



Document information

| Info | Content |
|-----------------|--|
| Keywords | LPC2141, LPC2142, LPC2144, LPC2146, LPC2148, LPC2000, LPC214x, ARM, ARM7, embedded, 32-bit, microcontroller, USB 2.0, USB device |
| Abstract | An initial LPC214x User Manual revision |

Revision history

| Rev | Date | Description |
|-----|----------|-----------------|
| 01 | 20050815 | Initial version |

Contact information

For additional information, please visit: <http://www.semiconductors.philips.com>

For sales office addresses, please send an email to: sales.addresses@www.semiconductors.philips.com

1.1 Introduction

The LPC2141/2/4/6/8 microcontrollers are based on a 32/16 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, LPC2141/2/4/6/8 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTS, SPI, SSP to I²Cs and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.

1.2 Features

- 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory.
128 bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software.
Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.
- EmbeddedICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip RealMonitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM.
In addition, the LPC2146/8 provide 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14 analog inputs, with conversion times as low as 2.44 µs per channel.
- Single 10-bit D/A converter provides variable analog output.
- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power real-time clock with independent power and dedicated 32 kHz clock input.
- Multiple serial interfaces including two UARTs (16C550), two Fast I²C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.

- 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 µs.
- On-chip integrated oscillator operates with an external crystal in range from 1 MHz to 30 MHz and with an external oscillator up to 50 MHz.
- Power saving modes include Idle and Power-down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).
- Single power supply chip with Power-On Reset (POR) and BOD circuits:
 - CPU operating voltage range of 3.0 V to 3.6 V ($3.3\text{ V} \pm 10\%$) with 5 V tolerant I/O pads.

1.3 Applications

- Industrial control
- Medical systems
- Access control
- Point-of-sale
- Communication gateway
- Embedded soft modem
- General purpose applications

1.4 Device information

Table 1: LPC2141/2/4/6/8 device information

| Device | Number of pins | On-chip SRAM | Endpoint USB RAM | On-chip FLASH | Number of 10-bit ADC channels | Number of 10-bit DAC channels | Note |
|---------|----------------|-----------------------------|------------------|---------------|-------------------------------|-------------------------------|---------------------------------|
| LPC2141 | 64 | 8 kB | 2 kB | 32 kB | 6 | - | - |
| LPC2142 | 64 | 16 kB | 2 kB | 64 kB | 6 | 1 | - |
| LPC2144 | 64 | 16 kB | 2 kB | 128 kB | 14 | 1 | UART1 with full modem interface |
| LPC2146 | 64 | 32 kB + 8 kB ^[1] | 2 kB | 256 kB | 14 | 1 | UART1 with full modem interface |
| LPC2148 | 64 | 32 kB + 8 kB ^[1] | 2 kB | 512 kB | 14 | 1 | UART1 with full modem interface |

[1] While the USB DMA is the primary user of the additional 8 kB RAM, this RAM is also accessible at any time by the CPU as a general purpose RAM for data and code storage.

1.5 Architectural overview

The LPC2141/2/4/6/8 consists of an ARM7TDMI-S CPU with emulation support, the ARM7 Local Bus for interface to on-chip memory controllers, the AMBA Advanced High-performance Bus (AHB) for interface to the interrupt controller, and the VLSI

Peripheral Bus (VPB, a compatible superset of ARM's AMBA Advanced Peripheral Bus) for connection to on-chip peripheral functions. The LPC2141/24/6/8 configures the ARM7TDMI-S processor in little-endian byte order.

AHB peripherals are allocated a 2 megabyte range of addresses at the very top of the 4 gigabyte ARM memory space. Each AHB peripheral is allocated a 16 kB address space within the AHB address space. LPC2141/2/4/6/8 peripheral functions (other than the interrupt controller) are connected to the VPB bus. The AHB to VPB bridge interfaces the VPB bus to the AHB bus. VPB peripherals are also allocated a 2 megabyte range of addresses, beginning at the 3.5 gigabyte address point. Each VPB peripheral is allocated a 16 kB address space within the VPB address space.

The connection of on-chip peripherals to device pins is controlled by a Pin Connect Block (see chapter "Pin Connect Block" on page 75). This must be configured by software to fit specific application requirements for the use of peripheral functions and pins.

1.6 ARM7TDMI-S processor

The ARM7TDMI-S 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 instruction set.
- A 16-bit THUMB instruction set.

The THUMB set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because THUMB code operates on the same 32-bit register set as ARM code.

THUMB code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system.

The ARM7TDMI-S processor is described in detail in the ARM7TDMI-S Datasheet that can be found on official ARM website.

1.7 On-chip Flash memory system

The LPC2141/2/4/6/8 incorporate a 32 kB, 64 kB, 128 kB, 256 kB, and 512 kB Flash memory system respectively. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways: over the serial built-in JTAG interface, using In System Programming (ISP) and UART0, or by means of In Application Programming (IAP) capabilities. The application program, using the IAP functions, may also erase and/or program the Flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc. When the LPC2141/2/4/6/8 on-chip bootloader is used, 32 kB, 64 kB, 128 kB, 256 kB, and 500 kB of Flash memory is available for user code.

The LPC2141/2/4/6/8 Flash memory provides minimum of 100,000 erase/write cycles and 20 years of data-retention.

1.8 On-chip Static RAM (SRAM)

On-chip Static RAM (SRAM) may be used for code and/or data storage. The on-chip SRAM may be accessed as 8-bits, 16-bits, and 32-bits. The LPC2141/2/4/6/8 provide 8/16/32 kB of static RAM respectively.

The LPC2141/2/4/6/8 SRAM is designed to be accessed as a byte-addressed memory. Word and halfword accesses to the memory ignore the alignment of the address and access the naturally-aligned value that is addressed (so a memory access ignores address bits 0 and 1 for word accesses, and ignores bit 0 for halfword accesses). Therefore valid reads and writes require data accessed as halfwords to originate from addresses with address line 0 being 0 (addresses ending with 0, 2, 4, 6, 8, A, C, and E in hexadecimal notation) and data accessed as words to originate from addresses with address lines 0 and 1 being 0 (addresses ending with 0, 4, 8, and C in hexadecimal notation). This rule applies to both off and on-chip memory usage.

The SRAM controller incorporates a write-back buffer in order to prevent CPU stalls during back-to-back writes. The write-back buffer always holds the last data sent by software to the SRAM. This data is only written to the SRAM when another write is requested by software (the data is only written to the SRAM when software does another write). If a chip reset occurs, actual SRAM contents will not reflect the most recent write request (i.e. after a "warm" chip reset, the SRAM does not reflect the last write operation). Any software that checks SRAM contents after reset must take this into account. Two identical writes to a location guarantee that the data will be present after a Reset. Alternatively, a dummy write operation before entering idle or power-down mode will similarly guarantee that the last data written will be present in SRAM after a subsequent Reset.

1.9 Block diagram

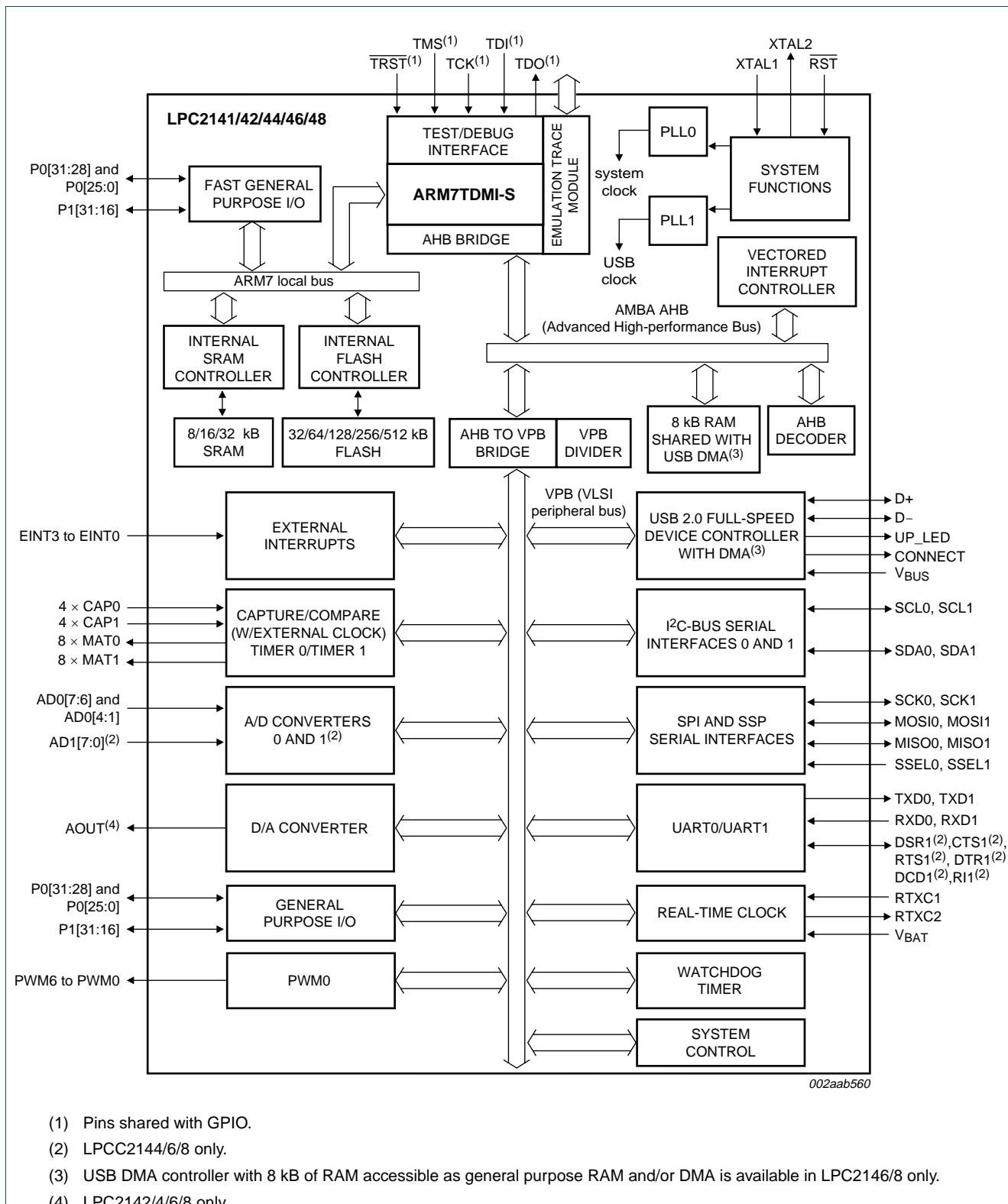
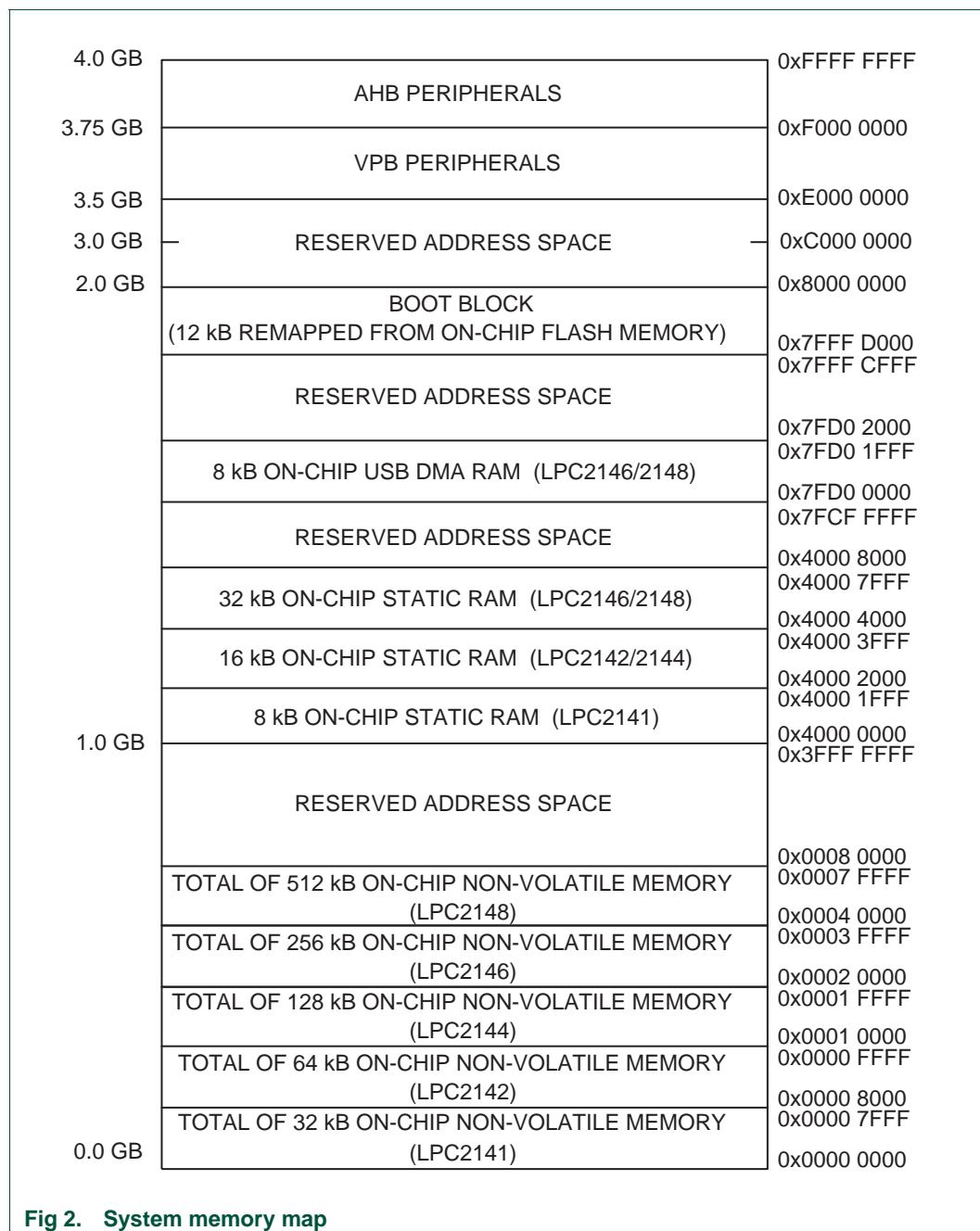
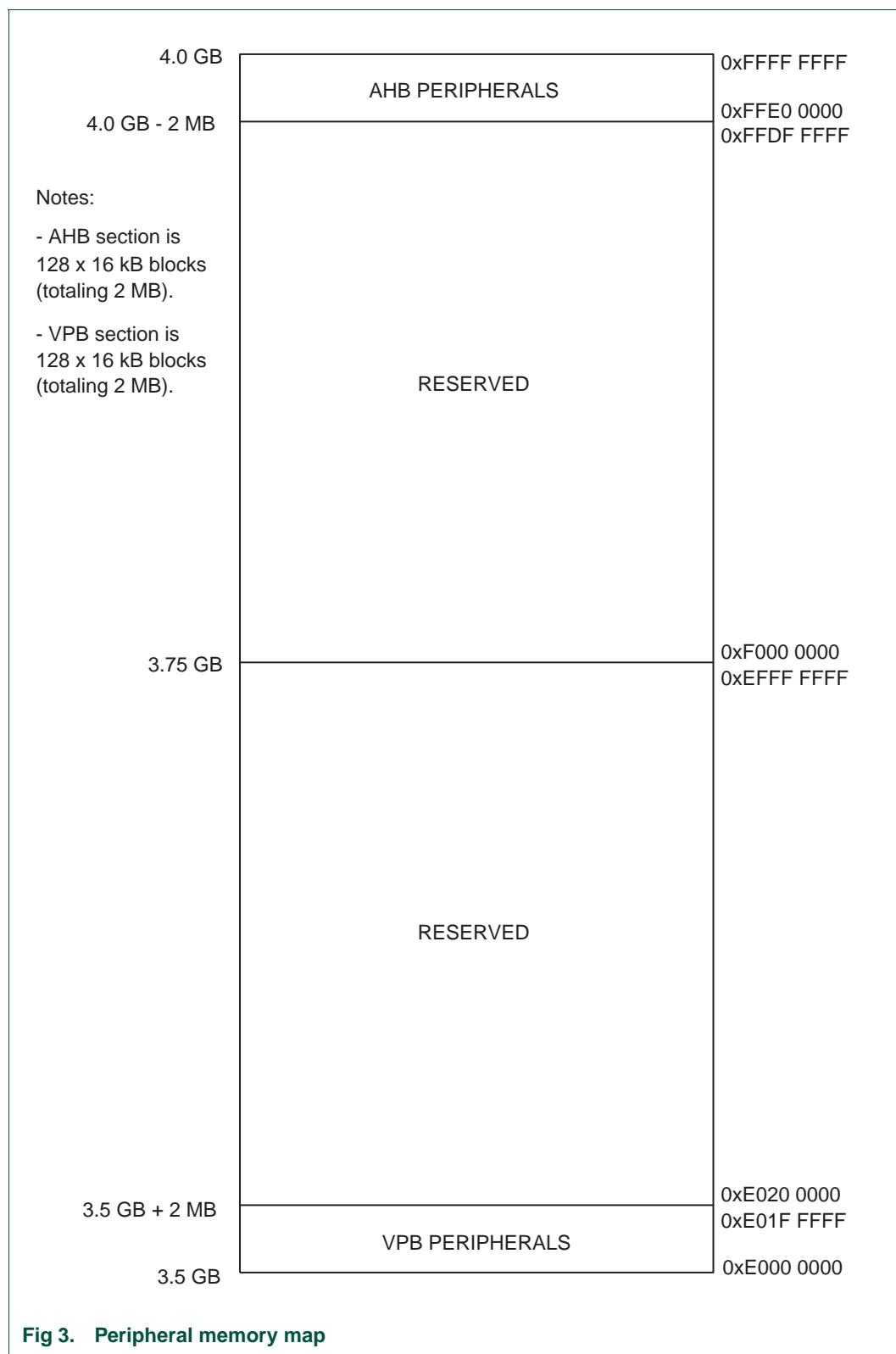


Fig 1. LPC2141/2/4/6/8 block diagram

2.1 Memory maps

The LPC2141/2/4/6/8 incorporates several distinct memory regions, shown in the following figures. [Figure 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 3. Peripheral memory map**

Figures 3 through 4 and Table 2 show different views of the peripheral address space. Both the AHB and VPB 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.

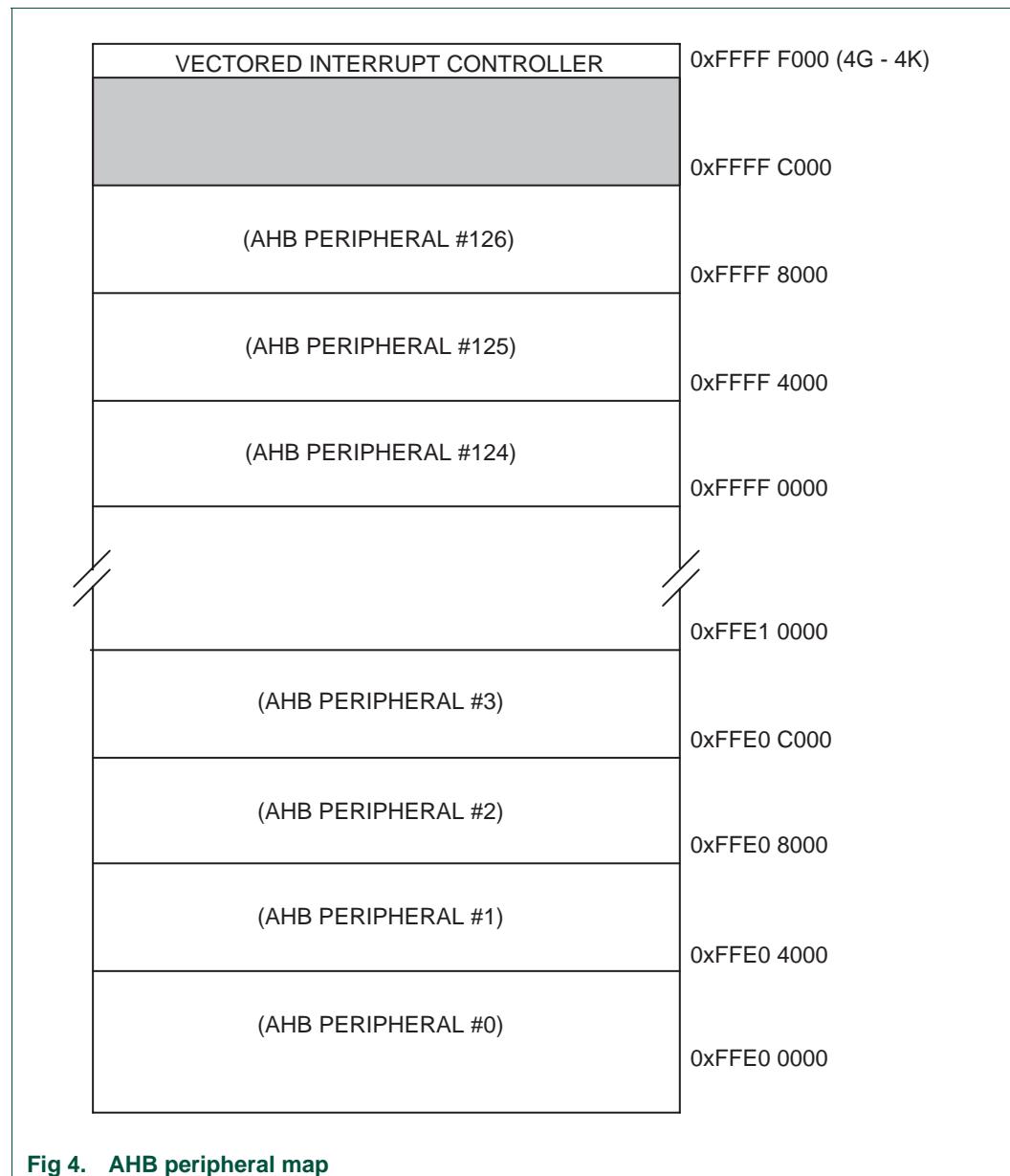


Fig 4. AHB peripheral map

Table 2: VPB peripheries and base addresses

| VPB peripheral | Base address | Peripheral name |
|----------------|--------------|-----------------|
| 0 | 0xE000 0000 | Watchdog timer |
| 1 | 0xE000 4000 | Timer 0 |
| 2 | 0xE000 8000 | Timer 1 |

Table 2: VPB peripheries and base addresses

| VPB peripheral | Base address | Peripheral name |
|----------------|----------------------------|----------------------|
| 3 | 0xE000 C000 | UART0 |
| 4 | 0xE001 0000 | UART1 |
| 5 | 0xE001 4000 | PWM |
| 6 | 0xE001 8000 | Not used |
| 7 | 0xE001 C000 | I ² C0 |
| 8 | 0xE002 0000 | SPI0 |
| 9 | 0xE002 4000 | RTC |
| 10 | 0xE002 8000 | GPIO |
| 11 | 0xE002 C000 | Pin connect block |
| 12 | 0xE003 0000 | Not used |
| 13 | 0xE003 4000 | ADC0 |
| 14 - 22 | 0xE003 8000 0xE005 8000 | Not used |
| 23 | 0xE005 C000 | I ² C1 |
| 24 | 0xE006 0000 | ADC1 |
| 25 | 0xE006 4000 | Not used |
| 26 | 0xE006 8000 | SSP |
| 27 | 0xE006 C000 | DAC |
| 28 - 35 | 0xE007 0000 0xE008 C000 | Not used |
| 36 | 0xE009 0000 | USB |
| 37 - 126 | 0xE009 4000 0xE01F 8000 | Not used |
| 127 | 0xE01F C000 | System Control Block |

2.2 LPC2141/2142/2144/2146/2148 memory re-mapping and boot block

2.2.1 Memory map concepts and operating modes

The basic concept on the LPC2141/2/4/6/8 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 3](#) below), a small portion of the Boot Block and SRAM spaces need to be re-mapped in order to allow alternative uses of interrupts in the different operating modes described in [Table 4](#). Re-mapping of the interrupts is accomplished via the Memory Mapping Control feature ([Section 3.7 "Memory mapping control" on page 26](#)).

Table 3: 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 291. |
| 0x0000 0018 | IRQ |
| 0x0000 001C | FIQ |

Table 4: LPC2141/2/4/6/8 memory mapping modes

| Mode | Activation | Usage |
|------------------|-------------------------------------|--|
| Boot Loader mode | Hardware activation by any Reset | The Boot Loader always executes after any reset. The Boot Block interrupt vectors are mapped to the bottom of memory to allow handling exceptions and using interrupts during the Boot Loading process. |
| User Flash mode | Software activation by Boot code | Activated by 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. |

2.2.2 Memory re-mapping

In order to allow for compatibility with future derivatives, the entire Boot Block 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 Block (which would require changing the Boot Loader code itself) or changing the mapping of the Boot Block interrupt vectors. Memory spaces other than the interrupt vectors remain in fixed locations.

[Figure 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. The re-mapped 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 Block must contain branches to the actual interrupt handlers, or to other instructions that accomplish the branch to the interrupt handlers.

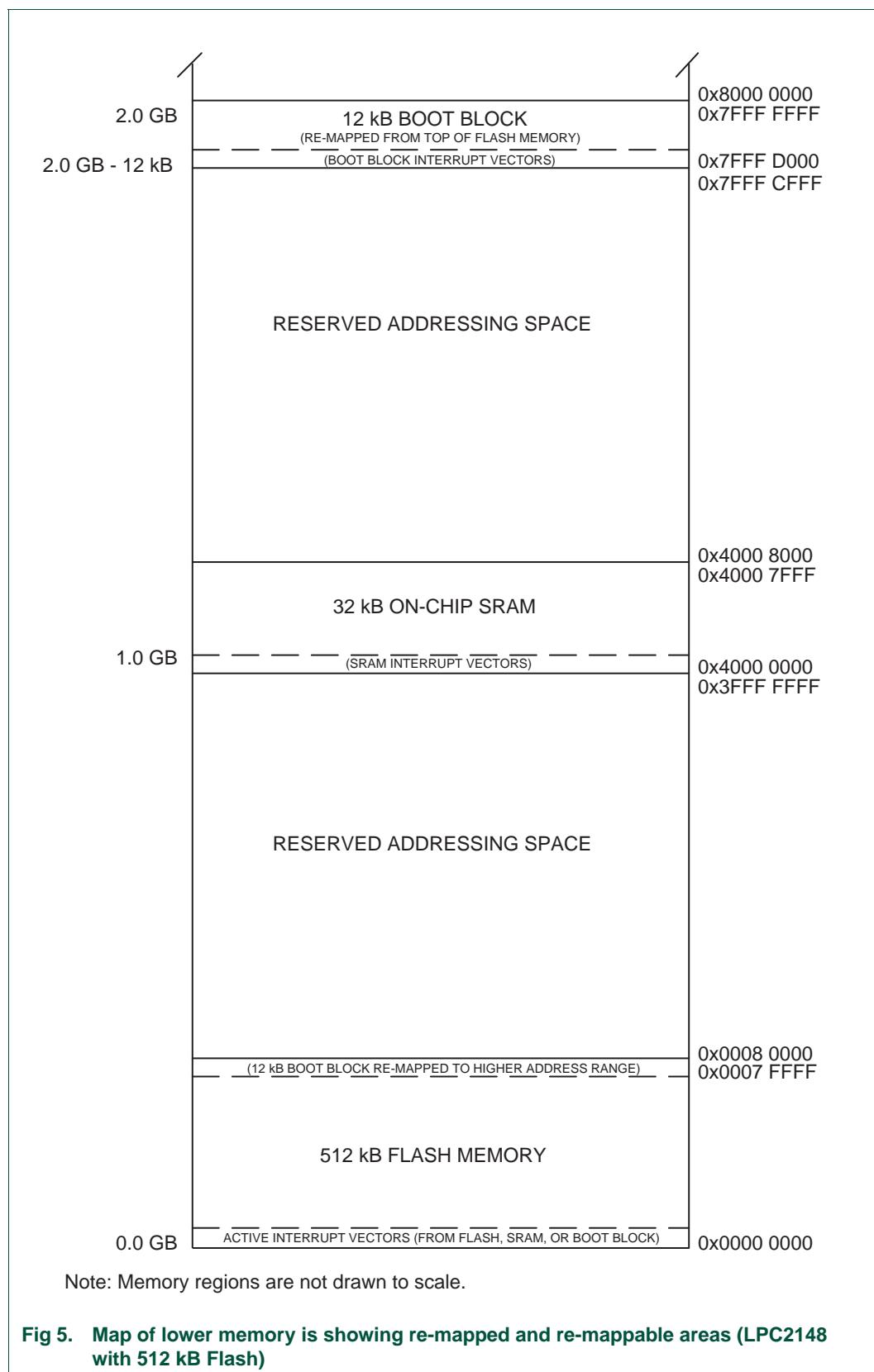
There are three reasons this configuration was chosen:

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 Block 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 Block 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 3.7 "Memory mapping control" on page 26](#).



2.3 Prefetch abort and data abort exceptions

The LPC2141/2/4/6/8 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 LPC2141/2/4/6/8, this is:
 - Address space between On-Chip Non-Volatile Memory and On-Chip SRAM, labelled "Reserved Address Space" in [Figure 2](#). For 32 kB Flash device this is memory address range from 0x0000 8000 to 0x3FFF FFFF, for 64 kB Flash device this is memory address range from 0x0001 0000 to 0x3FFF FFFF, for 128 kB Flash device this is memory address range from 0x0002 0000 to 0x3FFF FFFF, for 256 kB Flash device this is memory address range from 0x0004 0000 to 0x3FFF FFFF while for 512 kB Flash device this range is from 0x0008 0000 to 0x3FFF FFFF.
 - Address space between On-Chip Static RAM and the Boot Block. Labelled "Reserved Address Space" in [Figure 2](#). For 8 kB SRAM device this is memory address range from 0x4000 2000 to 0x7FFF CFFF, for 16 kB SRAM device this is memory address range from 0x4000 4000 to 0x7FFF CFFF. For 32 kB SRAM device this range is from 0x4000 8000 to 0x7FCF FFFF where the 8 kB USB DMA RAM starts, and from 0x7FD0 2000 to 0x7FFF CFFF.
 - Address space between 0x8000 0000 and 0xDFFF FFFF, labelled "Reserved Adress Space".
 - Reserved regions of the AHB and VPB spaces. See [Figure 3](#).
- Unassigned AHB peripheral spaces. See [Figure 4](#).
- Unassigned VPB peripheral spaces. See [Table 2](#).

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 VPB peripheral address.

Within the address space of an existing VPB 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 LPC2141/2/4/6/8 documentation and are not a supported feature.

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.



3.1 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:

- Crystal Oscillator
- External Interrupt Inputs
- Miscellaneous System Controls and Status
- Memory Mapping Control
- PLL
- Power Control
- Reset
- VPB Divider
- Wakeup Timer

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

3.2 Pin description

[Table 5](#) shows pins that are associated with System Control block functions.

Table 5: Pin summary

| Pin name | Pin direction | Pin description |
|----------|---------------|--|
| X1 | Input | Crystal Oscillator Input - Input to the oscillator and internal clock generator circuits |
| X2 | Output | Crystal Oscillator Output - Output from the oscillator amplifier |
| 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. Pins P0.1 and P0.16 can be selected to perform EINT0 function. |
| EINT1 | Input | External Interrupt Input 1 - See the EINT0 description above. Pins P0.3 and P0.14 can be selected to perform EINT1 function. Important: LOW level on pin P0.14 immediately after reset is considered as an external hardware request to start the ISP command handler. More details on ISP and Serial Boot Loader can be found in "Flash Memory System and Programming" chapter on page 291. |

Table 5: Pin summary

| Pin name | Pin direction | Pin description |
|----------|---------------|--|
| EINT2 | Input | External Interrupt Input 2 - See the EINT0 description above. Pins P0.7 and P0.15 can be selected to perform EINT2 function. |
| EINT3 | Input | External Interrupt Input 3 - See the EINT0 description above. Pins P0.9, P0.20 and P0.30 can be selected to perform EINT3 function. |
| RESET | Input | External Reset input - A LOW on this pin resets the chip, causing I/O ports and peripherals to take on their default states, and the processor to begin execution at address 0x0000 0000. |

3.3 Register description

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

Table 6: Summary of system control registers

| Name | Description | Access | Reset value ^[1] | Address |
|--------------------------------|--------------------------------------|--------|----------------------------|-------------|
| External Interrupts | | | | |
| EXTINT | External Interrupt Flag Register | R/W | 0 | 0xE01F C140 |
| INTWAKE | Interrupt Wakeup Register | R/W | 0 | 0xE01F C144 |
| EXTMODE | External Interrupt Mode Register | R/W | 0 | 0xE01F C148 |
| EXTPOLAR | External Interrupt Polarity Register | R/W | 0 | 0xE01F C14C |
| Memory Mapping Control | | | | |
| MEMMAP | Memory Mapping Control | R/W | 0 | 0xE01F C040 |
| Phase Locked Loop | | | | |
| PLL0CON | PLL0 Control Register | R/W | 0 | 0xE01F C080 |
| PLL0CFG | PLL0 Configuration Register | R/W | 0 | 0xE01F C084 |
| PLL0STAT | PLL0 Status Register | RO | 0 | 0xE01F C088 |
| PLL0FEED | PLL0 Feed Register | WO | NA | 0xE01F C08C |
| PLL1CON | PLL1 (USB) Control Register | R/W | 0 | 0xE01F C0A0 |
| PLL1CFG | PLL1 (USB) Configuration Register | R/W | 0 | 0xE01F C0A4 |
| PLL1STAT | PLL1 (USB) Status Register | RO | 0 | 0xE01F C0A8 |
| PLL1FEED | PLL1 (USB) Feed Register | WO | NA | 0xE01F C0AC |
| Power Control | | | | |
| PCON | Power Control Register | R/W | 0 | 0xE01F C0C0 |
| PCONP | Power Control for Peripherals | R/W | 0x03BE | 0xE01F C0C4 |
| VPB Divider | | | | |
| VPBDIV | VPB Divider Control | R/W | 0 | 0xE01F C100 |
| Reset | | | | |
| RSID | Reset Source Identification Register | R/W | 0 | 0xE01F C180 |
| Code Security/Debugging | | | | |

Table 6: Summary of system control registers

| Name | Description | Access | Reset value ^[1] | Address |
|---------------------------------------|-----------------------------------|--------|----------------------------|-------------|
| CSPR | Code Security Protection Register | RO | 0 | 0xE01F C184 |
| Syscon Miscellaneous Registers | | | | |
| SCS | System Controls and Status | R/W | 0 | 0xE01F C1A0 |

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

3.4 Crystal oscillator

While an input signal of 50-50 duty cycle within a frequency range from 1 MHz to 50 MHz can be used by the LPC2141/2/4/6/8 if supplied to its input XTAL1 pin, this microcontroller's onboard oscillator circuit supports external crystals in the range of 1 MHz to 30 MHz only. If the on-chip PLL system or the boot-loader is used, the input clock frequency is limited to an exclusive range of 10 MHz to 25 MHz.

The oscillator output frequency is called F_{OSC} and the ARM processor clock frequency is referred to as CCLK for purposes of rate equations, etc. elsewhere in this document. F_{OSC} and CCLK are the same value unless the PLL is running and connected. Refer to the [Section 3.8 "Phase Locked Loop \(PLL\)" on page 27](#) for details and frequency limitations.

The onboard oscillator in the LPC2141/2/4/6/8 can operate in one of two modes: slave mode and oscillation mode.

In slave mode the input clock signal should be coupled by means of a capacitor of 100 pF (C_C in [Figure 6](#), drawing a), with an amplitude of at least 200 mVrms. The X2 pin in this configuration can be left not connected. If slave mode is selected, the F_{OSC} signal of 50-50 duty cycle can range from 1 MHz to 50 MHz.

