	Size	Description	Access	Address
ILR	3	Interrupt Location. Reading this location indicates the source of an interrupt. Writing a one to the appropriate bit at this location clears the associated interrupt.	R/W	0xE002 4000
CTC	15	Clock Tick Counter. Value from the clock divider.	RO	0xE002 4004
CCR	4	Clock Control Register. Controls the function of the clock divider.	R/W	0xE002 4008
CIIR	8	Counter Increment Interrupt. Selects which counters will generate an interrupt when they are incremented.	R/W	0xE002 400C
AMR	8	Alarm Mask Register. Controls which of the alarm registers are masked.	R/W	0xE002 4010
CTIME0	32	Consolidated Time Register 0	RO	0xE002 4014
CTIME1	32	Consolidated Time Register 1	RO	0xE002 4018
CTIME2	32	Consolidated Time Register 2	RO	0xE002 401C

4.2.1 Interrupt Location Register (ILR - 0xE002 4000)

The Interrupt Location Register is a 2 bit register that specifies which blocks are generating an interrupt (see Table 28–498). Writing a one to the appropriate bit clears the corresponding interrupt. Writing a zero has no effect. This allows the programmer to read this register and write back the same value to clear only the interrupt that is detected by the read.

Table 498. Interrupt Location Register (ILR - address 0xE002 4000) bit description

Bit	Symbol	Description	Reset value
0	RTCCIF	When one, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.	NC
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.	NC
2	RTSSF	When one, the Counter Increment Sub-Seconds interrupt is generated. The interrupt rate is determined by the CISS register.	NC
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.2.2 Clock Tick Counter Register (CTCR - 0xE002 4004)

The Clock Tick Counter is read only. It can be reset to zero through the Clock Control Register (CCR). The CTC consists of the bits of the clock divider counter.

Table 499. Clock Tick Counter Register (CTCR - address 0xE002 4004) bit description

Bit	Symbol	Description	Reset value
14:0	Clock Tick Counter	Prior to the Seconds counter, the CTC counts 32,768 clocks per second. Due to the RTC Prescaler, these 32,768 time increments may not all be of the same duration. Refer to the Section 28–8.1 “Reference Clock Divider (Prescaler)” on page 556 for details.	NA
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.2.3 Clock Control Register (CCR - 0xE002 4008)

The clock register is a 4 bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 28–500.

Table 500. Clock Control Register (CCR - address 0xE002 4008) bit description

Bit	Symbol	Description	Reset value
0	CLKEN	Clock Enable. When this bit is a one the time counters are enabled. When it is a zero, they are disabled so that they may be initialized.	NA
1	CTCRST	CTC Reset. When one, the elements in the Clock Tick Counter are reset. The elements remain reset until CCR[1] is changed to zero.	NA
3:2	CTTEST	Test Enable. These bits should always be zero during normal operation.	NA
4	CLKSRC	If this bit is 0, the Clock Tick Counter takes its clock from the Prescaler, as on earlier devices in the NXP Embedded ARM family. If this bit is 1, the CTC takes its clock from the 32 kHz oscillator that’s connected to the RTCX1 and RTCX2 pins (see Section 28–10 “RTC external 32 kHz oscillator component selection” for hardware details).	NA
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.2.4 Counter Increment Interrupt Register (CIIR - 0xE002 400C)

The Counter Increment Interrupt Register (CIIR) gives the ability to generate an interrupt every time a counter is incremented. This interrupt remains valid until cleared by writing a one to bit zero of the Interrupt Location Register (ILR[0]).

Table 501. Counter Increment Interrupt Register (CIIR - address 0xE002 400C) bit description

Bit	Symbol	Description	Reset value
0	IMSEC	When 1, an increment of the Second value generates an interrupt.	NA
1	IMMIN	When 1, an increment of the Minute value generates an interrupt.	NA
2	IMHOUR	When 1, an increment of the Hour value generates an interrupt.	NA
3	IMDOM	When 1, an increment of the Day of Month value generates an interrupt.	NA
4	IMDOW	When 1, an increment of the Day of Week value generates an interrupt.	NA
5	IMDOY	When 1, an increment of the Day of Year value generates an interrupt.	NA
6	IMMON	When 1, an increment of the Month value generates an interrupt.	NA
7	IMYEAR	When 1, an increment of the Year value generates an interrupt.	NA

4.2.5 Counter Increment Select Mask Register (CISS - 0xE002 4040)

The CISS register provides a way to obtain millisecond-range periodic CPU interrupts from the Real Time Clock. This can allow freeing up one of the general purpose timers, or support power saving by putting the CPU into a reduced power mode between periodic interrupts.

Carry out signals from different stages of the Clock Tick Counter are used to generate the sub-second interrupts. The possibilities range from 16 counts of the CTC (about 488 microseconds), up to 2,048 counts of the CTC (about 62.5 milliseconds). The available counts and corresponding times are given in Table 28–502.

Table 502. Counter Increment Select Mask register (CISS - address 0xE002 4040) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SubSecSel		SubSecSelSub-Second Select. This field selects a count for the sub-second interrupt as follows:	NC
		000	An interrupt is generated on every 16 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 488 microseconds.	
		001	An interrupt is generated on every 32 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 977 microseconds.	
		010	An interrupt is generated on every 64 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 1.95 milliseconds.	
		011	An interrupt is generated on every 128 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 3.9 milliseconds.	
		100	An interrupt is generated on every 256 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 7.8 milliseconds.	
		101	An interrupt is generated on every 512 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 15.6 milliseconds.	
		110	An interrupt is generated on every 1024 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 31.25 milliseconds.	
		111	An interrupt is generated on every 2048 counts of the Clock Tick Counter. At 32.768 kHz, this generates an interrupt approximately every 62.5 milliseconds.	
6:3	Unused		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	SubSecEna		Subsecond interrupt enable.	NC
		0	The sub-second interrupt is disabled.	
		1	The sub-second interrupt is enabled.	

4.2.6 Alarm Mask Register (AMR - 0xE002 4010)

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 28–503 shows the relationship between the bits in the AMR and the alarms. For the alarm function, every non-masked alarm register must match the corresponding time counter for an interrupt to be generated. The interrupt is generated only when the counter comparison first changes from no match to match. The interrupt is removed when a one is written to the appropriate bit of the Interrupt Location Register (ILR). If all mask bits are set, then the alarm is disabled.

Table 503. Alarm Mask Register (AMR - address 0xE002 4010) bit description

Bit	Symbol	Description	Reset value
0	AMRSEC	When 1, the Second value is not compared for the alarm.	NA
1	AMRMIN	When 1, the Minutes value is not compared for the alarm.	NA
2	AMRHOUR	When 1, the Hour value is not compared for the alarm.	NA
3	AMRDOM	When 1, the Day of Month value is not compared for the alarm.	NA
4	AMRDOW	When 1, the Day of Week value is not compared for the alarm.	NA
5	AMRDOY	When 1, the Day of Year value is not compared for the alarm.	NA
6	AMRMON	When 1, the Month value is not compared for the alarm.	NA
7	AMRYEAR	When 1, the Year value is not compared for the alarm.	NA

4.3 Consolidated time registers

The values of the Time Counters can optionally be read in a consolidated format which allows the programmer to read all time counters with only three read operations. The various registers are packed into 32 bit values as shown in Table 28–504, Table 28–505, and Table 28–506. The least significant bit of each register is read back at bit 0, 8, 16, or 24.

The Consolidated Time Registers are read only. To write new values to the Time Counters, the Time Counter addresses should be used.

4.3.1 Consolidated Time Register 0 (CTIME0 - 0xE002 4014)

The Consolidated Time Register 0 contains the low order time values: Seconds, Minutes, Hours, and Day of Week.

Table 504. Consolidated Time register 0 (CTIME0 - address 0xE002 4014) bit description

Bit	Symbol	Description	Reset value
5:0	Seconds	Seconds value in the range of 0 to 59	NA
7:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
13:8	Minutes	Minutes value in the range of 0 to 59	NA
15:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
20:16	Hours	Hours value in the range of 0 to 23	NA
23:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	Day Of Week	Day of week value in the range of 0 to 6	NA
31:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.3.2 Consolidated Time Register 1 (CTIME1 - 0xE002 4018)

The Consolidate Time Register 1 contains the Day of Month, Month, and Year values.

Table 505. Consolidated Time register 1 (CTIME1 - address 0xE002 4018) bit description

Bit	Symbol	Description	Reset value
4:0	Day of Month	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).	NA
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:8	Month	Month value in the range of 1 to 12.	NA
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
27:16	Year	Year value in the range of 0 to 4095.	NA
31:28	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.3.3 Consolidated Time Register 2 (CTIME2 - 0xE002 401C)

The Consolidate Time Register 2 contains just the Day of Year value.

Table 506. Consolidated Time register 2 (CTIME2 - address 0xE002 401C) bit description

Bit	Symbol	Description	Reset value
11:0	Day of Year	Day of year value in the range of 1 to 365 (366 for leap years).	NA
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.4 Time Counter Group

The time value consists of the eight counters shown in Table 28–507 and Table 28–508. These counters can be read or written at the locations shown in Table 28–508.

Table 507. Time Counter relationships and values)

Counter	Size	Enabled by	Minimum value	Maximum value
Second	6	Clk1 (see Figure 28–118)	0	59
Minute	6	Second	0	59
Hour	5	Minute	0	23
Day of Month	5	Hour	1	28, 29, 30 or 31
Day of Week	3	Hour	0	6
Day of Year	9	Hour	1	365 or 366 (for leap year)
Month	4	Day of Month	1	12
Year	12	Month or day of Year	0	4095

Table 508. Time Counter registers

Name	Size	Description	Access	Address
SEC	6	Seconds value in the range of 0 to 59	R/W	0xE002 4020
MIN	6	Minutes value in the range of 0 to 59	R/W	0xE002 4024
HOUR	5	Hours value in the range of 0 to 23	R/W	0xE002 4028

Table 508. Time Counter registers

Name	Size	Description	Access	Address
DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).[1]	R/W	0xE002 402C
DOW	3	Day of week value in the range of 0 to 6[1]	R/W	0xE002 4030
DOY	9	Day of year value in the range of 1 to 365 (366 for leap years)[1]	R/W	0xE002 4034
MONTH	4	Month value in the range of 1 to 12	R/W	0xE002 4038
YEAR	12	Year value in the range of 0 to 4095	R/W	0xE002 403C

[1] These values are simply incremented at the appropriate intervals and reset at the defined overflow point. They are not calculated and must be correctly initialized in order to be meaningful.

4.4.1 Leap year calculation

The RTC does a simple bit comparison to see if the two lowest order bits of the year counter are zero. If true, then the RTC considers that year a leap year. The RTC considers all years evenly divisible by 4 as leap years. This algorithm is accurate from the year 1901 through the year 2099, but fails for the year 2100, which is not a leap year. The only effect of leap year on the RTC is to alter the length of the month of February for the month, day of month, and year counters.

5. Alarm register group

The alarm registers are shown in Table 28–509. The values in these registers are compared with the time counters. If all the unmasked (See Section 28–4.2.6 “Alarm Mask Register (AMR - 0xE002 4010)” on page 552) alarm registers match their corresponding time counters then an interrupt is generated. The interrupt is cleared when a one is written to bit one of the Interrupt Location Register (ILR[1]).

Table 509. Alarm registers

Name	Size	Description	Access	Address
ALSEC	6	Alarm value for Seconds	R/W	0xE002 4060
ALMIN	6	Alarm value for Minutes	R/W	0xE002 4064
ALHOUR	5	Alarm value for Hours	R/W	0xE002 4068
ALDOM	5	Alarm value for Day of Month	R/W	0xE002 406C
ALDOW	3	Alarm value for Day of Week	R/W	0xE002 4070
ALDOY	9	Alarm value for Day of Year	R/W	0xE002 4074
ALMON	4	Alarm value for Months	R/W	0xE002 4078
ALYEAR	12	Alarm value for Years	R/W	0xE002 407C

6. Alarm output

The RTC includes an alarm output pin that reflects both the alarm comparisons and interrupts from the RTC. This pin is in the RTC power domain, and therefore it is available during all power saving modes as long as power is supplied to VBAT. Since the Alarm pin combines the alarm and interrupt functions of the RTC, either a specific time/date/etc. or a

periodic interval can be provided to the outside world. For example, a time of day alarm could be used to tell external circuitry to turn on power to the LPC2400 in order to wake up from Deep Power Down mode.

7. RTC usage notes

The RTC may be clocked by either the 32.786 kHz RTC oscillator, or by the APB peripheral clock (PCLK) after adjustment by the reference clock divider.

If the RTC is used, VBAT must be connected to either pin $V_{DD(3V3)}$ or an independent power supply (external battery). Otherwise, VBAT should be tied to the ground (V_{SS}). No provision is made in the LPC2400 to retain RTC status upon the VBAT power loss, or to maintain time incrementation if the clock source is lost, interrupted, or altered.

Since the RTC operates using one of two available clocks (the APB clock (PCLK) or the 32 kHz signal coming from the RTCX1-2pins), any interruption of the selected clock will cause the time to drift away from the time value it would have provided otherwise. The variance could be to actual clock time if the RTC was initialized to that, or simply an error in elapsed time since the RTC was activated.

While the signal from RTCX1-2 pins can be used to supply the RTC clock at anytime, selecting the PCLK as the RTC clock and entering the Power Down mode will cause a lapse in the time update. Also, feeding the RTC with the PCLK and altering this timebase during system operation (by reconfiguring the PLL, the APB divider, or the RTC prescaler) will result in some form of accumulated time error. Accumulated time errors may occur in case RTC clock source is switched between the PCLK to the RTCX pins, too.

Once the 32 kHz signal from RTCX1-2 pins is selected as a clock source, the RTC can operate completely without the presence of the APB clock (PCLK). Therefore, power sensitive applications (i.e. battery powered application) utilizing the RTC will reduce the power consumption by using the signal from RTCX1-2 pins, and writing a 0 into the PCRTC bit in the PCONP power control register (see Section 4–7 “Power control” on page 43).

8. RTC clock generation

The RTC may be clocked by either the 32.786 kHz RTC oscillator, or by the APB peripheral clock (PCLK) after adjustment by the reference clock divider.

8.1 Reference Clock Divider (Prescaler)

The reference clock divider (hereafter referred to as the Prescaler) may be used when the RTC clock source is not supplied by the RTC oscillator, but comes from the APB peripheral clock (PCLK).

The Prescaler allows generation of a 32.768 kHz reference clock from any PCLK frequency greater than or equal to 65.536 kHz (2×32.768 kHz). This permits the RTC to always run at the proper rate regardless of the peripheral clock rate. Basically, the Prescaler divides PCLK by a value which contains both an integer portion and a fractional portion. The result is not a continuous output at a constant frequency, some clock periods will be one PCLK longer than others. However, the overall result can always be 32,768 counts per second.

The reference clock divider consists of a 13 bit integer counter and a 15 bit fractional counter. The reasons for these counter sizes are as follows:

1. For frequencies that are expected to be supported by the LPC2400, a 13 bit integer counter is required. This can be calculated as 160 MHz divided by 32,768 minus 1 equals 4881 with a remainder of 26,624. Thirteen bits are needed to hold the value 4881, but actually supports frequencies up to 268.4 MHz (32,768 × 8192).
2. The remainder value could be as large as 32,767, which requires 15 bits.

Table 510. Reference Clock Divider registers

Name	Size	Description	Access	Address
PREINT	13	Prescale Value, integer portion	R/W	0xE002 4080
PREFRAC	15	Prescale Value, fractional portion	R/W	0xE002 4084

8.2 Prescaler Integer Register (PREINT - 0xE002 4080)

This is the integer portion of the prescale value, calculated as:

PREINT = int (PCLK/32768) - 1. The value of PREINT must be greater than or equal to 1.