External components and models used in oscillation mode are shown in [Figure 6](#), drawings b and c, and in [Table 7](#). Since the feedback resistance is integrated on chip, only a crystal and the capacitances C_{X1} and C_{X2} need to be connected externally in case of fundamental mode oscillation (the fundamental frequency is represented by L , C_L and R_S). Capacitance C_P in [Figure 6](#), drawing c, represents the parallel package capacitance and should not be larger than 7 pF. Parameters F_C , C_L , R_S and C_P are supplied by the crystal manufacturer.

Choosing an oscillation mode as an on-board oscillator mode of operation limits F_{OSC} clock selection to 1 MHz to 30 MHz.

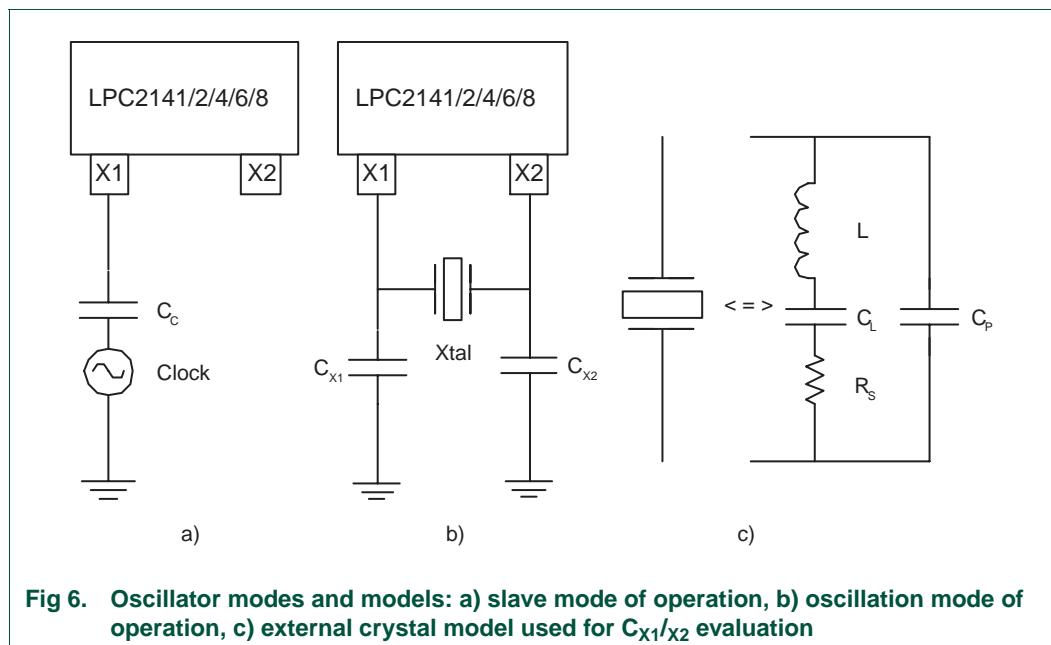


Fig 6. Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for $C_{x1/x2}$ evaluation

Table 7: Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters)

| Fundamental oscillation frequency f_{osc} | Crystal load capacitance C_L | Maximum crystal series resistance R_S | External load capacitors C_{x1}, C_{x2} |
|---|--------------------------------|---|---|
| 1 MHz - 5 MHz | 10 pF | NA | NA |
| | 20 pF | NA | NA |
| | 30 pF | < 300 Ω | 58 pF, 58 pF |
| 5 MHz - 10 MHz | 10 pF | < 300 Ω | 18 pF, 18 pF |
| | 20 pF | < 300 Ω | 38 pF, 38 pF |
| | 30 pF | < 300 Ω | 58 pF, 58 pF |
| 10 MHz - 15 MHz | 10 pF | < 300 Ω | 18 pF, 18 pF |
| | 20 pF | < 220 Ω | 38 pF, 38 pF |
| | 30 pF | < 140 Ω | 58 pF, 58 pF |
| 15 MHz - 20 MHz | 10 pF | < 220 Ω | 18 pF, 18 pF |
| | 20 pF | < 140 Ω | 38 pF, 38 pF |
| | 30 pF | < 80 Ω | 58 pF, 58 pF |
| 20 MHz - 25 MHz | 10 pF | < 160 Ω | 18 pF, 18 pF |
| | 20 pF | < 90 Ω | 38 pF, 38 pF |
| | 30 pF | < 50 Ω | 58 pF, 58 pF |
| 25 MHz - 30 MHz | 10 pF | < 130 Ω | 18 pF, 18 pF |
| | 20 pF | < 50 Ω | 38 pF, 38 pF |
| | 30 pF | NA | NA |

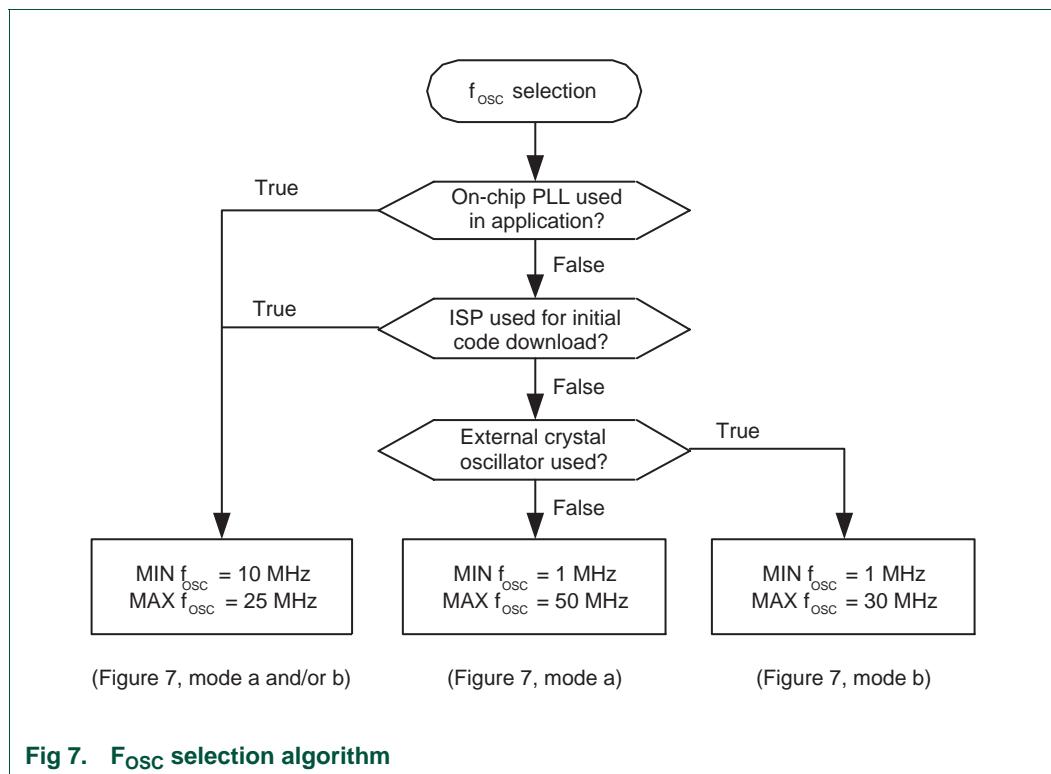


Fig 7. Fosc selection algorithm

3.5 External interrupt inputs

The LPC2141/2/4/6/8 includes four External Interrupt Inputs as selectable pin functions. The External Interrupt Inputs can optionally be used to wake up the processor from Power-down mode.

3.5.1 Register description

The external interrupt function has four registers associated with it. The EXTINT register contains the interrupt flags, and the EXTWAKEUP register contains bits that enable individual external interrupts to wake up the microcontroller from Power-down mode. The EXTMODE and EXTPOLAR registers specify the level and edge sensitivity parameters.

Table 8: 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 9 . | R/W | 0 | 0xE01F C140 |

Table 8: External interrupt registers

| Name | Description | Access | Reset value ^[1] | Address |
|----------|---|--------|----------------------------|-------------|
| INTWAKE | The Interrupt Wakeup Register contains four enable bits that control whether each external interrupt will cause the processor to wake up from Power-down mode. See Table 10 . | R/W | 0 | 0xE01F C144 |
| EXTMODE | The External Interrupt Mode Register controls whether each pin is edge- or level sensitive. | R/W | 0 | 0xE01F C148 |
| EXTPOLAR | The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt. | R/W | 0 | 0xE01F C14C |

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

3.5.2 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 this action is efficacious 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 the event that was just triggered by activity on the EINT pin will not be recognized in the future.

Important: whenever a change of external interrupt operating mode (i.e. active level/edge) is performed (including the initialization of an external interrupt), the corresponding bit in the EXTINT register must be cleared! For details see [Section 3.5.4 "External Interrupt Mode register \(EXTMODE - 0xE01F C148\)"](#) and [Section 3.5.5 "External Interrupt Polarity register \(EXTPOLAR - 0xE01F C14C\)"](#).

For example, if a system wakes up from power-down using a low level on external interrupt 0 pin, its post-wakeup code must reset the EINT0 bit in order to allow future entry into the power-down mode. If the 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 9: External Interrupt Flag register (EXTINT - address 0xE01F C140) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|---|-------------|
| 0 | EINT0 | <p>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.</p> <p>Up to two pins can be selected to perform the EINT0 function (see P0.1 and P0.16 description in "Pin Configuration" chapter page 66.)</p> <p>This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT0 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).</p> | 0 |
| 1 | EINT1 | <p>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.</p> <p>Up to two pins can be selected to perform the EINT1 function (see P0.3 and P0.14 description in "Pin Configuration" chapter on page 66.)</p> <p>This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT1 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).</p> | 0 |
| 2 | EINT2 | <p>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.</p> <p>Up to two pins can be selected to perform the EINT2 function (see P0.7 and P0.15 description in "Pin Configuration" chapter on page 66.)</p> <p>This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT2 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).</p> | 0 |
| 3 | EINT3 | <p>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.</p> <p>Up to three pins can be selected to perform the EINT3 function (see P0.9, P0.20 and P0.30 description in "Pin Configuration" chapter on page 66.)</p> <p>This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT3 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).</p> | 0 |
| 7:4 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

3.5.3 Interrupt Wakeup register (INTWAKE - 0xE01F C144)

Enable bits in the INTWAKE register allow the external interrupts and other sources 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).

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 bit in the External Interrupt Flag register ([Section 3.5.2 on page 21](#)).

Table 10: Interrupt Wakeup register (INTWAKE - address 0xE01F C144) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 0 | EXTWAKE0 | When one, assertion of EINT0 will wake up the processor from Power-down mode. | 0 |
| 1 | EXTWAKE1 | When one, assertion of EINT1 will wake up the processor from Power-down mode. | 0 |
| 2 | EXTWAKE2 | When one, assertion of EINT2 will wake up the processor from Power-down mode. | 0 |
| 3 | EXTWAKE3 | When one, assertion of EINT3 will wake up the processor from Power-down mode. | 0 |
| 4 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 5 | USBWAKE | When one, activity of the USB bus (USB_need_clock = 1) 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 Section 14.7.1 "USB Interrupt Status register (USBIIntSt - 0xE01F C1C0)" on page 200 and Section 3.8.8 "PLL and Power-down mode" on page 32 . | 0 |
| 13:4 | - | 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, a BOD interrupt will wake up the processor from Power-down mode. | 0 |
| 15 | RTCWAKE | When one, assertion of an RTC interrupt will wake up the processor from Power-down mode. | 0 |

3.5.4 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 75) and enabled via the VICIntEnable register ([Section 5.4.4 "Interrupt Enable register \(VICIntEnable - 0xFFFF F010\)" on page 54](#)) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

Note: Software should only change a bit in this register when its interrupt is disabled in the VICIntEnable register, and should write the corresponding 1 to the EXTINT register before enabling (initializing) or re-enabling the interrupt, to clear the EXTINT bit that could be set by changing the mode.

Table 11: External Interrupt Mode register (EXTMODE - address 0xE01F C148) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|--|-------------|
| 0 | EXTMODE0 | 0 | Level-sensitivity is selected for EINT0. | 0 |
| | | 1 | EINT0 is edge sensitive. | |
| 1 | EXTMODE1 | 0 | Level-sensitivity is selected for EINT1. | 0 |

Table 11: External Interrupt Mode register (EXTMODE - address 0xE01F C148) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|--|-------------|
| | | 1 | EINT1 is edge sensitive. | |
| 2 | EXTMODE2 | 0 | Level-sensitivity is selected for EINT2. | 0 |
| | | 1 | EINT2 is edge sensitive. | |
| 3 | EXTMODE3 | 0 | Level-sensitivity is selected for EINT3. | 0 |
| | | 1 | 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 |

3.5.5 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 75) and enabled in the VICIntEnable register ([Section 5.4.4 "Interrupt Enable register \(VICIntEnable - 0xFFFF F010\)" on page 54](#)) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

Note: Software should only change a bit in this register when its interrupt is disabled in the VICIntEnable register, and should write the corresponding 1 to the EXTINT register before enabling (initializing) or re-enabling the interrupt, to clear the EXTINT bit that could be set by changing the polarity.

Table 12: External Interrupt Polarity register (EXTPOLAR - address 0xE01F C14C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|-----------|-------|--|-------------|
| 0 | EXTPOLAR0 | 0 | EINT0 is low-active or falling-edge sensitive (depending on EXTMODE0). | 0 |
| | | 1 | EINT0 is high-active or rising-edge sensitive (depending on EXTMODE0). | |
| 1 | EXTPOLAR1 | 0 | EINT1 is low-active or falling-edge sensitive (depending on EXTMODE1). | 0 |
| | | 1 | EINT1 is high-active or rising-edge sensitive (depending on EXTMODE1). | |
| 2 | EXTPOLAR2 | 0 | EINT2 is low-active or falling-edge sensitive (depending on EXTMODE2). | 0 |
| | | 1 | EINT2 is high-active or rising-edge sensitive (depending on EXTMODE2). | |
| 3 | EXTPOLAR3 | 0 | EINT3 is low-active or falling-edge sensitive (depending on EXTMODE3). | 0 |
| | | 1 | 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 |

3.5.6 Multiple external interrupt pins

Software can select multiple pins for each of EINT3:0 in the Pin Select registers, which are described in chapter Pin Connect Block on page 75. The external interrupt logic for each of EINT3:0 receives the state of all of its associated pins from the pins' receivers, along with signals that indicate whether each pin is selected for the EINT function. The external interrupt logic handles the case when more than one pin is so selected, differently according to the state of its Mode and Polarity bits:

- In Low-Active Level Sensitive mode, the states of all pins selected for the same EINTx functionality are digitally combined using a positive logic AND gate.
- In High-Active Level Sensitive mode, the states of all pins selected for the same EINTx functionality are digitally combined using a positive logic OR gate.
- In Edge Sensitive mode, regardless of polarity, the pin with the lowest GPIO port number is used. (Selecting multiple pins for an EINTx in edge-sensitive mode could be considered a programming error.)

The signal derived by this logic processing multiple external interrupt pins is the EINT_i signal in the following logic schematic [Figure 8](#).

For example, if the EINT3 function is selected in the PINSEL0 and PINSEL1 registers for pins P0.9, P0.20 and P0.30, and EINT3 is configured to be low level sensitive, the inputs from all three pins will be logically ANDed. When more than one EINT pin is logically ORed, the interrupt service routine can read the states of the pins from the GPIO port using the IO0PIN and IO1PIN registers, to determine which pin(s) caused the interrupt.

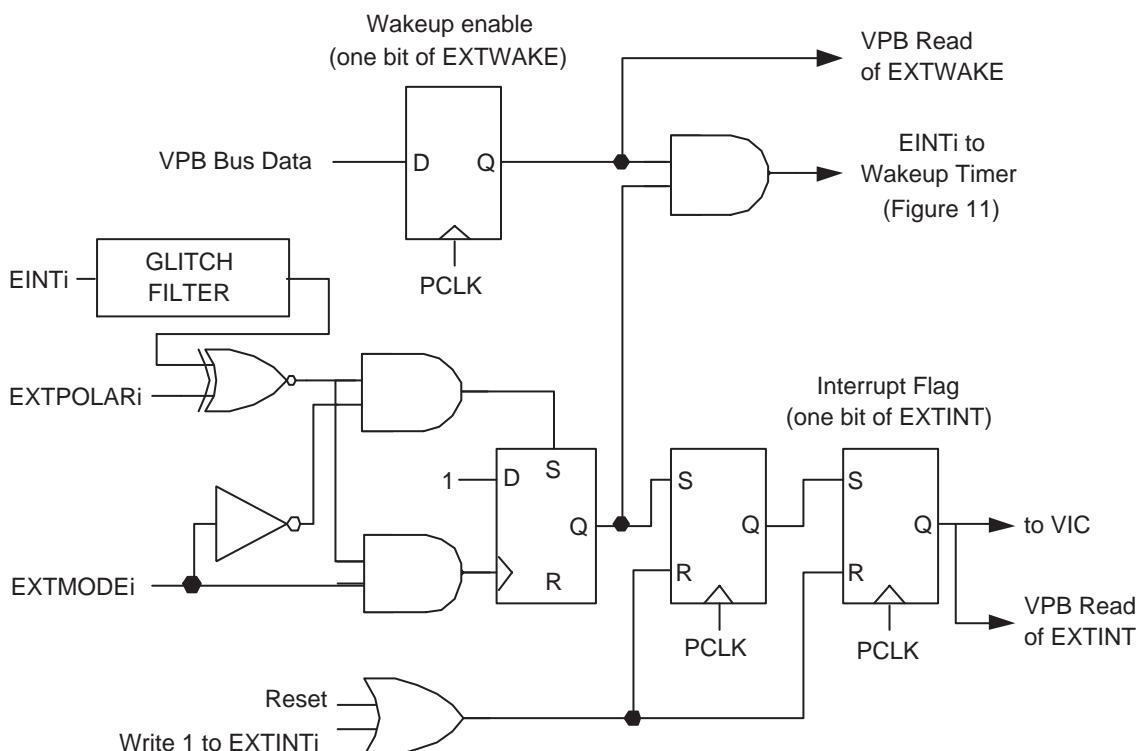


Fig 8. External interrupt logic

3.6 Other system controls

Some aspects of controlling LPC2141/2/4/6/8 operation that do not fit into peripheral or other registers are grouped here.

3.6.1 System Control and Status flags register (SCS - 0xE01F C1A0)

Table 13: System Control and Status flags register (SCS - address 0xE01F C1A0) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|--|---|-------------|
| 0 | GPIO0M | GPIO port 0 mode selection. | | 0 |
| | | 0 | GPIO port 0 is accessed via VPB addresses in a fashion compatible with previous LCP2000 devices. | |
| | | 1 | High speed GPIO is enabled on GPIO port 0, accessed via addresses in the on-chip memory range. This mode includes the port masking feature described in the GPIO chapter on page page 81. | |
| 1 | GPIO1M | GPIO port 1 mode selection. | | 0 |
| | | 0 | GPIO port 1 is accessed via VPB addresses in a fashion compatible with previous LCP2000 devices. | |
| | | 1 | High speed GPIO is enabled on GPIO port 1, accessed via addresses in the on-chip memory range. This mode includes the port masking feature described in the GPIO chapter on page page 81. | |
| 31:2 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | | NA |

3.7 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.

3.7.1 Memory Mapping control register (MEMMAP - 0xE01F C040)

Whenever an exception handling is necessary, the microcontroller will fetch an instruction residing on the exception corresponding address as described in [Table 3 “ARM exception vector locations” on page 12](#). The MEMMAP register determines the source of data that will fill this table.

Table 14: 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 Block. | 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 | Reserved. Do not use this option. | |
| 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 |

3.7.2 Memory mapping control usage notes

The 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, the ARM core will always fetch 32-bit data "residing" on 0x0000 0008 see [Table 3 "ARM exception vector locations" on page 12](#). This means that when MEMMAP[1:0]=10 (User RAM Mode), a read/fetch from 0x0000 0008 will provide data stored in 0x4000 0008. In case of MEMMAP[1:0]=00 (Boot Loader Mode), a read/fetch from 0x0000 0008 will provide data available also at 0x7FFF E008 (Boot Block remapped from on-chip Bootloader).

3.8 Phase Locked Loop (PLL)

There are two PLL modules in the LPC2141/2/4/6/8 microcontroller. The PLL0 is used to generate the CCLK clock (system clock) while the PLL1 has to supply the clock for the USB at the fixed rate of 48 MHz. Structurally these two PLLs are identical with exception of the PLL interrupt capabilities reserved only for the PLL0.

The PLL0 and PLL1 accept an input clock frequency in the range of 10 MHz to 25 MHz only. The input frequency is multiplied up the range of 10 MHz to 60 MHz for the CCLK and 48 MHz for the USB clock using a Current Controlled Oscillators (CCO). The multiplier can be an integer value from 1 to 32 (in practice, the multiplier value cannot be higher than 6 on the LPC2141/2/4/6/8 due to the upper frequency limit of the CPU). The CCO operates in the range of 156 MHz to 320 MHz, so there is an additional divider in the loop to keep the CCO within its frequency range while the PLL is providing the desired output frequency. The output divider may be set to divide by 2, 4, 8, or 16 to produce the output clock. Since the minimum output divider value is 2, it is insured that the PLL output has a 50% duty cycle. A block diagram of the PLL is shown in [Figure 9](#).

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, are dependent on the PLL0 when it is providing the chip clock, accidental changes to the PLL setup could result in unexpected behavior of

the microcontroller. The same concern is present with the PLL1 and the USB. 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.

Both PLLs are turned off and bypassed following a chip Reset and when by entering Power-down mode. The PLL is enabled by software only. The program must configure and activate the PLL, wait for the PLL to Lock, then connect to the PLL as a clock source.

3.8.1 Register description

The PLL is controlled by the registers shown in [Table 15](#). More detailed descriptions follow.

Warning: Improper setting of the PLL0 and PLL1 values may result in incorrect operation of the device and the USB module!

Table 15: PLL registers

| Generic name | Description | Access | Reset value ^[1] | System clock (PLL0) Address & Name | USB 48 MHz clock (PLL1) Address & Name |
|--------------|---|--------|----------------------------|---------------------------------------|---|
| 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 PLL0CON | 0xE01F C0A0 PLL1CON |
| 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 PLL0CFG | 0xE01F C0A4 PLL1CFG |
| 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 status of the PLL. | RO | 0 | 0xE01F C088 PLL0STAT | 0xE01F C0A8 PLL1STAT |
| 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 PLL0FEED | 0xE01F C0AC PLL1FEED |

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

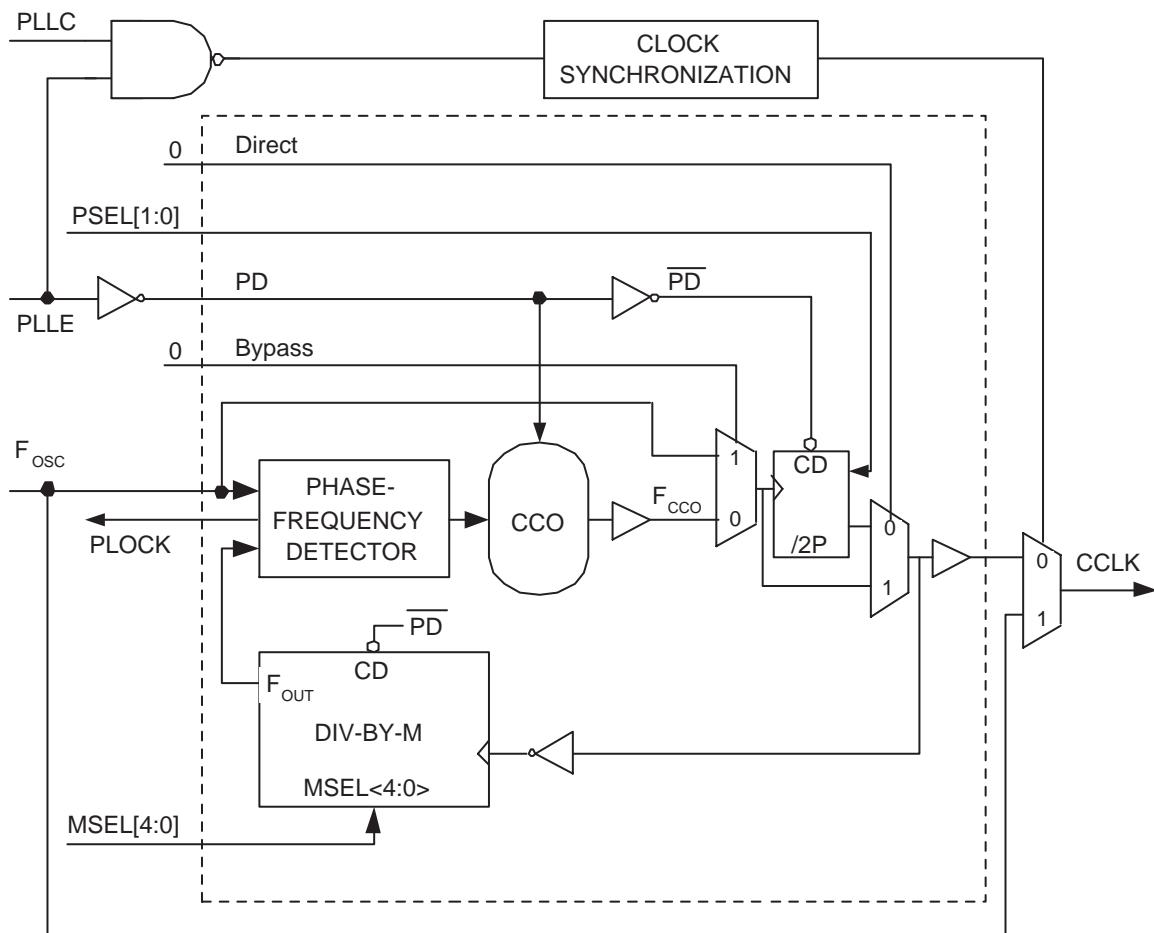


Fig 9. PLL block diagram

3.8.2 PLL Control register (PLL0CON - 0xE01F C080, PLL1CON - 0xE01F C0A0)

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 3.8.7 “PLL Feed register \(PLL0FEED - 0xE01F C08C, PLL1FEED - 0xE01F C0AC\)”](#) and [Section 3.8.3 “PLL Configuration register \(PLL0CFG - 0xE01F C084, PLL1CFG - 0xE01F C0A4\)”](#) on page 30).

Table 16: PLL Control register (PLL0CON - address 0xE01F C080, PLL1CON - address 0xE01F C0A0) 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 18 . | 0 |
| 1 | PLLC | PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the microcontroller. Otherwise, the oscillator clock is used directly by the microcontroller. See PLLSTAT register, Table 18 . | 0 |
| 7:2 | - | Reserved, user software should not write ones to reserved bits. The NA 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.

3.8.3 PLL Configuration register (PLL0CFG - 0xE01F C084, PLL1CFG - 0xE01F C0A4)

The PLLCFG register contains the PLL multiplier and divider values. Changes to the PLLCFG register do not take effect until a correct PLL feed sequence has been given (see [Section 3.8.7 "PLL Feed register \(PLL0FEED - 0xE01F C08C, PLL1FEED - 0xE01F C0AC\)" on page 32](#)). Calculations for the PLL frequency, and multiplier and divider values are found in the PLL Frequency Calculation section on page 33.

Table 17: PLL Configuration register (PLL0CFG - address 0xE01F C084, PLL1CFG - address 0xE01F C0A4) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 4:0 | MSEL | PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations. Note: For details on selecting the right value for MSEL see Section 3.8.9 "PLL frequency calculation" on page 33 . | 0 |
| 6:5 | PSEL | PLL Divider value. Supplies the value "P" in the PLL frequency calculations. Note: For details on selecting the right value for PSEL see Section 3.8.9 "PLL frequency calculation" on page 33 . | 0 |
| 7 | - | Reserved, user software should not write ones to reserved bits. The NA value read from a reserved bit is not defined. | NA |

3.8.4 PLL Status register (PLL0STAT - 0xE01F C088, PLL1STAT - 0xE01F C0A8)

The read-only PLLSTAT register provides the actual PLL parameters that are in effect at the time it is read, as well as the PLL status. PLLSTAT may disagree with values found in PLLCON and PLLCFG because changes to those registers do not take effect until a proper PLL feed has occurred (see [Section 3.8.7 “PLL Feed register \(PLL0FEED - 0xE01F C08C, PLL1FEED - 0xE01F C0AC\)”](#)).

Table 18: PLL Status register (PLL0STAT - address 0xE01F C088, PLL1STAT - address 0xE01F C0A8) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 4:0 | MSEL | Read-back for the PLL Multiplier value. This is the value currently used by the PLL. | 0 |
| 6:5 | PSEL | Read-back for the PLL Divider value. This is the value currently used by the PLL. | 0 |
| 7 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 8 | 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 |
| 9 | 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 microcontroller. When either PLLC or PLLE is zero, the PLL is bypassed and the oscillator clock is used directly by the microcontroller. This bit is automatically cleared when Power-down mode is activated. | 0 |
| 10 | PLOCK | Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency. | 0 |
| 15:11 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

3.8.5 PLL Interrupt

The PLOCK bit in the PLLSTAT register 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 (PLOCK = 1), the PLL may be connected, and the interrupt disabled. For details on how to enable and disable the PLL interrupt, see [Section 5.4.4 “Interrupt Enable register \(VICIntEnable - 0xFFFF F010\)” on page 54](#) and [Section 5.4.5 “Interrupt Enable Clear register \(VICIntEnClear - 0xFFFF F014\)” on page 55](#).

PLL interrupt is available only in PLL0, i.e. the PLL that generates the CCLK. USB dedicated PLL1 does not have this capability.

3.8.6 PLL Modes

The combinations of PLLE and PLLC are shown in [Table 19](#).

Table 19: PLL Control bit combinations

| PLLC | PLLE | PLL Function |
|------|------|---|
| 0 | 0 | PLL is turned off and disconnected. The CCLK equals the unmodified clock input. This combination can not be used in case of the PLL1 since there will be no 48 MHz clock and the USB can not operate. |
| 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. CCLK/system clock is sourced from the PLL0 and the USB clock is sourced from the PLL1. |

3.8.7 PLL Feed register (PLL0FEED - 0xE01F C08C, PLL1FEED - 0xE01F C0AC)

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 VPB 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 20: PLL Feed register (PLL0FEED - address 0xE01F C08C, PLL1FEED - address 0xE01F C0AC) 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 |

3.8.8 PLL and Power-down mode

Power-down mode automatically turns off and disconnects activated PLL(s). 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.

If activity on the USB data lines is not selected to wake up the microcontroller from Power-down mode (see [Section 3.5.3 “Interrupt Wakeup register \(INTWAKE - 0xE01F C144\)” on page 22](#)), both the system and the USB PLL will be automatically be turned off and disconnected when Power-down mode is invoked, as described above. However, in case USBWAKE = 1 and USB_need_clock = 1 it is not possible to go into Power-down mode and any attempt to set the PD bit will fail, leaving the PLLs in the current state.

3.8.9 PLL frequency calculation

The PLL equations use the following parameters:

Table 21: Elements determining PLL's frequency

| Element | Description |
|---------|--|
| Fosc | the frequency from the crystal oscillator/external oscillator |
| Fcco | the frequency of the PLL current controlled oscillator |
| CCLK | the PLL output frequency (also the processor clock frequency) |
| M | PLL Multiplier value from the MSEL bits in the PLLCFG register |
| P | PLL Divider value from the PSEL bits in the PLLCFG register |

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

$$\text{CCLK} = M \times F_{\text{osc}} \text{ or } \text{CCLK} = F_{\text{cco}} / (2 \times P)$$

The CCO frequency can be computed as:

$$F_{\text{cco}} = \text{CCLK} \times 2 \times P \text{ or } F_{\text{cco}} = F_{\text{osc}} \times M \times 2 \times P$$

The PLL inputs and settings must meet the following:

- F_{osc} is in the range of 10 MHz to 25 MHz.
- CCLK is in the range of 10 MHz to F_{max} (the maximum allowed frequency for the microcontroller - determined by the system microcontroller it is embedded in).
- F_{cco} is in the range of 156 MHz to 320 MHz.

3.8.10 Procedure for determining PLL settings

If a particular application uses the PLL0, its configuration may be determined as follows:

1. 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 than the processor (see [Section 3.11 “VPB divider” on page 40](#)).
2. Choose an oscillator frequency (F_{osc}). CCLK must be the whole (non-fractional) multiple of F_{osc} .
3. Calculate the value of M to configure the MSEL bits. $M = \text{CCLK} / F_{\text{osc}}$. M must be in the range of 1 to 32. The value written to the MSEL bits in PLLCFG is $M - 1$ (see [Table 23](#)).
4. Find a value for P to configure the PSEL bits, such that F_{cco} is within its defined frequency limits. F_{cco} is calculated using the equation given above. P must have one of the values 1, 2, 4, or 8. The value written to the PSEL bits in PLLCFG is 00 for $P = 1$; 01 for $P = 2$; 10 for $P = 4$; 11 for $P = 8$ (see [Table 22](#)).

Important: if a particular application is using the USB peripheral, the PLL1 must be configured since this is the only available source of the 48 MHz clock required by the USB. This limits the selection of F_{osc} to either 12 MHz, 16 MHz or 24 MHz.

Table 22: PLL Divider values

| PSEL Bits (PLLCFG bits [6:5]) | Value of P |
|-------------------------------|------------|
| 00 | 1 |
| 01 | 2 |
| 10 | 4 |
| 11 | 8 |

Table 23: PLL Multiplier values

| MSEL Bits (PLLCFG bits [4:0]) | Value of M |
|-------------------------------|------------|
| 00000 | 1 |
| 00001 | 2 |
| 00010 | 3 |
| 00011 | 4 |
| ... | ... |
| 11110 | 31 |
| 11111 | 32 |

3.8.11 PLL0 and PLL1 configuring examples

Example 1: an application not using the USB - configuring the PLL0

System design asks for $F_{OSC} = 10$ MHz and requires $CCLK = 60$ MHz.

Based on these specifications, $M = CCLK / Fosc = 60$ MHz / 10 MHz = 6. Consequently, $M - 1 = 5$ will be written as $PLLCFG[4:0]$.

Value for P can be derived from $P = F_{CCO} / (CCLK \times 2)$, using condition that F_{CCO} must be in range of 156 MHz to 320 MHz. Assuming the lowest allowed frequency for $F_{CCO} = 156$ MHz, $P = 156$ MHz / (2 x 60 MHz) = 1.3. The highest F_{CCO} frequency criteria produces $P = 2.67$. The only solution for P that satisfies both of these requirements and is listed in [Table 22](#) is $P = 2$. Therefore, $PLLCFG[6:5] = 1$ will be used.

Example 2: an application using the USB - configuring the PLL1

System design asks for $F_{OSC} = 12$ MHz and requires the USB clock of 48 MHz.

Based on these specifications, $M = 48$ MHz / $Fosc = 48$ MHz / 12 MHz = 4. Consequently, $M - 1 = 3$ will be written as $PLLCFG[4:0]$.

Value for P can be derived from $P = F_{CCO} / (48 \text{ MHz} \times 2)$, using condition that F_{CCO} must be in range of 156 MHz to 320 MHz. Assuming the lowest allowed frequency for $F_{CCO} = 156$ MHz, $P = 156$ MHz / (2 x 48 MHz) = 1.625. The highest F_{CCO} frequency criteria produces $P = 3.33$. Solution for P that satisfy both of these requirements and are listed in [Table 22](#) are $P = 2$ and $P = 3$. Therefore, either of these two values can be used to program $PLLCFG[6:5]$ in the PLL1.

Example 2 has illustrated the way PLL1 should be configured. Since PLL0 and PLL1 are independent, the PLL0 can be configured using the approach described in Example 1.

3.9 Power control

The LPC2141/2/4/6/8 supports two reduced power modes: Idle mode and Power-down mode. 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 power used by the processor itself, memory systems and related controllers, and internal buses.

In Power-down mode, the oscillator is shut down and the chip receives no internal clocks. The processor state and registers, peripheral registers, and internal SRAM values are preserved throughout Power-down mode and the logic levels of chip pins remain static. The Power-down 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, Power-down mode reduces chip power consumption to nearly zero.

Entry to Power-down and Idle modes must be coordinated with program execution. Wakeup from Power-down or Idle modes via an interrupt resumes program execution in such a way that no instructions are lost, incomplete, or repeated. Wake up from Power-down mode is discussed further in [Section 3.12 “Wakeup timer” on page 41](#).

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.

3.9.1 Register description

The Power Control function contains two registers, as shown in [Table 24](#). More detailed descriptions follow.

Table 24: 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 microcontroller. See Table 25 . | R/W | 0x00 | 0xE01F C0C0 |
| 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 | 0x0018 17BE | 0xE01F C0C4 |

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

3.9.2 Power Control register (PCON - 0xE01F COCO)

The PCON register contains two bits. Writing a one to the corresponding bit causes entry to either the Power-down or Idle mode. If both bits are set, Power-down mode is entered.

Table 25: Power Control register (PCON - address 0xE01F COCO) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 0 | IDL | Idle mode - when 1, this bit 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. | 0 |
| 1 | PD | Power-down mode - when 1, this bit causes the oscillator and all on-chip clocks to be stopped. A wakeup condition from an external interrupt can cause the oscillator to restart, the PD bit to be cleared, and the processor to resume execution. IMPORTANT: PD bit can be set to 1 at any time if USBWAKE = 0. In case of USBWAKE = 1, it is possible to set PD to 1 only if USB_need_clock = 0. Having both USBWAKE and USB_need_clock equal 1 prevents the microcontroller from entering Power-down mode. (For additional details see Section 3.5.3 "Interrupt Wakeup register (INTWAKE - 0xE01F C144)" on page 22 and Section 14.7.1 "USB Interrupt Status register (USBIntSt - 0xE01F C1C0)" on page 200) | 0 |
| 2 | PDBOD | When PD is 1 and this bit is 0, Brown Out Detection (BOD) remains operative during Power-down mode, such that its Reset can release the microcontroller from Power-down mode ^[1] . When PD and this bit are both 1, the BOD circuit is disabled during Power-down mode to conserve power. When PD is 0, the state of this bit has no effect. | 0 |
| 3 | BODPDM | When this bit is 1, the BOD circuitry will go into power down mode when chip power down is asserted, resulting in a further reduction in power. However, the possibility of using BOD as a wakeup source from Power Down mode will be lost. When this bit is 0, BOD stays active during Power Down mode. | 0 |
| 4 | BOGD | Brown Out Global Disable. When this bit is 1, the BOD circuitry is fully disabled at all times, and will not consume power. When 0, the BOD circuitry is enabled. | 0 |
| 5 | BORD | Brown Out Reset Disable. When this bit 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. | 0 |
| 7:6 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

[1] Since execution is delayed until after the Wakeup Timer has allowed the main oscillator to resume stable operation, there is no guarantee that execution will resume before V_{DD} has fallen below the lower BOD threshold, which prevents execution. If execution does resume, there is no guarantee of how long the microcontroller will continue execution before the lower BOD threshold terminates execution. These issues depend on the slope of the decline of V_{DD} . High decoupling capacitance (between V_{DD} and ground) in the vicinity of the microcontroller will improve the likelihood that software will be able to do what needs to be done when power is being lost.

3.9.3 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. Each bit in PCONP controls one of the peripherals. The bit numbers correspond to the related peripheral number as shown in the VPB peripheral map [Table 2 "VPB peripheries and base addresses"](#) in the "LPC2141/2/4/6/8 Memory Addressing" chapter.

If a peripheral control bit is 1, that peripheral is enabled. If a peripheral bit is 0, that peripheral is disabled to conserve power. For example if bit 19 is 1, the I²C1 interface is enabled. If bit 19 is 0, the I²C1 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 26: Power Control for Peripherals register (PCONP - address 0xE01F C0C4) 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 | 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 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 7 | PCI2C0 | The I ² C0 interface power/clock control bit. | 1 |
| 8 | PCSPI0 | The SPI0 interface power/clock control bit. | 1 |
| 9 | PCRTC | The RTC power/clock control bit. | 1 |
| 10 | PCSPI1 | The SSP interface power/clock control bit. | 1 |
| 11 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 12 | PCAD0 | A/D converter 0 (ADC0) power/clock control bit. Note: Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN. | 1 |
| 18:13 | - | 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 ² C1 interface power/clock control bit. | 1 |
| 20 | PCAD1 | A/D converter 1 (ADC1) power/clock control bit. Note: Clear the PDN bit in the AD1CR before clearing this bit, and set this bit before setting PDN. | 1 |
| 30:21 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 31 | PUSB | USB power/clock control bit. | 0 |

3.9.4 Power control usage notes

After every reset, the PCONP register contains the value that enables all interfaces and peripherals controlled by the PCONP to be enabled. Therefore, apart from proper configuring via peripheral dedicated registers, the user's application has no need to access the PCONP in order to start using any 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.

3.10 Reset

Reset has two sources on the LPC2141/2/4/6/8: the RESET pin and Watchdog Reset. The RESET pin is a Schmitt trigger input pin with an additional glitch filter. Assertion of chip Reset by any source starts the Wakeup Timer (see description in [Section 3.12 "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 on-chip circuitry has completed its initialization. The relationship between Reset, the oscillator, and the Wakeup Timer are shown in [Figure 10](#).

The Reset glitch filter allows the processor to ignore external reset pulses that are very short, and also determines the minimum duration of RESET that must be asserted in order to guarantee a chip reset. Once asserted, RESET pin can be deasserted only when crystal oscillator is fully running and an adequate signal is present on the X1 pin of the microcontroller. Assuming that an external crystal is used in the crystal oscillator subsystem, after power on, the RESET pin should be asserted for 10 ms. For all subsequent resets when crystal oscillator is already running and stable signal is on the X1 pin, the RESET pin needs to be asserted for 300 ns only.

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.

External and internal Resets have some small differences. An external Reset causes the value of certain pins to be latched to configure the part. External circuitry cannot determine when an internal Reset occurs in order to allow setting up those special pins, so those latches are not reloaded during an internal Reset. Pins that are examined during an external Reset for various purposes are: P1.20/TRACESYNC, P1.26/RTCK (see chapters "Pin Configuration" on page 66 and "Pin Connect Block" on page 75). Pin P0.14 (see "Flash Memory System and Programming" chapter on page 291) is examined by on-chip bootloader when this code is executed after every Reset.

It is possible for a chip Reset to occur during a Flash programming or erase operation. The Flash memory will interrupt the ongoing operation and hold off the completion of Reset to the CPU until internal Flash high voltages have settled.

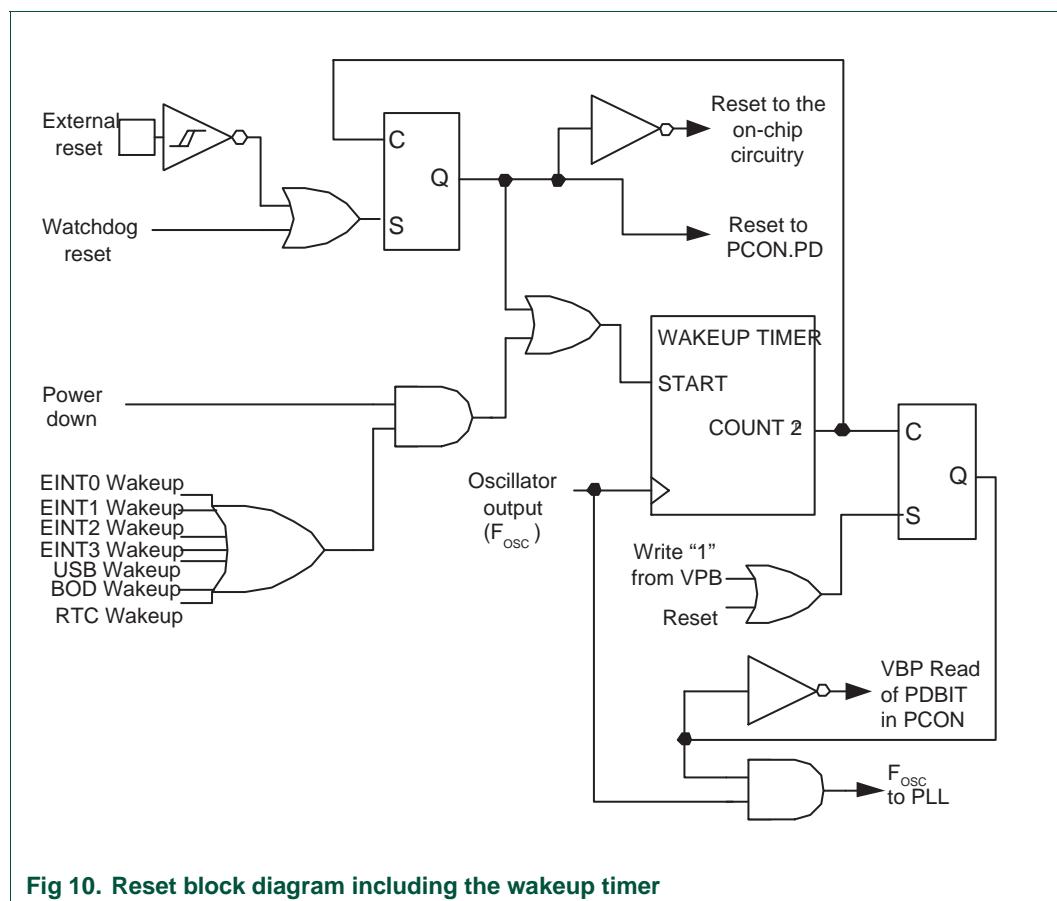


Fig 10. Reset block diagram including the wakeup timer

3.10.1 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 27: Reset Source identification Register (RSIR - address 0xE01F C180) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|---|-------------|
| 0 | POR | Power-On Reset (POR) event 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 <u>RESET</u> signal sets this bit. This bit is cleared by POR, but is not affected by WDT or BOD reset. | see text |

Table 27: Reset Source identification Register (RSIR - 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. Also, if the V_{DD} voltage rises continuously from below 1 V to a level above 2.6 V, the BODR will be set to 1, too. This bit is not affected by External Reset nor Watchdog Reset. Note: only in case a reset occurs and the bit POR = 0, the BODR bit indicates if the V_{DD} 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 |

3.11 VPB divider

The VPB Divider determines the relationship between the processor clock (CCLK) and the clock used by peripheral devices (PCLK). The VPB Divider serves two purposes.

The first is to provide peripherals with desired PCLK via VPB bus so that they can operate at the speed chosen for the ARM processor. In order to achieve this, the VPB bus may be slowed down to one half or one fourth of the processor clock rate. Because the VPB bus must work properly at power up (and its timing cannot be altered if it does not work since the VPB divider control registers reside on the VPB bus), the default condition at reset is for the VPB bus to run at one quarter speed.

The second purpose of the VPB Divider is to allow power savings when an application does not require any peripherals to run at the full processor rate.

The connection of the VPB Divider relative to the oscillator and the processor clock is shown in [Figure 11](#). Because the VPB Divider is connected to the PLL output, the PLL remains active (if it was running) during Idle mode.

3.11.1 Register description

Only one register is used to control the VPB Divider.

Table 28: VPB divider register map

| Name | Description | Access | Reset value ^[1] | Address |
|--------|--|--------|----------------------------|-------------|
| VPBDIV | Controls the rate of the VPB clock in relation to the processor clock. | R/W | 0x00 | 0xE01F C100 |

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

3.11.2 VPBDIV register (VPBDIV - 0xE01F C100)

The VPB Divider register contains two bits, allowing three divider values, as shown in [Table 29](#).

Table 29: VPB Divider register (VPBDIV - address 0xE01F C100) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--------|-------|--|-------------|
| 1:0 | VPBDIV | 00 | VPB bus clock is one fourth of the processor clock. | 00 |
| | | 01 | VPB bus clock is the same as the processor clock. | |
| | | 10 | VPB bus clock is one half of the processor clock. | |
| | | 11 | Reserved. If this value is written to the VPBDIV register, it has no effect (the previous setting is retained). | |
| 7:2 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

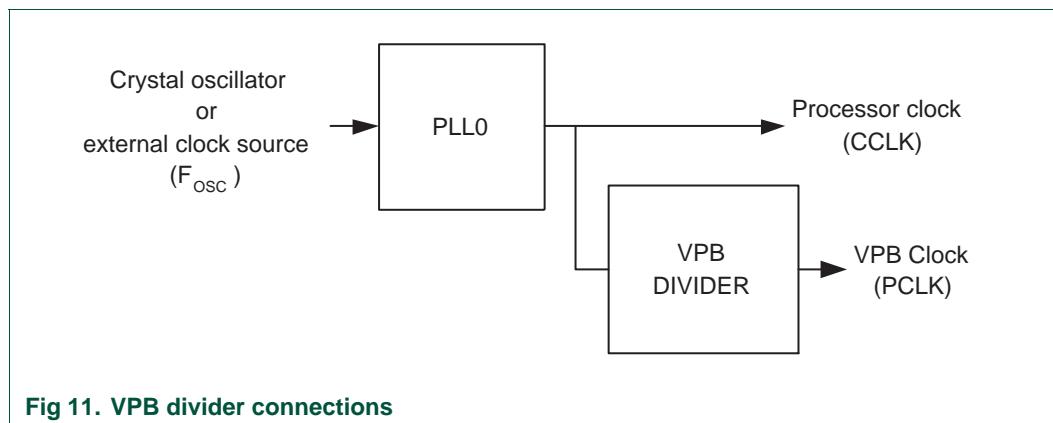


Fig 11. VPB divider connections

3.12 Wakeup timer

The purpose of the wakeup timer is to ensure that the oscillator and other analog functions required for chip operation are fully functional before the processor is allowed 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} 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 4096 clocks, then enables the on-chip circuitry to initialize. When the onboard modules initialization is complete, the processor is released to execute instructions if the external Reset has been deasserted. In the case where an external clock source is used in the system (as opposed to a crystal connected to the oscillator pins), the possibility that there could be little or no delay for oscillator start-up must be considered. The Wakeup Timer design then ensures that any other required chip functions will be operational prior to the beginning of program execution.

Any of the various Resets can bring the microcontroller out of power-down mode, as can the external interrupts EINT3:0, plus the RTC interrupt if the RTC is operating from its own oscillator on the RTCX1-2 pins. When one of these interrupts is enabled for wakeup and its selected event occurs, an oscillator wakeup cycle is started. The actual interrupt (if any) occurs after the wakeup timer expires, and is handled by the Vectored Interrupt Controller.

However, the pin multiplexing on the LPC2141/2/4/6/8 (see chapters "Pin Configuration" on page 66 and "Pin Connect Block" on page 75) was designed to allow other peripherals to, in effect, bring the device out of Power-down mode. The following pin-function pairings allow interrupts from events relating to UART0 or 1, SPI 0 or 1, or the I²C: RxD0 / EINT0, SDA / EINT1, SSEL0 / EINT2, RxD1 / EINT3, DCD1 / EINT1, RI1 / EINT2, SSEL1 / EINT3.

To put the device in Power-down mode and allow activity on one or more of these buses or lines to power it back up, software should reprogram the pin function to External Interrupt, select the appropriate mode and polarity for the Interrupt, and then select Power-down mode. Upon wakeup software should restore the pin multiplexing to the peripheral function.

All of the bus- or line-activity indications in the list above happen to be low-active. If software wants the device to come out of power -down mode in response to activity on more than one pin that share the same EINT_i channel, it should program low-level sensitivity for that channel, because only in level mode will the channel logically OR the signals to wake the device.

The only flaw in this scheme is that the time to restart the oscillator prevents the LPC2141/2/4/6/8 from capturing the bus or line activity that wakes it up. Idle mode is more appropriate than power-down mode for devices that must capture and respond to external activity in a timely manner.

To summarize: on the LPC2141/2/4/6/8, the Wakeup Timer enforces a minimum reset duration based on the crystal oscillator, and is activated whenever there is a wakeup from Power-down mode or any type of Reset.

3.13 Brown-out detection

The LPC2141/2/4/6/8 includes 2-stage monitoring of the voltage on the V_{DD} pins. If this voltage falls below 2.9 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 (see [Section 5.4.4 "Interrupt Enable register \(VICIntEnable - 0xFFFF F010\)" on page 54](#)); if not, software can monitor the signal by reading the Raw Interrupt Status register (see [Section 5.4.3 "Raw Interrupt status register \(VICRawIntr - 0xFFFF F008\)" on page 54](#)).

The second stage of low-voltage detection asserts Reset to inactivate the LPC2141/2/4/6/8 when the voltage on the V_{DD} pins falls below 2.6 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 2.9 V and 2.6 V thresholds include some hysteresis. In normal operation, this hysteresis allows the 2.9 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 LPC2141/2/4/6/8 out of Power-Down mode (which is itself not a guaranteed operation -- see [Section 3.9.2 "Power Control register \(PCON - 0xE01F COCO\)"](#)), 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 RISR being 0. Since all other wakeup conditions have latching flags (see [Section 3.5.2 "External Interrupt Flag register \(EXTINT - 0xE01F C140\)"](#) and [Section 19.4.3 "Interrupt Location Register \(ILR - 0xE002 4000\)" on page 277](#)), a wakeup of this type, without any apparent cause, can be assumed to be a Brown-Out that has gone away.

3.14 Code security vs. debugging

Applications in development typically need the debugging and tracing facilities in the LPC2141/2/4/6/8. 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 LPC2141/2/4/6/8 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 the "Flash Memory System and Programming" chapter on page 291.



4.1 Introduction

The MAM block in the LPC2141/2/4/6/8 maximizes the performance of the ARM processor when it is running code in Flash memory, but does so using a single Flash bank.

4.2 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 LPC2141/2/4/6/8 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.

4.3 MAM 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 12](#) 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.

4.3.1 Flash memory bank

There is one bank of Flash memory with the LPC2141/2/4/6/8 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.

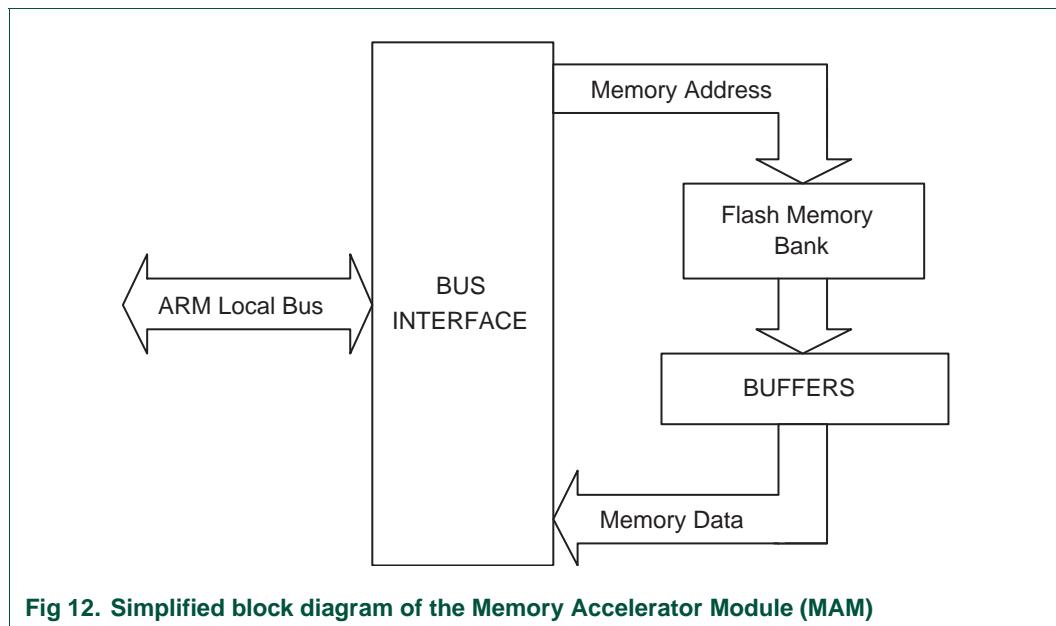


Fig 12. Simplified block diagram of the Memory Accelerator Module (MAM)

4.3.2 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.

Each Data access that is not in the Data latch causes a Flash fetch of 4 words of data, which are captured in the Data latch. This speeds up sequential Data operations, but has little or no effect on random accesses.

4.3.3 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 insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the Flash memory.

In order to preclude the possibility of stale data being read from the Flash memory, the LPC2141/2/4/6/8 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.

4.4 MAM 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 note 2 below). 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 30: 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] |
| 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 31: 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.

4.5 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.

4.6 Register description

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

Table 32: Summary of MAM 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 33 . | 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.

4.7 MAM Control Register (MAMCR - 0xE01F C000)

Two configuration bits select the three MAM operating modes, as shown in [Table 33](#). Following Reset, MAM functions are disabled. Changing the MAM operating mode causes the MAM to invalidate all of the holding latches, resulting in new reads of Flash information as required.

Table 33: MAM Control Register (MAMCR - address 0xE01F C000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------------|-------|--|-------------|
| 1:0 | MAM_mode_control | 00 | MAM functions disabled | 0 |
| | | 01 | MAM functions partially enabled | |
| | | 10 | MAM functions fully enabled | |
| | | 11 | Reserved. Not to be used in the application. | |
| 7:2 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

4.8 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 34: MAM Timing register (MAMTIM - address 0xE01F C004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------------------|-------|---|-------------|
| 2:0 | MAM_fetch_cycle_timing | 000 | 0 - Reserved. | 07 |
| | | 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 | |

Table 34: MAM Timing register (MAMTIM - address 0xE01F C004) bit description

| Bit | Symbol | Value | Description | Reset value |
|---|--------|-------|--|-------------|
| | | 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 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

4.9 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.



5.1 Features

- ARM PrimeCell™ Vectored Interrupt Controller
- 32 interrupt request inputs
- 16 vectored IRQ interrupts
- 16 priority levels dynamically assigned to interrupt requests
- Software interrupt generation

5.2 Description

The Vectored Interrupt Controller (VIC) takes 32 interrupt request inputs and programmably assigns them into 3 categories, FIQ, vectored IRQ, and non-vectored IRQ. 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 IRQs have the middle priority, but only 16 of the 32 requests can be assigned to this category. Any of the 32 requests can be assigned to any of the 16 vectored IRQ slots, among which slot 0 has the highest priority and slot 15 has the lowest.

Non-vectored IRQs have the lowest priority.

The VIC ORs the requests from all the vectored and non-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 there. If any of the vectored IRQs are requesting, the VIC provides the address of the highest-priority requesting IRQs service routine, otherwise it provides the address of a default routine that is shared by all the non-vectored IRQs. The default routine can read another VIC register to see what IRQs are active.

All registers in the VIC are word registers. Byte and halfword reads and write are not supported.

Additional information on the Vectored Interrupt Controller is available in the ARM PrimeCell™ Vectored Interrupt Controller (PL190) documentation.

5.3 Register description

The VIC implements the registers shown in [Table 35](#). More detailed descriptions follow.

Table 35: 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 | 0 | 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 | 0 | 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 | 0 | 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 |
| VICVectAddr | Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read. | R/W | 0 | 0xFFFF F030 |
| VICDefVectAddr | Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectorized IRQs. | R/W | 0 | 0xFFFF F034 |
| VICVectAddr0 | Vector address 0 register. Vector Address Registers 0-15 hold the addresses of the Interrupt Service routines (ISRs) for the 16 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 |

Table 35: VIC register map

| Name | Description | Access | Reset value ^[1] | Address |
|---------------|---|--------|----------------------------|-------------|
| 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 |
| VICVectCtl0 | Vector control 0 register. Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest. | R/W | 0 | 0xFFFF F200 |
| VICVectCtl1 | Vector control 1 register. | R/W | 0 | 0xFFFF F204 |
| VICVectCtl2 | Vector control 2 register. | R/W | 0 | 0xFFFF F208 |
| VICVectCtl3 | Vector control 3 register. | R/W | 0 | 0xFFFF F20C |
| VICVectCtl4 | Vector control 4 register. | R/W | 0 | 0xFFFF F210 |
| VICVectCtl5 | Vector control 5 register. | R/W | 0 | 0xFFFF F214 |
| VICVectCtl6 | Vector control 6 register. | R/W | 0 | 0xFFFF F218 |
| VICVectCtl7 | Vector control 7 register. | R/W | 0 | 0xFFFF F21C |
| VICVectCtl8 | Vector control 8 register. | R/W | 0 | 0xFFFF F220 |
| VICVectCtl9 | Vector control 9 register. | R/W | 0 | 0xFFFF F224 |
| VICVectCtl10 | Vector control 10 register. | R/W | 0 | 0xFFFF F228 |
| VICVectCtl11 | Vector control 11 register. | R/W | 0 | 0xFFFF F22C |
| VICVectCtl12 | Vector control 12 register. | R/W | 0 | 0xFFFF F230 |
| VICVectCtl13 | Vector control 13 register. | R/W | 0 | 0xFFFF F234 |
| VICVectCtl14 | Vector control 14 register. | R/W | 0 | 0xFFFF F238 |
| VICVectCtl15 | Vector control 15 register. | R/W | 0 | 0xFFFF F23C |

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

5.4 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.

5.4.1 Software Interrupt register (VICSoftInt - 0xFFFF F018)

The contents of this register are ORed with the 32 interrupt requests from the various peripherals, before any other logic is applied.

Table 36: Software Interrupt register (VICSoftInt - address 0xFFFF F018) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|
| Symbol | - | - | - | - | - | - | - | - |
| Access | R/W |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Table 37: Software Interrupt register (VICSoftInt - address 0xFFFF F018) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------------------------------------|-------|---|-------------|
| 31:0 | See VICSoftInt 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 5.4.2). | 0 |
| | | 1 | Force the interrupt request with this bit number. | |

5.4.2 Software Interrupt Clear register (VICSoftIntClear - 0xFFFF F01C)

This register allows software to clear one or more bits in the Software Interrupt register, without having to first read it.

Table 38: Software Interrupt Clear register (VICSoftIntClear - address 0xFFFF F01C) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | - | - | - | - | - | - | - |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |

Table 39: Software Interrupt Clear register (VICSoftIntClear - address 0xFFFF F01C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---|-------|--|-------------|
| 31:0 | See VICSoftIntClear 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, thus releasing the forcing of this request. | |

5.4.3 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 40: Raw Interrupt status register (VICRawIntr - address 0xFFFF F008) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | - | - | - | - | - | - | - |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |

Table 41: Raw Interrupt status register (VICRawIntr - address 0xFFFF F008) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------------------------------------|-------|--|-------------|
| 31:0 | See VICRawIntr bit allocation table. | 0 | The interrupt request or software interrupt with this bit number is negated. | 0 |
| | | 1 | The interrupt request or software interrupt with this bit number is negated. | |

5.4.4 Interrupt Enable register (VICIntEnable - 0xFFFF F010)

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

Table 42: Interrupt Enable register (VICIntEnable - address 0xFFFF F010) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | - | - | - | - | - | - | - |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Table 43: Interrupt Enable register (VICIntEnable - address 0xFFFF F010) bit description

| Bit | Symbol | Description | Reset value |
|------|--|--|-------------|
| 31:0 | See VICIntEnable 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 5.4.5 “Interrupt Enable Clear register (VICIntEnClear - 0xFFFF F014)” on page 55 and Table 45 below for how to disable interrupts. | 0 |

5.4.5 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 5.4.4 “Interrupt Enable register \(VICIntEnable - 0xFFFF F010\)” on page 54](#)), without having to first read it.

Table 44: Software Interrupt Clear register (VICIntEnClear - address 0xFFFF F014) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | - | - | - | - | - | - | - |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |

Table 45: Software Interrupt Clear register (VICIntEnClear - address 0xFFFF F014) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---|-------|---|-------------|
| 31:0 | See VICIntEnClear bit allocation table. | 0 | Writing a 0 leaves the corresponding bit in VICIntEnable unchanged. | 0 |

5.4.6 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 46: Interrupt Select register (VICIntSelect - address 0xFFFF F00C) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|
| Symbol | - | - | - | - | - | - | - | - |
| Access | R/W |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Table 47: Interrupt Select register (VICIntSelect - address 0xFFFF F00C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--|-------|---|-------------|
| 31:0 | See VICIntSelect 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. | |

5.4.7 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. It does not differentiate between vectored and non-vectored IRQs.

Table 48: IRQ Status register (VICIRQStatus - address 0xFFFF F000) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | - | - | - | - | - | - | - |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |

Table 49: IRQ Status register (VICIRQStatus - address 0xFFFF F000) bit description

| Bit | Symbol | Description | Reset value |
|------|--|--|-------------|
| 31:0 | See VICIRQStatus bit allocation table. | A bit read as 1 indicates a corresponding interrupt request being enabled, classified as IRQ, and asserted | 0 |

5.4.8 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 50: FIQ Status register (VICFIQStatus - address 0xFFFF F004) bit allocation

Reset value: 0x0000 0000

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------|-------|-------|--------|--------|----------|----------|-------|-------|
| Symbol | - | - | - | - | - | - | - | - |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Symbol | - | USB | AD1 | BOD | I2C1 | AD0 | EINT3 | EINT2 |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Symbol | EINT1 | EINT0 | RTC | PLL | SPI1/SSP | SPI0 | I2C0 | PWM0 |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | UART1 | UART0 | TIMER1 | TIMER0 | ARMCore1 | ARMCore0 | - | WDT |
| Access | RO | RO | RO | RO | RO | RO | RO | RO |

Table 51: FIQ Status register (VICFIQStatus - address 0xFFFF F004) bit description

| Bit | Symbol | Description | Reset value |
|------|--|--|-------------|
| 31:0 | See VICFIQStatus bit allocation table. | A bit read as 1 indicates a corresponding interrupt request being enabled, classified as FIQ, and asserted | 0 |

5.4.9 Vector Control registers 0-15 (VICVectCtl0-15 - 0xFFFF F200-23C)

These are a read/write accessible registers. Each of these registers controls one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest. Note that disabling a vectored IRQ slot in one of the VICVectCtl registers does not disable the interrupt itself, the interrupt is simply changed to the non-vectored form.

Table 52: Vector Control registers 0-15 (VICVectCtl0-15 - 0xFFFF F200-23C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------------------------|---|-------------|
| 4:0 | int_request/sw_int_assig | The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted. | 0 |
| 5 | IRQslot_en | When 1, this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted. | 0 |
| 31:6 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

5.4.10 Vector Address registers 0-15 (VICVectAddr0-15 - 0xFFFF F100-13C)

These are a read/write accessible registers. These registers hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.

Table 53: Vector Address registers (VICVectAddr0-15 - addresses 0xFFFF F100-13C) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|--|-------------|
| 31:0 | IRQ_vector | When one or more interrupt request or software interrupt is (are) enabled, classified as IRQ, asserted, and assigned to an enabled vectored IRQ slot, the value from this register for the highest-priority such slot will be provided when the IRQ service routine reads the Vector Address register -VICVectAddr (Section 5.4.10). | 0x0000 0000 |

5.4.11 Default Vector Address register (VICDefVectAddr - 0xFFFF F034)

This is a read/write accessible register. This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.

Table 54: Default Vector Address register (VICDefVectAddr - address 0xFFFF F034) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|---|-------------|
| 31:0 | IRQ_vector | When an IRQ service routine reads the Vector Address register (VICVectAddr), and no IRQ slot responds as described above, this address is returned. | 0x0000 0000 |

5.4.12 Vector Address register (VICVectAddr - 0xFFFF F030)

This is a read/write accessible register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.

Table 55: Vector Address register (VICVectAddr - address 0xFFFF F030) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|--|-------------|
| 31:0 | IRQ_vector | If any of the interrupt requests or software interrupts that are assigned to a vectored IRQ slot is (are) enabled, classified as IRQ, and asserted, reading from this register returns the address in the Vector Address Register for the highest-priority such slot (lowest-numbered) such slot. Otherwise it returns the address in the Default Vector Address Register. Writing to this register does not set the value for future reads from it. Rather, this register should be written near the end of an ISR, to update the priority hardware. | 0x0000 0000 |

5.4.13 Protection Enable register (VICProtection - 0xFFFF F020)

This is a read/write accessible register. It controls access to the VIC registers by software running in User mode.

Table 56: 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 NA value read from a reserved bit is not defined. | |

5.5 Interrupt sources

[Table 57](#) lists the interrupt sources for each peripheral function. Each peripheral device has one interrupt line connected to the Vectored Interrupt Controller, but may have several internal interrupt flags. Individual interrupt flags may also represent more than one interrupt source.

Table 57: Connection of interrupt sources to the Vectored Interrupt Controller (VIC)

| 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 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3) | 4 | 0x0000 0010 |
| TIMER1 | Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3) | 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 |
| UART1 | Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI) [1] | 7 | 0x0000 0080 |
| PWM0 | Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6) | 8 | 0x0000 0100 |
| I ² C0 | SI (state change) | 9 | 0x0000 0200 |
| SPI0 | SPI Interrupt Flag (SPIF) Mode Fault (MODF) | 10 | 0x0000 0400 |
| SPI1 (SSP) | TX FIFO at least half empty (TXRIS) Rx FIFO at least half full (RXRIS) Receive Timeout condition (RTRIS) Receive overrun (RORRIS) | 11 | 0x0000 0800 |
| PLL | PLL Lock (PLOCK) | 12 | 0x0000 1000 |
| RTC | Counter Increment (RTCCIF) Alarm (RTCALF) | 13 | 0x0000 2000 |
| System Control | External Interrupt 0 (EINT0) External Interrupt 1 (EINT1) External Interrupt 2 (EINT2) External Interrupt 3 (EINT3) | 14 15 16 17 | 0x0000 4000 0x0000 8000 0x0001 0000 0x0002 0000 |
| ADC0 | A/D Converter 0 end of conversion | 18 | 0x0004 0000 |
| I ² C1 | SI (state change) | 19 | 0x0008 0000 |

Table 57: Connection of interrupt sources to the Vectored Interrupt Controller (VIC)

| Block | Flag(s) | VIC Channel # and Hex Mask | |
|-------|--|----------------------------|-------------|
| BOD | Brown Out detect | 20 | 0x0010 0000 |
| ADC1 | A/D Converter 1 end of conversion ^[1] | 21 | 0x0020 0000 |
| USB | USB interrupts, DMA interrupt ^[1] | 22 | 0x0040 0000 |

[1] LPC2144/6/8 Only.