Table 511: Prescaler Integer register (PREINT - address 0xE002 4080) bit description

Bit	Symbol	Description	Reset Value
12:0	Prescaler Integer	Contains the integer portion of the RTC prescaler value.	0
15:13	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.3 Prescaler Fraction Register (PREFRAC - 0xE002 4084)

This is the fractional portion of the prescale value, and may be calculated as:

PREFRAC = PCLK - ((PREINT + 1) x 32768).

Table 512: Prescaler Integer register (PREFRAC - address 0xE002 4084) bit description

Bit	Symbol	Description	Reset Value
14:0	Prescaler Fraction	Contains the integer portion of the RTC prescaler value.	0
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.4 Example of Prescaler Usage

In a simplistic case, the PCLK frequency is 65.537 kHz. So:

PREINT = int (PCLK / 32768) - 1 = 1 and
PREFRAC = PCLK - ([PREINT + 1] x 32768) = 1

With this prescaler setting, exactly 32,768 clocks per second will be provided to the RTC by counting 2 PCLKs 32,767 times, and 3 PCLKs once.

In a more realistic case, the PCLK frequency is 10 MHz. Then,

PREINT = int (PCLK / 32768) - 1 = 304 and
PREFRAC = PCLK - ([PREINT + 1] x 32768) = 5,760.

In this case, 5,760 of the prescaler output clocks will be 306 (305+1) PCLKs long, the rest will be 305 PCLKs long.

In a similar manner, any PCLK rate greater than 65.536 kHz (as long as it is an even number of cycles per second) may be turned into a 32 kHz reference clock for the RTC. The only caveat is that if PREFRAC does not contain a zero, then not all of the 32,768 per second clocks are of the same length. Some of the clocks are one PCLK longer than others. While the longer pulses are distributed as evenly as possible among the remaining pulses, this "jitter" could possibly be of concern in an application that wishes to observe the contents of the Clock Tick Counter (CTC) directly(Section 28–4.2.2 “Clock Tick Counter Register (CTCR - 0xE002 4004)” on page 550).

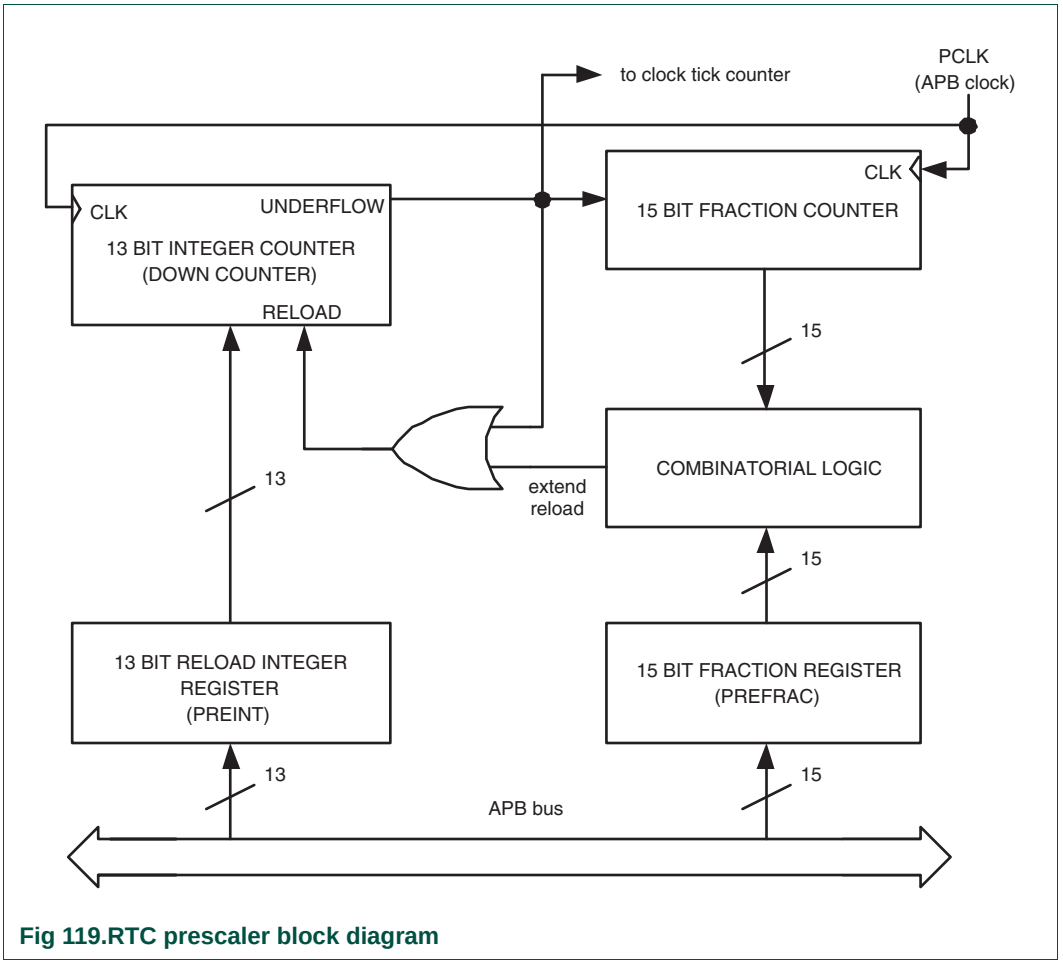


Fig 119.RTC prescaler block diagram

8.5 Prescaler operation

The Prescaler block labelled "Combination Logic" in Figure 28–119 determines when the decrement of the 13 bit PREINT counter is extended by one PCLK. In order to both insert the correct number of longer cycles, and to distribute them evenly, the combinatorial Logic associates each bit in PREFRAC with a combination in the 15 bit Fraction Counter. These associations are shown in the following Table 28–513.

For example, if PREFRAC bit 14 is a one (representing the fraction 1/2), then half of the cycles counted by the 13 bit counter need to be longer. When there is a 1 in the LSB of the Fraction Counter, the logic causes every alternate count (whenever the LSB of the Fraction Counter=1) to be extended by one PCLK, evenly distributing the pulse widths. Similarly, a one in PREFRAC bit 13 (representing the fraction 1/4) will cause every fourth cycle (whenever the two LSBs of the Fraction Counter = 10) counted by the 13 bit counter to be longer.

Table 513. Prescaler cases where the Integer Counter reload value is incremented

Fraction Counter	PREFRAC Bit														
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
--- ---- ---- ---1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- --10	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- -100	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- 1000	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---1 0000	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-
--- ---- --10 0000	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
--- ---- -100 0000	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-
--- ---- 1000 0000	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-
--- ---1 0000 0000	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-
--- --10 0000 0000	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
--- -100 0000 0000	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
--- 1000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-
--1 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-
-10 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-
100 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

9. Battery RAM

The Battery RAM is a 2 kbyte static RAM residing on the APB bus. The address range is 0xE008 4000 to 0xE008 47FF. The Battery RAM is powered from the VBAT pin along with the RTC, both of which exist in a power domain that is isolated from the rest of the chip. This allows them to operate while the main chip power has been removed.

10. RTC external 32 kHz oscillator component selection

The RTC external oscillator circuit is shown in Figure 28–120. Since the feedback resistance is integrated on chip, only a crystal, the capacitances C_{X1} and C_{X2} need to be connected externally to the microcontroller.

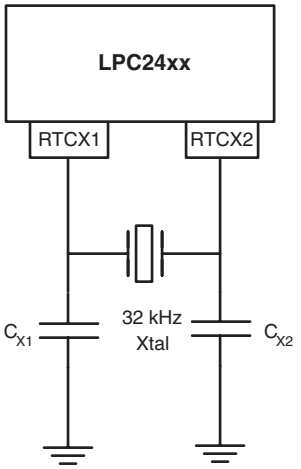


Fig 120.RTC 32 kHz crystal oscillator circuit

Table 28–514 gives the crystal parameters that should be used. C_L is the typical load capacitance of the crystal and is usually specified by the crystal manufacturer. The actual C_L influences oscillation frequency. When using a crystal that is manufactured for a different load capacitance, the circuit will oscillate at a slightly different frequency (depending on the quality of the crystal) compared to the specified one. Therefore for an accurate time reference it is advised to use the load capacitors as specified in Table 28–514 that belong to a specific C_L. The value of external capacitances C_{X1} and C_{X2} specified in this table are calculated from the internal parasitic capacitances and the C_L. Parasitics from PCB and package are not taken into account.

Table 514. Recommended values for the RTC external 32 kHz oscillator $C_{X1/X2}$ components

Crystal load capacitance C_L	Maximum crystal series resistance R_S	External load capacitors C_{X1}, C_{X2}
11 pF	< 100 kΩ	18 pF, 18 pF
13 pF	< 100 kΩ	22 pF, 22 pF
15 pF	< 100 kΩ	27 pF, 27 pF

1. Flash boot loader

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

2. Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip Flash memory, using the boot loader software and UART0 serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip Flash memory, as directed by the end-user application code.

3. Applications

The Flash boot loader provides both In-System and In-Application programming interfaces for programming the on-chip Flash memory.

4. Description

The Flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A LOW level after reset at the P2.10 pin is considered as an external hardware request to start the ISP command handler. Assuming that power supply pins are on their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before P2.10 is sampled and the decision on whether to continue with user code or ISP handler is made. If P2.10 is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P2.10 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

Pin P2.10 that is used as hardware request for ISP requires special attention. Since P2.10 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

When ISP mode is entered after a power on reset, the IRC and PLL are used to generate CCLK of 14.748 MHz. This may not be the case when ISP is invoked by the user application (see Section 29–9.8 “Reinvoke ISP” on page 580).

4.1 Memory map after any reset

The Flash portion of the boot block is 8 kB in size and resides in the top portion (starting from 0x0007 E000) of the on-chip Flash memory. After any reset the entire boot block is also mapped to the top of the on-chip memory space i.e. the boot block is also visible in the memory region starting from the address 0x7FFF E000. The Flash boot loader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in this chapter. The interrupt vectors residing in the boot block of the on-chip Flash memory also become active after reset, i.e., the bottom 64 bytes of the boot block are also visible in the memory region starting from the address 0x0000 0000. The reset vector contains a jump instruction to the entry point of the flash boot loader software.

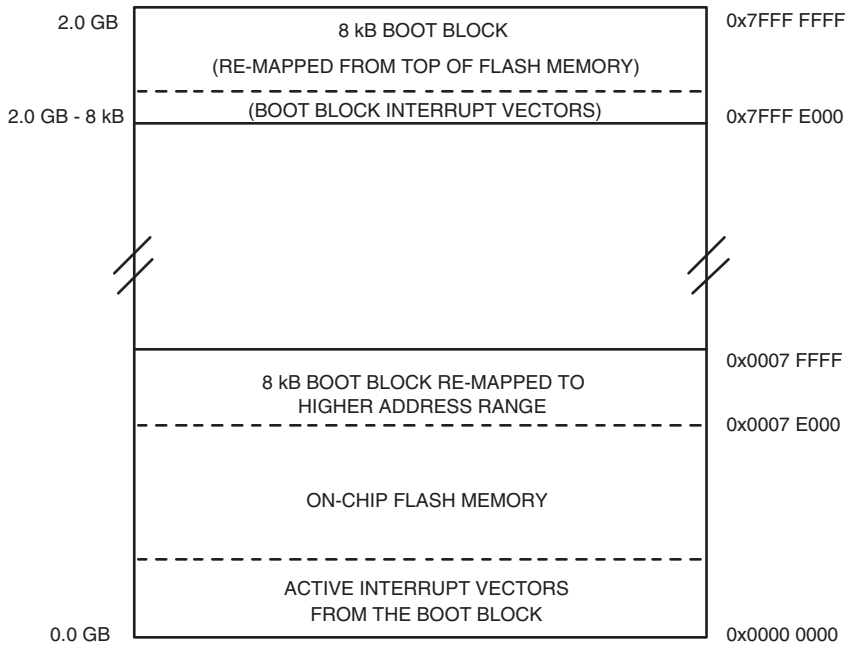


Fig 121.Map of lower memory after reset

4.1.1 Criterion for Valid User Code

Criterion for valid user code: The reserved ARM interrupt vector location (0x0000 0014) should contain the 2's complement of the check-sum of the remaining interrupt vectors. This causes the checksum of all of the vectors together to be 0. The boot loader code disables the overlaying of the interrupt vectors from the boot block, then checksums the interrupt vectors in sector 0 of the Flash. If the signatures match then the execution control is transferred to the user code by loading the program counter with 0x0000 0000. Hence the user Flash reset vector should contain a jump instruction to the entry point of the user application code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. The host should send a '?' (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also

sends an ASCII string ("Synchronized<CR><LF>") to the Host. In response to this host should send the same string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. Host should respond by sending the crystal frequency (in kHz) at which the part is running. For example, if the part is running at 10 MHz , the response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the CCLK frequency should be greater than or equal to 10 MHz.

For more details on Reset, PLL and startup/boot code interaction see Section 4–5.2 “PLL and startup/boot code interaction”.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in Flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 29–8 “ISP commands” on page 568.

4.2 Communication protocol

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

4.2.1 ISP command format

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

4.2.2 ISP response format

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

4.2.3 ISP data format

The data stream is in UU-encode format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

A description of UU-encode is available at the wotsit webpage.

4.2.4 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

4.2.5 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

4.2.6 Interrupts during ISP

The boot block interrupt vectors located in the boot block of the Flash are active after any reset.

4.2.7 Interrupts during IAP

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

4.2.8 RAM used by ISP command handler

ISP commands use on-chip RAM from 0x4000 0120 to 0x4000 01FF. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 256 bytes and it grows downwards.

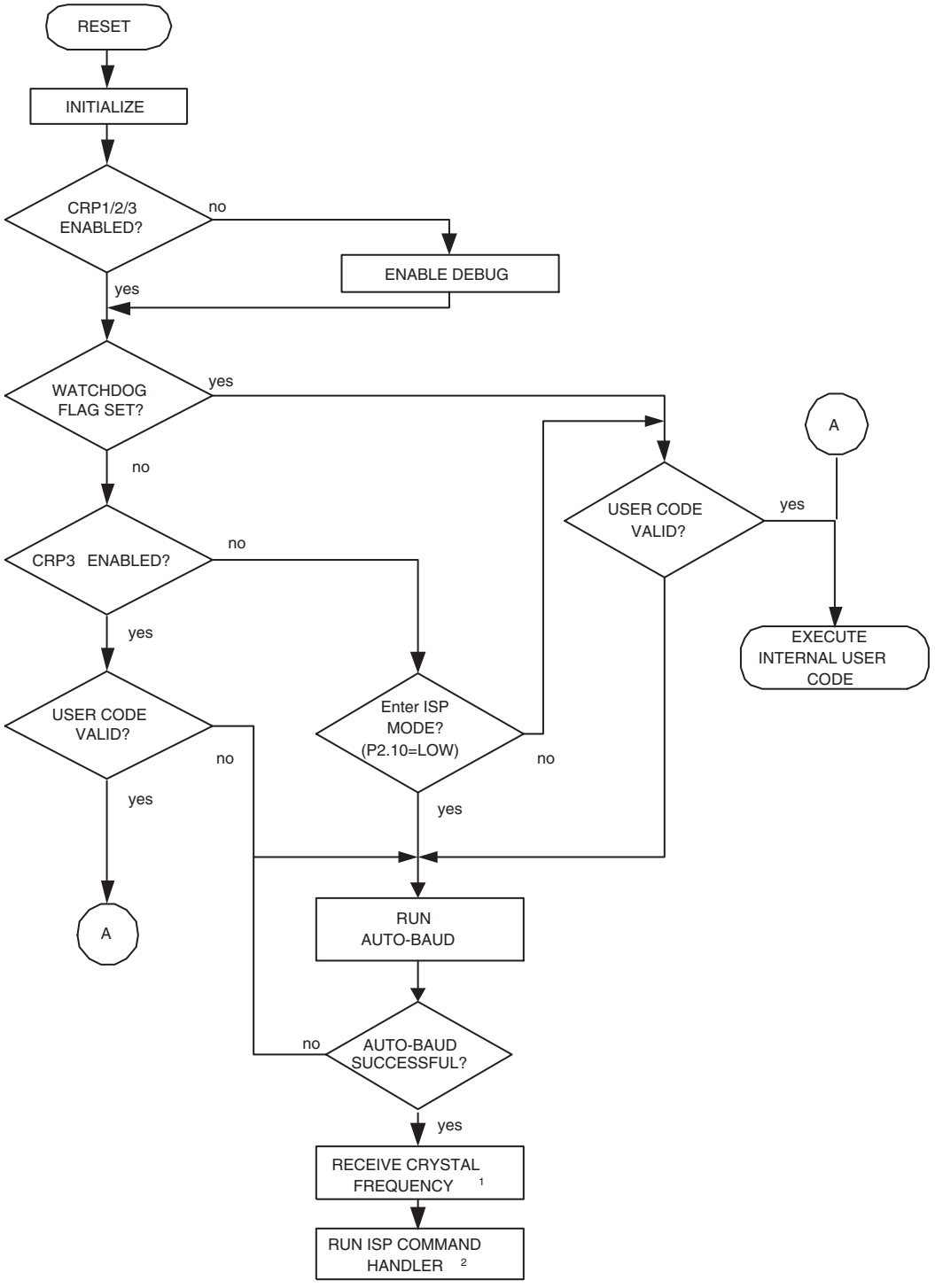
4.2.9 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes and it grows downwards.