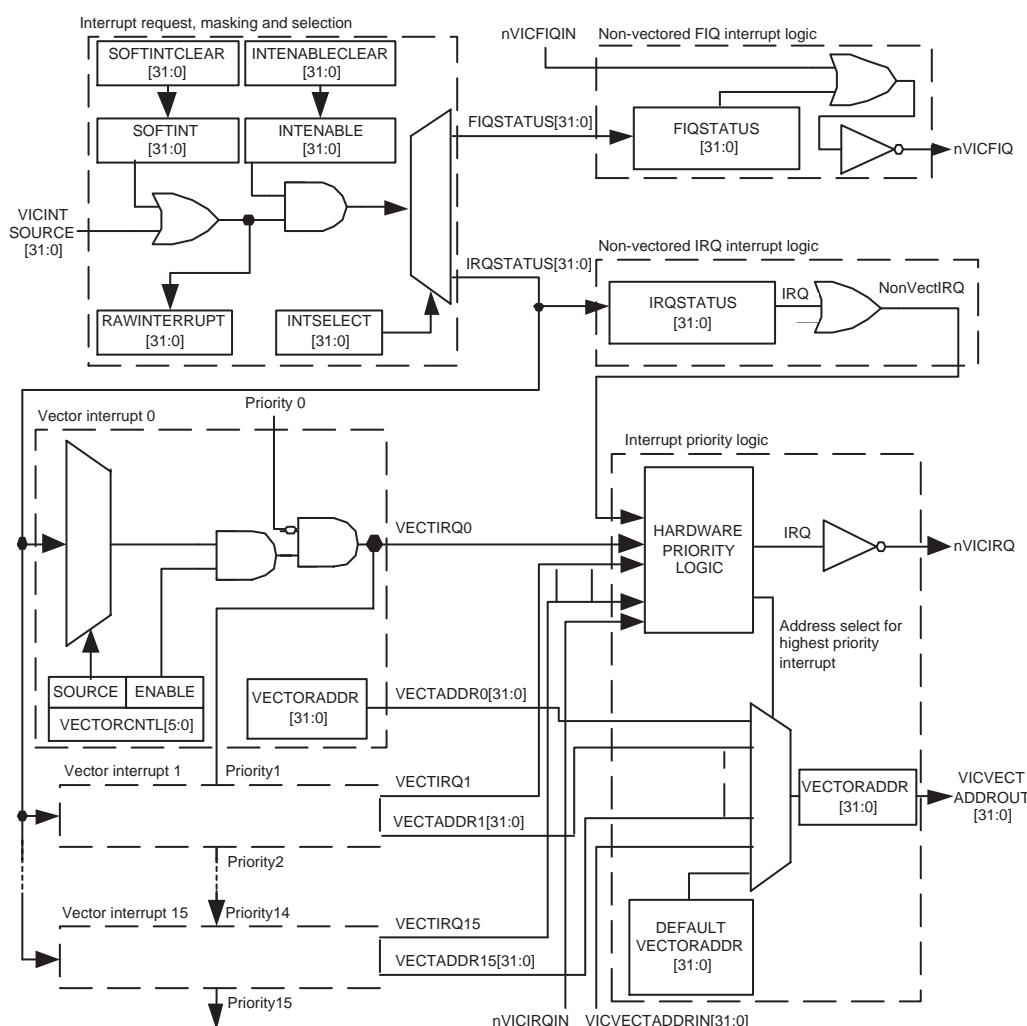


Fig 13. Block diagram of the Vectored Interrupt Controller (VIC)

5.6 Spurious interrupts

Spurious interrupts are possible in the ARM7TDMI based microcontrollers such as the LPC2141/2/4/6/8 due to asynchronous interrupt handling. The asynchronous character of the interrupt processing has its roots in the interaction of the core and the VIC. If the VIC state is changed between the moments when the core detects an interrupt, and the core actually processes an interrupt, problems may be generated.

Real-life applications may experience the following scenarios:

1. VIC decides there is an IRQ interrupt and sends the IRQ signal to the core.
2. Core latches the IRQ state.
3. Processing continues for a few cycles due to pipelining.
4. Core loads IRQ address from VIC.

Furthermore, It is possible that the VIC state has changed during step 3. For example, VIC was modified so that the interrupt that triggered the sequence starting with step 1) is no longer pending -interrupt got disabled in the executed code. In this case, the VIC will not be able to clearly identify the interrupt that generated the interrupt request, and as a result the VIC will return the default interrupt VicDefVectAddr (0xFFFF F034).

This potentially disastrous chain of events can be prevented in two ways:

1. Application code should be set up in a way to prevent the spurious interrupts from occurring. Simple guarding of changes to the VIC may not be enough since, for example, glitches on level sensitive interrupts can also cause spurious interrupts.
2. VIC default handler should be set up and tested properly.

5.6.1 Details and case studies on spurious interrupts

This chapter contains details that can be obtained from the official ARM website (<http://www.arm.com>), FAQ section under the "Technical Support" link: <http://www.arm.com/support/faqip/3677.html>.

What happens if an interrupt occurs as it is being disabled?

Applies to: ARM7TDMI

If an interrupt is received by the core during execution of an instruction that disables interrupts, the ARM7 family will still take the interrupt. This occurs for both IRQ and FIQ interrupts.

For example, consider the following instruction sequence:

```
MRS r0, cpsr  
ORR r0, r0, #I_Bit:OR:F_Bit ; disable IRQ and FIQ interrupts  
MSR cpsr_c, r0
```

If an IRQ interrupt is received during execution of the MSR instruction, then the behavior will be as follows:

- The IRQ interrupt is latched.

- The MSR cpsr, r0 executes to completion setting both the I bit and the F bit in the CPSR.
- The IRQ interrupt is taken because the core was committed to taking the interrupt exception before the I bit was set in the CPSR.
- The CPSR (with the I bit and F bit set) is moved to the SPSR_IRQ.

This means that, on entry to the IRQ interrupt service routine, you can see the unusual effect that an IRQ interrupt has just been taken while the I bit in the SPSR is set. In the example above, the F bit will also be set in both the CPSR and SPSR. This means that FIQs are disabled upon entry to the IRQ service routine, and will remain so until explicitly re-enabled. FIQs will not be reenabled automatically by the IRQ return sequence.

Although the example shows both IRQ and FIQ interrupts being disabled, similar behavior occurs when only one of the two interrupt types is being disabled. The fact that the core processes the IRQ after completion of the MSR instruction which disables IRQs does not normally cause a problem, since an interrupt arriving just one cycle earlier would be expected to be taken. When the interrupt routine returns with an instruction like:

```
SUBS pc, lr, #4
```

the SPSR_IRQ is restored to the CPSR. The CPSR will now have the I bit and F bit set, and therefore execution will continue with all interrupts disabled. However, this can cause problems in the following cases:

Problem 1: A particular routine maybe called as an IRQ handler, or as a regular subroutine. In the latter case, the system guarantees that IRQs would have been disabled prior to the routine being called. The routine exploits this restriction to determine how it was called (by examining the I bit of the SPSR), and returns using the appropriate instruction. If the routine is entered due to an IRQ being received during execution of the MSR instruction which disables IRQs, then the I bit in the SPSR will be set. The routine would therefore assume that it could not have been entered via an IRQ.

Problem 2: FIQs and IRQs are both disabled by the same write to the CPSR. In this case, if an IRQ is received during the CPSR write, FIQs will be disabled for the execution time of the IRQ handler. This may not be acceptable in a system where FIQs must not be disabled for more than a few cycles.

5.6.2 Workaround

There are 3 suggested workarounds. Which of these is most applicable will depend upon the requirements of the particular system.

5.6.3 Solution 1: test for an IRQ received during a write to disable IRQs

Add code similar to the following at the start of the interrupt routine.

```
SUB      lr, lr, #4      ; Adjust LR to point to return
STMFD   sp!, {..., lr}  ; Get some free regs
MRS     lr, SPSR        ; See if we got an interrupt while
TST     lr, #I_Bit       ; interrupts were disabled.
LDMNEFD sp!, {..., pc}^ ; If so, just return immediately.
                           ; The interrupt will remain pending since we haven't
                           ; acknowledged it and will be reissued when interrupts
```

```
; are next enabled.  
; Rest of interrupt routine
```

This code will test for the situation where the IRQ was received during a write to disable IRQs. If this is the case, the code returns immediately - resulting in the IRQ not being acknowledged (cleared), and further IRQs being disabled.

Similar code may also be applied to the FIQ handler, in order to resolve the first issue.

This is the recommended workaround, as it overcomes both problems mentioned above. However, in the case of problem two, it does add several cycles to the maximum length of time FIQs will be disabled.

5.6.4 Solution 2: disable IRQs and FIQs using separate writes to the CPSR

```
MRS r0, cpsr  
ORR r0, r0, #I_Bit ;disable IRQs  
MSR cpsr_c, r0  
ORR r0, r0, #F_Bit ;disable FIQs  
MSR cpsr_c, r0
```

This is the best workaround where the maximum time for which FIQs are disabled is critical (it does not increase this time at all). However, it does not solve problem one, and requires extra instructions at every point where IRQs and FIQs are disabled together.

5.6.5 Solution 3: re-enable FIQs at the beginning of the IRQ handler

As the required state of all bits in the c field of the CPSR are known, this can be most efficiently be achieved by writing an immediate value to CPSR_C, for example:

```
MSR cpsr_c, #I_Bit:OR:irq_MODE ;IRQ should be disabled  
;FIQ enabled  
;ARM state, IRQ mode
```

This requires only the IRQ handler to be modified, and FIQs may be re-enabled more quickly than by using workaround 1. However, this should only be used if the system can guarantee that FIQs are never disabled while IRQs are enabled. It does not address problem one.

5.7 VIC usage notes

If user code is running from an on-chip RAM and an application uses interrupts, interrupt vectors must be re-mapped to on-chip address 0x0. This is necessary because all the exception vectors are located at addresses 0x0 and above. This is easily achieved by configuring the MEMMAP register (see [Section 3.7.1 “Memory Mapping control register \(MEMMAP - 0xE01F C040\)” on page 26](#)) to User RAM mode. Application code should be linked such that at 0x4000 0000 the Interrupt Vector Table (IVT) will reside.

Although multiple sources can be selected (VICIntSelect) to generate FIQ request, only one interrupt service routine should be dedicated to service all available/present FIQ request(s). Therefore, if more than one interrupt sources are classified as FIQ the FIQ interrupt service routine must read VICFIQStatus to decide based on this content what to

do and how to process the interrupt request. However, it is recommended that only one interrupt source should be classified as FIQ. Classifying more than one interrupt sources as FIQ will increase the interrupt latency.

Following the completion of the desired interrupt service routine, clearing of the interrupt flag on the peripheral level will propagate to corresponding bits in VIC registers (VICRawIntr, VICFIQStatus and VICIRQStatus). Also, before the next interrupt can be serviced, it is necessary that write is performed into the VICVectAddr register before the return from interrupt is executed. This write will clear the respective interrupt flag in the internal interrupt priority hardware.

In order to disable the interrupt at the VIC you need to clear corresponding bit in the VICIntEnClr register, which in turn clears the related bit in the VICIntEnable register. This also applies to the VICSoftInt and VICSoftIntClear in which VICSoftIntClear will clear the respective bits in VICSoftInt. For example, if VICSoftInt = 0x0000 0005 and bit 0 has to be cleared, VICSoftIntClear = 0x0000 0001 will accomplish this. Before the new clear operation on the same bit in VICSoftInt using writing into VICSoftIntClear is performed in the future, VICSoftIntClear = 0x0000 0000 must be assigned. Therefore writing 1 to any bit in Clear register will have one-time-effect in the destination register.

If the watchdog is enabled for interrupt on underflow or invalid feed sequence only then there is no way of clearing the interrupt. The only way you could perform return from interrupt is by disabling the interrupt at the VIC (using VICIntEnClr).

Example:

Assuming that UART0 and SPI0 are generating interrupt requests that are classified as vectored IRQs (UART0 being on the higher level than SPI0), while UART1 and I²C are generating non-vectored IRQs, the following could be one possibility for VIC setup:

```

VICIntSelect = 0x0000 0000      ; SPI0, I2C, UART1 and UART0 are IRQ =>
                                ; bit10, bit9, bit7 and bit6=0
VICIntEnable = 0x0000 06C0      ; SPI0, I2C, UART1 and UART0 are enabled interrupts =>
                                ; bit10, bit9, bit 7 and bit6=1
VICDefVectAddr = 0x...          ; holds address at what routine for servicing
                                ; non-vectored IRQs (i.e. UART1 and I2C) starts
VICVectAddr0 = 0x...            ; holds address where UART0 IRQ service routine starts
VICVectAddr1 = 0x...            ; holds address where SPI0 IRQ service routine starts
VICVectCnt10 = 0x0000 0026      ; interrupt source with index 6 (UART0) is enabled as
                                ; the one with priority 0 (the highest)
VICVectCnt11 = 0x0000 002A      ; interrupt source with index 10 (SPI0) is enabled
                                ; as the one with priority 1

```

After any of IRQ requests (SPI0, I²C, UART0 or UART1) is made, microcontroller will redirect code execution to the address specified at location 0x0000 0018. For vectored and non-vectored IRQ's the following instruction could be placed at 0x0000 0018:

```
LDR pc, [pc,#-0xFF0]
```

This instruction loads PC with the address that is present in VICVectAddr register.

In case UART0 request has been made, VICVectAddr will be identical to VICVectAddr0, while in case SPI0 request has been made value from VICVectAddr1 will be found here. If neither UART0 nor SPI0 have generated IRQ request but UART1 and/or I²C were the reason, content of VICVectAddr will be identical to VICDefVectAddr.

6.1 LPC2141/2142/2144/2146/2148 pinout

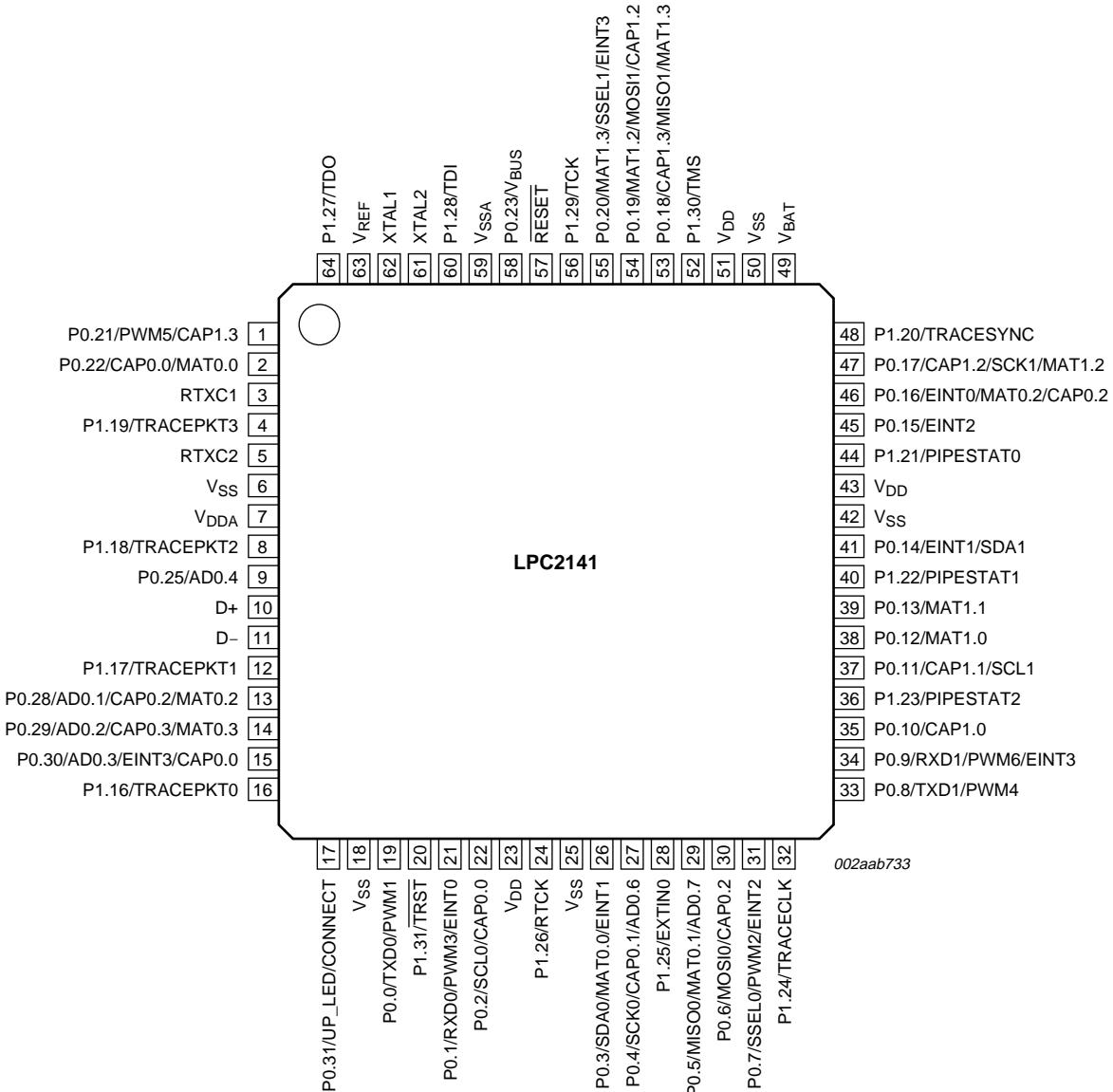


Fig 14. LPC2141 64-pin package

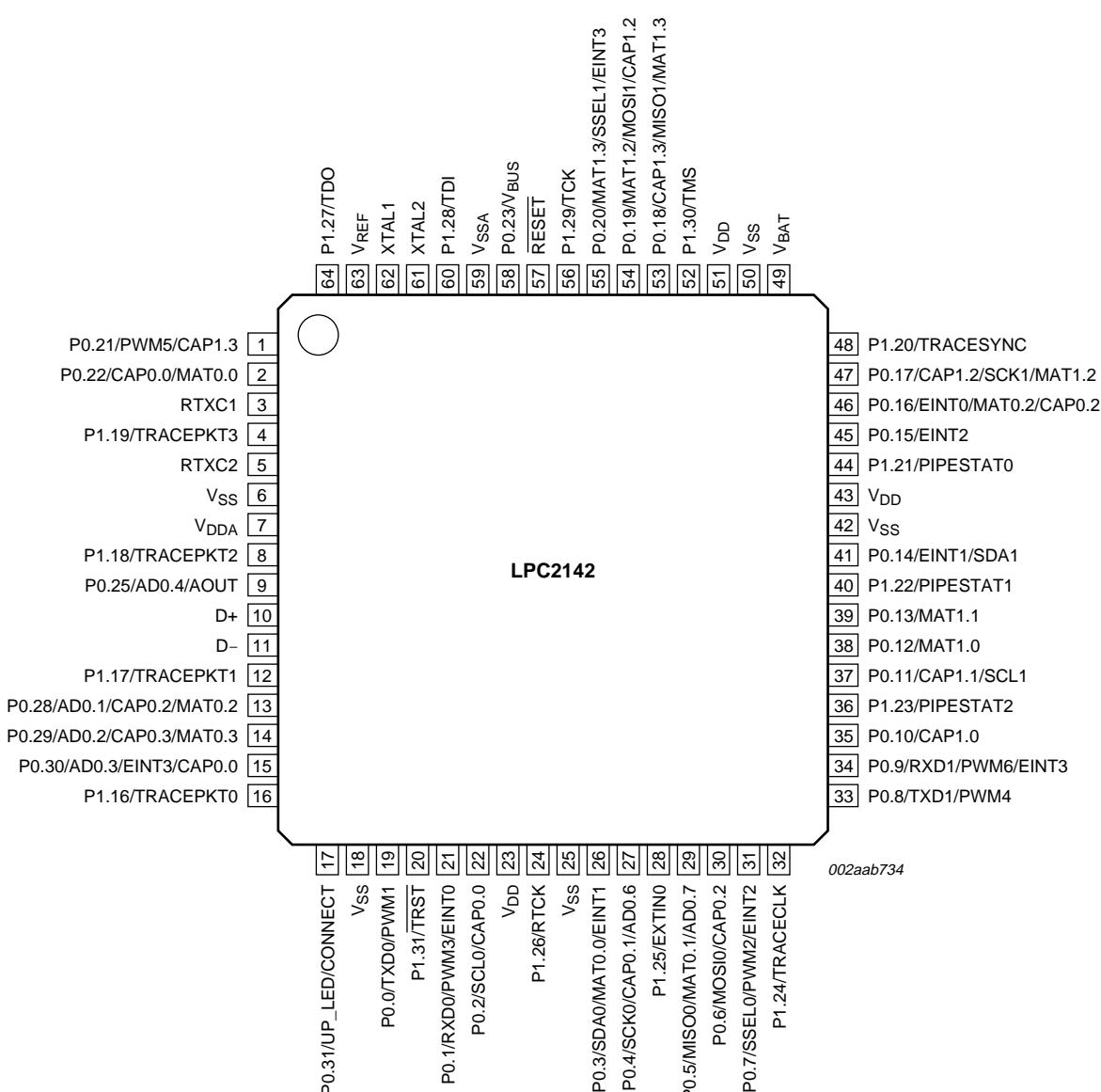


Fig 15. LPC2142 64-pin package

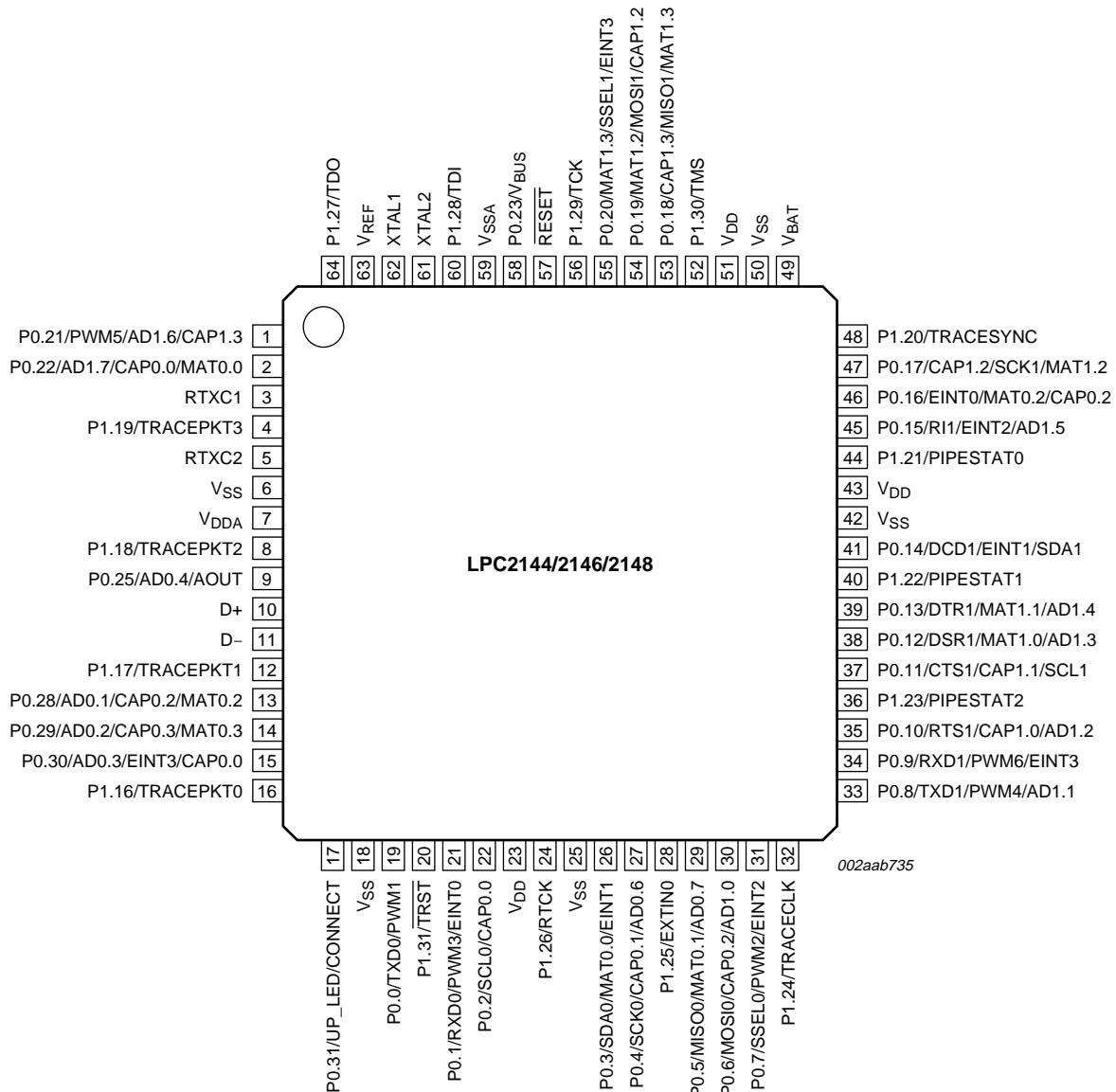


Fig 16. LPC2144/6/8 64-pin package

6.2 Pin description for LPC2141/2/4/6/8

Pin description for LPC2141/2/4/6/8 and a brief explanation of corresponding functions are shown in the following table.

Table 58: Pin description

| Symbol | Pin | 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. Total of 28 pins of the Port 0 can be used as a general purpose bi-directional digital I/Os while P0.31 provides digital output functions only. The operation of port 0 pins depends upon the pin function selected via the pin connect block. Pins P0.24, P0.26 and P0.27 are not available. |
| P0.0/TXD0/ PWM1 | 19 ^[1] | I/O | P0.0 — General purpose digital input/output pin |
| | | O | TXD0 — Transmitter output for UART0 |
| | | O | PWM1 — Pulse Width Modulator output 1 |
| P0.1/RxD0/ PWM3/EINT0 | 21 ^[2] | I/O | P0.1 — General purpose digital input/output pin |
| | | I | RxD0 — Receiver input for UART0 |
| | | O | PWM3 — Pulse Width Modulator output 3 |
| | | I | EINT0 — External interrupt 0 input |
| P0.2/SCL0/ CAP0.0 | 22 ^[3] | I/O | P0.2 — General purpose digital input/output pin |
| | | I/O | SCL0 — I ² C0 clock input/output. Open drain output (for I ² C compliance) |
| | | I | CAP0.0 — Capture input for Timer 0, channel 0 |
| P0.3/SDA0/ MAT0.0/EINT1 | 26 ^[3] | I/O | P0.3 — General purpose digital input/output pin |
| | | I/O | SDA0 — I ² C0 data input/output. Open drain output (for I ² C compliance) |
| | | O | MAT0.0 — Match output for Timer 0, channel 0 |
| | | I | EINT1 — External interrupt 1 input |
| P0.4/SCK0/ CAP0.1/AD0.6 | 27 ^[4] | I/O | P0.4 — General purpose digital input/output pin |
| | | I/O | SCK0 — Serial clock for SPI0. SPI clock output from master or input to slave |
| | | I | CAP0.1 — Capture input for Timer 0, channel 0 |
| | | I | AD0.6 — A/D converter 0, input 6. This analog input is always connected to its pin |
| P0.5/MISO0/ MAT0.1/AD0.7 | 29 ^[4] | I/O | P0.5 — General purpose digital input/output pin |
| | | I/O | MISO0 — Master In Slave OUT for SPI0. Data input to SPI master or data output from SPI slave |
| | | O | MAT0.1 — Match output for Timer 0, channel 1 |
| | | I | AD0.7 — A/D converter 0, input 7. This analog input is always connected to its pin |
| P0.6/MOSI0/ CAP0.2/AD1.0 | 30 ^[4] | I/O | P0.6 — General purpose digital input/output pin |
| | | I/O | MOSI0 — Master Out Slave In for SPI0. Data output from SPI master or data input to SPI slave |
| | | I | CAP0.2 — Capture input for Timer 0, channel 2 |
| | | I | AD1.0 — A/D converter 1, input 0. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |
| P0.7/SSEL0/ PWM2/EINT2 | 31 ^[2] | I/O | P0.7 — General purpose digital input/output pin |
| | | I | SSEL0 — Slave Select for SPI0. Selects the SPI interface as a slave |
| | | O | PWM2 — Pulse Width Modulator output 2 |
| | | I | EINT2 — External interrupt 2 input |

Table 58: Pin description ...continued

| Symbol | Pin | Type | Description |
|-----------------------------|-------------------|------|--|
| P0.8/TXD1/ PWM4/AD1.1 | 33 ^[4] | I/O | P0.8 — General purpose digital input/output pin |
| | | O | TXD1 — Transmitter output for UART1 |
| | | O | PWM4 — Pulse Width Modulator output 4 |
| | | I | AD1.1 — A/D converter 1, input 1. This analog input is always connected to its pin. Available in LPC2144/6/8 only |
| P0.9/RxD1/ PWM6/EINT3 | 34 ^[2] | I/O | P0.9 — General purpose digital input/output pin |
| | | I | RxD1 — Receiver input for UART1 |
| | | O | PWM6 — Pulse Width Modulator output 6 |
| | | I | EINT3 — External interrupt 3 input |
| P0.10/RTS1/ CAP1.0/AD1.2 | 35 ^[4] | I/O | P0.10 — General purpose digital input/output pin |
| | | O | RTS1 — Request to Send output for UART1. Available in LPC2144/6/8 only. |
| | | I | CAP1.0 — Capture input for Timer 1, channel 0 |
| | | I | AD1.2 — A/D converter 1, input 2. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |
| P0.11/CTS1/ CAP1.1/SCL1 | 37 ^[3] | I/O | P0.11 — General purpose digital input/output pin |
| | | I | CTS1 — Clear to Send input for UART1. Available in LPC2144/6/8 only. |
| | | I | CAP1.1 — Capture input for Timer 1, channel 1. |
| | | I/O | SCL1 — I ² C1 clock input/output. Open drain output (for I ² C compliance) |
| P0.12/DSR1/ MAT1.0/AD1.3 | 38 ^[4] | I/O | P0.12 — General purpose digital input/output pin |
| | | I | DSR1 — Data Set Ready input for UART1. Available in LPC2144/6/8 only. |
| | | O | MAT1.0 — Match output for Timer 1, channel 0. |
| | | I | AD1.3 — A/D converter input 3. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |
| P0.13/DTR1/ MAT1.1/AD1.4 | 39 ^[4] | I/O | P0.13 — General purpose digital input/output pin |
| | | O | DTR1 — Data Terminal Ready output for UART1. Available in LPC2144/6/8 only. |
| | | O | MAT1.1 — Match output for Timer 1, channel 1. |
| | | I | AD1.4 — A/D converter input 4. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |
| P0.14/DCD1/ EINT1/SDA1 | 41 ^[3] | I/O | P0.14 — General purpose digital input/output pin |
| | | I | DCD1 — Data Carrier Detect input for UART1. Available in LPC2144/6/8 only. |
| | | I | EINT1 — External interrupt 1 input |
| | | I/O | SDA1 — I ² C1 data input/output. Open drain output (for I ² C compliance) Note: LOW on this pin while RESET is LOW forces on-chip boot-loader to take over control of the part after reset. |
| P0.15/RI1/ EINT2/AD1.5 | 45 ^[4] | I/O | P0.15 — General purpose digital input/output pin |
| | | I | RI1 — Ring Indicator input for UART1. Available in LPC2144/6/8 only. |
| | | I | EINT2 — External interrupt 2 input. |
| | | I | AD1.5 — A/D converter 1, input 5. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |

Table 58: Pin description ...continued

| Symbol | Pin | Type | Description |
|-------------------------------|-------------------|------|---|
| P0.16/EINT0/ MAT0.2/CAP0.2 | 46 ^[2] | I/O | P0.16 — General purpose digital input/output pin |
| | | I | EINT0 — External interrupt 0 input. |
| | | O | MAT0.2 — Match output for Timer 0, channel 2. |
| | | I | CAP0.2 — Capture input for Timer 0, channel 2. |
| P0.17/CAP1.2/ SCK1/MAT1.2 | 47 ^[1] | I/O | P0.17 — General purpose digital input/output pin |
| | | I | CAP1.2 — Capture input for Timer 1, channel 2. |
| | | I/O | SCK1 — Serial Clock for SSP. Clock output from master or input to slave. |
| | | O | MAT1.2 — Match output for Timer 1, channel 2. |
| P0.18/CAP1.3/ MISO1/MAT1.3 | 53 ^[1] | I/O | P0.18 — General purpose digital input/output pin |
| | | I | CAP1.3 — Capture input for Timer 1, channel 3. |
| | | I/O | MISO1 — Master In Slave Out for SSP. Data input to SPI master or data output from SSP slave. |
| | | O | MAT1.3 — Match output for Timer 1, channel 3. |
| P0.19/MAT1.2/ MOSI1/CAP1.2 | 54 ^[1] | I/O | P0.19 — General purpose digital input/output pin |
| | | O | MAT1.2 — Match output for Timer 1, channel 2. |
| | | I/O | MOSI1 — Master Out Slave In for SSP. Data output from SSP master or data input to SSP slave. |
| | | I | CAP1.2 — Capture input for Timer 1, channel 2. |
| P0.20/MAT1.3/ SSEL1/EINT3 | 55 ^[2] | I/O | P0.20 — General purpose digital input/output pin |
| | | O | MAT1.3 — Match output for Timer 1, channel 3. |
| | | I | SSEL1 — Slave Select for SSP. Selects the SSP interface as a slave. |
| | | I | EINT3 — External interrupt 3 input. |
| P0.21/PWM5/ AD1.6/CAP1.3 | 1 ^[4] | I/O | P0.21 — General purpose digital input/output pin |
| | | O | PWM5 — Pulse Width Modulator output 5. |
| | | I | AD1.6 — A/D converter 1, input 6. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |
| | | I | CAP1.3 — Capture input for Timer 1, channel 3. |
| P0.22/AD1.7/ CAP0.0/MAT0.0 | 2 ^[4] | I/O | P0.22 — General purpose digital input/output pin. |
| | | I | AD1.7 — A/D converter 1, input 7. This analog input is always connected to its pin. Available in LPC2144/6/8 only. |
| | | I | CAP0.0 — Capture input for Timer 0, channel 0. |
| | | O | MAT0.0 — Match output for Timer 0, channel 0. |
| P0.23 | 58 ^[1] | I/O | P0.23 — General purpose digital input/output pin. |
| | | I | V_{BUS} — Indicates the presence of USB bus power. |
| P0.25/AD0.4/ Aout | 9 ^[5] | I/O | P0.25 — General purpose digital input/output pin |
| | | I | AD0.4 — A/D converter 0, input 4. This analog input is always connected to its pin. |
| | | O | Aout — D/A converter output. Available in LPC2142/4/6/8 only. |

Table 58: Pin description ...continued

| Symbol | Pin | Type | Description |
|-------------------------------|-------------------|------|--|
| P0.28/AD0.1/ CAP0.2/MAT0.2 | 13 ^[4] | I/O | P0.28 — General purpose digital input/output pin |
| | | I | AD0.1 — A/D converter 0, input 1. This analog input is always connected to its pin. |
| | | I | CAP0.2 — Capture input for Timer 0, channel 2. |
| | | O | MAT0.2 — Match output for Timer 0, channel 2. |
| P0.29/AD0.2/ CAP0.3/MAT0.3 | 14 ^[4] | I/O | P0.29 — General purpose digital input/output pin |
| | | I | AD0.2 — A/D converter 0, input 2. This analog input is always connected to its pin. |
| | | I | CAP0.3 — Capture input for Timer 0, Channel 3. |
| | | O | MAT0.3 — Match output for Timer 0, channel 3. |
| P0.30/AD0.3/ EINT3/CAP0.0 | 15 ^[4] | I/O | P0.30 — General purpose digital input/output pin. |
| | | I | AD0.3 — A/D converter 0, input 3. This analog input is always connected to its pin. |
| | | I | EINT3 — External interrupt 3 input. |
| | | I | CAP0.0 — Capture input for Timer 0, channel 0. |
| P0.31 | 17 ^[6] | O | P0.31 — General purpose output only digital pin (GPO). |
| | | O | UP_LED — USB Good Link 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 | CONNECT — Signal used to switch an external 1.5 kΩ resistor under the software control (active state for this signal is LOW). Used with the Soft Connect USB feature. Note: This pin MUST NOT be externally pulled LOW when RESET pin is LOW or the JTAG port will be disabled. |
| P1.0 to P1.31 | | I/O | Port 1: Port 1 is a 32-bit bi-directional 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. Pins 0 through 15 of port 1 are not available. |
| P1.16/ TRACEPKT0 | 16 ^[6] | I/O | P1.16 — General purpose digital input/output pin |
| | | O | TRACEPKT0 — Trace Packet, bit 0. Standard I/O port with internal pull-up. |
| P1.17/ TRACEPKT1 | 12 ^[6] | I/O | P1.17 — General purpose digital input/output pin |
| | | O | TRACEPKT1 — Trace Packet, bit 1. Standard I/O port with internal pull-up. |
| P1.18/ TRACEPKT2 | 8 ^[6] | I/O | P1.18 — General purpose digital input/output pin |
| | | O | TRACEPKT2 — Trace Packet, bit 2. Standard I/O port with internal pull-up. |
| P1.19/ TRACEPKT3 | 4 ^[6] | I/O | P1.19 — General purpose digital input/output pin |
| | | O | TRACEPKT3 — Trace Packet, bit 3. Standard I/O port with internal pull-up. |
| P1.20/ TRACESYNC | 48 ^[6] | I/O | P1.20 — General purpose digital input/output pin |
| | | O | TRACESYNC — Trace Synchronization. Standard I/O port with internal pull-up. Note: LOW on this pin while RESET is LOW enables pins P1.25:16 to operate as Trace port after reset |
| P1.21/ PIPESTAT0 | 44 ^[6] | I/O | P1.21 — General purpose digital input/output pin |
| | | O | PIPESTAT0 — Pipeline Status, bit 0. Standard I/O port with internal pull-up. |

Table 58: Pin description ...continued

| Symbol | Pin | Type | Description |
|---------------------------------|----------------------|------|--|
| P1.22/ PIPESTAT1 | 40 ^[6] | I/O | P1.22 — General purpose digital input/output pin |
| | | O | PIPESTAT1 — Pipeline Status, bit 1. Standard I/O port with internal pull-up. |
| P1.23/ PIPESTAT2 | 36 ^[6] | I/O | P1.23 — General purpose digital input/output pin |
| | | O | PIPESTAT2 — Pipeline Status, bit 2. Standard I/O port with internal pull-up. |
| P1.24/ TRACECLK | 32 ^[6] | I/O | P1.24 — General purpose digital input/output pin |
| | | O | TRACECLK — Trace Clock. Standard I/O port with internal pull-up. |
| P1.25/EXTIN0 | 28 ^[6] | I/O | P1.25 — General purpose digital input/output pin |
| | | I | EXTIN0 — External Trigger Input. Standard I/O with internal pull-up. |
| P1.26/RTCK | 24 ^[6] | I/O | P1.26 — General purpose digital input/output pin |
| | | I/O | RTCK — Returned Test Clock output. Extra signal added to the JTAG port. Assists debugger synchronization when processor frequency varies. Bi-directional pin with internal pull-up. Note: LOW on this pin while $\overline{\text{RESET}}$ is LOW enables pins P1.31:26 to operate as Debug port after reset |
| P1.27/TDO | 64 ^[6] | I/O | P1.27 — General purpose digital input/output pin |
| | | O | TDO — Test Data out for JTAG interface. |
| P1.28/TDI | 60 ^[6] | I/O | P1.28 — General purpose digital input/output pin |
| | | I | TDI — Test Data in for JTAG interface. |
| P1.29/TCK | 56 ^[6] | I/O | P1.29 — General purpose digital input/output pin |
| | | I | TCK — Test Clock for JTAG interface. |
| P1.30/TMS | 52 ^[6] | I/O | P1.30 — General purpose digital input/output pin |
| | | I | TMS — Test Mode Select for JTAG interface. |
| P1.31/ $\overline{\text{TRST}}$ | 20 ^[6] | I/O | P1.31 — General purpose digital input/output pin |
| | | I | $\overline{\text{TRST}}$ — Test Reset for JTAG interface. |
| D+ | 10 ^[7] | I/O | USB bidirectional D+ line. |
| D- | 10 ^[7] | I/O | USB bidirectional D- line. |
| RESET | 57 ^[8] | 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. |
| XTAL1 | 62 ^[9] | I | Input to the oscillator circuit and internal clock generator circuits. |
| XTAL2 | 61 ^[9] | O | Output from the oscillator amplifier. |
| RTXC1 | 3 ^[9] | I | Input to the RTC oscillator circuit. |
| RTXC2 | 5 ^[9] | O | Output from the RTC oscillator circuit. |
| V _{SS} | 6, 18, 25, 42, 50 | I | Ground: 0 V reference |
| V _{SSA} | 59 | 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} | 23, 43, 51 | I | 3.3 V Power Supply: This is the power supply voltage for the core and I/O ports. |

Table 58: Pin description ...continued

| Symbol | Pin | Type | Description |
|-----------|-----|------|---|
| V_{DDA} | 7 | I | Analog 3.3 V Power Supply: This should be nominally the same voltage as V_{DD} but should be isolated to minimize noise and error. This voltage is used to power the ADC(s). |
| V_{REF} | 63 | I | A/D Converter Reference: This should be nominally the same voltage as V_{DD} but should be isolated to minimize noise and error. Level on this pin is used as a reference for A/D convertor. |
| V_{BAT} | 49 | 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 and 10 ns slew rate control.

[2] 5 V tolerant pad providing digital I/O functions with TTL levels and hysteresis and 10 ns slew rate control. If configured for an input function, this pad utilizes built-in glitch filter that blocks pulses shorter than 3 ns.

[3] Open-drain 5 V tolerant digital I/O I2C-bus 400 kHz specification compatible pad. It requires external pull-up to provide an output functionality.

[4] 5 V tolerant pad providing digital I/O (with TTL levels and hysteresis and 10 ns slew rate control) and analog input function. If configured for an input function, this pad utilizes built-in glitch filter that blocks pulses shorter than 3 ns. When configured as an ADC input, digital section of the pad is disabled.

[5] 5 V tolerant pad providing digital I/O (with TTL levels and hysteresis and 10 ns slew rate control) and analog output function. When configured as the DAC output, digital section of the pad is disabled.

[6] 5 V tolerant pad with built-in pull-up resistor providing digital I/O functions with TTL levels and hysteresis and 10 ns slew rate control. The pull-up resistor's value typically ranges from 60 kΩ to 300 kΩ.

[7] Pad is designed in accordance with the Universal Serial Bus (USB) specification, revision 2.0 (Full-speed and Low-speed mode only).

[8] 5 V tolerant pad providing digital input (with TTL levels and hysteresis) function only.

[9] Pad provides special analog functionality.



7.1 Features

- Allows individual pin configuration.

7.2 Applications

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

7.3 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.

The only partial exception from the above rule of exclusion is the case of inputs to the A/D converter. Regardless of the function that is selected for the port pin that also hosts the A/D input, this A/D input can be read at any time and variations of the voltage level on this pin will be reflected in the A/D readings. However, valid analog reading(s) can be obtained if and only if the function of an analog input is selected. Only in this case proper interface circuit is active in between the physical pin and the A/D module. In all other cases, a part of digital logic necessary for the digital function to be performed will be active, and will disrupt proper behavior of the A/D.

7.4 Register description

The Pin Control Module contains 2 registers as shown in [Table 59](#) below.

Table 59: Pin connect block register map

| Name | Description | Access | Reset value ^[1] | Address |
|---------|---------------------------------|------------|------------------------------|-------------|
| PINSEL0 | Pin function select register 0. | Read/Write | 0x0000 0000 | 0xE002 C000 |
| PINSEL1 | Pin function select register 1. | Read/Write | 0x0000 0000 | 0xE002 C004 |
| PINSEL2 | Pin function select register 2. | Read/Write | See Table 62 | 0xE002 C014 |

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

7.4.1 Pin Function Select Register 0 (PINSEL0 - 0xE002 C000)

The PINSEL0 register controls the functions of the pins as per the settings listed in [Table 63](#). The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 60: Pin function Select register 0 (PINSEL0 - address 0xE002 C000) bit description

| Bit | Symbol | Value | Function | Reset value |
|-------|--------|-------|---|-------------|
| 1:0 | P0.0 | 00 | GPIO Port 0.0 | 0 |
| | | 01 | TXD (UART0) | |
| | | 10 | PWM1 | |
| | | 11 | Reserved | |
| 3:2 | P0.1 | 00 | GPIO Port 0.1 | 0 |
| | | 01 | RxD (UART0) | |
| | | 10 | PWM3 | |
| | | 11 | EINT0 | |
| 5:4 | P0.2 | 00 | GPIO Port 0.2 | 0 |
| | | 01 | SCL0 (I ² C0) | |
| | | 10 | Capture 0.0 (Timer 0) | |
| | | 11 | Reserved | |
| 7:6 | P0.3 | 00 | GPIO Port 0.3 | 0 |
| | | 01 | SDA0 (I ² C0) | |
| | | 10 | Match 0.0 (Timer 0) | |
| | | 11 | EINT1 | |
| 9:8 | P0.4 | 00 | GPIO Port 0.4 | 0 |
| | | 01 | SCK0 (SPI0) | |
| | | 10 | Capture 0.1 (Timer 0) | |
| | | 11 | AD0.6 | |
| 11:10 | P0.5 | 00 | GPIO Port 0.5 | 0 |
| | | 01 | MISO0 (SPI0) | |
| | | 10 | Match 0.1 (Timer 0) | |
| | | 11 | AD0.7 | |
| 13:12 | P0.6 | 00 | GPIO Port 0.6 | 0 |
| | | 01 | MOSI0 (SPI0) | |
| | | 10 | Capture 0.2 (Timer 0) | |
| | | 11 | Reserved [1] [2] or AD1.0 [3] | |
| 15:14 | P0.7 | 00 | GPIO Port 0.7 | 0 |
| | | 01 | SSEL0 (SPI0) | |
| | | 10 | PWM2 | |
| | | 11 | EINT2 | |
| 17:16 | P0.8 | 00 | GPIO Port 0.8 | 0 |
| | | 01 | TXD UART1 | |
| | | 10 | PWM4 | |
| | | 11 | Reserved [1] [2] or AD1.1 [3] | |

Table 60: Pin function Select register 0 (PINSEL0 - address 0xE002 C000) bit description

| Bit | Symbol | Value | Function | Reset value |
|-------|--------|-------|--|-------------|
| 19:18 | P0.9 | 00 | GPIO Port 0.9 | 0 |
| | | 01 | RxD (UART1) | |
| | | 10 | PWM6 | |
| | | 11 | EINT3 | |
| 21:20 | P0.10 | 00 | GPIO Port 0.10 | 0 |
| | | 01 | Reserved ^{[1][2]} or RTS (UART1) ^[3] | |
| | | 10 | Capture 1.0 (Timer 1) | |
| | | 11 | Reserved ^{[1][2]} or AD1.2 ^[3] | |
| 23:22 | P0.11 | 00 | GPIO Port 0.11 | 0 |
| | | 01 | Reserved ^{[1][2]} or CTS (UART1) ^[3] | |
| | | 10 | Capture 1.1 (Timer 1) | |
| | | 11 | SCL1 (I ² C1) | |
| 25:24 | P0.12 | 00 | GPIO Port 0.12 | 0 |
| | | 01 | Reserved ^{[1][2]} or DSR (UART1) ^[3] | |
| | | 10 | Match 1.0 (Timer 1) | |
| | | 11 | Reserved ^{[1][2]} or AD1.3 ^[3] | |
| 27:26 | P0.13 | 00 | GPIO Port 0.13 | 0 |
| | | 01 | Reserved ^{[1][2]} or DTR (UART1) ^[3] | |
| | | 10 | Match 1.1 (Timer 1) | |
| | | 11 | Reserved ^{[1][2]} or AD1.4 ^[3] | |
| 29:28 | P0.14 | 00 | GPIO Port 0.14 | 0 |
| | | 01 | Reserved ^{[1][2]} or DCD (UART1) ^[3] | |
| | | 10 | EINT1 | |
| | | 11 | SDA1 (I ² C1) | |
| 31:30 | P0.15 | 00 | GPIO Port 0.15 | 0 |
| | | 01 | Reserved ^{[1][2]} or RI (UART1) ^[3] | |
| | | 10 | EINT2 | |
| | | 11 | Reserved ^{[1][2]} or AD1.5 ^[3] | |

[1] Available on LPC2141.

[2] Available on LPC2142.

[3] Available on LPC2144/6/8.

7.4.2 Pin function Select register 1 (PINSEL1 - 0xE002 C004)

The PINSEL1 register controls the functions of the pins as per the settings listed in following tables. The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

Table 61: Pin function Select register 1 (PINSEL1 - address 0xE002 C004) bit description

| Bit | Symbol | Value | Function | Reset value |
|------------|---------------|--------------|--|--------------------|
| 1:0 | P0.16 | 00 | GPIO Port 0.16 | 0 |
| | | 01 | EINT0 | |
| | | 10 | Match 0.2 (Timer 0) | |
| | | 11 | Capture 0.2 (Timer 0) | |
| 3:2 | P0.17 | 00 | GPIO Port 0.17 | 0 |
| | | 01 | Capture 1.2 (Timer 1) | |
| | | 10 | SCK1 (SSP) | |
| | | 11 | Match 1.2 (Timer 1) | |
| 5:4 | P0.18 | 00 | GPIO Port 0.18 | 0 |
| | | 01 | Capture 1.3 (Timer 1) | |
| | | 10 | MISO1 (SSP) | |
| | | 11 | Match 1.3 (Timer 1) | |
| 7:6 | P0.19 | 00 | GPIO Port 0.19 | 0 |
| | | 01 | Match 1.2 (Timer 1) | |
| | | 10 | MOSI1 (SSP) | |
| | | 11 | Capture 1.2 (Timer 1) | |
| 9:8 | P0.20 | 00 | GPIO Port 0.20 | 0 |
| | | 01 | Match 1.3 (Timer 1) | |
| | | 10 | SSEL1 (SSP) | |
| | | 11 | EINT3 | |
| 11:10 | P0.21 | 00 | GPIO Port 0.21 | 0 |
| | | 01 | PWM5 | |
| | | 10 | Reserved ^{[1][2]} or AD1.6 ^[3] | |
| | | 11 | Capture 1.3 (Timer 1) | |
| 13:12 | P0.22 | 00 | GPIO Port 0.22 | 0 |
| | | 01 | Reserved ^{[1][2]} or AD1.7 ^[3] | |
| | | 10 | Capture 0.0 (Timer 0) | |
| | | 11 | Match 0.0 (Timer 0) | |
| 15:14 | P0.23 | 00 | GPIO Port 0.23 | 0 |
| | | 01 | V _{BUS} | |
| | | 10 | Reserved | |
| | | 11 | Reserved | |
| 17:16 | P0.24 | 00 | Reserved | 0 |
| | | 01 | Reserved | |
| | | 10 | Reserved | |
| | | 11 | Reserved | |
| 19:18 | P0.25 | 00 | GPIO Port 0.25 | 0 |
| | | 01 | AD0.4 | |
| | | 10 | Reserved ^[1] or Aout(DAC) ^{[2][3]} | |
| | | 11 | Reserved | |

Table 61: Pin function Select register 1 (PINSEL1 - address 0xE002 C004) bit description

| Bit | Symbol | Value | Function | Reset value |
|-------|--------|-------|-----------------------|-------------|
| 21:20 | P0.26 | 00 | Reserved | 0 |
| | | 01 | Reserved | |
| | | 10 | Reserved | |
| | | 11 | Reserved | |
| 23:22 | P0.27 | 00 | Reserved | 0 |
| | | 01 | Reserved | |
| | | 10 | Reserved | |
| | | 11 | Reserved | |
| 25:24 | P0.28 | 00 | GPIO Port 0.28 | 0 |
| | | 01 | AD0.1 | |
| | | 10 | Capture 0.2 (Timer 0) | |
| | | 11 | Match 0.2 (Timer 0) | |
| 27:26 | P0.29 | 00 | GPIO Port 0.29 | 0 |
| | | 01 | AD0.2 | |
| | | 10 | Capture 0.3 (Timer 0) | |
| | | 11 | Match 0.3 (Timer 0) | |
| 29:28 | P0.30 | 00 | GPIO Port 0.30 | 0 |
| | | 01 | AD0.3 | |
| | | 10 | EINT3 | |
| | | 11 | Capture 0.0 (Timer 0) | |
| 31:30 | P0.31 | 00 | GPO Port only | 0 |
| | | 01 | UP_LED | |
| | | 10 | CONNECT | |
| | | 11 | Reserved | |

[1] Available on LPC2141.

[2] Available on LPC2142.

[3] Available on LPC2144/6/8.

7.4.3 Pin function Select register 2 (PINSEL2 - 0xE002 C014)

The PINSEL2 register controls the functions of the pins as per the settings listed in [Table 62](#). The direction control bit in the IO1DIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

Warning: use read-modify-write operation when accessing PINSEL2 register. Accidental write of 0 to bit 2 and/or bit 3 results in loss of debug and/or trace functionality! Changing of either bit 4 or bit 5 from 1 to 0 may cause an incorrect code execution!

Table 62: Pin function Select register 2 (PINSEL2 - 0xE002 C014) bit description

| Bit | Symbol | Value | Function | 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 | GPIO/DEBUG | 0 | Pins P1.36-26 are used as GPIO pins. | P1.26/RTCK |
| | | 1 | Pins P1.36-26 are used as a Debug port. | |
| 3 | GPIO/TRACE | 0 | Pins P1.25-16 are used as GPIO pins. | P1.20/ TRACESYNC |
| | | 1 | Pins P1.25-16 are used as a Trace port. | |
| 31:4 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

7.4.4 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 63: Pin function select register bits

| PINSEL0 and PINSEL1 Values | Function | Value after Reset |
|----------------------------|---|-------------------|
| 00 | Primary (default) function, typically GPIO port | 00 |
| 01 | First alternate function | |
| 10 | Second alternate function | |
| 11 | Reserved | |

The direction control bit in the IO0DIR/IO1DIR register 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.

8.1 Features

- Every physical GPIO port is accessible via either the group of registers providing an enhanced features and accelerated port access or the legacy group of registers
- 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 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 VPB bus

8.2 Applications

- General purpose I/O
- Driving LEDs, or other indicators
- Controlling off-chip devices
- Sensing digital inputs

8.3 Pin description

Table 64: GPIO pin description

| Pin | Type | Description |
|-------------|--------------|---|
| P0.0-P3.1 | Input/Output | General purpose input/output. The number of GPIOs actually available depends on the use of alternate functions. |
| P1.16-P1.31 | Output | |

8.4 Register description

LPC2141/2/4/6/8 has two 32-bit General Purpose I/O ports. Total of 30 input/output and a single output only pin out of 32 pins are available on PORT0. PORT1 has up to 16 pins available for GPIO functions. PORT0 and PORT1 are controlled via two groups of 4 registers as shown in [Table 65](#) and [Table 66](#).

Legacy registers shown in [Table 65](#) allow backward compatibility with earlier family devices, using existing code. The functions and relative timing of older GPIO implementations is preserved.

The registers in [Table 66](#) represent the enhanced GPIO features available on the LPC2141/2/4/6/8. All of these registers are located directly on the local bus of the CPU for the fastest possible read and write timing. An additional feature has been added that provides byte addressability of all GPIO registers. A mask register allows treating groups of bits in a single GPIO port separately from other bits on the same port.

User must select whether a GPIO will be accessed via registers that provide enhanced features or a legacy set of registers (see [Section 3.6.1 "System Control and Status flags register \(SCS - 0xE01F C1A0\)" 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 65: GPIO register map (legacy VPB accessible registers)

| Generic Name | Description | Access | Reset value ^[1] | PORT0 Address & Name | PORT1 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. | R/W | NA | 0xE002 8000 IO0PIN | 0xE002 8010 IO1PIN |
| 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 | 0x0000 0000 | 0xE002 8004 IO0SET | 0xE002 8014 IO1SET |
| IODIR | GPIO Port Direction control register. This register individually controls the direction of each port pin. | R/W | 0x0000 0000 | 0xE002 8008 IO0DIR | 0xE002 8018 IO1DIR |
| 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 | 0x0000 0000 | 0xE002 800C IO0CLR | 0xE002 801C IO1CLR |

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

Table 66: GPIO register map (local bus accessible registers - enhanced GPIO features)

| Generic Name | Description | Access | Reset value ^[1] | PORT0 Address & Name | PORT1 Address & Name |
|--------------|--|--------|----------------------------|-------------------------|-------------------------|
| FIODIR | Fast GPIO Port Direction control register. This register individually controls the direction of each port pin. | R/W | 0x0000 0000 | 0x3FFF C000 FIO0DIR | 0x3FFF C020 FIO1DIR |
| 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 | 0x0000 0000 | 0x3FFF C010 FIO0MASK | 0x3FFF C030 FIO1MASK |
| 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 is not configured as an input to ADC). The value read is masked by ANDing with FIOMASK. Writing to this register places corresponding values in all bits enabled by ones in FIOMASK. | R/W | 0x0000 0000 | 0x3FFF C014 FIO0PIN | 0x3FFF C034 FIO1PIN |
| 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 ones in FIOMASK can be altered. | R/W | 0x0000 0000 | 0x3FFF C018 FIO0SET | 0x3FFF C038 FIO1SET |
| 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 ones in FIOMASK0 can be altered. | WO | 0x0000 0000 | 0x3FFF C01C FIO0CLR | 0x3FFF C03C FIO1CLR |

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

8.4.1 GPIO port Direction register (IODIR, Port 0: IO0DIR - 0xE002 8008 and Port 1: IO1DIR - 0xE002 8018; FIODIR, Port 0: FIO0DIR - 0x3FFF C000 and Port 1:FIO1DIR - 0x3FFF C020)

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 and FIO1DIR registers.

Table 67: GPIO port 0 Direction register (IO0DIR - address 0xE002 8008) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 31:0 | P0xDIR | | Slow GPIO Direction control bits. Bit 0 controls P0.0 ... bit 30 controls P0.30. | 0x0000 0000 |
| | | 0 | Controlled pin is input. | |
| | | 1 | Controlled pin is output. | |

Table 68: GPIO port 1 Direction register (IO1DIR - address 0xE002 8018) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 31:0 | P1xDIR | | Slow GPIO Direction control bits. Bit 0 in IO1DIR controls P1.0 ... Bit 30 in IO1DIR controls P1.30. | 0x0000 0000 |
| | | 0 | Controlled pin is input. | |
| | | 1 | Controlled pin is output. | |

Table 69: Fast GPIO port 0 Direction register (FIO0DIR - address 0x3FFF C000) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---------|-------|--|-------------|
| 31:0 | FP0xDIR | | Fast GPIO Direction control bits. Bit 0 in FIO0DIR controls P0.0 ... Bit 30 in FIO0DIR controls P0.30. | 0x0000 0000 |
| | | 0 | Controlled pin is input. | |
| | | 1 | Controlled pin is output. | |

Table 70: Fast GPIO port 1 Direction register (FIO1DIR - address 0x3FFF C020) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---------|-------|--|-------------|
| 31:0 | FP1xDIR | | Fast GPIO Direction control bits. Bit 0 in FIO1DIR controls P1.0 ... Bit 30 in FIO1DIR controls P1.30. | 0x0000 0000 |
| | | 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 71](#) and [Table 72](#), 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 71: Fast GPIO port 0 Direction control byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|---|-------------|
| FIO0DIR0 | 8 (byte) | 0x3FFF C000 | Fast GPIO Port 0 Direction control register 0. Bit 0 in FIO0DIR0 register corresponds to P0.0 ... bit 7 to P0.7. | 0x00 |
| FIO0DIR1 | 8 (byte) | 0x3FFF C001 | Fast GPIO Port 0 Direction control register 1. Bit 0 in FIO0DIR1 register corresponds to P0.8 ... bit 7 to P0.15. | 0x00 |
| FIO0DIR2 | 8 (byte) | 0x3FFF C002 | Fast GPIO Port 0 Direction control register 2. Bit 0 in FIO0DIR2 register corresponds to P0.16 ... bit 7 to P0.23. | 0x00 |
| FIO0DIR3 | 8 (byte) | 0x3FFF C003 | Fast GPIO Port 0 Direction control register 3. Bit 0 in FIO0DIR3 register corresponds to P0.24 ... bit 7 to P0.31. | 0x00 |
| FIO0DIRL | 16 (half-word) | 0x3FFF C000 | Fast GPIO Port 0 Direction control Lower half-word register. Bit 0 in FIO0DIRL register corresponds to P0.0 ... bit 15 to P0.15. | 0x0000 |
| FIO0DIRU | 16 (half-word) | 0x3FFF C002 | Fast GPIO Port 0 Direction control Upper half-word register. Bit 0 in FIO0DIRU register corresponds to P0.16 ... bit 15 to P0.31. | 0x0000 |

Table 72: Fast GPIO port 1 Direction control byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|---|-------------|
| FIO1DIR0 | 8 (byte) | 0x3FFF C020 | Fast GPIO Port 1 Direction control register 0. Bit 0 in FIO1DIR0 register corresponds to P1.0 ... bit 7 to P1.7. | 0x00 |
| FIO1DIR1 | 8 (byte) | 0x3FFF C021 | Fast GPIO Port 1 Direction control register 1. Bit 0 in FIO1DIR1 register corresponds to P1.8 ... bit 7 to P1.15. | 0x00 |
| FIO1DIR2 | 8 (byte) | 0x3FFF C022 | Fast GPIO Port 1 Direction control register 2. Bit 0 in FIO1DIR2 register corresponds to P1.16 ... bit 7 to P1.23. | 0x00 |
| FIO1DIR3 | 8 (byte) | 0x3FFF C023 | Fast GPIO Port 1 Direction control register 3. Bit 0 in FIO1DIR3 register corresponds to P1.24 ... bit 7 to P1.31. | 0x00 |
| FIO1DIRL | 16 (half-word) | 0x3FFF C020 | Fast GPIO Port 1 Direction control Lower half-word register. Bit 0 in FIO1DIRL register corresponds to P1.0 ... bit 15 to P1.15. | 0x0000 |
| FIO1DIRU | 16 (half-word) | 0x3FFF C022 | Fast GPIO Port 1 Direction control Upper half-word register. Bit 0 in FIO1DIRU register corresponds to P1.16 ... bit 15 to P1.31. | 0x0000 |

8.4.2 Fast GPIO port Mask register (FIOMASK, Port 0: FIO0MASK - 0x3FFF C010 and Port 1:FIO1MASK - 0x3FFF C030)

This register is available in the enhanced group of registers only. It is used to select port's pins that will and will not be affected by a write accesses to the FIOPIN, FIOSET or FIOSLR 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 8.5 "GPIO usage notes" on page 92](#)

Table 73: Fast GPIO port 0 Mask register (FIO0MASK - address 0x3FFF C010) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|----------|-------|---|-------------|
| 31:0 | FP0xMASK | | Fast GPIO physical pin access control. | 0x0000 0000 |
| | | 0 | Pin is affected by writes to the FIOSET, FIOCLR, and FIOPIN registers. Current state of the pin will be observable in the FIOPIN register. | |
| | | 1 | Physical pin is unaffected by writes into the FIOSET, FIOCLR and FIOPIN registers. When the FIOPIN register is read, this bit will not be updated with the state of the physical pin. | |

Table 74: Fast GPIO port 1 Mask register (FIO1MASK - address 0x3FFF C030) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|----------|-------|---|-------------|
| 31:0 | FP1xMASK | | Fast GPIO physical pin access control. | 0x0000 0000 |
| | | 0 | Pin is affected by writes to the FIOSET, FIOCLR, and FIOPIN registers. Current state of the pin will be observable in the FIOPIN register. | |
| | | 1 | Physical pin is unaffected by writes into the FIOSET, FIOCLR and FIOPIN registers. 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 75](#) and [Table 76](#), 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 75: Fast GPIO port 0 Mask byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|---|-------------|
| FIO0MASK0 | 8 (byte) | 0x3FFF C010 | Fast GPIO Port 0 Mask register 0. Bit 0 in FIO0MASK0 register corresponds to P0.0 ... bit 7 to P0.7. | 0x00 |
| FIO0MASK1 | 8 (byte) | 0x3FFF C011 | Fast GPIO Port 0 Mask register 1. Bit 0 in FIO0MASK1 register corresponds to P0.8 ... bit 7 to P0.15. | 0x00 |
| FIO0MASK2 | 8 (byte) | 0x3FFF C012 | Fast GPIO Port 0 Mask register 2. Bit 0 in FIO0MASK2 register corresponds to P0.16 ... bit 7 to P0.23. | 0x00 |
| FIO0MASK3 | 8 (byte) | 0x3FFF C013 | Fast GPIO Port 0 Mask register 3. Bit 0 in FIO0MASK3 register corresponds to P0.24 ... bit 7 to P0.31. | 0x00 |
| FIO0MASKL | 16 (half-word) | 0x3FFF C001 | Fast GPIO Port 0 Mask Lower half-word register. Bit 0 in FIO0MASKL register corresponds to P0.0 ... bit 15 to P0.15. | 0x0000 |
| FIO0MASKU | 16 (half-word) | 0x3FFF C012 | Fast GPIO Port 0 Mask Upper half-word register. Bit 0 in FIO0MASKU register corresponds to P0.16 ... bit 15 to P0.31. | 0x0000 |

Table 76: Fast GPIO port 1 Mask byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|---|-------------|
| FIO1MASK0 | 8 (byte) | 0x3FFF C010 | Fast GPIO Port 1 Mask register 0. Bit 0 in FIO1MASK0 register corresponds to P1.0 ... bit 7 to P1.7. | 0x00 |
| FIO1MASK1 | 8 (byte) | 0x3FFF C011 | Fast GPIO Port 1 Mask register 1. Bit 0 in FIO1MASK1 register corresponds to P1.8 ... bit 7 to P1.15. | 0x00 |
| FIO1MASK2 | 8 (byte) | 0x3FFF C012 | Fast GPIO Port 1 Mask register 2. Bit 0 in FIO1MASK2 register corresponds to P1.16 ... bit 7 to P1.23. | 0x00 |
| FIO1MASK3 | 8 (byte) | 0x3FFF C013 | Fast GPIO Port 1 Mask register 3. Bit 0 in FIO1MASK3 register corresponds to P1.24 ... bit 7 to P1.31. | 0x00 |
| FIO1MASKL | 16 (half-word) | 0x3FFF C001 | Fast GPIO Port 1 Mask Lower half-word register. Bit 0 in FIO1MASKL register corresponds to P1.0 ... bit 15 to P1.15. | 0x0000 |
| FIO1MASKU | 16 (half-word) | 0x3FFF C012 | Fast GPIO Port 1 Mask Upper half-word register. Bit 0 in FIO1MASKU register corresponds to P1.16 ... bit 15 to P1.31. | 0x0000 |

8.4.3 GPIO port Pin value register (IOPIN, Port 0: IO0PIN - 0xE002 8000 and Port 1: IO1PIN - 0xE002 8010; FIOPIN, Port 0: FIO0PIN - 0x3FFF C014 and Port 1: FIO1PIN - 0x3FFF C034)

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 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 and FIO1PIN registers. Access to a port pins via the FIOPIN register is conditioned by the corresponding FIOMASK register (see [Section 8.4.2 “Fast GPIO port Mask register \(FIOMASK, Port 0: FIO0MASK - 0x3FFF C010 and Port 1:FIO1MASK - 0x3FFF C030”](#)).

Only pins masked with zeros in the Mask register (see [Section 8.4.2 “Fast GPIO port Mask register \(FIOMASK, Port 0: FIO0MASK - 0x3FFF C010 and Port 1:FIO1MASK - 0x3FFF C030”](#)) will be correlated to the current content of the Fast GPIO port pin value register.

Table 77: GPIO port 0 Pin value register (IO0PIN - address 0xE002 8000) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 31:0 | P0xVAL | Slow GPIO pin value bits. Bit 0 in IO0PIN corresponds to P0.0 ... Bit 31 in IO0PIN corresponds to P0.31. | NA |

Table 78: GPIO port 1 Pin value register (IO1PIN - address 0xE002 8010) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 31:0 | P1xVAL | Slow GPIO pin value bits. Bit 0 in IO1PIN corresponds to P1.0 ... Bit 31 in IO1PIN corresponds to P1.31. | NA |

Table 79: Fast GPIO port 0 Pin value register (FIO0PIN - address 0x3FFF C014) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|--|-------------|
| 31:0 | FP0xVAL | Fast GPIO pin value bits. Bit 0 in FIO0PIN corresponds to P0.0 ... Bit 31 in FIO0PIN corresponds to P0.31. | NA |

Table 80: Fast GPIO port 1 Pin value register (FIO1PIN - address 0x3FFF C034) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|--|-------------|
| 31:0 | FP1xVAL | Fast GPIO pin value bits. Bit 0 in FIO1PIN corresponds to P1.0 ... Bit 31 in FIO1PIN corresponds to P1.31. | NA |

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 81](#) and [Table 82](#), 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 81: Fast GPIO port 0 Pin value byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|---|-------------|
| FIO0PIN0 | 8 (byte) | 0x3FFF C014 | Fast GPIO Port 0 Pin value register 0. Bit 0 in FIO0PIN0 register corresponds to P0.0 ... bit 7 to P0.7. | 0x00 |
| FIO0PIN1 | 8 (byte) | 0x3FFF C015 | Fast GPIO Port 0 Pin value register 1. Bit 0 in FIO0PIN1 register corresponds to P0.8 ... bit 7 to P0.15. | 0x00 |
| FIO0PIN2 | 8 (byte) | 0x3FFF C016 | Fast GPIO Port 0 Pin value register 2. Bit 0 in FIO0PIN2 register corresponds to P0.16 ... bit 7 to P0.23. | 0x00 |
| FIO0PIN3 | 8 (byte) | 0x3FFF C017 | Fast GPIO Port 0 Pin value register 3. Bit 0 in FIO0PIN3 register corresponds to P0.24 ... bit 7 to P0.31. | 0x00 |
| FIO0PINL | 16 (half-word) | 0x3FFF C014 | Fast GPIO Port 0 Pin value Lower half-word register. Bit 0 in FIO0PINL register corresponds to P0.0 ... bit 15 to P0.15. | 0x0000 |
| FIO0PINU | 16 (half-word) | 0x3FFF C016 | Fast GPIO Port 0 Pin value Upper half-word register. Bit 0 in FIO0PINU register corresponds to P0.16 ... bit 15 to P0.31. | 0x0000 |

Table 82: Fast GPIO port 1 Pin value byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|---|-------------|
| FIO1PIN0 | 8 (byte) | 0x3FFF C034 | Fast GPIO Port 1 Pin value register 0. Bit 0 in FIO1PIN0 register corresponds to P1.0 ... bit 7 to P1.7. | 0x00 |
| FIO1PIN1 | 8 (byte) | 0x3FFF C035 | Fast GPIO Port 1 Pin value register 1. Bit 0 in FIO1PIN1 register corresponds to P1.8 ... bit 7 to P1.15. | 0x00 |
| FIO1PIN2 | 8 (byte) | 0x3FFF C036 | Fast GPIO Port 1 Pin value register 2. Bit 0 in FIO1PIN2 register corresponds to P1.16 ... bit 7 to P1.23. | 0x00 |
| FIO1PIN3 | 8 (byte) | 0x3FFF C037 | Fast GPIO Port 1 Pin value register 3. Bit 0 in FIO1PIN3 register corresponds to P1.24 ... bit 7 to P1.31. | 0x00 |
| FIO1PINL | 16 (half-word) | 0x3FFF C034 | Fast GPIO Port 1 Pin value Lower half-word register. Bit 0 in FIO1PINL register corresponds to P1.0 ... bit 15 to P1.15. | 0x0000 |
| FIO1PINU | 16 (half-word) | 0x3FFF C036 | Fast GPIO Port 1 Pin value Upper half-word register. Bit 0 in FIO1PINU register corresponds to P1.16 ... bit 15 to P1.31. | 0x0000 |

8.4.4 GPIO port output Set register (IOSET, Port 0: IO0SET - 0xE002 8004 and Port 1: IO1SET - 0xE002 8014; FIOSET, Port 0: FIO0SET - 0x3FFF C018 and Port 1: FIO1SET - 0x3FFF C038)

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 and FIO1SET registers. Access to a port pins via the FIOSET register is conditioned by the corresponding FIOMASK register (see [Section 8.4.2 "Fast GPIO port Mask register \(FIOMASK, Port 0: FIO0MASK - 0x3FFF C010 and Port 1:FIO1MASK - 0x3FFF C030\)"](#)).