4.2.10 RAM used by RealMonitor

The RealMonitor uses on-chip RAM from 0x4000 0040 to 0x4000 011F. The user could use this area if RealMonitor based debug is not required. The Flash boot loader does not initialize the stack for RealMonitor.

5. Boot process flowchart



(1) For details on handling the crystal frequency, see Section 29–9.8 “Reinvoke ISP” on page 580

(2) For details on available ISP commands based on the CRP settings see Section 29–7 “Code Read Protection (CRP)”

Fig 122.Boot process flowchart

6. Sector numbers

Some IAP and ISP commands operate on "sectors" and specify sector numbers. The following table indicate the correspondence between sector numbers and memory addresses for LPC2400 devices. IAP, ISP, and RealMonitor routines are located in the boot block. The boot block is present at addresses 0x0007 E000 to 0x0007 FFFF. ISP and IAP commands do not allow write/erase/go operation on the boot block. Because of the boot block, the amount of Flash available for user code and data is 504 K bytes.

Table 515. Sectors in a LPC2400 device

Sector number	Sector size [kB]	Address range
0	4	0X0000 0000 - 0X0000 0FFF
1	4	0X0000 1000 - 0X0000 1FFF
2	4	0X0000 2000 - 0X0000 2FFF
3	4	0X0000 3000 - 0X0000 3FFF
4	4	0X0000 4000 - 0X0000 4FFF
5	4	0X0000 5000 - 0X0000 5FFF
6	4	0X0000 6000 - 0X0000 6FFF
7	4	0X0000 7000 - 0X0000 7FFF
8	32	0x0000 8000 - 0X0000 FFFF
9	32	0x0001 0000 - 0X0001 7FFF
10 (0x0A)	32	0x0001 8000 - 0X0001 FFFF
11 (0x0B)	32	0x0002 0000 - 0X0002 7FFF
12 (0x0C)	32	0x0002 8000 - 0X0002 FFFF
13 (0x0D)	32	0x0003 0000 - 0X0003 7FFF
14 (0X0E)	32	0x0003 8000 - 0X0003 FFFF
15 (0x0F)	32	0x0004 0000 - 0X0004 7FFF
16 (0x10)	32	0x0004 8000 - 0X0004 FFFF
17 (0x11)	32	0x0005 0000 - 0X0005 7FFF
18 (0x12)	32	0x0005 8000 - 0X0005 FFFF
19 (0x13)	32	0x0006 0000 - 0X0006 7FFF
20 (0x14)	32	0x0006 8000 - 0X0006 FFFF
21 (0x15)	32	0x0007 0000 - 0X0007 7FFF
22 (0x16)	4	0x0007 8000 - 0X0007 8FFF
23 (0x17)	4	0x0007 9000 - 0X0007 9FFF
24 (0x18)	4	0x0007 A000 - 0X0007 AFFF
25 (0x19)	4	0x0007 B000 - 0X0007 BFFF
26 (0x1A)	4	0x0007 C000 - 0X0007 CFFF
27 (0x1B)	4	0x0007 D000 - 0X0007 DFFF

7. Code Read Protection (CRP)

Code Read Protection is a mechanism that allows user to enable different levels of security in the system so that access to the on-chip Flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in Flash location at 0x000001FC. IAP commands are not affected by the code read protection.

Starting with bootloader version 3.2 three levels of CRP are implemented. Earlier bootloader versions had only CRP2 option implemented.

Important: any CRP change becomes effective only after the device has gone through a power cycle.

Table 516. Code Rad Protection options

Name	Pattern programmed in 0x000001FC	Description
CRP1	0x12345678	Access to chip via the JTAG pins is disabled. This mode allows partial Flash update using the following ISP commands and restrictions: • Write to RAM command can not access RAM below 0x40000200 • Copy RAM to Flash command can not write to Sector 0 • Erase command can erase Sector 0 only when all sectors are selected for erase • Compare command is disabled This mode is useful when CRP is required and Flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the Flash.
CRP2	0x87654321	Access to chip via the JTAG pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to Flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the JTAG pins is disabled. ISP entry by pulling P2.10 LOW is disabled if a valid user code is present in Flash sector 0. This mode effectively disables ISP overide using P2.10 pin. It is up to the user's application to provide need Flash update mechanism using IAP calls or call reinvoke ISP command to enable Flash update via UART0. **Caution: when this mode is invoked, no future factory testing can be performed on the device.**

Table 517. Code Read Protection hardware/software interaction

CRP option	User Code Valid	P2.10 pin at reset	JTAG enabled	LPC2400 enters ISP mode	partial Flash Update in ISP mode
No	No	X	Yes	Yes	Yes
No	Yes	High	Yes	No	NA
No	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes
CRP2	Yes	High	No	Yes	NA
CRP2	Yes	Low	No	Yes	No
CRP3	Yes	High	No	No	NA
CRP3	Yes	Low	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

In case a CRP mode is enabled and access to the chip is allowed via the ISP, an unsupoorted or restricted ISP command will be terminated with return code CODE_READ_PROTECTION_ENABLED.

8. ISP commands

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

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

Table 518. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 29–519
Set Baud Rate	B <Baud Rate> <stop bit>	Table 29–520
Echo	A <setting>	Table 29–522
Write to RAM	W <start address> <number of bytes>	Table 29–523
Read Memory	R <address> <number of bytes>	Table 29–524
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 29–525
Copy RAM to Flash	C <Flash address> <RAM address> <number of bytes>	Table 29–526
Go	G <address> <Mode>	Table 29–527
Erase sector(s)	E <start sector number> <end sector number>	Table 29–528
Blank check sector(s)	I <start sector number> <end sector number>	Table 29–529

Table 518. ISP command summary

ISP Command	Usage	Described in
Read Part ID	J	Table 29–530
Read Boot code version	K	Table 29–532
Compare	M <address1> <address2> <number of bytes>	Table 29–533

8.1 Unlock <Unlock code>

Table 519. ISP Unlock command

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

8.2 Set Baud Rate <Baud Rate> <stop bit>

Table 520. ISP Set Baud Rate command

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 | 230400 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

Table 521. Correlation between possible ISP baudrates and CCLK frequency (in MHz)

ISP Baudrate .vs. CCLK Frequency	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456[1]	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

[1] ISP entry after reset uses the on chip IRC and PLL to run the device at CCLK = 14.748 MHz

8.3 Echo <setting>

Table 522. ISP Echo command

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

8.4 Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 523. ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. Data should be in UU-encoded format. This command is blocked when code read protection is enabled.
Example	"W 1073742336 4<CR><LF>" writes 4 bytes of data to address 0x4000 0200.

8.5 Read Memory <address> <no. of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with

"OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 524. ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (UU-encoded)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or Flash memory. This command is blocked when code read protection is enabled.
Example	"R 1073741824 4<CR><LF>" reads 4 bytes of data from address 0x4000 0000.

8.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 525. ISP Prepare sector(s) for write operation command

Command	P
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot block can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the Flash sector 0.

8.7 Copy RAM to Flash <Flash address> <RAM address> <no of bytes>

Table 526. ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination Flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the Flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot block cannot be written by this command. This command is blocked when code read protection is enabled.
Example	"C 0 1073774592 512<CR><LF>" copies 512 bytes from the RAM address 0x4000 8000 to the Flash address 0.

8.8 Go <address> <mode>

Table 527. ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode) | A (Execute program in ARM mode).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or Flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled.
Example	"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.

8.9 Erase sector(s) <start sector number> <end sector number>

Table 528. ISP Erase sector command

Command	E
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip Flash memory. The boot block can not be erased using this command. This command only allows erasure of all user sectors when the code read protection is enabled.
Example	"E 2 3<CR><LF>" erases the Flash sectors 2 and 3.

8.10 Blank check sector(s) <sector number> <end sector number>

Table 529. ISP Blank check sector command

Command	I
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR |
Description	This command is used to blank check one or more sectors of on-chip Flash memory. **Blank check on sector 0 always fails as first 64 bytes are re-mapped to Flash boot block.**
Example	"I 2 3<CR><LF>" blank checks the Flash sectors 2 and 3.

8.11 Read Part Identification number

Table 530. ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 29–531 "LPC2400 part Identification numbers").
Description	This command is used to read the part identification number.

Table 531. LPC2400 part Identification numbers

Device	ASCII/dec coding	Hex coding
2468	100925237	0x0603 FF35

8.12 Read Boot code version number

Table 532. ISP Read Boot Code version number command

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

8.13 Compare <address1> <address2> <no of bytes>

Table 533. ISP Compare command

Command	M
Input	**Address1 (DST):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Number of Bytes:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR |
Description	This command is used to compare the memory contents at two locations. **Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. First 64 bytes are re-mapped to Flash boot sector**
Example	"M 8192 1073741824 4<CR><LF>" compares 4 bytes from the RAM address 0x4000 0000 to the 4 bytes from the Flash address 0x2000.

8.14 ISP Return Codes

Table 534. ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.

Table 534. ISP Return Codes Summary

Return Code	Mnemonic	Description
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.
19	CODE_READ_PROTECTION_ENABLED	Code read protection enabled.

9. IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. Result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case if number of results are more than number of parameters. Parameter passing is illustrated in the Figure 29–123. The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 2, returned by the "Blankcheck sector(s)" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 0x7FFF FFF0 location and it is thumb code.

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

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x7ffffff1
```

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

```
unsigned long command[5];
```

```
unsigned long result[2];
```

or

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x……
result= (unsigned long *) 0x……
```

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

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

Setting function pointer:

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, result);
```

The IAP call could be simplified further by using the symbol definition file feature supported by ARM Linker in ADS (ARM Developer Suite). You could also call the IAP routine using assembly code.

The following symbol definitions can be used to link IAP routine and user application:

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
0x7fffffb0 A rm_prefetchabort_handler
0x7fffffc0 A rm_dataabort_handler
0x7fffffd0 A rm_irqhandler
0x7fffffe0 A rm_irqhandler2
0x7ffffff0 T iap_entry
```

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

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

Table 535. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50_{10}	Table 29–536
Copy RAM to Flash	51_{10}	Table 29–537
Erase sector(s)	52_{10}	Table 29–538
Blank check sector(s)	53_{10}	Table 29–539
Read Part ID	54_{10}	Table 29–540
Read Boot code version	55_{10}	Table 29–541
Compare	56_{10}	Table 29–542
Reinvoke ISP	57_{10}	Table 29–543

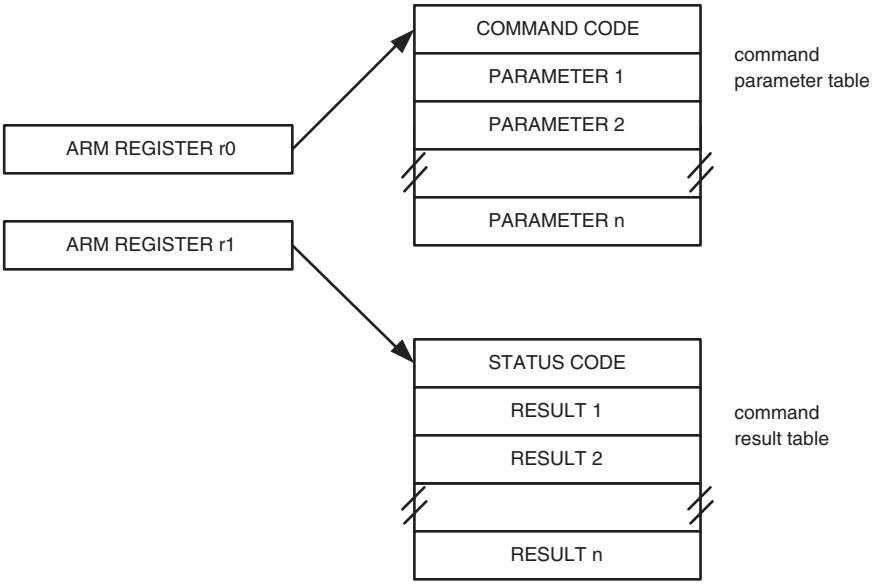


Fig 123.IAP parameter passing

9.1 Prepare sector(s) for write operation

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

Table 536. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Input	**Command code: 50_{10}** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).

Table 536. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.

9.2 Copy RAM to Flash

Table 537. IAP Copy RAM to Flash command

Command	Copy RAM to Flash
Input	**Command code: 51_{10}** **Param0(DST):** Destination Flash address where data bytes are to be written. This address should be a 256 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **Param3:** System Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY |
Result	None
Description	This command is used to program the Flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command.

9.3 Erase Sector(s)

Table 538. IAP Erase Sector(s) command

Command	Erase Sector(s)
Input	**Command code: 52_{10}** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number). **Param2:** System Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip Flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers.

9.4 Blank check sector(s)

Table 539. IAP Blank check sector(s) command

Command	Blank check sector(s)
Input	**Command code: 53_{10}** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Return Code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	**Result0:** Offset of the first non blank word location if the Status Code is SECTOR_NOT_BLANK. **Result1:** Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip Flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.

9.5 Read Part Identification number

Table 540. IAP Read Part Identification command

Command	Read part identification number
Input	**Command code: 54_{10}** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** Part Identification Number.
Description	This command is used to read the part identification number.

9.6 Read Boot code version number

Table 541. IAP Read Boot Code version number command

Command	Read boot code version number
Input	**Command code: 55_{10}** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

9.7 Compare <address1> <address2> <no of bytes>

Table 542. IAP Compare command

Command	Compare
Input	**Command code: 56_{10}** **Param0(DST):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param1(SRC):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param2:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations. **The result may not be correct when the source or destination includes any of the first 64 bytes starting from address zero. The first 64 bytes can be re-mapped to RAM.**

9.8 Reinvoke ISP

Table 543. Reinvoke ISP

Command	Compare
Input	**Command code: 57_{10}**

Table 543. Reinvoke ISP

Command	Compare
Return Code	None
Result	**None.**
Description	This command is used to invoke the bootloader in ISP mode. It maps boot vectors, sets PCLK = CCLK / 4, configures UART0 pins Rx and Tx, restets TIMER1 and resets the U0FDR (see Section 16–3.12). This command may be used when a valid user program is present in the internal Flash memory and the P2.10 pin is not accessible to force the ISP mode. The command does not disable the PLL hence it is possible to invoke the bootloader when the part is running off the PLL. In such case the ISP utility should pass the CCLK (crystal or PLL output depending on the clock source selection Section 4–4.1) frequency after autobaud handshake. Another option is to disable the PLL and select the IRC as the clock source before making this IAP call. In this case frequency sent by ISP is ignored and IRC and PLL are used to generate CCLK = 14.748 MHz.

9.9 IAP Status Codes

Table 544. IAP Status Codes Summary

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

10. JTAG Flash programming interface

Debug tools can write parts of the Flash image to the RAM and then execute the IAP call "Copy RAM to Flash" repeatedly with proper offset.

1. Introduction

The General Purpose DMA Controller (GPDMA) is an AMBA AHB compliant peripheral allowing selected LPC2400 peripherals to have DMA support.

2. Features of the GPDMA

- Two DMA channels. Each channel can support a unidirectional transfer.
- The GPDMA provides 16 peripheral DMA request lines. Some of these are connected to peripheral functions that support DMA: the SD/MMC, two SSP, and I2S interfaces.
- Single DMA and burst DMA request signals. Each peripheral connected to the GPDMA can assert either a burst DMA request or a single DMA request. The DMA burst size is set by programming the GPDMA.
- Memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers.
- 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. Each DMA channel has a specific hardware priority. DMA channel 0 has the highest priority and channel 1 has the lowest priority. If requests from two channels become active at the same time the channel with the highest priority is serviced first.
- AHB slave DMA programming interface. The GPDMA is programmed by writing to the DMA control registers over the AHB slave interface.
- One AHB bus master for transferring data. This interface transfers data when a DMA request goes active.
- 32 bit AHB master bus width.
- Incrementing or non-incrementing addressing for source and destination.
- Programmable DMA burst size. The DMA burst size can be programmed to more efficiently transfer data. Usually the burst size is set to half the size of the FIFO in the peripheral.
- Internal four-word FIFO per channel.
- Supports 8, 16, and 32 bit wide transactions.
- Big-endian and little-endian support. The GPDMA 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.
- Interrupt masking. The DMA error and DMA terminal count interrupt requests can be masked.
- Raw interrupt status. The DMA error and DMA count raw interrupt status can be read prior to masking.
- Test registers for use in block and integration system level testing.

- Identification registers that uniquely identify the GPDMA. These can be used by an operating system to automatically configure itself.

3. Functional overview

This chapter describes the major functional blocks of the GPDMA. It contains the following sections:

- GPDMA functional description
- System considerations
- System connectivity
- Use with memory management unit based systems

3.1 GPDMA functional description

The GPDMA 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 30–124 shows a block diagram of the GPDMA.

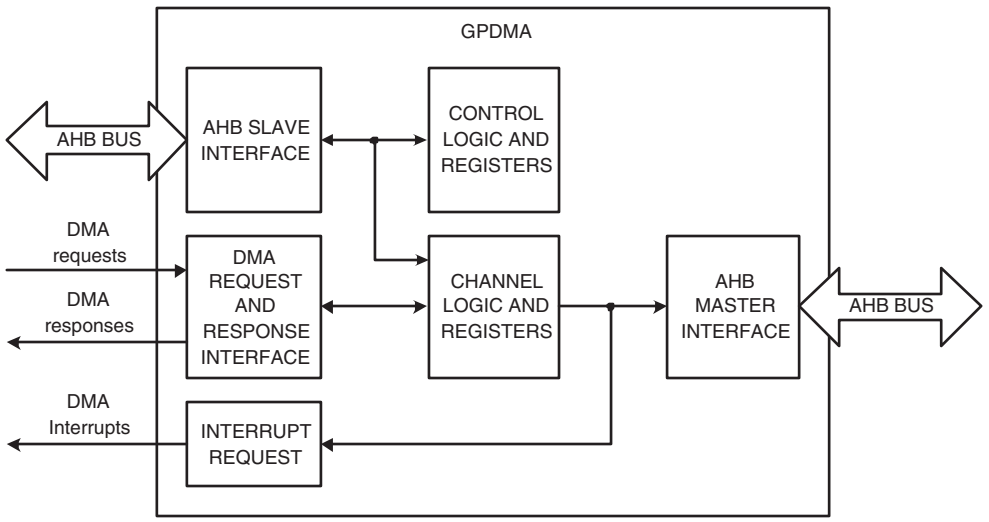


Fig 124.GPDMA block diagram

The functions of the GPDMA are described in the following sections:

- AHB slave interface
- Control logic and register bank
- DMA request and response interface
- Channel logic and channel register bank

- Interrupt request
- AHB master interface
- Channel hardware
- DMA request priority

3.1.1 AHB Slave Interface

All transactions on the AHB slave programming bus of the GPDMA are 32 bit wide.

3.1.2 Control Logic and Register Bank

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

3.1.3 DMA Request and Response Interface

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

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.

3.1.5 Interrupt Request

The interrupt request generates interrupts to the ARM processor.

3.1.6 AHB Master Interface

The GPDMA contains a full AHB master. See Figure 30–125 for an example showing the GPDMA connected into a system.

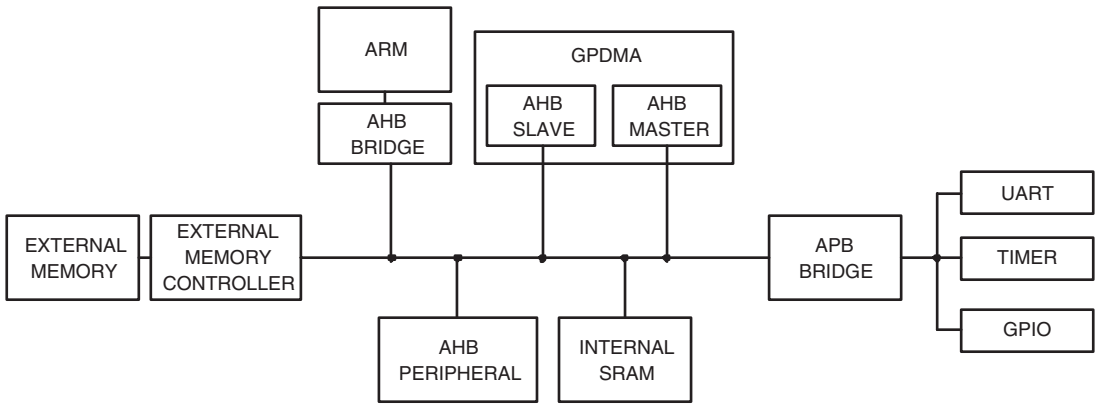


Fig 125.Example of GPDMA in a system

The AHB master is capable of dealing with all types of AHB transactions, including:

- Split, retry, and error responses from slaves. If a peripheral performs a split or retry, the GPDMA 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.

3.1.7 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 GPDMA packs or unpacks data as appropriate.

3.1.8 Endian behavior

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

Internally the GPDMA 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 you do not require byte swapping then avoid using different endianness between the source and destination addresses.

Table 30–545 shows endian behavior for different source and destination combinations.

Table 545. 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
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
				2/[15:8]	43		
				3/[23:16]	65		
				4/[31:24]	87		

Table 545. 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	32	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				2/[23:16]	65	3/[23:16]	65656565
				2/[31:24]	87	4/[31:24]	87878787
Little	Little	32	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	32	32	1/[7:0]	21	1/[31:0]	87654321
				2/[15:8]	43		
				3/[23:16]	65		
				4/[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
				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	16	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	16	32	1/[31:24]	12	1/[31:0]	12345678
				2/[23:16]	34		
				3/[15:8]	56		
				4/[7:0]	78		

Table 545. Endian behavior

Source Endian	Destination Endian	Source Width	Destination Width	Source Transfer no/ byte Lane	Source Data	Destination Transfer no/ byte Lane	Destination Data
Big	Big	32	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	32	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	32	32	1/[31:24]	12	1/[31:0]	12345678
				2/[23:16]	34		
				3/[15:8]	56		
				4/[7:0]	78		

3.1.9 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 GPDMA 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.

3.1.10 Channel hardware

Each stream is supported by a dedicated hardware channel, including source and destination controllers, and 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.

3.1.11 DMA request priority

DMA channel priority is fixed. DMA channel 0 has the highest priority and DMA channel 1 has the lowest priority.

If the GPDMA 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. Channel 1 in the GPDMA is designed so that it cannot saturate the AHB bus. If it goes active, the GPDMA relinquishes control of the bus (for a bus cycle), after four transfers of the programmed size (irrespective of the size of transfer). This enables other AHB masters to access the bus.

It is recommended that memory-to-memory transactions use the low priority channel. Otherwise other (lower priority) AHB bus masters are prevented from accessing the bus during GPDMA memory-to-memory transfer.

3.1.12 Interrupt generation

A combined interrupt output is generated as an OR function of the individual interrupt requests of the GPDMA, and is connected to the LPC2400 interrupt controller.

3.1.13 The completion of the DMA transfer indication

The completion of the DMA transfer is indicated by:

1. The transfer count reaching 0 if the GPDMA is performing flow control, OR
2. The peripheral setting the DMA Last Word Request Input (DMACLSREQ) or the DMA Last Burst Request Input (DMALBREQ) if the peripheral is performing flow control.

According to Table 30–546 “DMA Connections”, SSP0, SSP1 and I2S do not use DMA Last Word Request Input nor DMA Last Burst Request Input. Therefore there will be no indication of completion if SSP0, SSP1 and I2S are performing the flow control.

3.2 DMA system connections

The connection of the GPDMA to the supported peripheral devices depends on the DMA functions implemented in those peripherals. Table 30–546 shows the DMA Request numbers used by the supported peripherals.

Table 546. DMA Connections

Peripheral Function	DMA Single Request Input	DMA Burst Request Input	DMA Last Word Request Input	DMA Last Burst Request Input
SSP0 Tx	0	0	-	-
SSP0 Rx	1	1	-	-
SSP1 Tx	2	2	-	-
SSP1 Rx	3	3	-	-
SD/MMC	4	4	4	4
I2S channel 0	-	5	-	-
I2S channel 1	-	6	-	-

4. Programmer’s model

This chapter describes the GPDMA registers and provides details required when programming the microcontroller. It contains the following sections:

- About the programmer’s model.
- Programming the GPDMA.
- Summary of GPDMA registers.
- Register descriptions.
- Address generation.
- Scatter/gather.
- Interrupt requests.
- GPDMA data flow.

5. About the programmer's model

The GPDMA enables peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral, and memory-to-memory transactions. Each DMA stream is configured to provide unidirectional DMA transfers for a single source and destination. The source and destination areas can each be either a memory region or a peripheral which supports the GPDMA, and must be accessible through AHB1.

6. Programming the GPDMA

The following applies to the registers used in the GPDMA:

- Reserved or unused address locations must not be accessed because this can result in unpredictable behavior of the device.
- Reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.
- All register bits are reset to a logic 0 by a system or power-on reset unless otherwise stated in the relevant text.
- Unless otherwise stated in the relevant text, all registers support read and write accesses. A write updates the contents of a register and a read returns the contents of the register.
- All registers defined in this document can only be accessed using word reads and word writes (i.e. 32 bit accesses), unless otherwise stated in the relevant text.

6.1 Enabling the GPDMA

To enable the GPDMA set the DMA Enable bit in the DMACConfiguration Register (Section 30–8.13 "Configuration Register (DMACConfiguration - 0xFFE0 4030)".

6.2 Disabling the GPDMA

To disable the GPDMA:

1. Read the DMACEnbldChns Register and ensure that all the DMA channels have been disabled. If any channels are active, see Section 30–6.4 "Disabling a DMA channel".
2. Disable the GPDMA by writing 0 to the DMA Enable bit in the DMACConfiguration Register (Section 30–9.6 "Channel Configuration Registers (DMACC0Configuration - 0xFFE0 4110 and DMACC1Configuration - 0xFFE0 4130)").

6.3 Enabling a DMA channel

To enable the DMA channel set the Channel Enable bit in the relevant DMA channel Configuration Register (Section 30–9.6 "Channel Configuration Registers (DMACC0Configuration - 0xFFE0 4110 and DMACC1Configuration - 0xFFE0 4130)").

Note: The channel must be fully initialized before it is enabled. Additionally, you must set the Enable bit of the GPDMA before any channels are enabled.

6.4 Disabling a DMA channel

You can disable a DMA channel in the following ways:

- Write directly to the Channel Enable bit. Any outstanding data in the FIFOs is lost if this method is used.
- Use the Active and Halt bits in conjunction with the Channel Enable bit.
- Wait until the transfer completes. The channel is then automatically disabled.

6.5 Disabling a DMA channel without losing data in the FIFO

To disable a DMA channel without losing data in the FIFO:

1. Set the Halt bit in the relevant channel Configuration Register (Section 30–9.6 “Channel Configuration Registers (DMACC0Configuration - 0xFFE0 4110 and DMACC1Configuration - 0xFFE0 4130)”). This causes any further DMA requests to be ignored.
2. 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 which has to be transferred.
3. Clear the Channel Enable bit in the relevant channel Configuration Register.

6.6 Setup a new DMA transfer

To set up a new DMA transfer:

1. If the channel is not set aside for the DMA transaction:
   - Read the DMACEnbldChns Register and find out which channels are inactive (see Section 30–8.8 “Enabled Channel Register (DMACEnbldChns - 0xFFE0 401C)”).
   - Choose an inactive channel that has the required priority.
2. Program the GPDMA.

6.7 Disabling a DMA channel and losing data in the FIFO

Clear the relevant Channel Enable bit in the relevant channel Configuration Register (Section 30–9.6 “Channel Configuration Registers (DMACC0Configuration - 0xFFE0 4110 and DMACC1Configuration - 0xFFE0 4130)”). The current AHB transfer, if one is in progress, completes and the channel is disabled. Any data in the FIFO is lost.

6.8 Halting a DMA transfer

Set the Halt bit in the relevant DMA channel Configuration Register. The current source request is serviced. Any further source DMA requests are ignored until the Halt bit is cleared.

6.9 Programming a DMA channel

To program a DMA channel:

1. Choose a free DMA channel with the priority required. DMA channel 0 has the highest priority and DMA channel 1 the lowest priority.

2. Clear any pending interrupts on the channel to be used by writing to the DMACIntTCClr Register (Section 30–8.3 “Interrupt Terminal Count Clear Register (DMACIntClear - 0xFFE0 4008)”) and DMACIntErrClr Register (Section 30–8.5 “Interrupt Error Clear Register (DMACIntErrClr - 0xFFE0 4010)”). The previous channel operation might have left interrupts active.
3. Write the source address into the DMACCxSrcAddr Register (Section 30–9.1 “Channel Source Address Registers (DMACC0SrcAddr - 0xFFE0 4100 and DMACC1SrcAddr - 0xFFE0 4120)”).
4. Write the destination address into the DMACCxDestAddr Register (Section 30–9.2 “Channel Destination Address Registers (DMACC0DestAddr - 0xFFE0 4104 and DMACC1DestAddr - 0xFFE0 4124)”).
5. Write the address of the next Linked List Item (LLI) into the DMACCxLLI Register (Section 30–9.3 “Channel Linked List Item Registers (DMACC0LLI - 0xFFE0 4108 and DMACC1LLI - 0xFFE0 4128)”). If the transfer consists of a single packet of data then 0 must be written into this register.
6. Write the control information into the DMACCxControl Register (Section 30–9.4 “Channel Control Registers (DMACC0Control - 0xFFE0 410C and DMACC0Control - 0xFFE0 412C)”).
7. Write the channel configuration information into the DMACCxConfiguration Register (Section 30–9.6 “Channel Configuration Registers (DMACC0Configuration - 0xFFE0 4110 and DMACC1Configuration - 0xFFE0 4130)”). If the Enable bit is set then the DMA channel is automatically enabled.

7. Summary of GPDMA registers

The GPDMA registers are shown in Table 30–547.