Table 83: GPIO port 0 output Set register (IO0SET - address 0xE002 8004) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | P0xSET | Slow GPIO output value Set bits. Bit 0 in IO0SET corresponds to P0.0 ... Bit 31 in IO0SET corresponds to P0.31. | 0x0000 0000 |

Table 84: GPIO port 1 output Set register (IO1SET - address 0xE002 8014) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | P1xSET | Slow GPIO output value Set bits. Bit 0 in IO1SET corresponds to P1.0 ... Bit 31 in IO1SET corresponds to P1.31. | 0x0000 0000 |

Table 85: Fast GPIO port 0 output Set register (FIO0SET - address 0x3FFF C018) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|---|-------------|
| 31:0 | FP0xSET | Fast GPIO output value Set bits. Bit 0 in FIO0SET corresponds to P0.0 ... Bit 31 in FIO0SET corresponds to P0.31. | 0x0000 0000 |

Table 86: Fast GPIO port 1 output Set register (FIO1SET - address 0x3FFF C038) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|---|-------------|
| 31:0 | FP1xSET | Fast GPIO output value Set bits. Bit 0 in FIO1SET corresponds to P1.0 ... Bit 31 in FIO1SET corresponds to P1.31. | 0x0000 0000 |

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 87](#) and [Table 88](#), 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 87: Fast GPIO port 0 output Set byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|--|-------------|
| FIO0SET0 | 8 (byte) | 0x3FFF C018 | Fast GPIO Port 0 output Set register 0. Bit 0 in FIO0SET0 register corresponds to P0.0 ... bit 7 to P0.7. | 0x00 |
| FIO0SET1 | 8 (byte) | 0x3FFF C019 | Fast GPIO Port 0 output Set register 1. Bit 0 in FIO0SET1 register corresponds to P0.8 ... bit 7 to P0.15. | 0x00 |
| FIO0SET2 | 8 (byte) | 0x3FFF C01A | Fast GPIO Port 0 output Set register 2. Bit 0 in FIO0SET2 register corresponds to P0.16 ... bit 7 to P0.23. | 0x00 |
| FIO0SET3 | 8 (byte) | 0x3FFF C01B | Fast GPIO Port 0 output Set register 3. Bit 0 in FIO0SET3 register corresponds to P0.24 ... bit 7 to P0.31. | 0x00 |
| FIO0SETL | 16 (half-word) | 0x3FFF C018 | Fast GPIO Port 0 output Set Lower half-word register. Bit 0 in FIO0SETL register corresponds to P0.0 ... bit 15 to P0.15. | 0x0000 |
| FIO0SETU | 16 (half-word) | 0x3FFF C01A | Fast GPIO Port 0 output Set Upper half-word register. Bit 0 in FIO0SETU register corresponds to P0.16 ... bit 15 to P0.31. | 0x0000 |

Table 88: Fast GPIO port 1 output Set byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|--|-------------|
| FIO1SET0 | 8 (byte) | 0x3FFF C038 | Fast GPIO Port 1 output Set register 0. Bit 0 in FIO1SET0 register corresponds to P1.0 ... bit 7 to P1.7. | 0x00 |
| FIO1SET1 | 8 (byte) | 0x3FFF C039 | Fast GPIO Port 1 output Set register 1. Bit 0 in FIO1SET1 register corresponds to P1.8 ... bit 7 to P1.15. | 0x00 |
| FIO1SET2 | 8 (byte) | 0x3FFF C03A | Fast GPIO Port 1 output Set register 2. Bit 0 in FIO1SET2 register corresponds to P1.16 ... bit 7 to P1.23. | 0x00 |
| FIO1SET3 | 8 (byte) | 0x3FFF C03B | Fast GPIO Port 1 output Set register 3. Bit 0 in FIO1SET3 register corresponds to P1.24 ... bit 7 to P1.31. | 0x00 |
| FIO1SETL | 16 (half-word) | 0x3FFF C038 | Fast GPIO Port 1 output Set Lower half-word register. Bit 0 in FIO1SETL register corresponds to P1.0 ... bit 15 to P1.15. | 0x0000 |
| FIO1SETU | 16 (half-word) | 0x3FFF C03A | Fast GPIO Port 1 output Set Upper half-word register. Bit 0 in FIO1SETU register corresponds to P1.16 ... bit 15 to P1.31. | 0x0000 |

8.4.5 GPIO port output Clear register (IOCLR, Port 0: IO0CLR - 0xE002 800C and Port 1: IO1CLR - 0xE002 801C; FIOCLR, Port 0: FIO0CLR - 0x3FFF C01C and Port 1: FIO1CLR - 0x3FFF C03C)

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 and FIO1CLR registers. Access to a port pins via the FIOCLR register is conditioned by the corresponding FIOMASK register (see [Section 8.4.2 "Fast GPIO port Mask register \(FIOMASK, Port 0: FIO0MASK - 0x3FFF C010 and Port 1:FIO1MASK - 0x3FFF C030\)".](#)

Table 89: GPIO port 0 output Clear register 0 (IO0CLR - address 0xE002 800C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | P0xCLR | Slow GPIO output value Clear bits. Bit 0 in IO0CLR corresponds to P0.0 ... Bit 31 in IO0CLR corresponds to P0.31. | 0x0000 0000 |

Table 90: GPIO port 1 output Clear register 1 (IO1CLR - address 0xE002 801C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | P1xCLR | Slow GPIO output value Clear bits. Bit 0 in IO1CLR corresponds to P1.0 ... Bit 31 in IO1CLR corresponds to P1.31. | 0x0000 0000 |

Table 91: Fast GPIO port 0 output Clear register 0 (FIO0CLR - address 0x3FFF C01C) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|---|-------------|
| 31:0 | FP0xCLR | Fast GPIO output value Clear bits. Bit 0 in FIO0CLR corresponds to P0.0 ... Bit 31 in FIO0CLR corresponds to P0.31. | 0x0000 0000 |

Table 92: Fast GPIO port 1 output Clear register 1 (FIO1CLR - address 0x3FFF C03C) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|---|-------------|
| 31:0 | FP1xCLR | Fast GPIO output value Clear bits. Bit 0 in FIO1CLR corresponds to P1.0 ... Bit 31 in FIO1CLR corresponds to P1.31. | 0x0000 0000 |

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 93](#) and [Table 94](#), 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 93: Fast GPIO port 0 output Clear byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|--|-------------|
| FIO0CLR0 | 8 (byte) | 0x3FFF C01C | Fast GPIO Port 0 output Clear register 0. Bit 0 in FIO0CLR0 register corresponds to P0.0 ... bit 7 to P0.7. | 0x00 |
| FIO0CLR1 | 8 (byte) | 0x3FFF C01D | Fast GPIO Port 0 output Clear register 1. Bit 0 in FIO0CLR1 register corresponds to P0.8 ... bit 7 to P0.15. | 0x00 |
| FIO0CLR2 | 8 (byte) | 0x3FFF C01E | Fast GPIO Port 0 output Clear register 2. Bit 0 in FIO0CLR2 register corresponds to P0.16 ... bit 7 to P0.23. | 0x00 |
| FIO0CLR3 | 8 (byte) | 0x3FFF C01F | Fast GPIO Port 0 output Clear register 3. Bit 0 in FIO0CLR3 register corresponds to P0.24 ... bit 7 to P0.31. | 0x00 |
| FIO0CLRL | 16 (half-word) | 0x3FFF C01C | Fast GPIO Port 0 output Clear Lower half-word register. Bit 0 in FIO0CLRL register corresponds to P0.0 ... bit 15 to P0.15. | 0x0000 |
| FIO0CLRU | 16 (half-word) | 0x3FFF C01E | Fast GPIO Port 0 output Clear Upper half-word register. Bit 0 in FIO0SETU register corresponds to P0.16 ... bit 15 to P0.31. | 0x0000 |

Table 94: Fast GPIO port 1 output Clear byte and half-word accessible register description

| Register name | Register length (bits) & access | Address | Description | Reset value |
|---------------|---------------------------------|-------------|--|-------------|
| FIO1CLR0 | 8 (byte) | 0x3FFF C03C | Fast GPIO Port 1 output Clear register 0. Bit 0 in FIO1CLR0 register corresponds to P1.0 ... bit 7 to P1.7. | 0x00 |
| FIO1CLR1 | 8 (byte) | 0x3FFF C03D | Fast GPIO Port 1 output Clear register 1. Bit 0 in FIO1CLR1 register corresponds to P1.8 ... bit 7 to P1.15. | 0x00 |
| FIO1CLR2 | 8 (byte) | 0x3FFF C03E | Fast GPIO Port 1 output Clear register 2. Bit 0 in FIO1CLR2 register corresponds to P1.16 ... bit 7 to P1.23. | 0x00 |
| FIO1CLR3 | 8 (byte) | 0x3FFF C03F | Fast GPIO Port 1 output Clear register 3. Bit 0 in FIO1CLR3 register corresponds to P1.24 ... bit 7 to P1.31. | 0x00 |
| FIO1CLRL | 16 (half-word) | 0x3FFF C03C | Fast GPIO Port 1 output Clear Lower half-word register. Bit 0 in FIO1CLRL register corresponds to P1.0 ... bit 15 to P1.15. | 0x0000 |
| FIO1CLRU | 16 (half-word) | 0x3FFF C03E | Fast GPIO Port 1 output Clear Upper half-word register. Bit 0 in FIO1CLRU register corresponds to P1.16 ... bit 15 to P1.31. | 0x0000 |

8.5 GPIO usage notes

8.5.1 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 case of a 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.

8.5.2 Example 2: an immediate 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]:

```
IOPIN = (IOPIN && 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;
```

8.5.3 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 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.

8.5.4 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 an assembly code and executed in the ARM mode.

Here is a code where the pin control section is written in assembly language for ARM. It illustrates the difference between the fast and slow GPIO port output capabilities. Once this code is compiled in the ARM mode, its execution from the on-chip Flash will yield the best results when the MAM module is configured as described in [Section 4.9 "MAM usage notes" on page 49](#). Execution from the on-chip SRAM is independent from the MAM setup.

```
ldr r0,=0xe01fc1a0 /*register address--enable fast port*/
mov r1,#0x1
str r1,[r0]          /*enable fast port0*/
ldr r1,=0xffffffff
ldr r0,=0x3fffc000 /*direction of fast port0*/
str r1,[r0]
ldr r0,=0xe0028018 /*direction of slow port 1*/
str r1,[r0]
ldr r0,=0x3fffc018 /*FIO0SET -- fast port0 register*/
ldr r1,=0x3fffc01c /*FIO0CLR0 -- fast port0 register*/
ldr r2,=0xC0010000 /*select fast port 0.16 for toggle*/
ldr r3,=0xE0028014 /*I01SET -- slow port1 register*/
ldr r4,=0xE002801C /*I01CLR -- slow port1 register*/
ldr r5,=0x00100000 /*select slow port 1.20 for toggle*/
/*Generate 2 pulses on the fast port*/
str r2,[r0]
```

```

str  r2,[r1]
str  r2,[r0]
str  r2,[r1]
/*Generate 2 pulses on the slow port*/
str  r5,[r3]
str  r5,[r4]
str  r5,[r3]
str  r5,[r4]
loop:b    loop

```

[Figure 17](#) illustrates the code from above executed from the LPC2148 Flash memory. The PLL generated $F_{CCLK} = 60$ MHz out of external $F_{OSC} = 12$ MHz. The MAM was fully enabled with $MEMCR = 2$ and $MEMTIM = 3$, and $VPBDIV = 1$ ($PCLK = CCLK$).

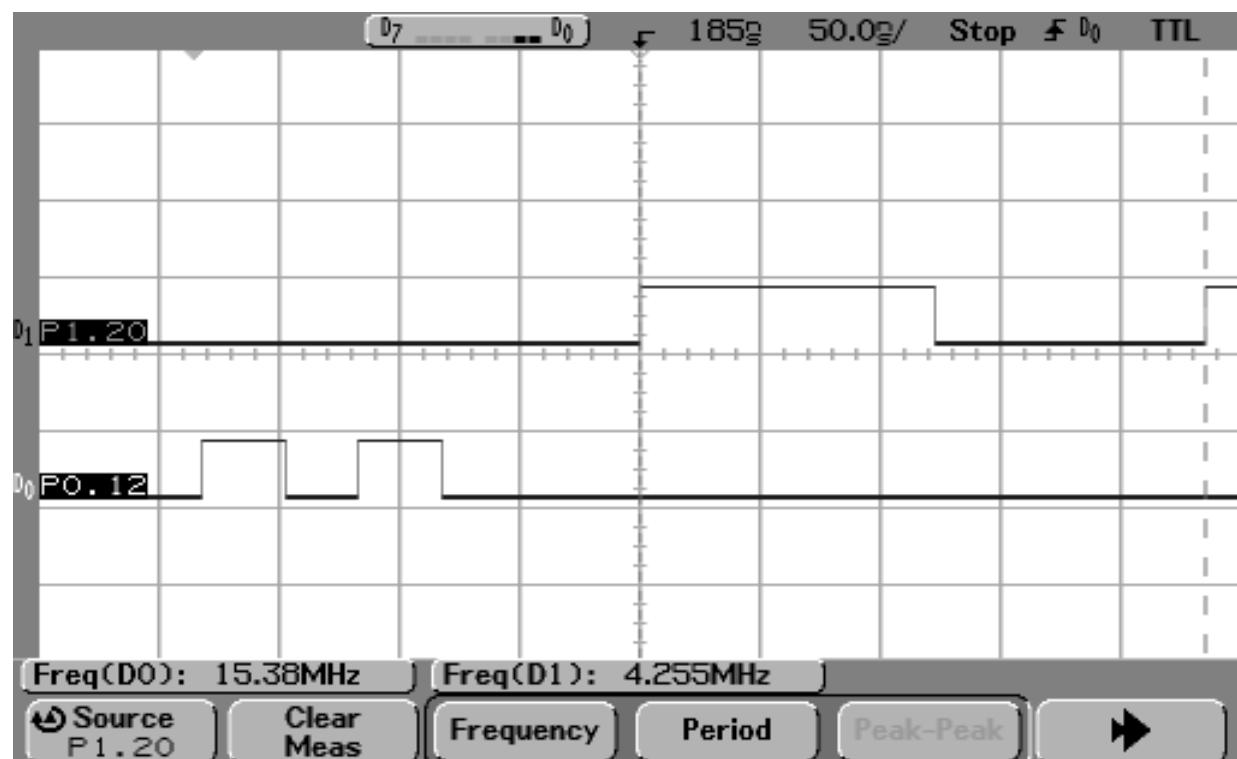


Fig 17. Illustration of the fast and slow GPIO access and output showing 3.5 x increase of the pin output frequency

9.1 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 fractional baud rate generator with autobauding capabilities.
- Mechanism that enables software and hardware flow control implementation.

9.2 Pin description

Table 95: UART0 pin description

| Pin | Type | Description |
|------|--------|---|
| RXD0 | Input | Serial Input. Serial receive data. |
| TXD0 | Output | Serial Output. Serial transmit data. |

9.3 Register description

UART0 contains registers organized as shown in [Table 96](#). The Divisor Latch Access Bit (DLAB) is contained in U0LCR[7] and enables access to the Divisor Latches.

Table 96: UART0 register map

| Name | Description | Bit functions and addresses | | | | | | | | Access | Reset value[1] | Address | | | |
|-------|-----------------------------|-----------------------------|-----------|--------------|-----------------|---------------|------------------|---------------------|--------------------|--------|----------------|----------------------|--|--|--|
| | | MSB LSB | | | | | | | | | | | | | |
| | | BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 | | | | | | |
| U0RBR | Receiver Buffer Register | 8-bit Read Data | | | | | | | | RO | NA | 0xE000 C000 (DLAB=0) | | | |
| U0THR | Transmit Holding Register | 8-bit Write Data | | | | | | | | WO | NA | 0xE000 C000 (DLAB=0) | | | |
| U0DLL | Divisor Latch LSB | 8-bit Data | | | | | | | | R/W | 0x01 | 0xE000 C000 (DLAB=1) | | | |
| U0DLM | Divisor Latch MSB | 8-bit Data | | | | | | | | R/W | 0x00 | 0xE000 C004 (DLAB=1) | | | |
| U0IER | Interrupt Enable Register | - | - | - | - | - | - | En.ABTO | En.ABEO | R/W | 0x00 | 0xE000 C004 (DLAB=0) | | | |
| | | - | - | - | - | - | - | En.RX Lin.St.Int | Enable THRE Int | | | | | | |
| U0IIR | Interrupt ID Reg. | - | - | - | - | - | - | ABTO Int | ABEO Int | RO | 0x01 | 0xE000 C008 | | | |
| | | FIFOs Enabled | | - | - | IIR3 | IIR2 | IIR1 | IIR0 | | | | | | |
| U0FCR | FIFO Control Register | RX Trigger | | - | - | - | TX FIFO Reset | RX FIFO Reset | FIFO Enable | WO | 0x00 | 0xE000 C008 | | | |
| U0LCR | Line Control Register | DLAB | Set Break | Stick Parity | Even Par.Selct. | Parity Enable | No. of Stop Bits | Word Length Select | | R/W | 0x00 | 0xE000 C00C | | | |
| U0LSR | Line Status Register | RX FIFO Error | TEMT | THRE | BI | FE | PE | OE | DR | RO | 0x60 | 0xE000 C014 | | | |
| U0SCR | Scratch Pad Reg. | 8-bit Data | | | | | | | | R/W | 0x00 | 0xE000 C01C | | | |
| U0ACR | Auto-baud Control Register | - | - | - | - | - | - | ABTO Int.Clr | ABEO Int.Clr | R/W | 0x00 | 0xE000 C020 | | | |
| | | - | - | - | - | - | - | Aut.Rstrt. | Mode | | | | | | |
| U0FDR | Fractional Divider Register | Reserved[31:8] | | | | | | | | | 0x10 | 0xE000 C028 | | | |
| | | MulVal | | | | DivAddVal | | | | | | | | | |
| U0TER | TX. Enable Reg. | TXEN | - | - | - | - | - | - | - | R/W | 0x80 | 0xE000 C030 | | | |

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

9.3.1 UART0 Receiver Buffer Register (U0RBR - 0xE000 C000, when DLAB = 0, Read Only)

The U0RBR is the top byte of the UART0 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 U0LCR must be zero in order to access the U0RBR. The U0RBR 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 U0RBR.

Table 97: UART0 Receiver Buffer Register (U0RBR - address 0xE000 C000, when DLAB = 0, Read Only) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 7:0 | RBR | The UART0 Receiver Buffer Register contains the oldest received byte in the UART0 Rx FIFO. | undefined |

9.3.2 UART0 Transmit Holding Register (U0THR - 0xE000 C000, when DLAB = 0, Write Only)

The U0THR is the top byte of the UART0 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 U0LCR must be zero in order to access the U0THR. The U0THR is always Write Only.

Table 98: UART0 Transmit Holding Register (U0THR - address 0xE000 C000, when DLAB = 0, Write Only) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 7:0 | THR | Writing to the UART0 Transmit Holding Register causes the data to be stored in the UART0 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available. | NA |

9.3.3 UART0 Divisor Latch Registers (U0DLL - 0xE000 C000 and U0DLM - 0xE000 C004, when DLAB = 1)

The UART0 Divisor Latch is part of the UART0 Fractional Baud Rate Generator and holds the value used to divide the clock supplied by the fractional prescaler in order to produce the baud rate clock, which must be 16x the desired baud rate ([Equation 1](#)). The U0DLL and U0DLM registers together form a 16 bit divisor where U0DLL contains the lower 8 bits of the divisor and U0DLM 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 U0LCR must be one in order to access the UART0 Divisor Latches.

Details on how to select the right value for U0DLL and U0DLM can be found later on in this chapter.

Table 99: UART0 Divisor Latch LSB register (U0DLL - address 0xE000 C000, when DLAB = 1) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|---|-------------|
| 7:0 | DLL | The UART0 Divisor Latch LSB Register, along with the U0DLM register, determines the baud rate of the UART0. | 0x01 |

Table 100: UART0 Divisor Latch MSB register (U0DLM - address 0xE000 C004, when DLAB = 1) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|---|-------------|
| 7:0 | DLM | The UART0 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART0. | 0x00 |

9.3.4 UART0 Fractional Divider Register (U0FDR - 0xE000 C028)

The UART0 Fractional Divider Register (U0FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at user's discretion. This pre-scaler takes the VPB clock and generates an output clock per specified fractional requirements.

Table 101: UART0 Fractional Divider Register (U0FDR - address 0xE000 C028) bit description

| Bit | Function | Description | Reset value |
|------|-----------|--|-------------|
| 3:0 | DIVADDVAL | Baudrate generation pre-scaler divisor value. If this field is 0, fractional baudrate generator will not impact the UART0 baudrate. | 0 |
| 7:4 | MULVAL | Baudrate pre-scaler multiplier value. This field must be greater or equal 1 for UART0 to operate properly, regardless of whether the fractional baudrate generator is used or not. | 1 |
| 31:8 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of UART0 disabled making sure that UART0 is fully software and hardware compatible with UARTs not equipped with this feature.

UART0 baudrate can be calculated as:

(1)

$$UART0_{baudrate} = \frac{PCLK}{16 \times (16 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where PCLK is the peripheral clock, U0DLM and U0DLL are the standard UART0 baud rate divider registers, and DIVADDVAL and MULVAL are UART0 fractional baudrate 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 U0FDR 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.

The value of the U0FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

Usage Note: For practical purposes, UART0 baudrate formula can be written in a way that identifies the part of a UART baudrate generated without the fractional baudrate generator, and the correction factor that this module adds:

(2)

$$UART0_{baudrate} = \frac{PCLK}{16 \times (16 \times U0DLM + U0DLL)} \times \frac{MulVal}{(MulVal + DivAddVal)}$$

Based on this representation, fractional baudrate generator contribution can also be described as a prescaling with a factor of MULVAL / (MULVAL + DIVADDVAL).

9.3.5 UART0 baudrate calculation

Example 1: Using UART0baudrate formula from above, it can be determined that system with PCLK = 20 MHz, U0DL = 130 (U0DLM = 0x00 and U0DLL = 0x82), DIVADDVAL = 0 and MULVAL = 1 will enable UART0 with UART0baudrate = 9615 bauds.

Example 2: Using UART0baudrate formula from above, it can be determined that system with PCLK = 20 MHz, U0DL = 93 (U0DLM = 0x00 and U0DLL = 0x5D), DIVADDVAL = 2 and MULVAL = 5 will enable UART0 with UART0baudrate = 9600 bauds.

Table 102: Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

| Desired baudrate | MULVAL = 0 DIVADDVAL = 0 | | Optimal MULVAL & DIVADDVAL | | |
|------------------|--------------------------|------------------------|----------------------------|-----------------------------|------------------------|
| | U0DLM:U0DLL | % error ^[3] | U0DLM:U0DLL | Fractional pre-scaler value | % error ^[3] |
| | hex ^[2] | dec ^[1] | dec ^[1] | MULDIV | MULDIV + DIVADDVAL |
| 50 | 61A8 | 25000 | 0.0000 | 25000 | 1/(1+0) |
| 75 | 411B | 16667 | 0.0020 | 12500 | 3/(3+1) |
| 110 | 2C64 | 11364 | 0.0032 | 6250 | 11/(11+9) |
| 134.5 | 244E | 9294 | 0.0034 | 3983 | 3/(3+4) |
| 150 | 208D | 8333 | 0.0040 | 6250 | 3/(3+1) |
| 300 | 1047 | 4167 | 0.0080 | 3125 | 3/(3+1) |
| 600 | 0823 | 2083 | 0.0160 | 1250 | 3/(3+2) |
| 1200 | 0412 | 1042 | 0.0320 | 625 | 3/(3+2) |
| 1800 | 02B6 | 694 | 0.0640 | 625 | 9/(9+1) |
| 2000 | 0271 | 625 | 0.0000 | 625 | 1/(1+0) |
| 2400 | 0209 | 521 | 0.0320 | 250 | 12/(12+13) |
| 3600 | 015B | 347 | 0.0640 | 248 | 5/(5+2) |
| 4800 | 0104 | 260 | 0.1600 | 125 | 12/(12+13) |

Table 102: Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

| Desired baudrate | MULVAL = 0 DIVADDVAL = 0 | | Optimal MULVAL & DIVADDVAL | | Fractional pre-scaler value MULDIV | % error ^[3] |
|------------------|-----------------------------------|--------------------|-----------------------------------|------------------------------|---------------------------------------|------------------------|
| | U0DLM:U0DLL hex ^[2] | dec ^[1] | U0DLM:U0DLL dec ^[1] | MULDIV MULDIV + DIVADDVAL | | |
| 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 baudrate.

9.3.6 UART0 Interrupt Enable Register (U0IER - 0xE000 C004, when DLAB = 0)

The U0IER is used to enable UART0 interrupt sources.

Table 103: UART0 Interrupt Enable Register (U0IER - address 0xE000 C004, when DLAB = 0) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------------------------------|-------|---|-------------|
| 0 | RBR Interrupt Enable | 0 | U0IER[0] enables the Receive Data Available interrupt for UART0. 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 | U0IER[1] enables the THRE interrupt for UART0. The status of this can be read from U0LSR[5]. | 0 |
| | | | 0 Disable the THRE interrupts. | |
| | | | 1 Enable the THRE interrupts. | |
| 2 | RX Line Status Interrupt Enable | 0 | U0IER[2] enables the UART0 RX line status interrupts. The status of this interrupt can be read from U0LSR[4:1]. | 0 |
| | | | 0 Disable the RX line status interrupts. | |
| | | | 1 Enable the RX line status interrupts. | |
| 7:4 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

Table 103: UART0 Interrupt Enable Register (U0IER - address 0xE000 C004, when DLAB = 0)
bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 8 | ABTOIntEn | 0 | U1IER8 enables the auto-baud time-out interrupt. | 0 |
| | | | Disable Auto-baud Time-out Interrupt. | |
| | | | Enable Auto-baud Time-out Interrupt. | |
| 9 | ABEOIntEn | 0 | U1IER9 enables the end of auto-baud interrupt. | 0 |
| | | | Disable End of Auto-baud Interrupt. | |
| | | | 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 |

9.3.7 UART0 Interrupt Identification Register (U0IIR - 0xE000 C008, Read Only)

The U0IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U0IIR access. If an interrupt occurs during an U0IIR access, the interrupt is recorded for the next U0IIR access.

Table 104: UART0 Interrupt Identification Register (U0IIR - address 0xE000 C008, read only)
bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------------------------|-------|--|-------------|
| 0 | Interrupt Pending | 0 | Note that U0IIR[0] is active low. The pending interrupt can be determined by evaluating U0IIR[3:1]. | 1 |
| | | | At least one interrupt is pending. | |
| | | | No pending interrupts. | |
| 3:1 | Interrupt Identification | 011 | U0IER[3:1] identifies an interrupt corresponding to the UART0 Rx FIFO. All other combinations of U0IER[3:1] not listed above are reserved (000,100,101,111). | 0 |
| | | | 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 U0FCR[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. | |
| 31:10 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

Interrupts are handled as described in [Table 105](#). Given the status of U0IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U0IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART0 RLS interrupt ($U0IIR[3:1] = 011$) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART0 Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART0 Rx error condition that set the interrupt can be observed via $U0LSR[4:1]$. The interrupt is cleared upon an $U0LSR$ read.

The UART0 RDA interrupt ($U0IIR[3:1] = 010$) shares the second level priority with the CTI interrupt ($U0IIR[3:1] = 110$). The RDA is activated when the UART0 Rx FIFO reaches the trigger level defined in $U0FCR[7:6]$ and is reset when the UART0 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 ($U0IIR[3:1] = 110$) is a second level interrupt and is set when the UART0 Rx FIFO contains at least one character and no UART0 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART0 Rx FIFO activity (read or write of UART0 RSR) will clear the interrupt. This interrupt is intended to flush the UART0 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 105: UART0 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] | $U0LSR$ Read ^[2] |
| 0100 | Second | RX Data Available | Rx data available or trigger level reached in FIFO ($U0FCR0=1$) | U0RBR Read ^[3] or UART0 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: $[(\text{word length}) \times 7 - 2] \times 8 + [(\text{trigger level} - \text{number of characters}) \times 8 + 1] \text{ RCLKs}$ | U0RBR Read ^[3] |
| 0010 | Third | THRE | THRE ^[2] | $U0IIR$ 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 9.3.10 "UART0 Line Status Register \(\$U0LSR\$ - 0xE000 C014, Read Only\)"](#)

[3] For details see [Section 9.3.1 "UART0 Receiver Buffer Register \(\$U0RBR\$ - 0xE000 C000, when DLAB = 0, Read Only\)"](#)

[4] For details see [Section 9.3.7 "UART0 Interrupt Identification Register \(\$U0IIR\$ - 0xE000 C008, Read Only\)"](#)
and [Section 9.3.2 "UART0 Transmit Holding Register \(\$U0THR\$ - 0xE000 C000, when DLAB = 0, Write Only\)"](#)

The UART0 THRE interrupt ($U0IIR[3:1] = 001$) is a third level interrupt and is activated when the UART0 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART0 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 U0THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U0THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART0 THR FIFO has held two or more characters at one time and currently, the U0THR is empty. The THRE interrupt is reset when a U0THR write occurs or a read of the U0IIR occurs and the THRE is the highest interrupt (U0IIR[3:1] = 001).

9.3.8 UART0 FIFO Control Register (U0FCR - 0xE000 C008)

The U0FCR controls the operation of the UART0 Rx and TX FIFOs.

Table 106: UART0 FIFO Control Register (U0FCR - address 0xE000 C008) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------------|-------|--|-------------|
| 0 | FIFO Enable | 0 | UART0 FIFOs are disabled. Must not be used in the application. | 0 |
| | | 1 | Active high enable for both UART0 Rx and TX FIFOs and U0FCR[7:1] access. This bit must be set for proper UART0 operation. Any transition on this bit will automatically clear the UART0 FIFOs. | |
| 1 | RX FIFO Reset | 0 | No impact on either of UART0 FIFOs. | 0 |
| | | 1 | Writing a logic 1 to U0FCR[1] will clear all bytes in UART0 Rx FIFO and reset the pointer logic. This bit is self-clearing. | |
| 2 | TX FIFO Reset | 0 | No impact on either of UART0 FIFOs. | 0 |
| | | 1 | Writing a logic 1 to U0FCR[2] will clear all bytes in UART0 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 | 00 | These two bits determine how many receiver UART0 FIFO characters must be written before an interrupt is activated. | 0 |
| | | | 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) | |

9.3.9 UART0 Line Control Register (U0LCR - 0xE000 C00C)

The U0LCR determines the format of the data character that is to be transmitted or received.

Table 107: UART0 Line Control Register (U0LCR - address 0xE000 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 | |

Table 107: UART0 Line Control Register (U0LCR - address 0xE000 C00C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------------------------------|-------|--|-------------|
| 2 | Stop Bit Select | 0 | 1 stop bit. | 0 |
| | | 1 | 2 stop bits (1.5 if U0LCR[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 U0LCR[6] is active high. | |
| 7 | Divisor Latch Access Bit (DLAB) | 0 | Disable access to Divisor Latches. | 0 |
| | | 1 | Enable access to Divisor Latches. | |

9.3.10 UART0 Line Status Register (U0LSR - 0xE000 C014, Read Only)

The U0LSR is a read-only register that provides status information on the UART0 TX and RX blocks.

Table 108: UART0 Line Status Register (U0LSR - address 0xE000 C014, read only) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------------------------|-------|--|-------------|
| 0 | Receiver Data Ready (RDR) | 0 | U0LSR0 is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty. | 0 |
| | | 0 | U0RBR is empty. | |
| | | 1 | U0RBR contains valid data. | |
| 1 | Overrun Error (OE) | 0 | The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost. | 0 |
| | | 0 | Overrun error status is inactive. | |
| | | 1 | Overrun error status is active. | |
| 2 | Parity Error (PE) | 0 | When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR[2]. Time of parity error detection is dependent on U0FCR[0]. | 0 |
| | | 0 | Note: A parity error is associated with the character at the top of the UART0 RBR FIFO. | |
| | | 1 | Parity error status is inactive. | |

Table 108: UART0 Line Status Register (U0LSR - address 0xE000 C014, read only) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--|-------|---|-------------|
| 3 | Framing Error (FE) | | When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR[3]. The time of the framing error detection is dependent on U0FCR0. 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 UART0 RBR FIFO. | 0 |
| | | 0 | Framing error status is inactive. | |
| | | 1 | Framing error status is active. | |
| 4 | Break Interrupt (BI) | | When RXD0 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 RXD0 goes to marking state (all 1's). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCR[0]. Note: The break interrupt is associated with the character at the top of the UART0 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 UART0 THR and is cleared on a U0THR write. | 1 |
| | | 0 | U0THR contains valid data. | |
| | | 1 | U0THR is empty. | |
| 6 | Transmitter Empty (TEMT) | | TEM is set when both U0THR and U0TSR are empty; TEM is cleared when either the U0TSR or the U0THR contain valid data. | 1 |
| | | 0 | U0THR and/or the U0TSR contains valid data. | |
| | | 1 | U0THR and the U0TSR are empty. | |
| 7 | Error in RX FIFO (RXFE) | | U0LSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO. | 0 |
| | | 0 | U0RBR contains no UART0 RX errors or U0FCR[0]=0. | |
| | | 1 | UART0 RBR contains at least one UART0 RX error. | |

9.3.11 UART0 Scratch pad register (U0SCR - 0xE000 C01C)

The U0SCR has no effect on the UART0 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 U0SCR has occurred.

Table 109: UART0 Scratch pad register (U0SCR - address 0xE000 C01C) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|----------------------------|-------------|
| 7:0 | Pad | A readable, writable byte. | 0x00 |

9.3.12 UART0 Auto-baud Control Register (U0ACR - 0xE000 C020)

The UART0 Auto-baud Control Register (U0ACR) 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 110: Auto-baud Control Register (U0ACR - 0xE000 C020) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-------------|-------|---|-------------|
| 0 | Start | 0 | 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 | 0 | 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) | |
| 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 U0IIR. 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 U0IIR. 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 |

9.3.13 Auto-baud

The UART0 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 U0DLM and U0DLL accordingly.

Auto-baud is started by setting the U0ACR Start bit. Auto-baud can be stopped by clearing the U0ACR 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 U0ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART0 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 UART0 Rx pin (the length of the start bit).

The U0ACR 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 UART0 Rx pin.

The auto-baud function can generate two interrupts.

- The U0IIR ABTOInt interrupt will get set if the interrupt is enabled (U0IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U0IIR ABEoInt interrupt will get set if the interrupt is enabled (U0IER ABEoIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U0ACR 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 UART0 Rx pin baud-rate, but the value of the U0FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U0DLM and U0DLL registers should be done before U0ACR register write. The minimum and the maximum baudrates supported by UART0 are function of PCLK, number of data bits, stop-bits and parity bits.

(3)

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART0_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

9.3.14 UART0 Transmit Enable Register (U0TER - 0xE000 C030)

LPC2141/2/4/6/8's U0TER enables implementation of software flow control. When TXEn=1, UART0 transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART0 transmission will stop.

[Table 111](#) describes how to use TXEn bit in order to achieve software flow control.

Table 111: UART0 Transmit Enable Register (U0TER - address 0xE000 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 |

9.3.15 Auto-baud Modes

When the software is expecting an "AT" command, it configures the UART0 with the expected character format and sets the U0ACR Start bit. The initial values in the divisor latches U0DLM and U0DLL don't care. Because of the "A" or "a" ASCII coding ("A" = 0x41, "a" = 0x61), the UART0 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U0ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U0ACR Start bit setting, the baud-rate measurement counter is reset and the UART0 U0RSR is reset. The U0RSR baud rate is switch to the highest rate.
2. A falling edge on UART0 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) UART0 input clock, guaranteeing the start bit is stored in the U0RSR.
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 UART0 input clock (PCLK).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART0 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART0 Rx pin.
6. The rate counter is loaded into U0DLM/U0DLL and the baud-rate will be switched to normal operation. After setting the U0DLM/U0DLL the end of auto-baud interrupt U0IIR ABEOInt will be set, if enabled. The U0RSR will now continue receiving the remaining bits of the "A/a" character.

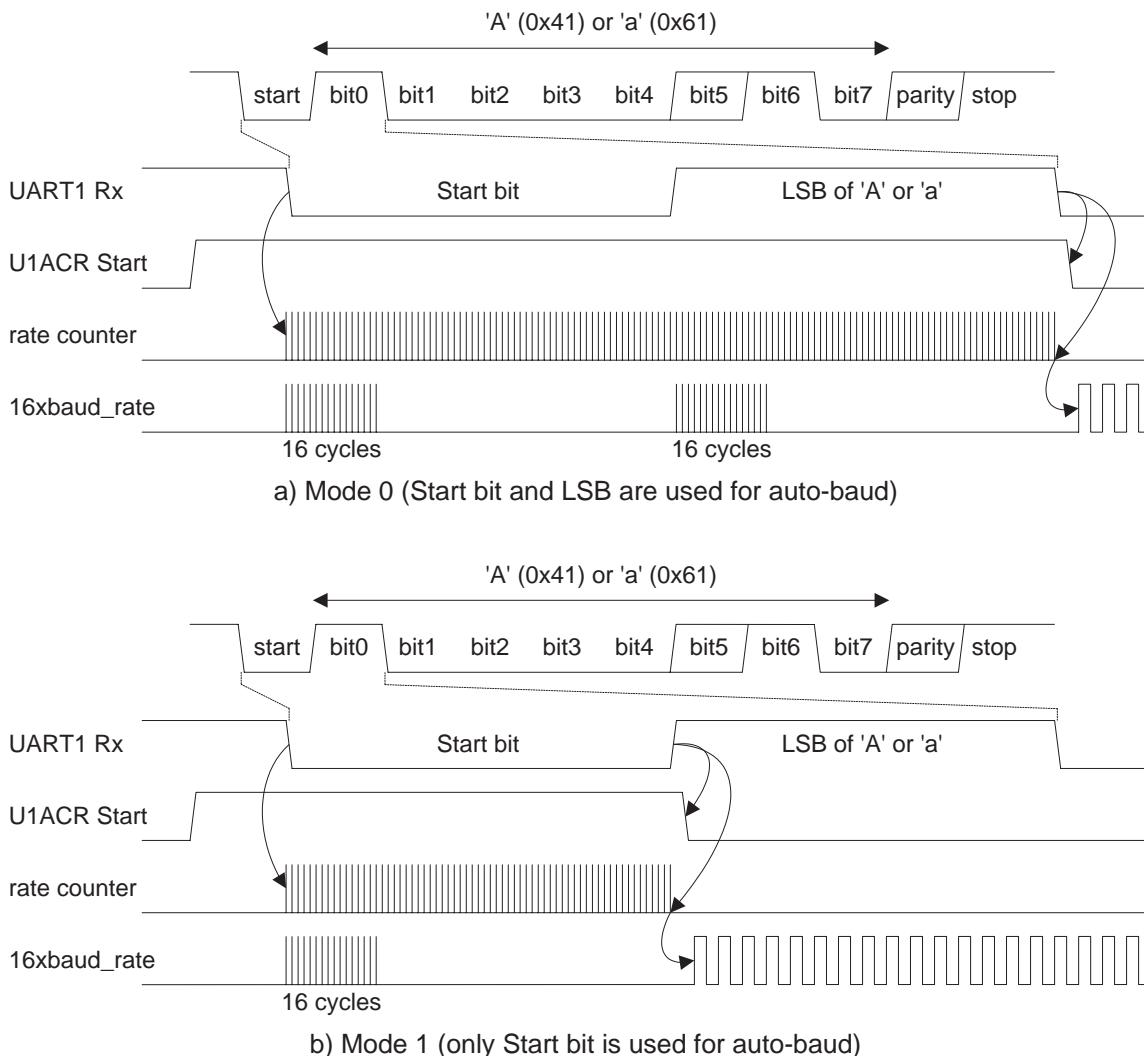


Fig 18. Autobaud Mode 0 and Mode 1 waveform

9.4 Architecture

The architecture of the UART0 is shown below in the block diagram.

The VPB interface provides a communications link between the CPU or host and the UART0.

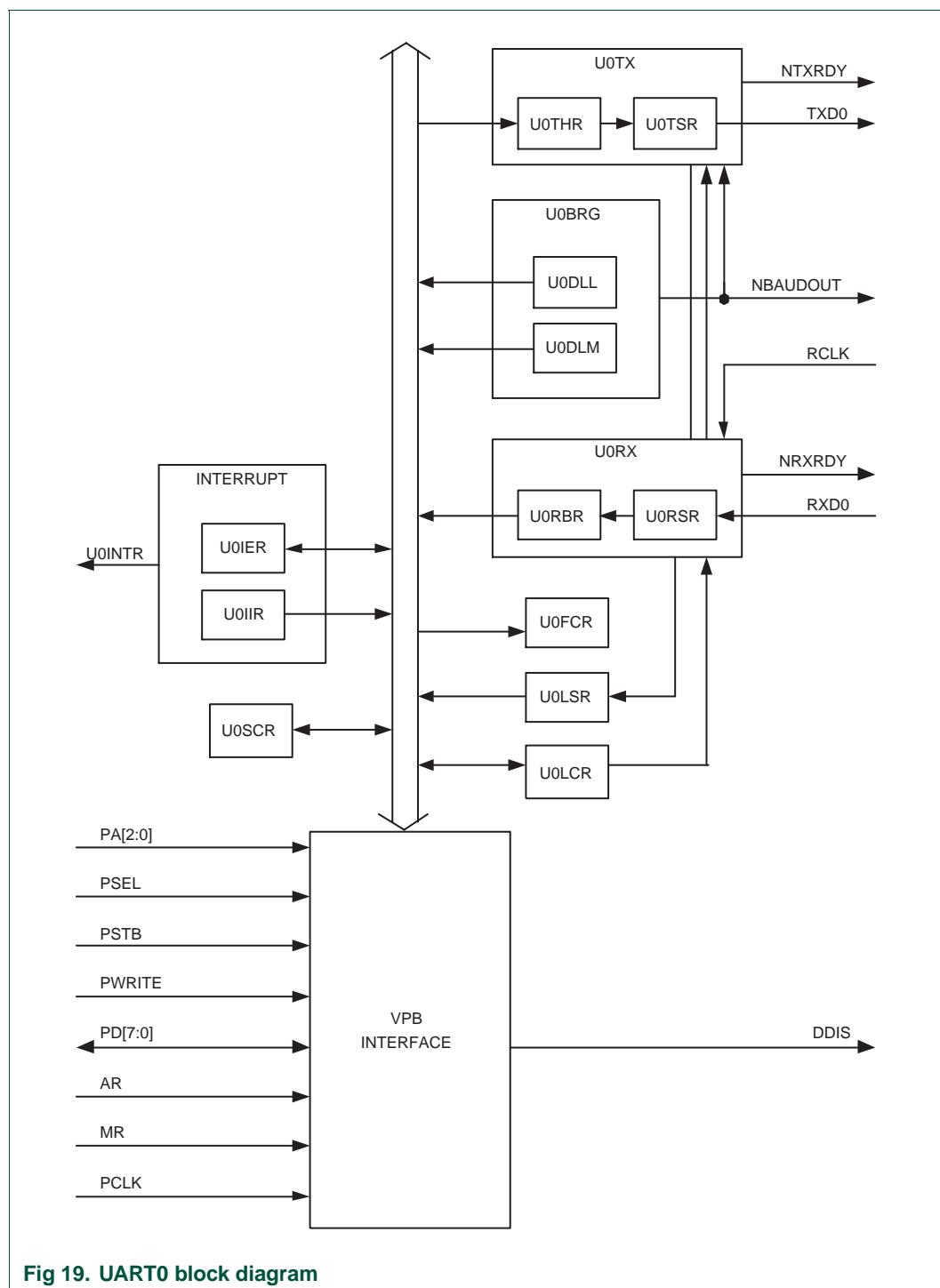
The UART0 receiver block, U0RX, monitors the serial input line, RXD0, for valid input. The UART0 RX Shift Register (U0RSR) accepts valid characters via RXD0. After a valid character is assembled in the U0RSR, it is passed to the UART0 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART0 transmitter block, U0TX, accepts data written by the CPU or host and buffers the data in the UART0 TX Holding Register FIFO (U0THR). The UART0 TX Shift Register (U0TSR) reads the data stored in the U0THR and assembles the data to transmit via the serial output pin, TXD0.

The UART0 Baud Rate Generator block, U0BRG, generates the timing enables used by the UART0 TX block. The U0BRG clock input source is the VPB clock (PCLK). The main clock is divided down per the divisor specified in the U0DLL and U0DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The interrupt interface contains registers U0IER and U0IIR. The interrupt interface receives several one clock wide enables from the U0TX and U0RX blocks.

Status information from the U0TX and U0RX is stored in the U0LSR. Control information for the U0TX and U0RX is stored in the U0LCR.





10.1 Features

- UART1 is identical to UART0, 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 fractional baud rate generator with autobauding capabilities.
- Mechanism that enables software and hardware flow control implementation.
- Standard modem interface signals included with flow control (auto-CTS/RTS) fully supported in hardware (LPC2144/6/8 only).

10.2 Pin description

Table 112: UART1 pin description

| Pin | Type | Description |
|--------------------|--------|--|
| RXD1 | Input | Serial Input. Serial receive data. |
| TXD1 | Output | Serial Output. Serial transmit data. |
| CTS ^[1] | 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). |
| DCD ^[1] | 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). |
| DSR ^[1] | 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). |
| DTR ^[1] | 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]. |
| RI ^[1] | 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). |
| RTS ^[1] | 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]. |

[1] LPC2144/6/8 only.

10.3 Register description

UART1 contains registers organized as shown in Table 76. The Divisor Latch Access Bit (DLAB) is contained in U1LCR[7] and enables access to the Divisor Latches.

Table 113: 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 | - | - | - | - | - | - | - | En.ABTO En.ABEO | R/W | 0x00 | 0xE001 0004 (DLAB=0) |
| | | En.CTS Int ^[2] | - | - | - | E.Modem St.Int ^[2] | En. RX Lin.St. Int | Enable THRE Int | En. RX Dat.Av.Int | | | |
| U1IIR | Interrupt ID Reg. | - | - | - | - | - | - | - | ABTO Int ABEO Int | RO | 0x01 | 0xE001 0008 |
| | | FIFOs Enabled | | - | - | IIR3 | IIR2 | IIR1 | IIR0 | | | |
| U1FCR | FIFO Control Register | RX Trigger | | - | - | - | TX FIFO Reset | RX FIFO Reset | FIFO Enable | WO | 0x00 | 0xE001 0008 |
| U1LCR | Line Control Register | DLAB | Set Break | Stick Parity | Even Par.Selct. | Parity Enable | No. of Stop Bits | Word Length Select | | R/W | 0x00 | 0xE001 000C |
| U1MCR ^[2] | Modem Ctrl. Reg. | CTSen | RTSen | - | LoopBck. | - | - | 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 ^[2] | Modem Status Register | DCD | RI | DSR | CTS | Delta DCD | Trailing Edge RI | Delta DSR | Delta CTS | RO | 0x00 | 0xE001 0018 |
| U1SCR | Scratch Pad Reg. | 8-bit Data | | | | | | | | R/W | 0x00 | 0xE001 001C |
| U1ACR | Auto-baud Control Register | - | - | - | - | - | - | ABTO IntClr | ABEO IntClr | R/W | 0x00 | 0xE001 0020 |
| | | - | - | - | - | - | - | Aut.Rstrt. | Mode Start | | | |
| U1FDR | Fractional Divider Register | Reserved[31:8] | | | | | | | | R/W | 0x10 | 0xE001 0028 |
| | | MulVal | | | | DivAddVal | | | | | | |
| U1TER | TX. Enable Reg. | TXEN | - | - | - | - | - | - | - | R/W | 0x80 | 0xE001 0030 |

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

[2] Modem specific features are available in LPC2144/6/8 only.

10.3.1 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 114: 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 |

10.3.2 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 115: 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 |

10.3.3 UART1 Divisor Latch Registers 0 and 1 (U1DLL - 0xE001 0000 and U1DLM - 0xE001 0004, when DLAB = 1)

The UART1 Divisor Latch is part of the UART1 Fractional Baud Rate Generator and holds the value used to divide the clock supplied by the fractional prescaler in order to produce the baud rate clock, which must be 16x the desired baud rate ([Equation 4](#)). 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.

Table 116: 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 117: 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 |

10.3.4 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. This pre-scaler takes the VPB clock and generates an output clock per specified fractional requirements.

Table 118: UART1 Fractional Divider Register (U1FDR - address 0xE001 0028) bit description

| Bit | Function | Description | Reset value |
|------|-----------|--|-------------|
| 3:0 | DIVADDVAL | Baudrate generation pre-scaler divisor value. If this field is 0, fractional baudrate generator will not impact the UART1 baudrate. | 0 |
| 7:4 | MULVAL | Baudrate pre-scaler multiplier value. This field must be greater or equal 1 for UART1 to operate properly, regardless of whether the fractional baudrate generator is used or not. | 1 |
| 31:8 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of UART1 disabled making sure that UART1 is fully software and hardware compatible with UARTs not equipped with this feature.

UART1 baudrate can be calculated as:

(4)

$$UART1_{baudrate} = \frac{PCLK}{16 \times (16 \times U1DLM + U1DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where PCLK is the peripheral clock, U1DLM and U1DLL are the standard UART1 baud rate divider registers, and DIVADDVAL and MULVAL are UART1 fractional baudrate 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.

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, UART1 baudrate formula can be written in a way that identifies the part of a UART baudrate generated without the fractional baudrate generator, and the correction factor that this module adds:

(5)

$$UART1_{baudrate} = \frac{PCLK}{16 \times (16 \times U1DLM + U1DLL)} \times \frac{MulVal}{(MulVal + DivAddVal)}$$

Based on this representation, fractional baudrate generator contribution can also be described as a prescaling with a factor of MULVAL / (MULVAL + DIVADDVAL).

10.3.5 UART1 baudrate calculation

Example 1: Using UART1baudrate formula from above, it can be determined that system with PCLK = 20 MHz, U1DL = 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 119: Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

| Desired baudrate | MULVAL = 0 DIVADDVAL = 0 | | Optimal MULVAL & DIVADDVAL | | |
|------------------|--------------------------|------------------------|----------------------------|-----------------------------|------------------------|
| | U1DLM:U1DLL | % error ^[3] | U1DLM:U1DLL | Fractional pre-scaler value | % error ^[3] |
| | hex ^[2] | dec ^[1] | dec ^[1] | MULDIV | MULDIV + DIVADDVAL |
| 50 | 61A8 | 25000 | 0.0000 | 25000 | 1/(1+0) |
| 75 | 411B | 16667 | 0.0020 | 12500 | 3/(3+1) |
| 110 | 2C64 | 11364 | 0.0032 | 6250 | 11/(11+9) |
| 134.5 | 244E | 9294 | 0.0034 | 3983 | 3/(3+4) |
| 150 | 208D | 8333 | 0.0040 | 6250 | 3/(3+1) |
| 300 | 1047 | 4167 | 0.0080 | 3125 | 3/(3+1) |
| 600 | 0823 | 2083 | 0.0160 | 1250 | 3/(3+2) |
| 1200 | 0412 | 1042 | 0.0320 | 625 | 3/(3+2) |
| 1800 | 02B6 | 694 | 0.0640 | 625 | 9/(9+1) |
| 2000 | 0271 | 625 | 0.0000 | 625 | 1/(1+0) |
| 2400 | 0209 | 521 | 0.0320 | 250 | 12/(12+13) |
| 3600 | 015B | 347 | 0.0640 | 248 | 5/(5+2) |
| 4800 | 0104 | 260 | 0.1600 | 125 | 12/(12+13) |

Table 119: Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

| Desired baudrate | MULVAL = 0 DIVADDVAL = 0 | | Optimal MULVAL & DIVADDVAL | | Fractional pre-scaler value | % error ^[3] |
|------------------|-----------------------------------|--------------------|-----------------------------------|--------|-----------------------------|------------------------|
| | U1DLM:U1DLL hex ^[2] | dec ^[1] | U1DLM:U1DLL dec ^[1] | MULDIV | | |
| | MULDIV + DIVADDVAL | | | | | |
| 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 baudrate.

10.3.6 UART1 Interrupt Enable Register (U1IER - 0xE001 0004, when DLAB = 0)

The U1IER is used to enable UART1 interrupt sources.

Table 120: UART1 Interrupt Enable Register (U1IER - address 0xE001 0004, when DLAB = 0) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---|-------|---|-------------|
| 0 | RBR Interrupt Enable | 0 | 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 | THRE Interrupt Enable | 0 | 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. | |
| 2 | RX Line Interrupt Enable | 0 | 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. | |
| 3 | Modem Status Interrupt Enable ^[1] | 0 | 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. | |

Table 120: UART1 Interrupt Enable Register (U1IER - address 0xE001 0004, when DLAB = 0) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|---|---|--|-------------|
| 6:4 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 7 | CTS Interrupt Enable ^[1] | 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. | 0 | |
| | | 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 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 |

[1] Available in LPC2144/6/8 only. In all other LPC214x parts this bit is Reserved.

10.3.7 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 121: UART1 Interrupt Identification Register (U1IIR - address 0xE001 0008, read only) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------------------|---|-------------|-------------|
| 0 | Interrupt Pending | 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 121: UART1 Interrupt Identification Register (U1IIR - address 0xE001 0008, read only)
bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------------------------|------------|---|-------------|
| 3:1 | 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. ^[1] | |
| 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 |

[1] LPC2144/6/8 only. For all other LPC214x devices '000' combination is Reserved.

Interrupts are handled as described in Table 83. 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 122: 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 ^[3] or PE ^[3] or FE ^[3] or BI ^[3] | U1LSR Read ^[3] |
| 0100 | Second | RX Data Available | Rx data available or trigger level reached in FIFO (U1FCR0=1) | U1RBR Read ^[4] 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: $[(\text{word length}) \times 7 - 2] \times 8 + [(\text{trigger level} - \text{number of characters}) \times 8 + 1] \text{ RCLKs}$ | U1RBR Read ^[4] |
| 0010 | Third | THRE | THRE ^[3] | U1IIR Read ^[5] (if source of interrupt) or THR write |
| 0000 ^[2] | Fourth | Modem Status | CTS or DSR or RI or DCD | MSR Read |

[1] Values "0000" (see [Table note 2](#)), "0011", "0101", "0111", "1000", "1001", "1010", "1011", "1101", "1110", "1111" are reserved.

[2] LPC2144/6/8 only.

[3] For details see [Section 10.3.11 "UART1 Line Status Register \(U1LSR - 0xE001 0014, Read Only\)"](#)

[4] For details see [Section 10.3.1 "UART1 Receiver Buffer Register \(U1RBR - 0xE001 0000, when DLAB = 0 Read Only\)"](#)

[5] For details see [Section 10.3.7 "UART1 Interrupt Identification Register \(U1IIR - 0xE001 0008, Read Only\)"](#) and [Section 10.3.2 "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).

The modem interrupt (U1IIR[3:1] = 000) is available in LPC2144/6/8 only. 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.

10.3.8 UART1 FIFO Control Register (U1FCR - 0xE001 0008)

The U1FCR controls the operation of the UART1 RX and TX FIFOs.

Table 123: UART1 FIFO Control Register (U1FCR - address 0xE001 0008) 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). | |

10.3.9 UART1 Line Control Register (U1LCR - 0xE001 000C)

The U1LCR determines the format of the data character that is to be transmitted or received.

Table 124: 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. | |
| 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. | |

Table 124: UART1 Line Control Register (U1LCR - address 0xE001 000C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------------------------------|-------|--|-------------|
| 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. | |

10.3.10 UART1 Modem Control Register (U1MCR - 0xE001 0010), LPC2144/6/8 only

The U1MCR enables the modem loopback mode and controls the modem output signals.

Table 125: UART1 Modem Control Register (U1MCR - address 0xE001 0010), LPC2144/6/8 only 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 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 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. | |
| 5:3 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 6 | RTSen | | Auto-RTS control bit. | 0 |
| | | 0 | Disable auto-RTS flow control. | |
| | | 1 | Enable auto-RTS flow control. | |
| 7 | CTSen | | Auto-CTS control bit. | 0 |
| | | 0 | Disable auto-CTS flow control. | |
| | | 1 | Enable auto-CTS flow control. | |

Auto-flow control (LPC2144/6/8 only)

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.

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 123 on page 122). The RTS1 output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

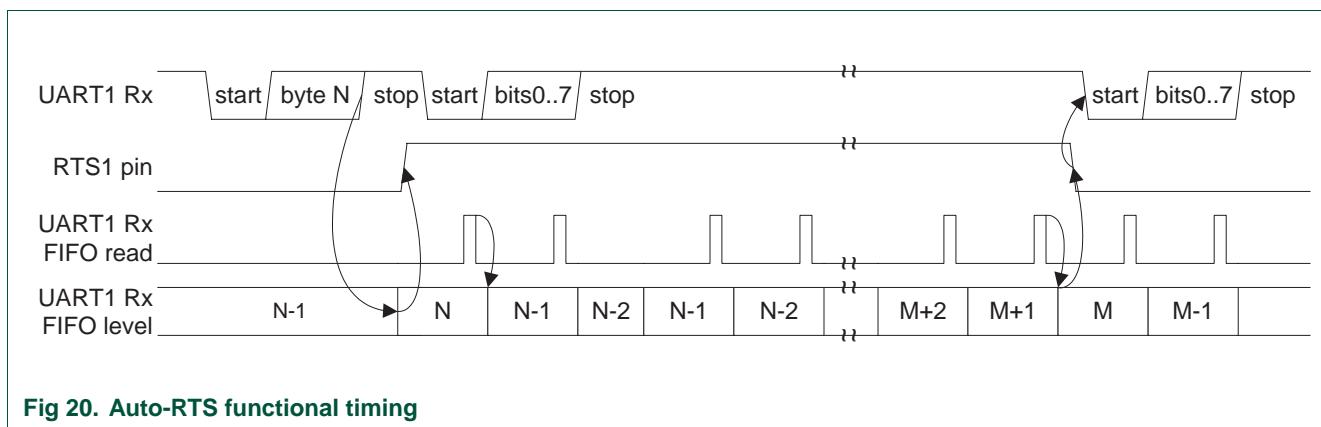


Fig 20. Auto-RTS functional timing

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 126](#) lists the conditions for generating a Modem Status interrupt.

Table 126: Modem status interrupt generation

| Enable Modem Status Interrupt (U1IER[3]) | CTS _n (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 21](#) illustrates the auto-CTS functional timing.

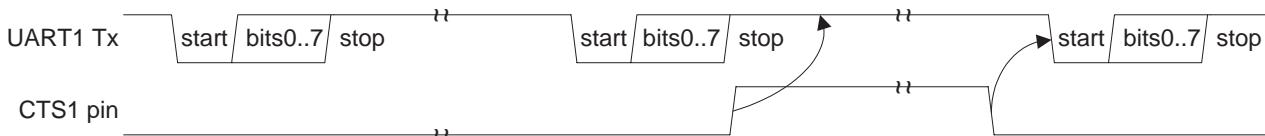


Fig 21. Auto-CTS functional timing

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.

10.3.11 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 127: 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. | |

Table 127: UART1 Line Status Register (U1LSR - address 0xE001 0014, read only) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---|--|--|--|
| 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. | 0 |
| 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]. | 0 0: Parity error status is inactive. 1: Parity error status is active. | 0 Note: A parity error is associated with the character at the top of the UART1 RBR FIFO. |
| 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. | 0 0: Framing error status is inactive. 1: Framing error status is active. | 0 Note: A framing error is associated with the character at the top of the UART1 RBR FIFO. |
| 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]. | 0 0: Break interrupt status is inactive. 1: Break interrupt status is active. | 0 Note: The break interrupt is associated with the character at the top of the UART1 RBR FIFO. |
| 5 | Transmitter Holding Register Empty (THRE) | THRE is set immediately upon detection of an empty UART1 THR and is cleared on a U1THR write. | 0 0: U1THR contains valid data. 1: U1THR is empty. | 1 |
| 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. | 0 0: U1THR and/or the U1TSR contains valid data. 1: U1THR and the U1TSR are empty. | 1 |
| 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. | 0 |

10.3.12 UART1 Modem Status Register (U1MSR - 0xE001 0018), LPC2144/6/8 only

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 128: UART1 Modem Status Register (U1MSR - address 0xE001 0018), LPC2144/6/8 only 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 |
| 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 |

10.3.13 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 129: UART1 Scratch pad register (U1SCR - address 0xE001 0014) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|----------------------------|-------------|
| 7:0 | Pad | A readable, writable byte. | 0x00 |

10.3.14 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 130: Auto-baud Control Register (U1ACR - 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) | |
| 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 U1IIR. 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 U1IIR. 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 |

10.3.15 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.

(6)

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART1_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

10.3.16 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 U1DLL 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.

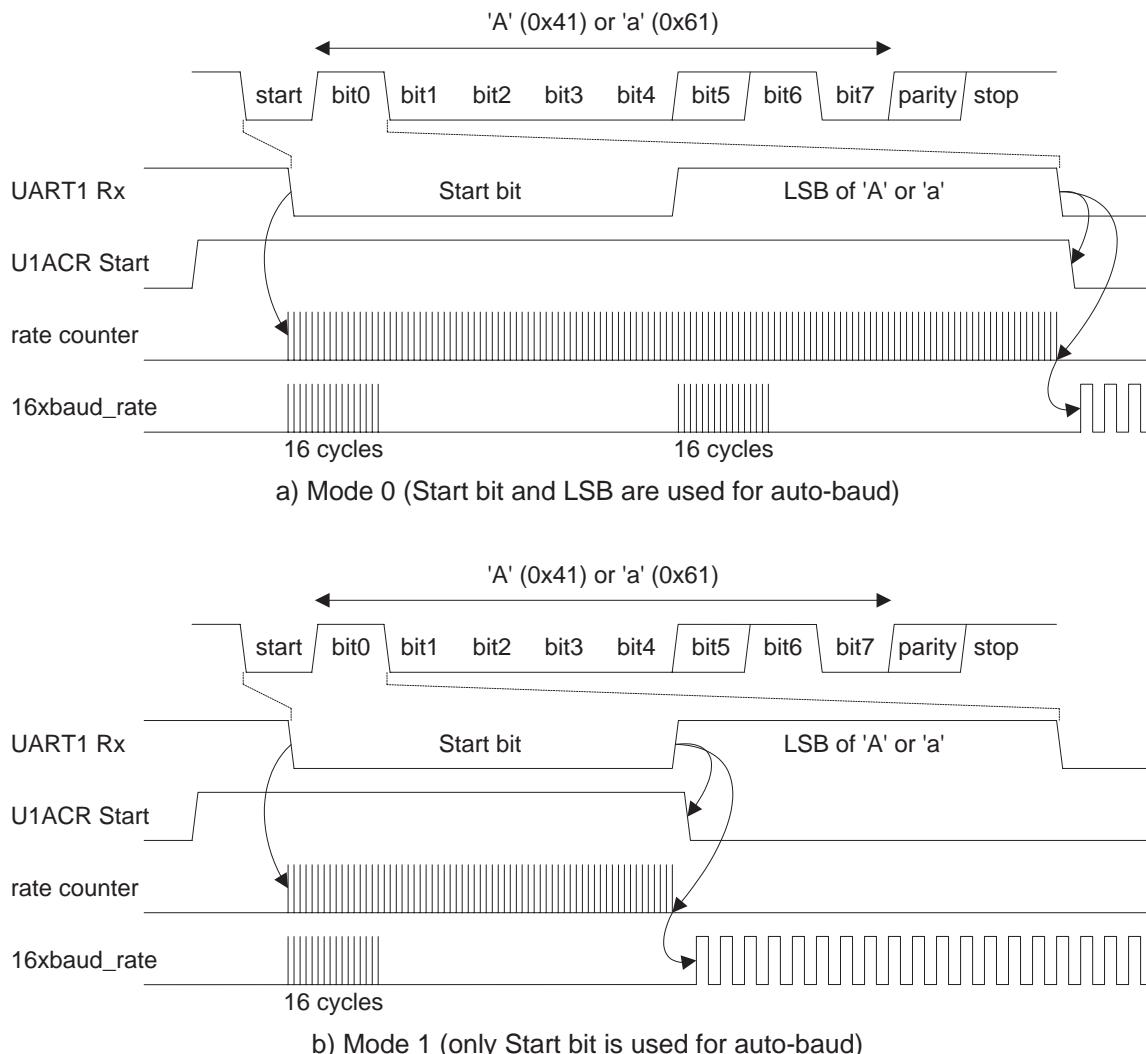


Fig 22. Autobaud Mode 0 and Mode 1 waveform

10.3.17 UART1 Transmit Enable Register (U1TER - 0xE001 0030)

LPC2141/2/4/6/8'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 131](#) describes how to use TXEn bit in order to achieve software flow control.

Table 131: 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 (LPC2144/6/8: CTS - otherwise any GPIO/external interrupt line) 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 |

10.4 Architecture

The architecture of the UART1 is shown below in the block diagram.

The VPB 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.

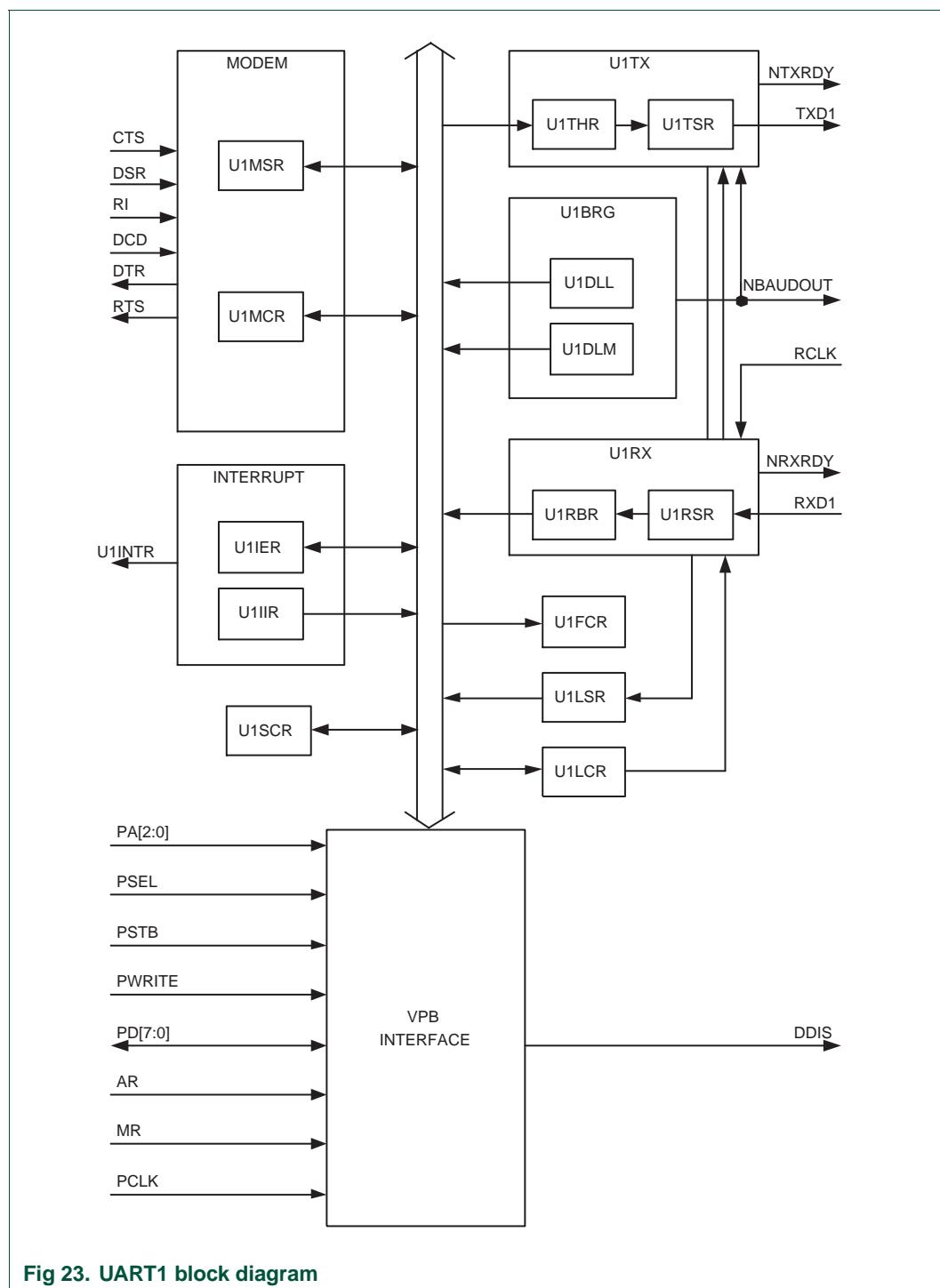
The UART1 transmitter block, U1TX, accepts data written by the CPU or host and buffers the data in the UART1 TX Holding Register FIFO (U1THR). The UART1 TX Shift Register (U1TSR) reads the data stored in the U1THR and assembles the data to transmit via the serial output pin, TXD1.

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 TX block. The U1BRG clock input source is the VPB 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.



11.1 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.

11.2 Applications

Interfaces to external I²C standard parts, such as serial RAMs, LCDs, tone generators, etc.

11.3 Description

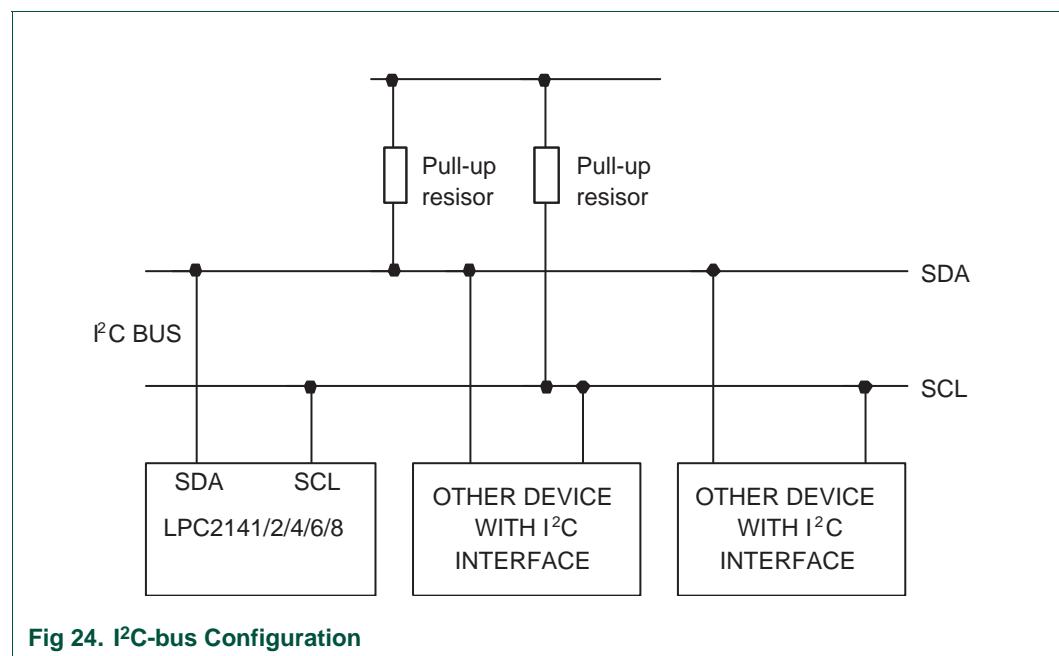
A typical I²C-bus configuration is shown in [Figure 24](#). Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I²C-bus:

- 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.

The LPC2141/2/4/6/8 I²C interfaces are byte oriented, and have four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The I²C interfaces compile with entire I²C specification, supporting the ability to turn power off to the LPC2141/2/4/6/8 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" in the microcontrollers datasheet). 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.