Table 547. GPDMA register map

Name	Description	Access	Reset Value	Address
General Registers				
DMACIntStatus	Interrupt Status Register	RO	0x0	0xFFE0 4000
DMACIntTCStatus	Interrupt Terminal Count Status Register	RO	0x0	0xFFE0 4004
DMACIntTCClear	Interrupt Terminal Count Clear Register	WO	-	0xFFE0 4008
DMACIntErrorStatus	Interrupt Error Status Register	RO	0x0	0xFFE0 400C
DMACIntErrClr	Interrupt Error Clear Register	WO	-	0xFFE0 4010
DMACRawIntTCStatus	Raw Interrupt Terminal Count Status Register	RO	-	0xFFE0 4014
DMACRawIntErrorStatus	Raw Error Interrupt Status Register	RO	-	0xFFE0 4018
DMACEnbldChns	Enabled Channel Register	RO	0x0	0xFFE0 401C
DMACSoftBReq	Software Burst Request Register	R/W	0x0000	0xFFE0 4020
DMACSoftSReq	Software Single Request Register	R/W	0x0000	0xFFE0 4024
DMACSoftLBReq	Software Last Burst Request Register	R/W	0x0000	0xFFE0 4028

Table 547. GPDMA register map

Name	Description	Access	Reset Value	Address
DMACSoftLSReq	Software Last Single Request Register	R/W	0x0000	0xFFE0 402C
DMACConfiguration	Configuration Register	R/W	0x0000 0000	0xFFE0 4030
DMACSync	Synchronization Register	R/W	0x0000	0xFFE0 4034
Channel 0 Registers				
DMACC0SrcAddr	Channel 0 Source Address Register	R/W	0x0000 0000	0xFFE0 4100
DMACC0DestAddr	Channel 0 Destination Address Register	R/W	0x0000 0000	0xFFE0 4104
DMACC0LLI	Channel 0 Linked List Item Register	R/W	0x0000 0000	0xFFE0 4108
DMACC0Control	Channel 0 Control Register	R/W	0x0000 0000	0xFFE0 410C
DMACC0Configuration	Channel 0 Configuration Register	R/W	0x00000 [1]	0xFFE0 4110
Channel 1 Registers				
DMACC1SrcAddr	Channel 1 Source Address Register	R/W	0x0000 0000	0xFFE0 4120
DMACC1DestAddr	Channel 1 Destination Address Register	R/W	0x0000 0000	0xFFE0 4124
DMACC1LLI	Channel 1 Linked List Item Register	R/W	0x0000 0000	0xFFE0 4128
DMACC1Control	Channel 1 Control Register	R/W	0x0000 0000	0xFFE0 412C
DMACC1Configuration	Channel 1 Configuration Register	R/W	0x00000 [1]	0xFFE0 4130

[1] Bit [17] is read-only.

8. Register descriptions

This section describes the registers of the GPDMA.

8.1 Interrupt Status Register (DMACIntStatus - 0xFFE0 4000)

The DMACIntStatus 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 30–548 shows the bit assignments of the DMACIntStatus Register.

Table 548. Interrupt Status register (DMACIntStatus - address 0xFFE0 4000) bit description

Bit	Symbol	Description	Reset Value
0	IntStatus0	Status of channel 0 interrupts after masking.	0
1	IntStatus1	Status of channel 1 interrupts after masking.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.2 Interrupt Terminal Count Status Register (DMACIntTCStatus - 0xFFE0 4004)

The DMACIntTCStatus Register is read-only and indicates the status of the terminal count after masking. Table 30–549 shows the bit assignments of the DMACIntTCStatus Register.

Table 549. Interrupt Terminal Count Status register (DMACIntTCStatus - address 0xFFE0 4004) bit description

Bit	Symbol	Description	Reset Value
0	IntTCStatus0	Terminal count interrupt request status for channel 0.	0
1	IntTCStatus1	Terminal count interrupt request status for channel 1.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.3 Interrupt Terminal Count Clear Register (DMACIntClear - 0xFFE0 4008)

The DMACIntTCClear Register is write-only and clears a terminal count interrupt request. When writing to this register, each data bit that is set 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 30–550 shows the bit assignments of the DMACIntTCClear Register.

Table 550. Interrupt Terminal Count Clear register (DMACIntClear - address 0xFFE0 4008) bit description

Bit	Symbol	Description	Reset Value
0	IntTCClear0	Writing a 1 clears the terminal count interrupt request for channel 0 (IntTCStatus0).	-
1	IntTCClear1	Writing a 1 clears the terminal count interrupt request for channel 1 (IntTCStatus1).	-
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.4 Interrupt Error Status Register (DMACIntErrorStatus - 0xFFE0 400C)

The DMACIntErrorStatus Register is read-only and indicates the status of the error request after masking. Table 30–551 shows the bit assignments of the DMACIntErrorStatus Register.

Table 551. Interrupt Error Status register (DMACIntErrorStatus - address 0xFFE0 400C) bit description

Bit	Symbol	Description	Reset Value
0	IntErrorStatus0	Interrupt error status for channel 0.	0x0
1	IntErrorStatus1	Interrupt error status for channel 1.	0x0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.5 Interrupt Error Clear Register (DMACIntErrClr - 0xFFE0 4010)

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 30–552 shows the bit assignments of the DMACIntErrClr Register.

Table 552. Interrupt Error Clear register (DMACIntErrClr - address 0xFFE0 4010) bit description

Bit	Symbol	Description	Reset Value
0	IntErrClr0	Writing a 1 clears the error interrupt request for channel 0 (IntErrorStatus0).	-
1	IntErrClr1	Writing a 1 clears the error interrupt request for channel 1 (IntErrorStatus1).	-
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.6 Raw Interrupt Terminal Count Status Register (DMACRawIntTCStatus - 0xFFE0 4014)

The DMACRawIntTCStatus Register is read-only and indicates which DMA channel is requesting a transfer complete (terminal count interrupt) prior to masking. A HIGH bit indicates that the terminal count interrupt request is active prior to masking. Table 30–553 shows the bit assignments of the DMACRawIntTCStatus Register.

Table 553. Raw Interrupt Terminal Count Status register (DMACRawIntTCStatus - address 0xFFE0 4014) bit description

Bit	Symbol	Description	Reset Value
0	RawIntTCStatus0	Status of the terminal count interrupt for channel 0 prior to masking.	-
1	RawIntTCStatus1	Status of the terminal count interrupt for channel 1 prior to masking.	-
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.7 Raw Error Interrupt Status Register (DMACRawIntErrorStatus - 0xFFE0 4018)

The DMACRawIntErrorStatus Register is read-only and indicates which DMA channel is requesting an error interrupt prior to masking. A HIGH bit indicates that the error interrupt request is active prior to masking. Table 30–554 shows the bit assignments of register of the DMACRawIntErrorStatus Register.

Table 554. Raw Error Interrupt Status register (DMACRawIntErrorStatus - address 0xFFE0 4018) bit description

Bit	Symbol	Description	Reset Value
0	RawIntErrorStatus0	Status of the error interrupt for channel 0 prior to masking.	-
1	RawIntErrorStatus1	Status of the error interrupt for channel 1 prior to masking.	-
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.8 Enabled Channel Register (DMACEnbldChns - 0xFFE0 401C)

The DMACEnbldChns Register is read-only and indicates which DMA channels are enabled, as indicated by the Enable bit in the DMACCxConfiguration Register. A HIGH bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer. Table 30–555 shows the bit assignments of the DMACEnbldChns Register.

Table 555. Enabled Channel register (DMACEnbldChns - address 0xFFE0 401C) bit description

Bit	Symbol	Description	Reset Value
0	EnabledChannels0	Enable status for Channel 0.	0
1	EnabledChannels1	Enable status for Channel 1.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.9 Software Burst Request Register (DMACSoftBReq - 0xFFE0 4020)

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. Writing 0 to this register has no effect. 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. Table 30–556 shows the bit assignments of the DMACSoftBReq Register.

Table 556. Software Burst Request register (DMACSoftBReq - address 0xFFE0 4020) bit description

Bit	Symbol	Description	Reset Value
0	SoftBReqSSP0Tx	Software burst request flag for SSP0 Tx.	0
1	SoftBReqSSP0Rx	Software burst request flag for SSP0 Rx.	0
2	SoftBReqSSP1Tx	Software burst request flag for SSP1 Tx.	0
3	SoftBReqSSP1Rx	Software burst request flag for SSP1 Rx.	0
4	SoftBReqSDMMC	Software burst request flag for SD/MMC.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

8.10 Software Single Request Register (DMACSoftSReq - 0xFFE0 4024)

The DMACSoftSReq Register is read/write and enables DMA 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. Writing 0 to this register has no effect. 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 30–557 shows the bit assignments of the DMACSoftSReq Register.

Table 557. Software Single Request register (DMACSoftSReq - address 0xFFE0 4024) bit description

Bit	Symbol	Description	Reset Value
0	SoftReqSSP0Tx	Single software request flag for SSP0 Tx.	0
1	SoftReqSSP0Rx	Single software request flag for SSP0 Rx.	0
2	SoftReqSSP1Tx	Single software request flag for SSP1 Tx.	0
3	SoftReqSSP1Rx	Single software request flag for SSP1 Rx.	0
4	SoftReqSDMMC	Single software request flag for SD/MMC.	0
5	SoftSReqI2S0	Single software request flag for i2S0.	0
6	SoftSReqI2S1	Single software request flag for i2S1.	0
31:7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.11 Software Last Burst Request Register (DMACSoftLBreq - 0xFFE0 4028)

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. Writing 0 to this register has no effect. 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 30–558 shows the bit assignments of the DMACSoftLBReq Register.

Table 558. Software Last Burst Request register (DMACSoftLBReq - address 0xFFE0 4028) bit description

Bit	Symbol	Description	Reset Value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	SoftLBReqSDMMC	Software last burst request flags for SD/MMC.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.12 Software Last Single Request Register (DMACSoftLSReq - 0xFFE0 402C)

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. Writing 0 to this register has no effect. 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 30–559 shows the bit assignments of the DMACSoftLSReq Register.

Table 559. Software Last Single Request register (DMACSoftLSReq - address 0xFFE0 402C) bit description

Bit	Symbol	Description	Reset Value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	SoftLSReqSDMMC	Software last single request flags for SD/MMC.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.13 Configuration Register (DMACConfiguration - 0xFFE0 4030)

The DMACConfiguration Register is read/write and configures the operation of the GPDMA. 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 30–560 shows the bit assignments of the DMACConfiguration Register.

Table 560. Configuration register (DMACConfiguration - address 0xFFE0 4030) bit description

Bit	Symbol	Value	Description	Reset Value
0	E		GPDMA enable:	0
		0	Disabled. Disabling the GPDMA reduces power consumption.	
		1	Enabled.	
1	M		AHB Master endianness configuration:	0
		0	Little-endian mode.	
		1	Big-endian mode.	
31:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.14 Synchronization Register (DMACSync - 0xFFE0 4034)

The DMACSync Register is read/write and enables or disables synchronization logic for the DMA request signals. The DMA request signals consist of the DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0], and DMACLSREQ[15:0]. A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests. This register is reset to 0, synchronization logic enabled.

Table 30–561 shows the bit assignments of the DMACSync Register.

Table 561. Synchronization register (DMACSync - address 0xFFE0 4034) bit description

Bit	Symbol	Description	Reset Value
15:0	DMACSync	DMA synchronization logic for DMA request signals enabled or disabled. A LOW bit indicates that the synchronization logic for the DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0], and DMACLSREQ[15:0] request signals is enabled. A HIGH bit indicates that the synchronization logic is disabled.	0x0000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

9. Channel registers

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

- Two DMACCxSrcAddr Registers
- Two DMACCxDestAddr Registers
- Two DMACCxLLI Registers
- Two DMACCxControl Registers
- Two DMACCxConfiguration Registers

When performing scatter/gather DMA the first four registers are automatically updated.

9.1 Channel Source Address Registers (DMACC0SrcAddr - 0xFFE0 4100 and DMACC1SrcAddr - 0xFFE0 4120)

The two read/write DMACCxSrcAddr Registers contain the current source address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the appropriate channel is enabled. When the DMA channel is enabled this register is updated:

- As the source address is incremented.
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time software has processed the value read, the channel might 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 30–562 shows the bit assignments of the DMACCxSrcAddr Registers.

Table 562. Channel Source Address registers (DMACC0SrcAddr - address 0xFFE0 4100 and DMACC1SrcAddr - address 0xFFE0 4120) bit description

Bit	Symbol	Description	Reset Value
31:0	SrcAddr	DMA source address.	0x0000 0000

9.2 Channel Destination Address Registers (DMACC0DestAddr - 0xFFE0 4104 and DMACC1DestAddr - 0xFFE0 4124)

The two read/write DMACCxDestAddr Registers contain the current destination address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the channel is enabled. When the DMA channel is enabled the register is updated as the destination address is incremented and by following the linked list when a complete packet of data has been transferred. Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the channel might 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 30–563 shows the bit assignments of the DMACCxDestAddr Register.

Table 563. Channel Destination Address registers (DMACC0DestAddr - address 0xFFE0 4104 and DMACC1DestAddr - address 0xFFE0 4124) bit description

Bit	Symbol	Description	Reset Value
31:0	DestAddr	DMA destination address	0x0000 0000

9.3 Channel Linked List Item Registers (DMACC0LLI - 0xFFE0 4108 and DMACC1LLI - 0xFFE0 4128)

The two read/write DMACCxLLI Registers contain a word-aligned address of the next Linked List Item (LLI). If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled when all DMA transfers associated with it are completed.

Note: Programming this register when the DMA channel is enabled has unpredictable side effects.

Table 30–564 shows the bit assignments of the DMACCxLLI Register.

Table 564. Channel Linked List Item registers (DMACC0LLI - address 0xFFE0 4108 and DMACC1LLI - address 0xFFE0 4128) bit description

Bit	Symbol	Description	Reset Value
0	Reserved	Reserved, read as zero, do not modify.	NA
1	R	Reserved, and must be written as 0, masked on read.	0
31:2	LLI	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.	0

Note: To make loading the LLIs more efficient for some systems, the LLI data structures can be made four-word aligned.

9.4 Channel Control Registers (DMACC0Control - 0xFFE0 410C and DMACC0Control - 0xFFE0 412C)

The two read/write DMACCxControl Registers contain DMA channel control information such as the transfer size, burst size, and transfer width. Each register is programmed directly by software before the DMA channel is enabled. When the channel is enabled the register is updated by following the linked list when a complete packet of data has been transferred. Reading the register while the channel is active does not give useful information. This is because by the time software has processed the value read, the channel might have progressed. It is intended to be read only when a channel has stopped. Table 30–565 shows the bit assignments of the DMACCxControl Register.

Table 565. Channel Control registers (DMACC0Control - address 0xFFE0 410C and DMACC1Control - address 0xFFE0 412C) bit description

Bit	Symbol	Description	Reset Value
11:0	TransferSize	Transfer size. A write to this field sets the size of the transfer when the GPDMA is the flow controller.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 GPDMA is not the flow controller.	0
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.	0
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.	0
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.	0
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.	0
25:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26	SI	Source increment. When set the source address is incremented after each transfer.	0
27	DI	Destination increment. When set the destination address is incremented after each transfer.	0
30:28	Prot	Protection.	0
31	I	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.	0

Table 30–566 shows the value of the 3 bit DBSize or SBSize fields and the corresponding burst sizes.

Table 566. Source or destination burst size

Bit value of DBSize or SBSize	Source or distention burst transfer request size
000	1
001	4
010	8
011	16

Table 566. Source or destination burst size

Bit value of DBSize or SBSize	Source or distention burst transfer request size
100	32
101	64
110	128
111	256

Table 30–567 shows the value of the 3 bit SWidth or DWidth fields and the corresponding transfer width.

Table 567. Source or destination transfer width

Bit value of DBWidth or SBWidth	Source or distention burst transfer request size
000	Byte (8 bit)
001	Halfword (16 bit)
010	Word (32 bit)
011 and 1xxx	Reserved

9.5 Protection and Access Information

AHB access information is provided to the source and destination peripherals when a transfer occurs. The transfer information is provided by programming the DMA channel (the Prot bit of the DMACCxControl Register, and the Lock bit of the DMACCxConfiguration Register). These bits are programmed by software and peripherals can use this information if necessary. Three bits of information are provided, and Table 30–568 shows the purpose of the three protection bits.

Table 568. Protection bits

DMACC1Control Bit	Value	Description	Reset Value
28		Privileged or User. This bit controls the AHB HPROT[1] signal. Indicates that the access is in User, or privileged mode:	0
	0	User mode.	
	1	Privileged mode.	
29		Bufferable or not bufferable. This bit indicates that the access is bufferable. This bit can, for example, be used to indicate to an AMBA bridge that the read can complete in zero wait states on the source bus without waiting for it to arbitrate for the destination bus and for the slave to accept the data. This bit controls the AHB HPROT[2] signal. Indicates that the access is bufferable, or not bufferable:	0
	0	Not bufferable.	
	1	Bufferable.	

Table 568. Protection bits

DMACC1Control Bit	Value	Description	Reset Value
30		Cacheable or not cacheable. This indicates that the access is cacheable. This bit can, for example, be used to indicate to an AMBA bridge that when it saw the first read of a burst of eight it can transfer the whole burst of eight reads on the destination bus, rather than pass the transactions through one at a time. This bit controls the AHB HPROT[3] signal. Indicates that the access is cacheable or not cacheable:	0
	0	Not cacheable.	
	1	Cacheable.	

9.6 Channel Configuration Registers (DMACC0Configuration - 0xFFE0 4110 and DMACC1Configuration - 0xFFE0 4130)

The two DMACCxConfiguration Registers are read/write with the exception of bit[17] which is read-only. Used these to configure the DMA channel. The registers are not updated when a new LLI is requested. Table 30–569 shows the bit assignments of the DMACCxConfiguration Register.

Table 569. Channel Configuration registers (DMACC0Configuration - address 0xFFE0 4110 and DMACC1Configuration - address 0xFFE0 4130) bit description

Bit	Symbol	Value	Description	Reset Value
0	E		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 and the channel must be fully re-initialized. The channel is also disabled, and Channel Enable bit cleared, when the last LLI is reached or if a channel error is encountered. If a channel has to 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. Channel enable -- reading this bit indicates whether a channel is currently enabled or disabled:	0
		0	Channel disabled.	
		1	Channel enabled.	

Table 569. Channel Configuration registers (DMACC0Configuration - address 0xFFE0 4110 and DMACC1Configuration - address 0xFFE0 4130) bit description

Bit	Symbol	Value	Description	Reset Value
4:1	SrcPeripheral		Source peripheral. This value selects the DMA source request peripheral.This field is ignored if the source of the transfer is from memory.	0
		0000	SSP0 Tx	
		0001	SSP0 Rx	
		0010	SSP1 Tx	
		0011	SSP1 Rx	
		0100	SD/MMC	
		0101	I2S channel 0	
		0110	I2S channel 1	
		0111 or 1xxx	These values are reserved and should not be used.	
5	-	-	Reserved, do not modify, masked on read.	NA
9:6	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 the SrcPeripheral symbol description for values.	0
10	-	-	Reserved, do not modify, masked on read.	NA
13:11	FlowCntrl		Flow control and transfer type. This value indicates the flow controller and transfer type. The flow controller can be the GPDMA, 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.	0
14	IE		Interrupt error mask. When cleared this bit masks out the error interrupt of the relevant channel.	0
15	ITC		Terminal count interrupt mask. When cleared this bit masks out the terminal count interrupt of the relevant channel.	0
16	L		Lock. When set, this bit enables locked transfers.	0
17	A		Active. This value can be used with the Halt and Channel Enable bits to cleanly disable a DMA channel. Writing to this bit has no effect.	
		0	There is no data in the FIFO of the channel.	
		1	The channel FIFO has data.	
18	H		Halt. 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.	0
		0	Enable DMA requests.	
		1	Ignore further source DMA requests.	
31:19	-		Reserved, do not modify, masked on read.	NA

9.7 Lock control

Set the lock bit by programming bit 16 in the DMACCxConfiguration Register.

When a burst occurs, the AHB arbiter must not de-grant the master during the burst until the lock is deasserted. The GPDMA can be locked for a a single burst such as a long source fetch burst or a long destination drain burst. The GPDMA does not usually assert the lock continuously for a source fetch burst followed by a destination drain burst.

There are situations when the GPDMA asserts the lock for source transfers followed by destination transfers. This is possible when internal conditions in the GPDMA permit it to perform a source fetch followed by a destination drain back-to-back.

9.8 Flow control and transfer type

Table 30–570 lists the bit values of the three flow control and transfer type bits.

Table 570. 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.

10. Address generation

Address generation can be either incrementing or non-incrementing (address wrapping is not supported). Bursts do not cross the 1 kB address boundary.

11. 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 GPDMA.

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).

11.1 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 DMACCxConfiguration DMA channel Configuration Register is not part of the linked list item.

11.2 Programming the GPDMA for scatter/gather DMA

To program the GPDMA 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. The LLIs must be stored in the memory where the GPDMA has access to (i.e. AHB1 SRAM and external memory).
2. Choose a free DMA channel with the priority required. DMA channel 0 has the highest priority and DMA channel 1 the lowest priority.
3. Write the first linked list item, previously written to memory, to the relevant channel in the GPDMA.
4. Write the channel configuration information to the channel Configuration Register and set the Channel Enable bit. The GPDMA 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.

11.3 Example of scatter/gather DMA

See Figure 30–126 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.

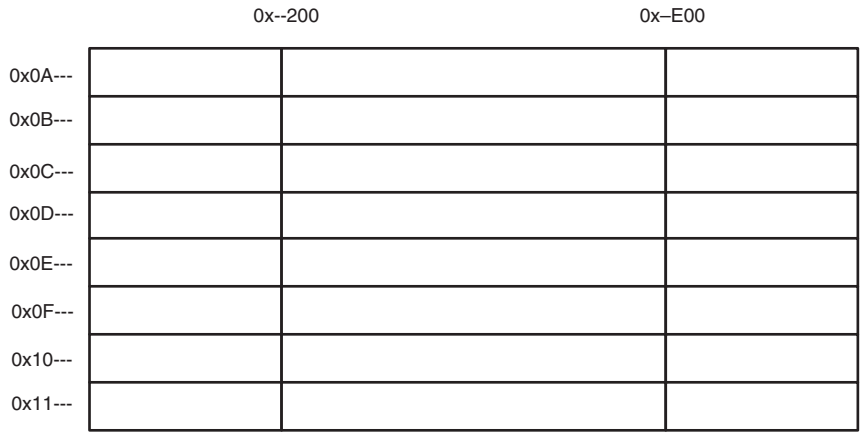


Fig 126.LLI example

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, 3 072 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, 3 072 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 GPDMA. 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, 3 072 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.

12. 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 DMACCxConfiguration Channel Registers. Interrupt status registers are provided which group the interrupt requests from all the DMA channels prior to interrupt masking (DMACRawIntTCStatus and DMACRawIntErrorStatus), and after interrupt masking (DMACIntTCStatus and DMACIntErrorStatus). The DMACIntStatus Register combines both the DMACIntTCStatus and DMACIntErrorStatus 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.

12.1 Hardware interrupt sequence flow

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

1. Read the DMACIntStatus Register to determine which channel generated the interrupt. If more than one request is active it is recommended that the highest priority channels be checked first.
2. Read the DMACIntTCStatus 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.
3. Read the DMACIntErrorStatus Register to determine whether the interrupt was generated due to an error occurring. A HIGH bit indicates that an error occurred.
4. Service the interrupt request.
5. 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.

12.2 Interrupt polling sequence flow

Used when the GPDMA interrupt request signal is either masked out, disabled in the interrupt controller or disabled in the processor. When polling the GPDMA, you must:

1. Read the DMACIntStatus Register. If none of the bits are HIGH repeat this step, otherwise go to step 2. If more than one request is active it is recommended that the highest priority channels be checked first.
2. Read the DMACIntTCStatus 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.
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.

13. GPDMA data flow

This section describes the GPDMA 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 GPDMA as the flow controller so there are eight possible control scenarios.

Table 30–571 indicates the request signals used for each type of transfer.

Table 571. DMA request signal usage

Transfer Direction	Request Generator	Flow Controller
Memory-to-peripheral	Peripheral	GPDMA
Memory-to-peripheral	Peripheral	Peripheral
Peripheral-to-memory	Peripheral	GPDMA
Peripheral-to-memory	Peripheral	Peripheral
Memory-to-memory	GPDMA	GPDMA
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	GPDMA

13.1 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 GPDMA starts transferring data when:
   - The DMA request goes active.
   - The DMA stream has the highest pending priority.
   - The GPDMA 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 GPDMA is performing the flow control.

6. If the transfer has completed (indicated by the transfer count reaching 0 if the GPDMA is performing flow control, or by the peripheral sending a DMA request if the peripheral is performing flow control):
   - The GPDMA 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 back to step 2. However, if DMACCxLLI is 0, the DMA stream is disabled and the flow sequence ends.

13.2 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 GPDMA starts transferring data when:
   - The DMA request goes active.
   - The DMA stream has the highest pending priority.
   - The GPDMA is the bus master of the AHB bus.
4. If an error occurs while transferring the data an error interrupt is generated, then finishes.
5. Decrement the transfer count if the GPDMA is performing the flow control.
6. If the transfer has completed (indicated by the transfer count reaching 0 if the GPDMA is performing flow control, or by the peripheral sending a DMA request if the peripheral is performing flow control):
   - The GPDMA 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 GPDMA FIFO, transfer data into the destination peripheral.
8. If an error occurs while transferring the data, an error interrupt is generated and disables the DMA stream, and the flow sequence ends.
9. If the transfer has completed it is indicated by the transfer count reaching 0 if the GPDMA is performing flow control, or by sending a DMA request if the peripheral is performing flow control. The following happens:
   - The GPDMA 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.

13.3 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 GPDMA 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.

14. Flow control

The peripheral that controls the length of the packet is known as the flow controller. The flow controller is usually the GPDMA 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 is transferred:

1. The GPDMA 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 GPDMA moves on to the next LLI.

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 ARM7TDMI-S core.
- The ARM7TDMI-S core or the System state can be examined, saved or changed depending on the type of instruction inserted.
- Allows instructions to execute at a slow debug speed or at a fast system speed.

2. Applications

The 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 convertor. EmbeddedICE protocol convertor converts the Remote Debug Protocol commands to the JTAG data needed to access the ARM7TDMI-S core present on the target system.

3. Description

The ARM7TDMI-S Debug Architecture uses the existing JTAG[3] 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 ARM7TDMI-S. A JTAG-style Test Access Port Controller controls the scan chains. In addition to the scan chains, the debug architecture uses EmbeddedICE logic which resides on chip with the ARM7TDMI-S core. The EmbeddedICE has its own scan chain that is used to insert watchpoints and breakpoints for the ARM7TDMI-S core. The EmbeddedICE 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 ARM7TDMI-S core. Execution is halted when a match occurs between the values programmed into the EmbeddedICE 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 ARM7TDMI 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

3.For more details refer to IEEE Standard 1149.1 - 1990 Standard Test Access Port and Boundary Scan Architecture.

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.

- 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 ARM7TDMI-S core has a Debug Communication Channel function in-built. 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 ARM7TDMI-S 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 logic.

- For more details refer to *IEEE Standard 1149.1 - 1990 Standard Test Access Port and Boundary Scan Architecture*.

4. Pin description

Table 572. EmbeddedICE 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 edgetriggered 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.
nTRST	Input	**Test Reset.** The nTRST pin can be used to reset the test logic within the EmbeddedICE logic.
RTCK	Output	**Returned Test Clock.** Extra signal added to the JTAG port. Required for designs based on ARM7TDMI-S processor core. Multi-ICE (Development system from ARM) uses this signal to maintain synchronization with targets having slow or widely varying clock frequency. For details refer to *"Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)"*. Board designers may need to connect a weak bias resistor to this pin as described below.

5. JTAG function select

The JTAG port may be used either for debug or for boundary scan. The state of the DBGEN pin determines which function is available. When DBGEN = 0, the JTAG port may be used for boundary scan. When DBGEN = 1, the JTAG port may be used for debug.

6. Register description

The EmbeddedICE logic contains 16 registers as shown in Table 31–573 below. The ARM7TDMI-S debug architecture is described in detail in "ARM7TDMI-S (rev 4) Technical Reference Manual" (ARM DDI 0234A) published by ARM Limited.

Table 573. EmbeddedICE logic registers

Name	Width	Description	Address
Debug Control	6	Force debug state, disable interrupts	00000
Debug Status	5	Status of debug	00001
Debug Comms Control Register	32	Debug communication control register	00100
Debug Comms Data Register	32	Debug communication data register	00101
Watchpoint 0 Address Value	32	Holds watchpoint 0 address value	01000
Watchpoint 0 Address Mask	32	Holds watchpoint 0 address mask	01001
Watchpoint 0 Data Value	32	Holds watchpoint 0 data value	01010
Watchpoint 0 Data Mask	32	Holds watchpoint 0 data mask	01011
Watchpoint 0 Control Value	9	Holds watchpoint 0 control value	01100
Watchpoint 0 Control Mask	8	Holds watchpoint 0 control mask	01101
Watchpoint 1 Address Value	32	Holds watchpoint 1 address value	10000
Watchpoint 1 Address Mask	32	Holds watchpoint 1 address mask	10001
Watchpoint 1 Data Value	32	Holds watchpoint 1 data value	10010
Watchpoint 1 Data Mask	32	Holds watchpoint 1 data mask	10011
Watchpoint 1 Control Value	9	Holds watchpoint 1 control value	10100
Watchpoint 1 Control Mask	8	Holds watchpoint 1 control mask	10101

7. Block diagram

The block diagram of the debug environment is shown below in Figure 31–127.

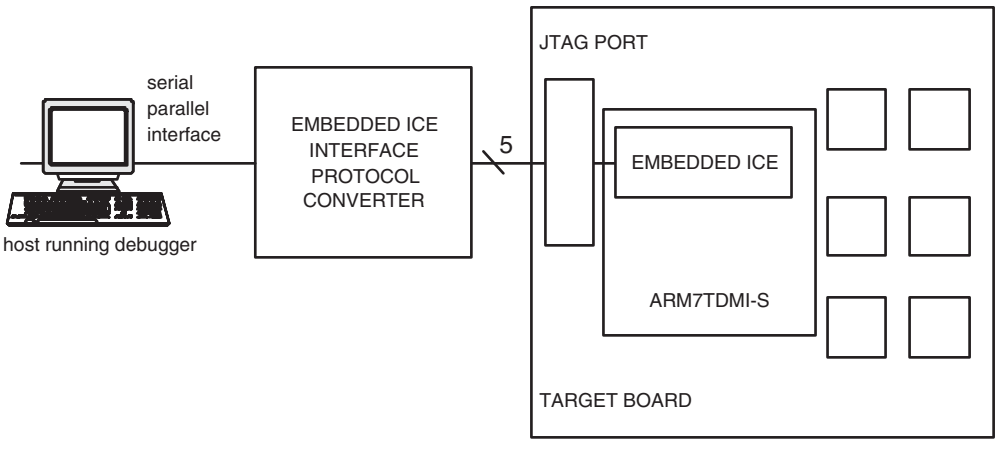


Fig 127.EmbeddedICE debug environment block diagram

1. Features

- Closely track the instructions that the ARM core is executing.
- One external trigger input.
- 10 pin interface.
- All registers are programmed through JTAG interface.
- Does not consume power when trace is not being used.
- THUMB instruction set support.

2. Applications

As the microcontroller has significant amounts of on-chip memories, 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.

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 external Trace Port Analyzer captures the trace information under software debugger control. Trace port can broadcast the Instruction trace information. 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. Trace information generation can be controlled by selecting the trigger resource. Trigger resources include address 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.

3.1 ETM configuration

The following standard configuration is selected for the ETM macrocell.

Table 574. ETM configuration

Resource number/type	Small[1]
Pairs of address comparators	1
Data Comparators	0 (Data tracing is not supported)
Memory Map Decoders	4
Counters	1
Sequencer Present	No

Table 574. ETM configuration

Resource number/type	Small[1]
External Inputs	2
External Outputs	0
FIFOFULL Present	Yes (Not wired)
FIFO depth	10 bytes
Trace Packet Width	4/8

[1] For details refer to ARM documentation "Embedded Trace Macrocell Specification (ARM IHI 0014E)".

4. Pin description

Table 575. ETM pin description

Pin Name	Type	Description
TRACECLK	Output	**Trace Clock.** The trace clock signal provides the clock for the trace port. PIPESTAT[2:0], TRACESYNC, and TRACEPKT[3:0] signals are referenced to the rising edge of the trace clock. This clock is not generated by the ETM block. It is to be derived from the system clock. The clock should be balanced to provide sufficient hold time for the trace data signals. Half rate clocking mode is supported. Trace data signals should be shifted by a clock phase from TRACECLK. Refer to Figure 3.14 page 3.26 and figure 3.15 page 3.27 in *"ETM7 Technical Reference Manual" (ARM DDI 0158B)*, for example circuits that implements both half-rateclocking and shifting of the trace data with respect to the clock. For TRACECLK timings refer to section 5.2 on page 5-13 in *"Embedded Trace Macrocell Specification" (ARM IHI 0014E)*.
PIPESTAT[2:0]	Output	**Pipe Line status.** The pipeline status signals provide a cycle-by-cycle indication of what is happening in the execution stage of the processor pipeline.
TRACESYNC	Output	**Trace synchronization.** The trace sync signal is used to indicate the first packet of a group of trace packets and is asserted HIGH only for the first packet of any branch address.
TRACEPKT[3:0]	Output	**Trace Packet.** The trace packet signals are used to output packaged address and data information related to the pipeline status. All packets are eight bits in length. A packet is output over two cycles. In the first cycle, Packet[3:0] is output and in the second cycle, Packet[7:4] is output.
EXTIN[0]	Input	**External Trigger Input**

5. Register description

The ETM contains 29 registers as shown in Table 32–576 below. They are described in detail in the *ARM IHI 0014E document published by ARM Limited*.

Table 576. ETM Registers

Name	Description	Access	Register Encoding
ETM Control	Controls the general operation of the ETM.	R/W	000 0000
ETM Configuration Code	Allows a debugger to read the number of each type of resource.	RO	000 0001
Trigger Event	Holds the controlling event.	WO	000 0010
Memory Map Decode Control	Eight bit register, used to statically configure the memory map decoder.	WO	000 0011
ETM Status	Holds the pending overflow status bit.	RO	000 0100
System Configuration	Holds the configuration information using the SYSOPT bus.	RO	000 0101
Trace Enable Control 3	Holds the trace on/off addresses.	WO	000 0110
Trace Enable Control 2	Holds the address of the comparison.	WO	000 0111
Trace Enable Event	Holds the enabling event.	WO	000 1000
Trace Enable Control 1	Holds the include and exclude regions.	WO	000 1001
FIFOFULL Region	Holds the include and exclude regions.	WO	000 1010
FIFOFULL Level	Holds the level below which the FIFO is considered full.	WO	000 1011
ViewData event	Holds the enabling event.	WO	000 1100
ViewData Control 1	Holds the include/exclude regions.	WO	000 1101
ViewData Control 2	Holds the include/exclude regions.	WO	000 1110
ViewData Control 3	Holds the include/exclude regions.	WO	000 1111
Address Comparator 1 to 16	Holds the address of the comparison.	WO	001 xxxx
Address Access Type 1 to 16	Holds the type of access and the size.	WO	010 xxxx
Reserved	-	-	000 xxxx
Reserved	-	-	100 xxxx
Initial Counter Value 1 to 4	Holds the initial value of the counter.	WO	101 00xx
Counter Enable 1 to 4	Holds the counter clock enable control and event.	WO	101 01xx
Counter reload 1 to 4	Holds the counter reload event.	WO	101 10xx
Counter Value 1 to 4	Holds the current counter value.	RO	101 11xx
Sequencer State and Control	Holds the next state triggering events.	-	110 00xx
External Output 1 to 4	Holds the controlling events for each output.	WO	110 10xx
Reserved	-	-	110 11xx
Reserved	-	-	111 0xxx
Reserved	-	-	111 1xxx

6. Reset state of multiplexed pins

On the LPC2400, the ETM pin functions are multiplexed with GPIO, PWM, UART, and CAN functions. In order to use the trace feature, the pins must be configured to select the function. Details may be found in "Pin Connect" chapter on page 119.

7. Block diagram

The block diagram of the ETM debug environment is shown below in Figure 32–128.

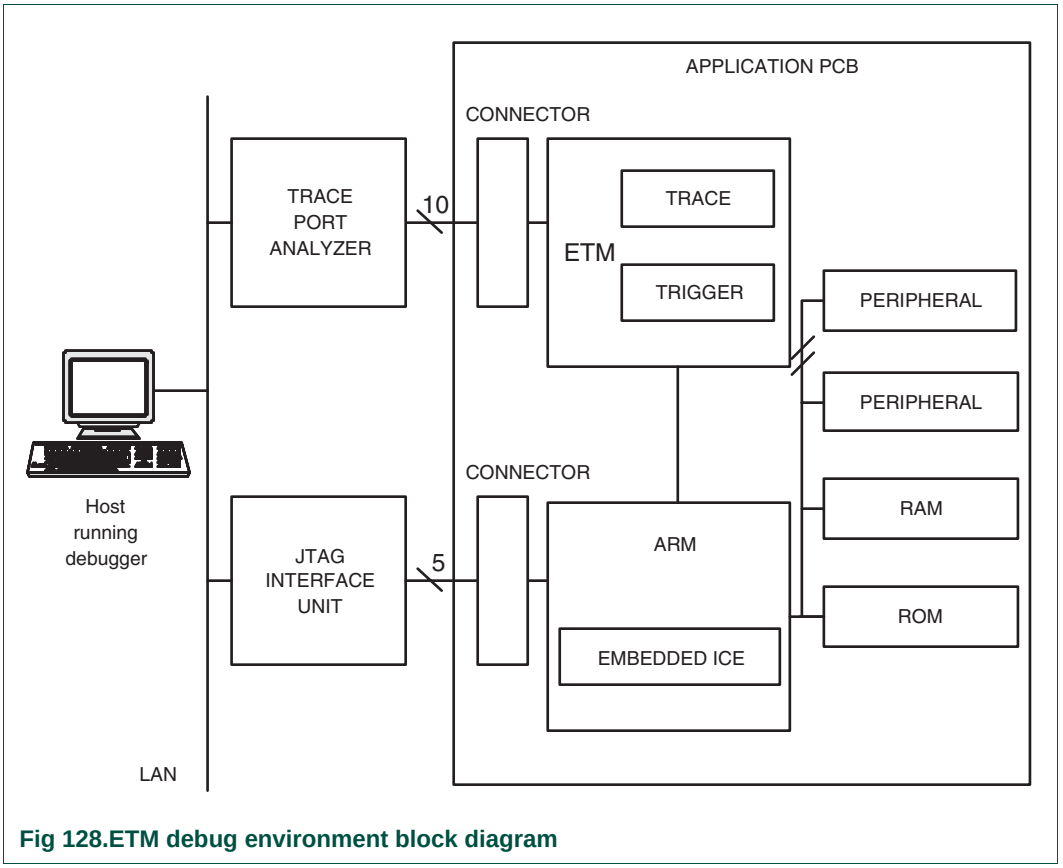


Fig 128.ETM debug environment block diagram

1. Features

Remark: RealMonitor is a configurable software module which enables real time debug. RealMonitor is developed by ARM Inc. Information presented in this chapter is taken from the ARM document RealMonitor Target Integration Guide (ARM DUI 0142A). It applies to a specific configuration of RealMonitor software programmed in the on-chip ROM boot memory of this device.

- Allows user to establish a debug session to a currently running system without halting or resetting the system.
- Allows user time-critical interrupt code to continue executing while other user application code is being debugged.

2. Applications

Real time debugging.

3. Description

RealMonitor is a lightweight debug monitor that allows interrupts to be serviced while user debug their foreground application. It communicates with the host using the DCC (Debug Communications Channel), which is present in the EmbeddedICE logic. RealMonitor provides advantages over the traditional methods for debugging applications in ARM systems. The traditional methods include:

- Angel (a target-based debug monitor).
- Multi-ICE or other JTAG unit and EmbeddedICE logic (a hardware-based debug solution).

Although both of these methods provide robust debugging environments, neither is suitable as a lightweight real-time monitor.

Angel is designed to load and debug independent applications that can run in a variety of modes, and communicate with the debug host using a variety of connections (such as a serial port or ethernet). Angel is required to save and restore full processor context, and the occurrence of interrupts can be delayed as a result. Angel, as a fully functional target-based debugger, is therefore too heavyweight to perform as a real-time monitor.

Multi-ICE is a hardware debug solution that operates using the EmbeddedICE unit that is built into most ARM processors. To perform debug tasks such as accessing memory or the processor registers, Multi-ICE must place the core into a debug state. While the processor is in this state, which can be millions of cycles, normal program execution is suspended, and interrupts cannot be serviced.

RealMonitor combines features and mechanisms from both Angel and Multi-ICE to provide the services and functions that are required. In particular, it contains both the Multi-ICE communication mechanisms (the DCC using JTAG), and Angel-like support for

processor context saving and restoring. RealMonitor is pre-programmed in the on-chip ROM memory (boot sector). When enabled It allows user to observe and debug while parts of application continue to run. Refer to Section 33–4 “How to enable RealMonitor” on page 621 for details.

3.1 RealMonitor components

As shown in Figure 33–129, RealMonitor is split in to two functional components:

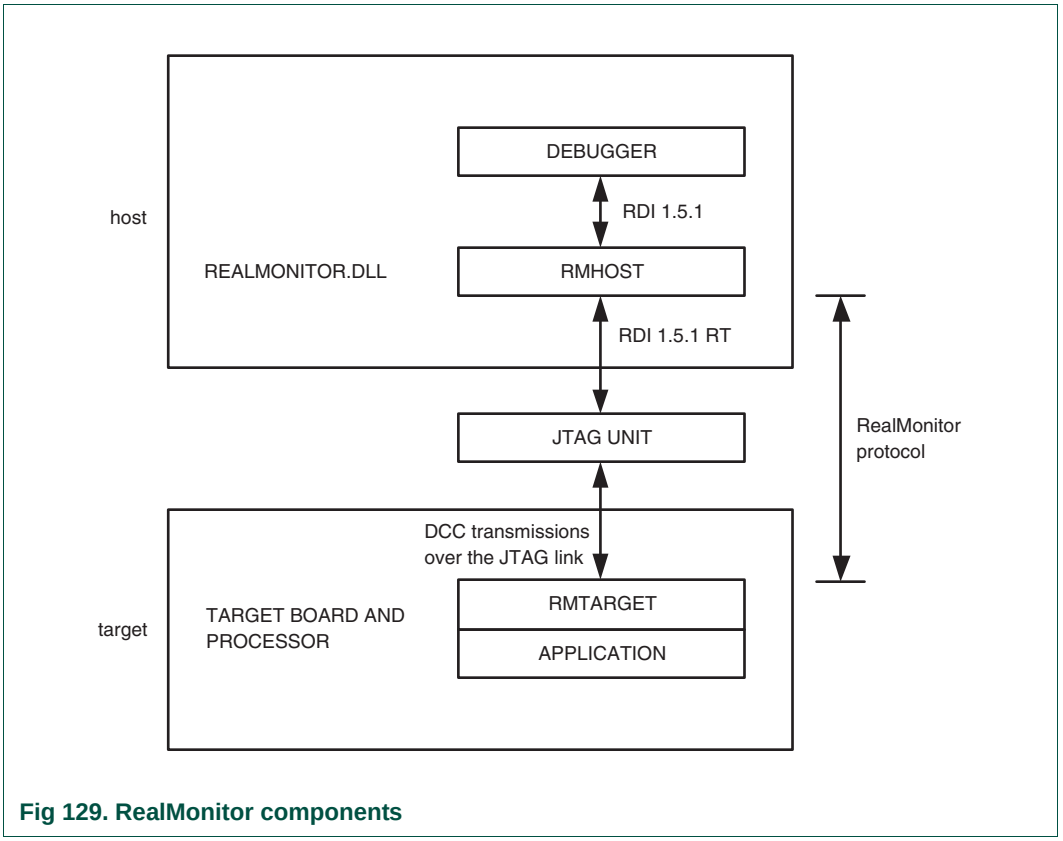


Fig 129. RealMonitor components

3.1.1 RMHost

This is located between a debugger and a JTAG unit. The RMHost controller, RealMonitor.dll, converts generic Remote Debug Interface (RDI) requests from the debugger into DCC-only RDI messages for the JTAG unit. For complete details on debugging a RealMonitor-integrated application from the host, see the ARM RMHost User Guide (ARM DUI 0137A).

3.1.2 RMTarget

This is pre-programmed in the on-chip ROM memory (boot sector), and runs on the target hardware. It uses the EmbeddedICE logic, and communicates with the host using the DCC. For more details on RMTarget functionality, see the RealMonitor Target Integration Guide (ARM DUI 0142A).

3.2 How RealMonitor works

In general terms, the RealMonitor operates as a state machine, as shown in Figure 33–130. RealMonitor switches between running and stopped states, in response to packets received by the host, or due to asynchronous events on the target. RMTarget supports the triggering of only one breakpoint, watchpoint, stop, or semihosting SWI at a time. There is no provision to allow nested events to be saved and restored. So, for example, if user application has stopped at one breakpoint, and another breakpoint occurs in an IRQ handler, RealMonitor enters a panic state. No debugging can be performed after RealMonitor enters this state.

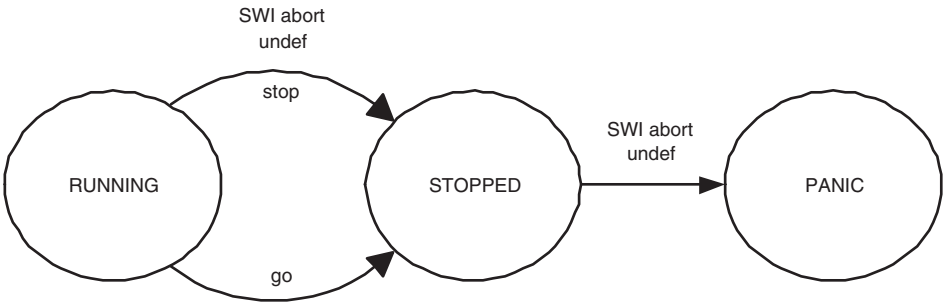


Fig 130.RealMonitor as a State Machine

A debugger such as the ARM eXtended Debugger (AXD) or other RealMonitor aware debugger, that runs on a host computer, can connect to the target to send commands and receive data. This communication between host and target is illustrated in Figure 33–129.

The target component of RealMonitor, RMTarget, communicates with the host component, RMHost, using the Debug Communications Channel (DCC), which is a reliable link whose data is carried over the JTAG connection.

While user application is running, RMTarget typically uses IRQs generated by the DCC. This means that if user application also wants to use IRQs, it must pass any DCC-generated interrupts to RealMonitor.

To allow nonstop debugging, the EmbeddedICE-RT logic in the processor generates a Prefetch Abort exception when a breakpoint is reached, or a Data Abort exception when a watchpoint is hit. These exceptions are handled by the RealMonitor exception handlers that inform the user, by way of the debugger, of the event. This allows user application to continue running without stopping the processor. RealMonitor considers user application to consist of two parts:

- A foreground application running continuously, typically in User, System, or SVC mode
- A background application containing interrupt and exception handlers that are triggered by certain events in user system, including:
  - IRQs or FIQs
  - Data and Prefetch aborts caused by user foreground application. This indicates an error in the application being debugged. In both cases the host is notified and the user application is stopped.

– Undef exception caused by the undefined instructions in user foreground application. This indicates an error in the application being debugged. RealMonitor stops the user application until a "Go" packet is received from the host.

When one of these exceptions occur that is not handled by user application, the following happens:

- RealMonitor enters a loop, polling the DCC. If the DCC read buffer is full, control is passed to rm_ReceiveData() (RealMonitor internal function). If the DCC write buffer is free, control is passed to rm_TransmitData() (RealMonitor internal function). If there is nothing else to do, the function returns to the caller. The ordering of the above comparisons gives reads from the DCC a higher priority than writes to the communications link.
- RealMonitor stops the foreground application. Both IRQs and FIQs continue to be serviced if they were enabled by the application at the time the foreground application was stopped.

4. How to enable RealMonitor

The following steps must be performed to enable RealMonitor. A code example which implements all the steps can be found at the end of this section.

4.1 Adding stacks

User must ensure that stacks are set up within application for each of the processor modes used by RealMonitor. For each mode, RealMonitor requires a fixed number of words of stack space. User must therefore allow sufficient stack space for both RealMonitor and application.

RealMonitor has the following stack requirements:

Table 577. RealMonitor stack requirement

Processor mode	RealMonitor Stack Usage (Bytes)
Undef	48
Prefetch Abort	16
Data Abort	16
IRQ	8

4.2 IRQ mode

A stack for this mode is always required. RealMonitor uses two words on entry to its interrupt handler. These are freed before nested interrupts are enabled.

4.3 Undef mode

A stack for this mode is always required. RealMonitor uses 12 words while processing an undefined instruction exception.

4.4 SVC mode

RealMonitor makes no use of this stack.

4.5 Prefetch Abort mode

RealMonitor uses four words on entry to its Prefetch abort interrupt handler.

4.6 Data Abort mode

RealMonitor uses four words on entry to its data abort interrupt handler.

4.7 User/System mode

RealMonitor makes no use of this stack.

4.8 FIQ mode

RealMonitor makes no use of this stack.

4.9 Handling exceptions

This section describes the importance of sharing exception handlers between RealMonitor and user application.

4.9.1 RealMonitor exception handling

To function properly, RealMonitor must be able to intercept certain interrupts and exceptions. Figure 33–131 illustrates how exceptions can be claimed by RealMonitor itself, or shared between RealMonitor and application. If user application requires the exception sharing, they must provide function (such as app_IRQDispatch ()). Depending on the nature of the exception, this handler can either:

- Pass control to the RealMonitor processing routine, such as rm_irqhandler2().
- Claim the exception for the application itself, such as app_IRQHandler ().

In a simple case where an application has no exception handlers of its own, the application can install the RealMonitor low-level exception handlers directly into the vector table of the processor. Although the irq handler must get the address of the Vectored Interrupt Controller. The easiest way to do this is to write a branch instruction (<address>) into the vector table, where the target of the branch is the start address of the relevant RealMonitor exception handler.

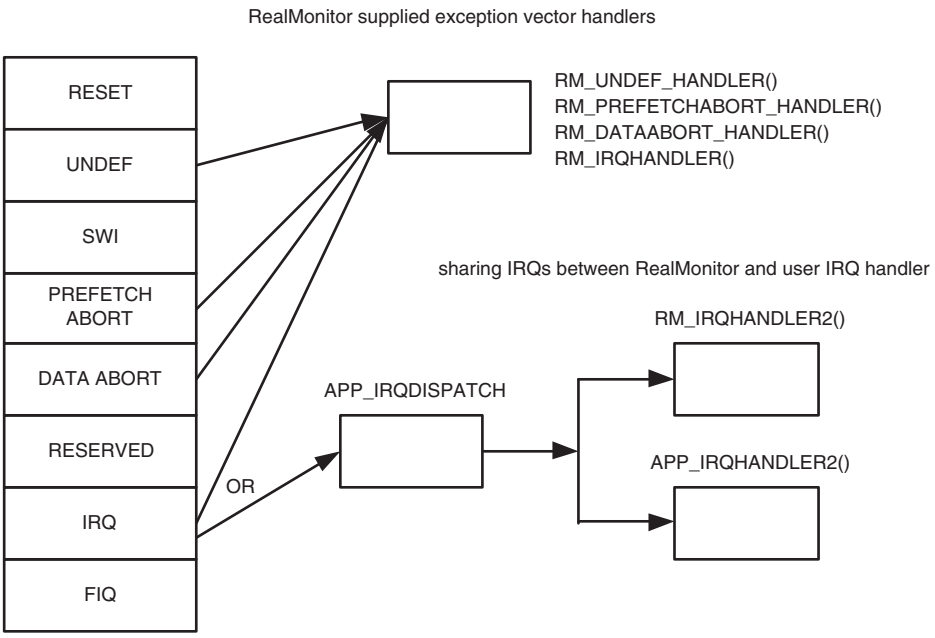


Fig 131. Exception handlers

4.10 RMTarget initialization

While the processor is in a privileged mode, and IRQs are disabled, user must include a line of code within the start-up sequence of application to call rm_init_entry().

4.11 Code example

The following example shows how to setup stack, VIC, initialize RealMonitor and share non vectored interrupts:

```
IMPORT rm_init_entry
IMPORT rm_prefetchabort_handler
IMPORT rm_dataabort_handler
IMPORT rm_irqhandler2
IMPORT rm_undef_handler
IMPORT User_Entry ;Entry point of user application.
CODE32
ENTRY
;Define exception table. Instruct linker to place code at address 0x0000 0000

AREA exception_table, CODE

LDR pc, Reset_Address
LDR pc, Undefined_Address
LDR pc, SWI_Address
LDR pc, Prefetch_Address
LDR pc, Abort_Address
```

```
     NOP ; Insert User code valid signature here.
     LDR pc, [pc, #-0x120] ;Load IRQ vector from VIC
     LDR PC, FIQ_Address

Reset_Address         DCD __init            ;Reset Entry point
Undefined_Address     DCD rm_undef_handler ;Provided by RealMonitor
SWI_Address           DCD 0                 ;User can put address of SWI handler here
Prefetch_Address      DCD rm_prefetchabort_handler    ;Provided by RealMonitor
Abort_Address         DCD rm_dataabort_handler        ;Provided by RealMonitor
FIQ_Address           DCD 0                 ;User can put address of FIQ handler here

AREA init_code, CODE

ram_end EQU 0x4000xxxx ; Top of on-chip RAM.
__init
; /*********************************************************************
; * Set up the stack pointers for various processor modes. Stack grows
; * downwards.
; *********************************************************************/
     LDR  r2, =ram_end ;Get top of RAM
     MRS r0, CPSR ;Save current processor mode

     ; Initialize the Undef mode stack for RealMonitor use
     BIC  r1, r0, #0x1f
     ORR  r1, r1, #0x1b
     MSR  CPSR_c, r1
     ;Keep top 32 bytes for flash programming routines.
     ;Refer to Flash Memory System and Programming chapter
     SUB  sp,r2,#0x1F

     ; Initialize the Abort mode stack for RealMonitor
     BIC  r1, r0, #0x1f
     ORR  r1, r1, #0x17
     MSR  CPSR_c, r1
     ;Keep 64 bytes for Undef mode stack
     SUB  sp,r2,#0x5F

     ; Initialize the IRQ mode stack for RealMonitor and User
     BIC  r1, r0, #0x1f
     ORR  r1, r1, #0x12
     MSR  CPSR_c, r1
     ;Keep 32 bytes for Abort mode stack
     SUB  sp,r2,#0x7F

     ; Return to the original mode.
     MSR  CPSR_c, r0

     ; Initialize the stack for user application
     ; Keep 256 bytes for IRQ mode stack
     SUB  sp,r2,#0x17F
```

```
; /*********************************************************************
; * Setup Vectored Interrupt controller. DCC Rx and Tx interrupts
; * generate Non Vectored IRQ request. rm_init_entry is aware
; * of the VIC and it enables the DBGCommRX and DBGCommTx interrupts.
; * Default vector address register is programmed with the address of
; * Non vectored app_irqDispatch mentioned in this example. User can setup
; * Vectored IRQs or FIQs here.
; *********************************************************************/

     VICBaseAddr          EQU 0xFFFFF000 ; VIC Base address
     VICDefVectAddrOffset  EQU 0x34

     LDR  r0, =VICBaseAddr
     LDR  r1, =app_irqDispatch
     STR  r1, [r0,#VICDefVectAddrOffset]

     BL   rm_init_entry   ;Initialize RealMonitor
     ;enable FIQ and IRQ in ARM Processor
     MRS  r1, CPSR        ; get the CPSR
     BIC  r1, r1, #0xC0   ; enable IRQs and FIQs
     MSR  CPSR_c, r1      ; update the CPSR
; /*********************************************************************
; * Get the address of the User entry point.
; *********************************************************************/
     LDR  lr, =User_Entry
     MOV  pc, lr
; /*********************************************************************
; * Non vectored irq handler (app_irqDispatch)
; *********************************************************************/

AREA app_irqDispatch, CODE
VICVectAddrOffset EQU 0x30
app_irqDispatch

     ;enable interrupt nesting
     STMFD sp!, {r12,r14}
     MRS  r12, spsr            ;Save SPSR in to r12
     MSR  cpsr_c,0x1F          ;Re-enable IRQ, go to system mode

;User should insert code here if non vectored Interrupt sharing is
;required. Each non vectored shared irq handler must return to
;the interrupted instruction by using the following code.
;    MSR  cpsr_c, #0x52                 ;Disable irq, move to IRQ mode
;    MSR  spsr, r12                     ;Restore SPSR from r12
;    STMFD sp!, {r0}
;    LDR  r0, =VICBaseAddr
;    STR  r1, [r0,#VICVectAddrOffset]   ;Acknowledge Non Vectored irq has finished
;    LDMFD sp!, {r12,r14,r0}            ;Restore registers
;    SUBS pc, r14, #4                   ;Return to the interrupted instruction

     ;user interrupt did not happen so call rm_irqhandler2. This handler
```

```
;is not aware of the VIC interrupt priority hardware so trick
;rm_irqhandler2 to return here

STMFD sp!, {ip,pc}
LDR  pc, rm_irqhandler2
;rm_irqhandler2 returns here
MSR  cpsr_c, #0x52                  ;Disable irq, move to IRQ mode
MSR  spsr, r12                      ;Restore SPSR from r12
STMFD sp!, {r0}
LDR  r0, =VICBaseAddr
STR  r1, [r0,#VICVectAddrOffset]    ;Acknowledge Non Vectored irq has finished
LDMFD sp!, {r12,r14,r0}             ;Restore registers
SUBS  pc, r14, #4                   ;Return to the interrupted instruction

END
```

5. RealMonitor build options

RealMonitor was built with the following options:

RM_OPT_DATALOGGING=FALSE

This option enables or disables support for any target-to-host packets sent on a non RealMonitor (third-party) channel.

RM_OPT_STOPSTART=TRUE

This option enables or disables support for all stop and start debugging features.

RM_OPT_SOFTBREAKPOINT=TRUE

This option enables or disables support for software breakpoints.

RM_OPT_HARDBREAKPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_HARDWATCHPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_SEMIHOSTING=FALSE

This option enables or disables support for SWI semi-hosting. Semi-hosting provides code running on an ARM target use of facilities on a host computer that is running an ARM debugger. Examples of such facilities include the keyboard input, screen output, and disk I/O.

RM_OPT_SAVE_FIQ_REGISTERS=TRUE

This option determines whether the FIQ-mode registers are saved into the registers block when RealMonitor stops.

RM_OPT_READBYTES=TRUE

RM_OPT_WRITEBYTES=TRUE

RM_OPT_READHALFWORDS=TRUE

RM_OPT_WRITEHALFWORDS=TRUE

RM_OPT_READWORDS=TRUE

RM_OPT_WRITEWORDS=TRUE

Enables/Disables support for 8/16/32 bit read/write.

RM_OPT_EXECUTECODE=FALSE

Enables/Disables support for executing code from "execute code" buffer. The code must be downloaded first.

RM_OPT_GETPC=TRUE

This option enables or disables support for the RealMonitor GetPC packet. Useful in code profiling when real monitor is used in interrupt mode.

RM_EXECUTECODE_SIZE=NA

"execute code" buffer size. Also refer to RM_OPT_EXECUTECODE option.

RM_OPT_GATHER_STATISTICS=FALSE

This option enables or disables the code for gathering statistics about the internal operation of RealMonitor.

RM_DEBUG=FALSE

This option enables or disables additional debugging and error-checking code in RealMonitor.

RM_OPT_BUILDIDENTIFIER=FALSE

This option determines whether a build identifier is built into the capabilities table of RMTarget. Capabilities table is stored in ROM.

RM_OPT_SDM_INFO=FALSE

SDM gives additional information about application board and processor to debug tools.

RM_OPT_MEMORYMAP=FALSE

This option determines whether a memory map of the board is built into the target and made available through the capabilities table

RM_OPT_USE_INTERRUPTS=TRUE

This option specifies whether RMTarget is built for interrupt-driven mode or polled mode.

RM_FIFOSIZE=NA

This option specifies the size, in words, of the data logging FIFO buffer.

CHAIN_VECTORS=FALSE

This option allows RMTarget to support vector chaining through µHAL (ARM HW abstraction API).

1. Abbreviations

Table 578. Acronyms and abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
BLS	Byte Lane Select
BOD	BrownOut Detection
CAN	Controller Area Network
DAC	Digital-to-Analog Converter
DCC	Debug Communication Channel
DMA	Direct Memory Access
DSP	Digital Signal Processing
EOP	End Of Packet
ETM	Embedded Trace Macrocell
GPIO	General Purpose Input/Output
JTAG	Joint Test Action Group
MII	Media Independent Interface
PHY	Physical Layer
PLL	Phase-Locked Loop
PWM	Pulse Width Modulator
RMII	Reduced Media Independent Interface
SE0	Single Ended Zero
SPI	Serial Peripheral Interface
SSI	Synchronous Serial Interface
SSP	Synchronous Serial Port
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. References

Optional section for document references. The bold reference title is optional.

[1] <reference title> — <reference description>

3. Legal information

3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

3.2 Disclaimers

General — 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.

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 medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a 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.

3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^2C-bus — logo is a trademark of NXP B.V.

SoftConnect — is a trademark of NXP B.V.

GoodLink — is a trademark of NXP B.V.

Notes

4. Tables

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

5. Figures

continued >>

continued >>

6. Contents

Chapter 1: Introductory information

Chapter 2: LPC2468 memory addressing

Chapter 3: System control

Chapter 4: Clocking and power control

continued >>

Chapter 5: External Memory Controller (EMC)

continued >>

Chapter 6: Memory Accelerator Module (MAM)

Chapter 7: Vectored Interrupt Controller (VIC)

Chapter 8: Pin configuration

Chapter 9: Pin connect block

continued >>

Chapter 10: General Purpose Input/Output ports (GPIO)

continued >>

Chapter 11: Ethernet

continued >>

Chapter 12: CAN controllers CAN1/2

continued >>

continued >>

Chapter 13: USB device controller

continued >>

continued >>

Chapter 14: USB host controller

Chapter 15: USB OTG controller

continued >>

Chapter 16: Universal Asynchronous Receiver Transmitter (UART) 0/2/3

Chapter 17: Universal Asynchronous Receiver Transmitter (UART) 1

continued >>

Chapter 18: SPI interface

Chapter 19: SSP0/1 Interface

Chapter 20: SD/MMC card interface

continued >>

Chapter 21: I²C interfaces I²C0, I²C1, I²C2

continued >>

Chapter 22: I2S Interface

Chapter 23: Timer0/1/2/3

continued >>

Chapter 24: WatchDog Timer (WDT)

Chapter 25: Pulse Width Modulator PWM0 and PWM1

Chapter 26: Analog-To-Digital Converter (ADC)

Chapter 27: Digital-To-Analog Converter (DAC)

continued >>

Chapter 28: Real Time Clock (RTC) battery RAM

Chapter 29: Flash memory programming firmware

continued >>

Chapter 30: General Purpose DMA controller (GPDMA)

continued >>

Chapter 31: EmbeddedICE

Chapter 32: EmbeddedTrace Module (ETM)

Chapter 33: RealMonitor

Chapter 34: Supplementary information

continued >>

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 18 December 2006
Document identifier: UM10237_1