11.4 Pin description

Table 132: I²C Pin Description

| Pin | Type | Description |
|--------|--------------|--------------------------------|
| SDA0,1 | Input/Output | I ² C Serial Data. |
| SCL0,1 | Input/Output | I ² C Serial Clock. |

11.5 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.

11.5.1 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 133](#). 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 133: I2C0CONSET and I2C1CONSET used to configure Master mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---|------|-----|-----|----|----|---|---|
| Symbol | - | I2EN | STA | STO | SI | AA | - | - |
| Value | - | 1 | 0 | 0 | 0 | 0 | - | - |

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I²C interface will enter master transmitter mode when software sets the STA bit. The I²C 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 148](#) to [Table 151](#).

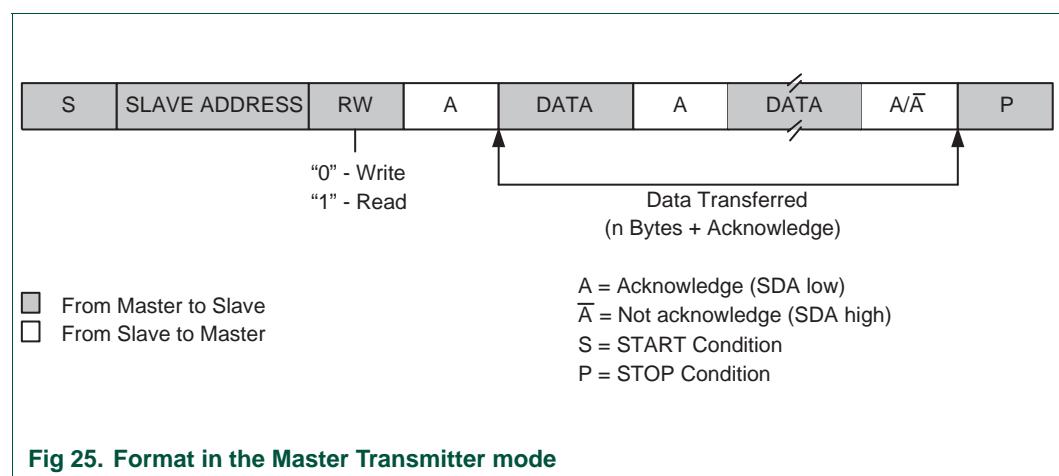
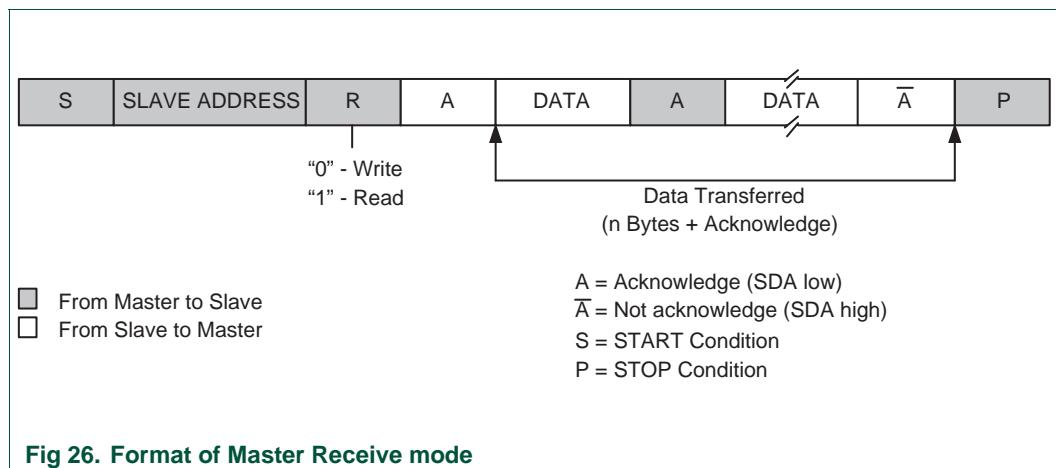


Fig 25. Format in the Master Transmitter mode

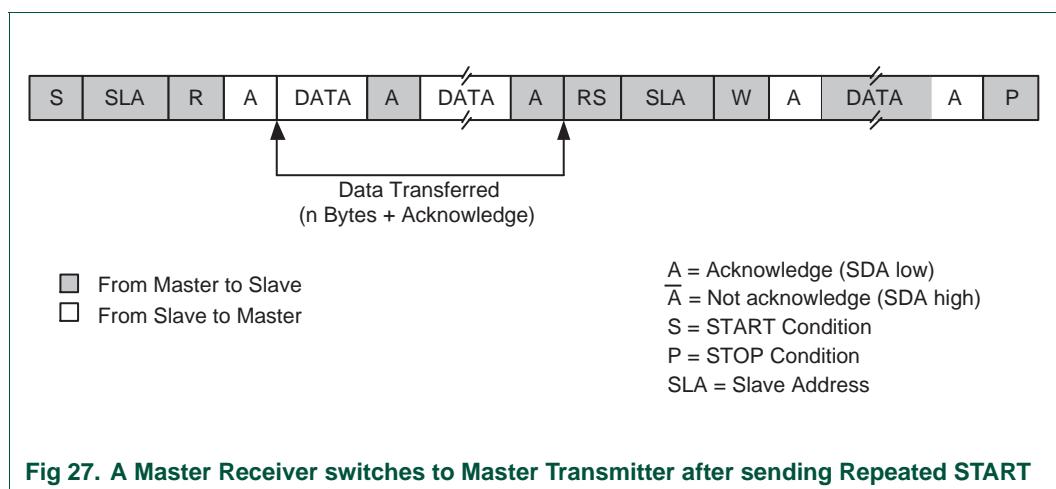
11.5.2 Master Receiver mode

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 I²C 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 149](#).



After a repeated START condition, I²C may switch to the master transmitter mode.



11.5.3 Slave Receiver mode

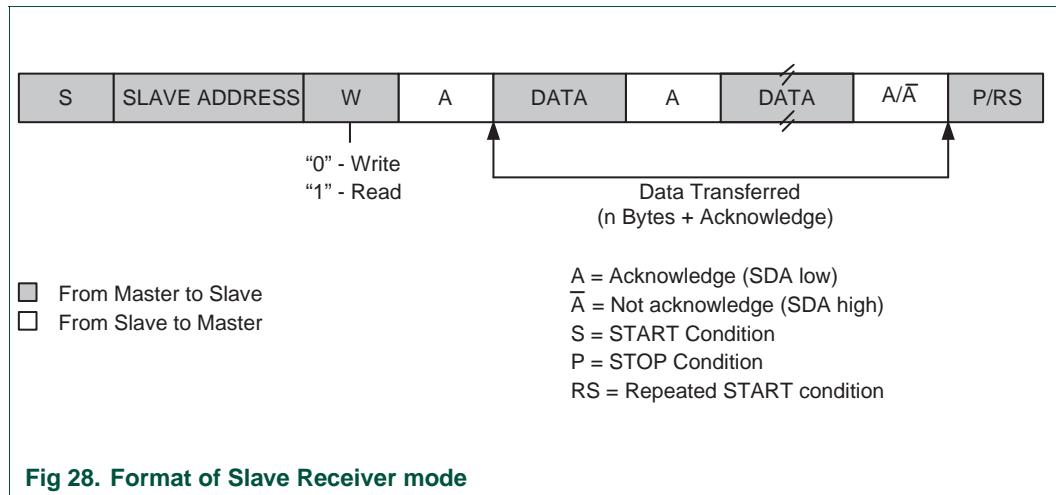
In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, user write the Slave Address register (I2ADR) and write the I²C Control Set register (I2CONSET) as shown in [Table 134](#).

Table 134: I2C0CONSET and I2C1CONSET used to configure Slave mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---|------|-----|-----|----|----|---|---|
| Symbol | - | I2EN | STA | STO | SI | AA | - | - |
| Value | - | 1 | 0 | 0 | 0 | 1 | - | - |

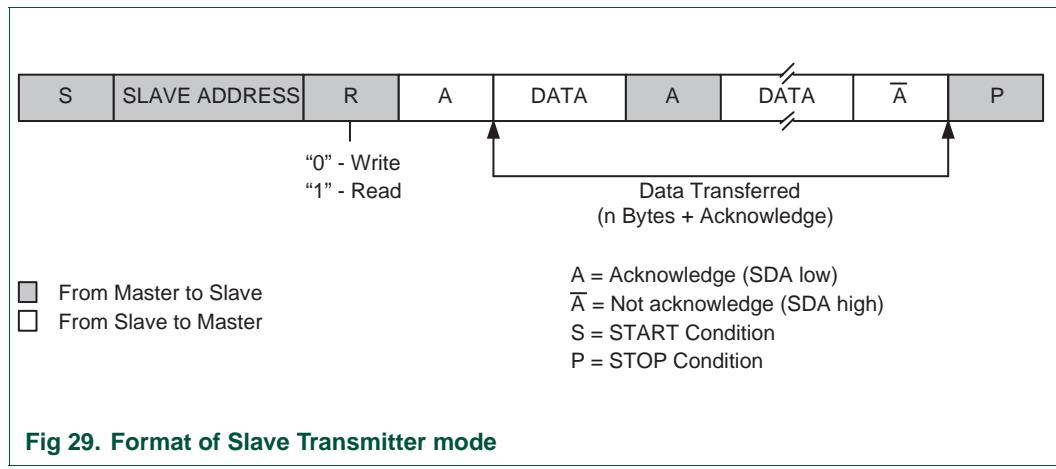
I2EN must be set to 1 to enable the I²C function. AA bit must be set to 1 to acknowledge its own slave address or the general call address. The STA, STO and SI bits are set to 0.

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 150](#) for the status codes and actions.



11.5.4 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, I²C may operate as a master and as a slave. 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. 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 I²C interface switches to the slave mode immediately and can detect its own slave address in the same serial transfer.



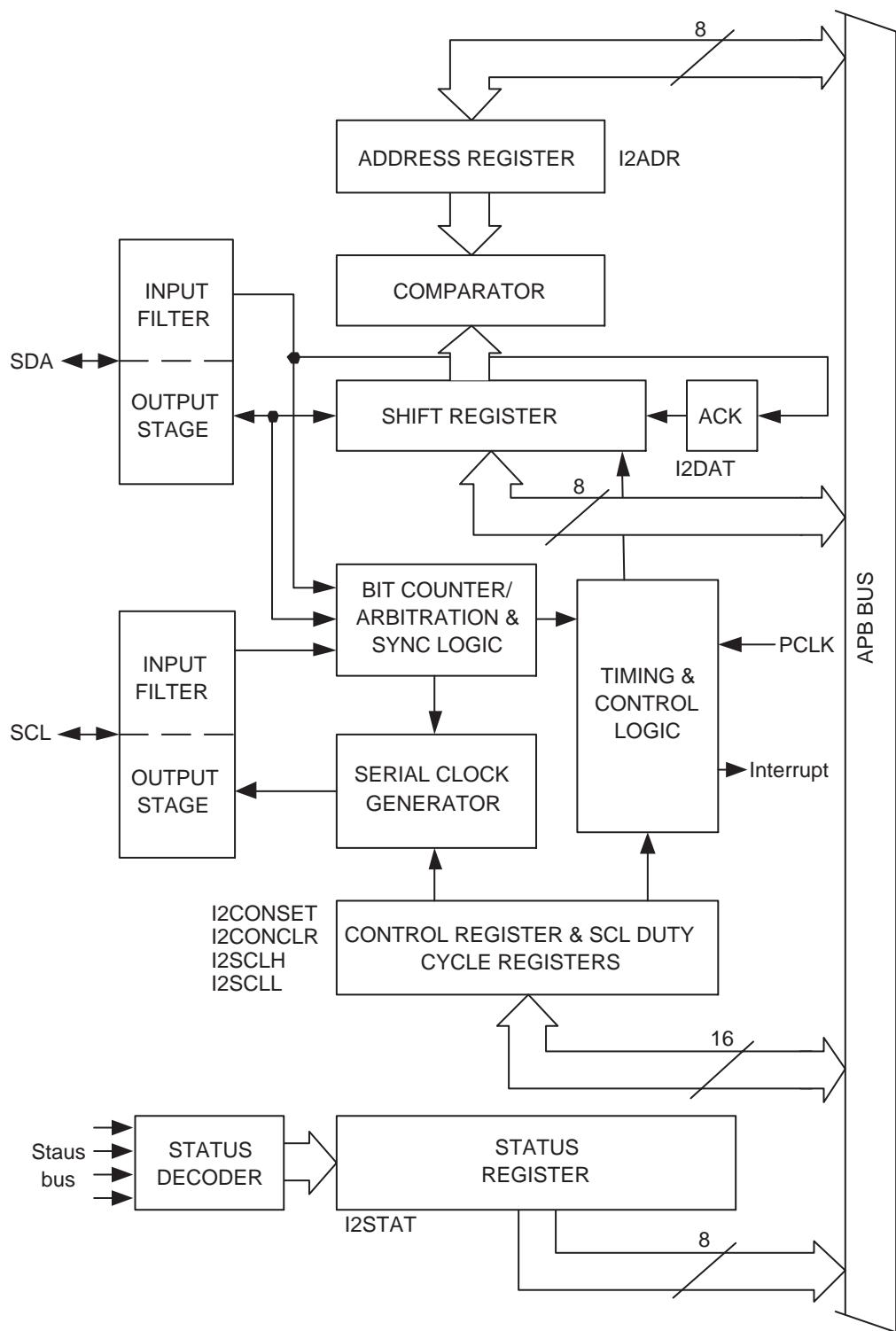
11.6 I²C Implementation and operation

[Figure 30](#) shows how the on-chip I²C-bus interface is implemented, and the following text describes the individual blocks.

11.6.1 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.

Fig 30. I²C serial interface block diagram

11.6.2 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.

11.6.3 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.

11.6.4 Shift register, I2DAT

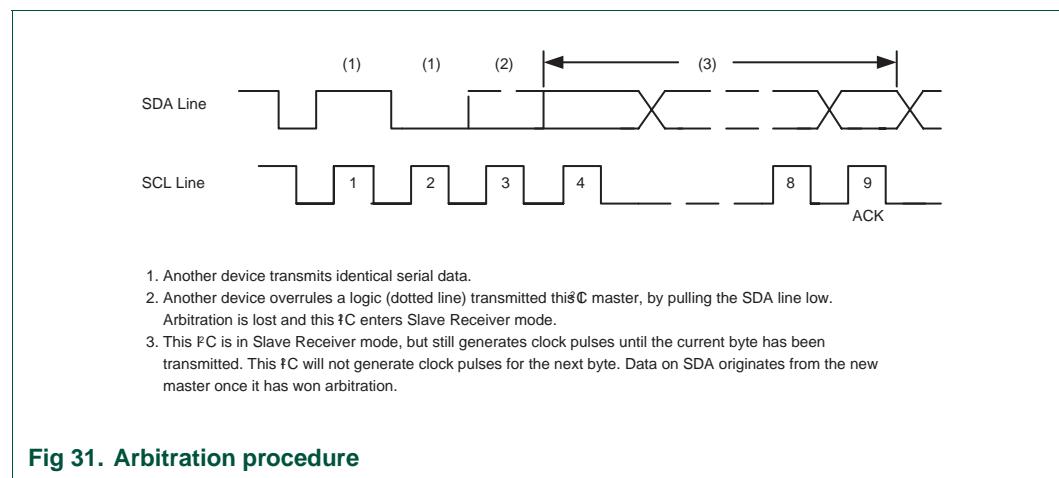
This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in I2DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB 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.

11.6.5 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.

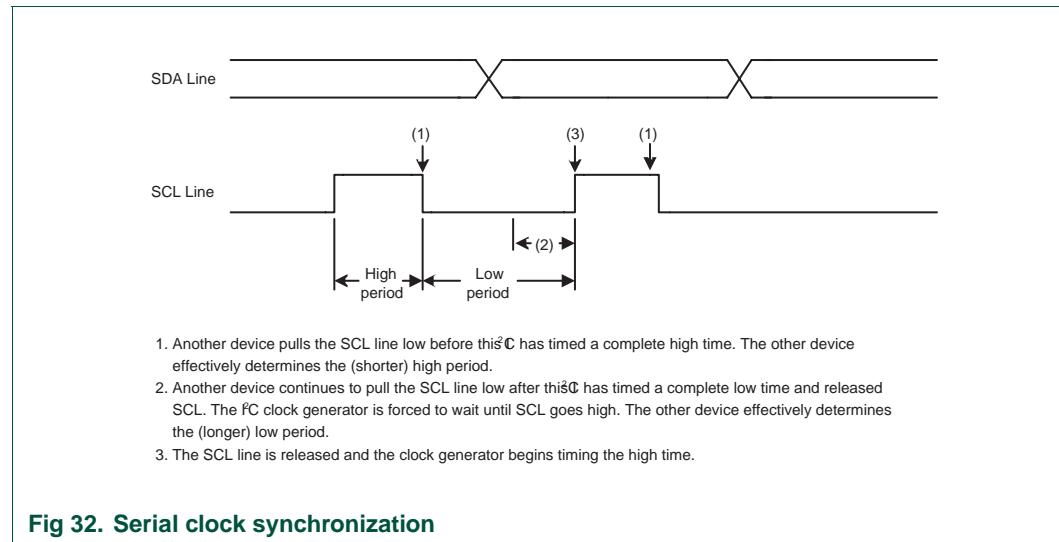
Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I²C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, the I²C block generates no further clock pulses.

[Figure 31](#) shows the 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 32](#) shows the synchronization procedure.



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.

11.6.6 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.

11.6.7 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.

11.6.8 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.

11.6.9 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).

11.7 Register description

Each I²C interface contains 7 registers as shown in [Table 135](#) below.

Table 135: I²C register map

| Name | Description | Access | Reset value ^[1] | I ² C0 Address and Name | I ² C1 Address and Name |
|----------|---|--------|----------------------------|------------------------------------|------------------------------------|
| I2CONSET | I²C Control Set Register. When a one is written to a bit of this register, the corresponding bit in the I ² C control register is set. Writing a zero has no effect on the corresponding bit in the I ² C control register. | R/W | 0x00 | 0xE001 C000 I2C0CONSET | 0xE005 C000 I2C1CONSET |
| I2STAT | I²C Status Register. During I ² C operation, this register provides detailed status codes that allow software to determine the next action needed. | RO | 0xF8 | 0xE001 C004 I2C0STAT | 0xE005 C004 I2C1STAT |
| 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 | 0xE001 C008 I2C0DAT | 0xE005 C008 I2C1DAT |
| I2ADR | I²C Slave Address Register. Contains the 7-bit slave address for operation of the I ² C 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 | 0xE001 C00C I2C0ADR | 0xE005 C00C I2C1ADR |
| I2SCLH | SCH Duty Cycle Register High Half Word. Determines the high time of the I ² C clock. | R/W | 0x04 | 0xE001 C010 I2C0SCLH | 0xE005 C010 I2C1SCLH |
| I2SCLL | SCL Duty Cycle Register Low Half Word. Determines the low time of the I ² C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I ² C master and certain times used in slave mode. | R/W | 0x04 | 0xE001 C014 I2C0SCLL | 0xE005 C014 I2C1SCLL |
| I2CONCLR | I²C Control Clear Register. When a one is written to a bit of this register, the corresponding bit in the I ² C control register is cleared. Writing a zero has no effect on the corresponding bit in the I ² C control register. | WO | NA | 0xE001 C018 I2C0CONCLR | 0xE005 C018 I2C1CONCLR |

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

11.7.1 I²C Control Set register (I2CONSET: I2C0, I2C0CONSET - 0xE001 C000 and I2C1, I2C1CONSET - 0xE005 C000)

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 136: I²C Control Set register (I2CONSET: I2C0, I2C0CONSET - address 0xE001 C000 and I2C1, I2C1CONSET - address 0xE005 C000) 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 | I ² C interrupt flag. | 0 |
| 4 | STO | STOP flag. See the text below. | 0 |
| 5 | STA | START flag. See the text below. | 0 |
| 6 | I2EN | I ² C 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.

11.7.2 I²C Control Clear register (I2CONCLR: I2C0, I2C0CONCLR - 0xE001 C018 and I2C1, I2C1CONCLR - 0xE005 C018)

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 137: I²C Control Set register (I2CONCLR: I2C0, I2C0CONCLR - address 0xE001 C018 and I2C1, I2C1CONCLR - address 0xE005 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 |
| 2 | AAC | Assert acknowledge Clear bit. | |
| 3 | SIC | I ² C 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 |

Table 137: I²C Control Set register (I2CONCLR: I2C0, I2C0CONCLR - address 0xE001 C018 and I2C1, I2C1CONCLR - address 0xE005 C018) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 5 | STAC | START flag Clear bit. | 0 |
| 6 | I2ENC | I ² C 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.

I2ENC 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.

11.7.3 I²C Status register (I2STAT: I2C0, I2C0STAT - 0xE001 C004 and I2C1, I2C1STAT - 0xE005 C004)

Each I²C Status register reflects the condition of the corresponding I²C interface. The I²C Status register is Read-Only.

Table 138: I²C Status register (I2STAT: I2C0, I2C0STAT - address 0xE001 C004 and I2C1, I2C1STAT - address 0xE005 C004) 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 I ² C 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 148](#) to [Table 151](#).

11.7.4 I²C Data register (I2DAT: I2C0, I2C0DAT - 0xE001 C008 and I2C1, I2C1DAT - 0xE005 C008)

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 139: I²C Data register (I2DAT: I2C0, I2C0DAT - address 0xE001 C008 and I2C1, I2C1DAT - address 0xE005 C008) 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 |

11.7.5 I²C Slave Address register (I2ADR: I2C0, I2C0ADR - 0xE001 C00C and I2C1, I2C1ADR - address 0xE005 C00C)

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 140: I²C Slave Address register (I2ADR: I2C0, I2C0ADR - address 0xE001 C00C and I2C1, I2C1ADR - address 0xE005 C00C) bit description

| Bit | Symbol | Description | Reset value |
|-----|---------|---|-------------|
| 0 | GC | General Call enable bit. | 0 |
| 7:1 | Address | The I ² C device address for slave mode. | 0x00 |

11.7.6 I²C SCL High duty cycle register (I2SCLH: I2C0, I2C0SCLH - 0xE001 C010 and I2C1, I2C1SCLH - 0xE005 C010)

Table 141: I²C SCL High Duty Cycle register (I2SCLH: I2C0, I2C0SCLH - address 0xE001 C010 and I2C1, I2C1SCLH - address 0xE005 C010) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 15:0 | SCLH | Count for SCL HIGH time period selection. | 0x0004 |

11.7.7 I²C SCL Low duty cycle register (I2SCLL: I2C0 - I2C0SCLL: 0xE001 C014; I2C1 - I2C1SCLL: 0xE005 C014)

Table 142: I²C SCL Low Duty Cycle register (I2SCLL: I2C0, I2C0SCLL - address 0xE001 C014 and I2C1, I2C1SCLL - address 0xE005 C014) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 15:0 | SCLL | Count for SCL LOW time period selection. | 0x0004 |

11.7.8 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 (PCLK is the frequency of the peripheral bus VPB):

(7)

$$I^2C_{bitfrequency} = \frac{PCLK}{I2CSCLH + I2CSCLL}$$

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 143](#) gives some examples of I²C-bus rates based on PCLK frequency and I2SCLL and I2SCLH values.

Table 143: Example I²C clock rates

| I ² SCLL + I ² SCLH | I ² C 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 |

11.8 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 33 to 37. Table 144 lists abbreviations used in these figures when describing the I²C operating modes.

Table 144: 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) |
| <u>A</u> | Not acknowledge bit (high level at SDA) |
| Data | 8-bit data byte |
| P | Stop condition |

In Figures 33 to 37, 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 148 to Table 152.

11.8.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see [Figure 33](#)). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 145: 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 148](#). After a repeated start condition (state 0x10). The I²C block may switch to the master receiver mode by loading I2DAT with SLA+R).

11.8.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see [Figure 34](#)). 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 149](#). After a repeated start condition (state 0x10), the I²C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

11.8.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see [Figure 35](#)). To initiate the slave receiver mode, I2ADR and I2CON must be loaded as follows:

Table 146: 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 I²C block will respond when addressed by a master. If the LSB (GC) is set, the I²C block will respond to the general call address (0x00); otherwise it ignores the general call address.

Table 147: 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 I²C-bus rate settings do not affect the I²C block in the slave mode. I2EN must be set to logic 1 to enable the I²C block. The AA bit must be set to enable the I²C 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 I²C block waits until it is addressed by its own slave address followed by the data direction bit which must be "0" (W) for the I²C 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 I²C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I²C block will return a not acknowledge (logic 1) to SDA after the next received data byte. 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.

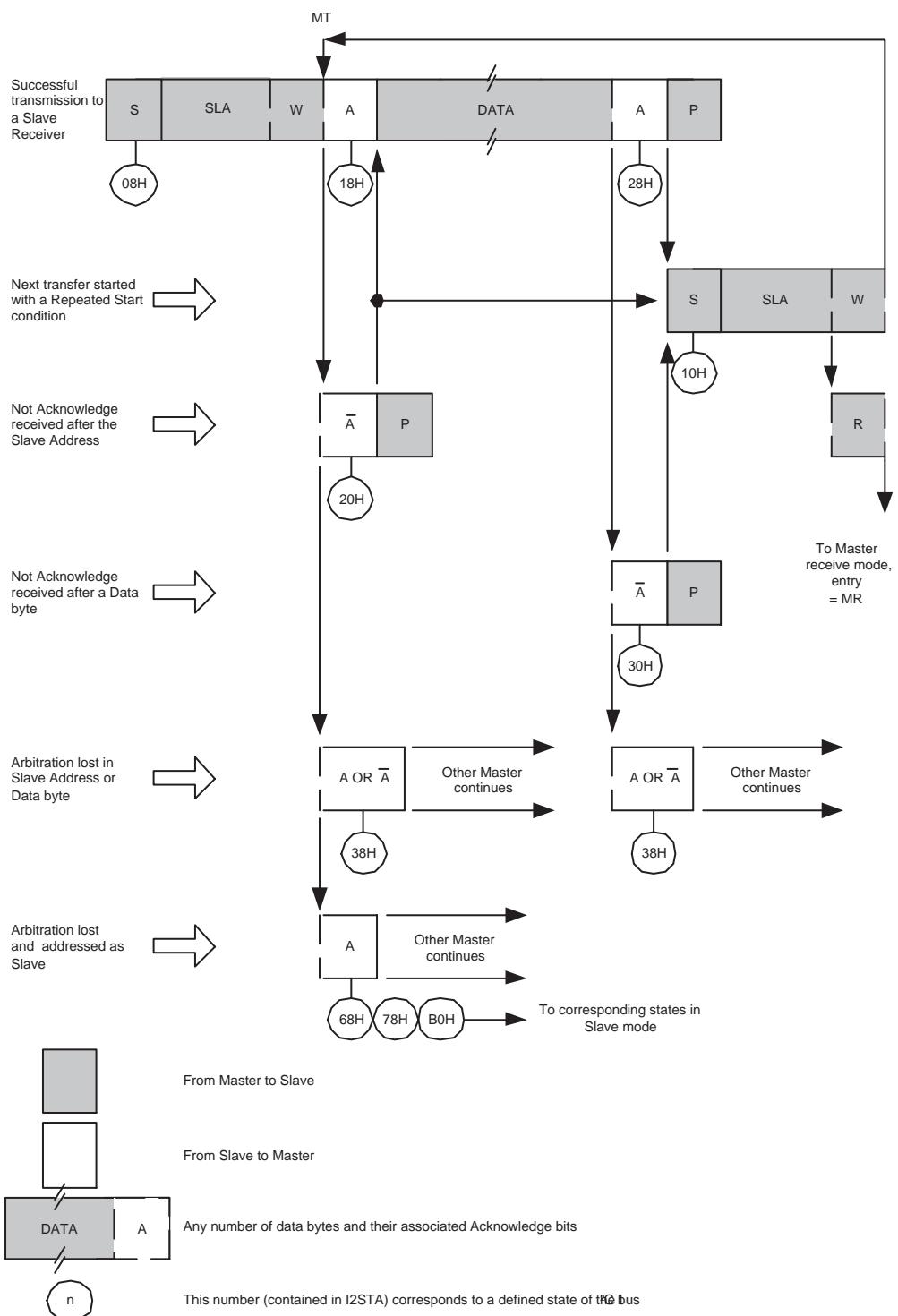
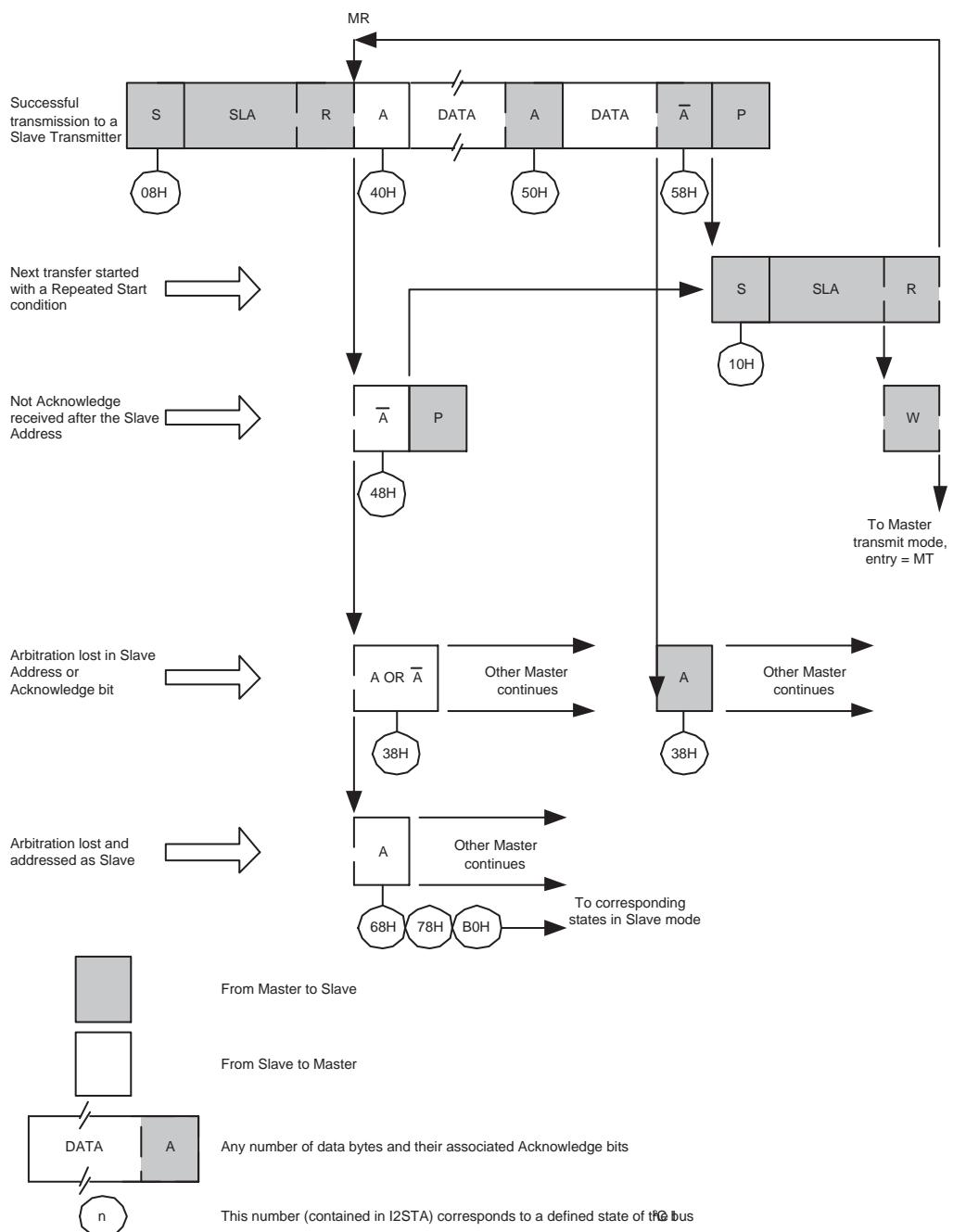
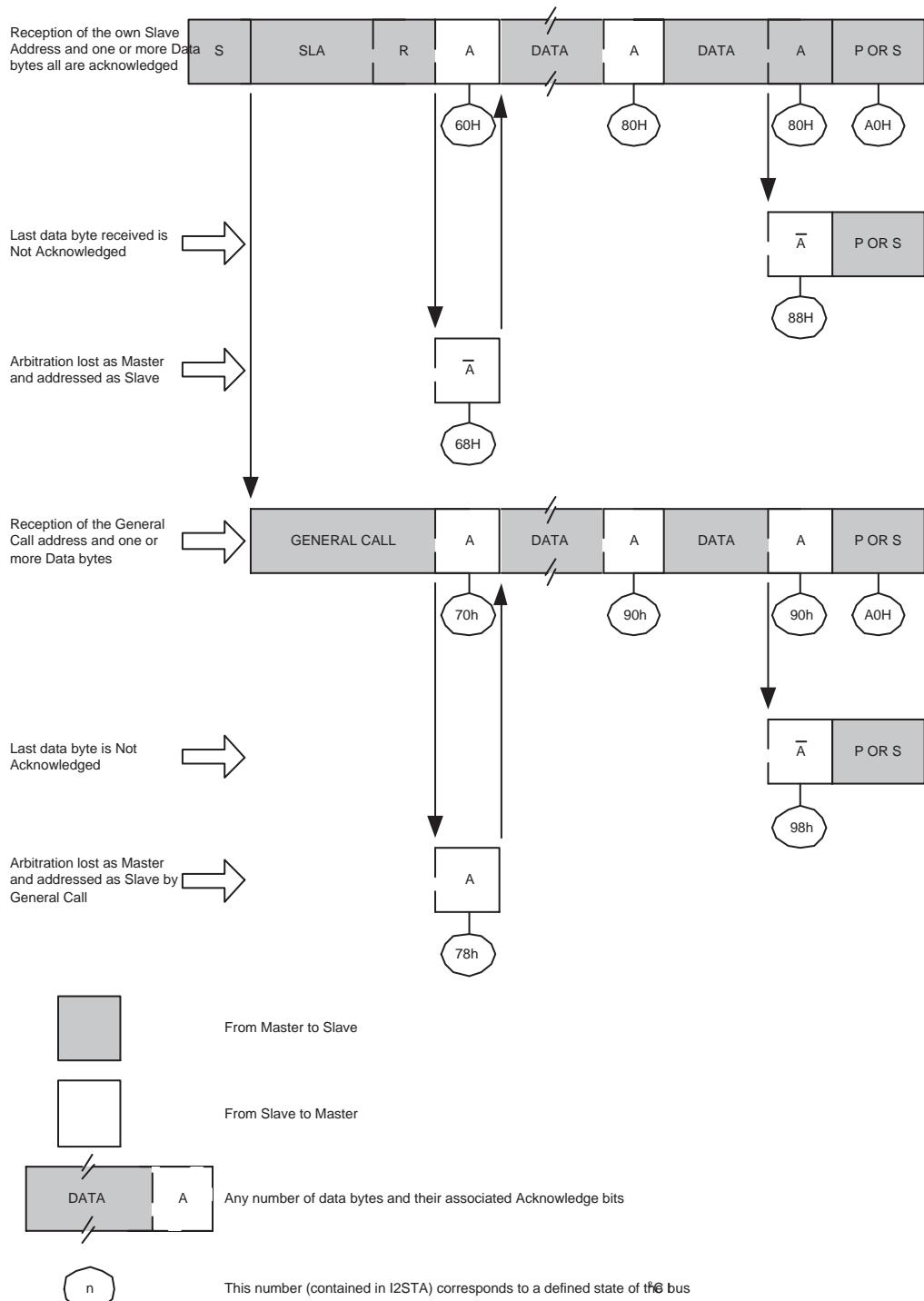


Fig 33. Format and States in the Master Transmitter mode

**Fig 34. Format and States in the Master Receiver mode**

**Fig 35. Format and States in the Slave Receiver mode**

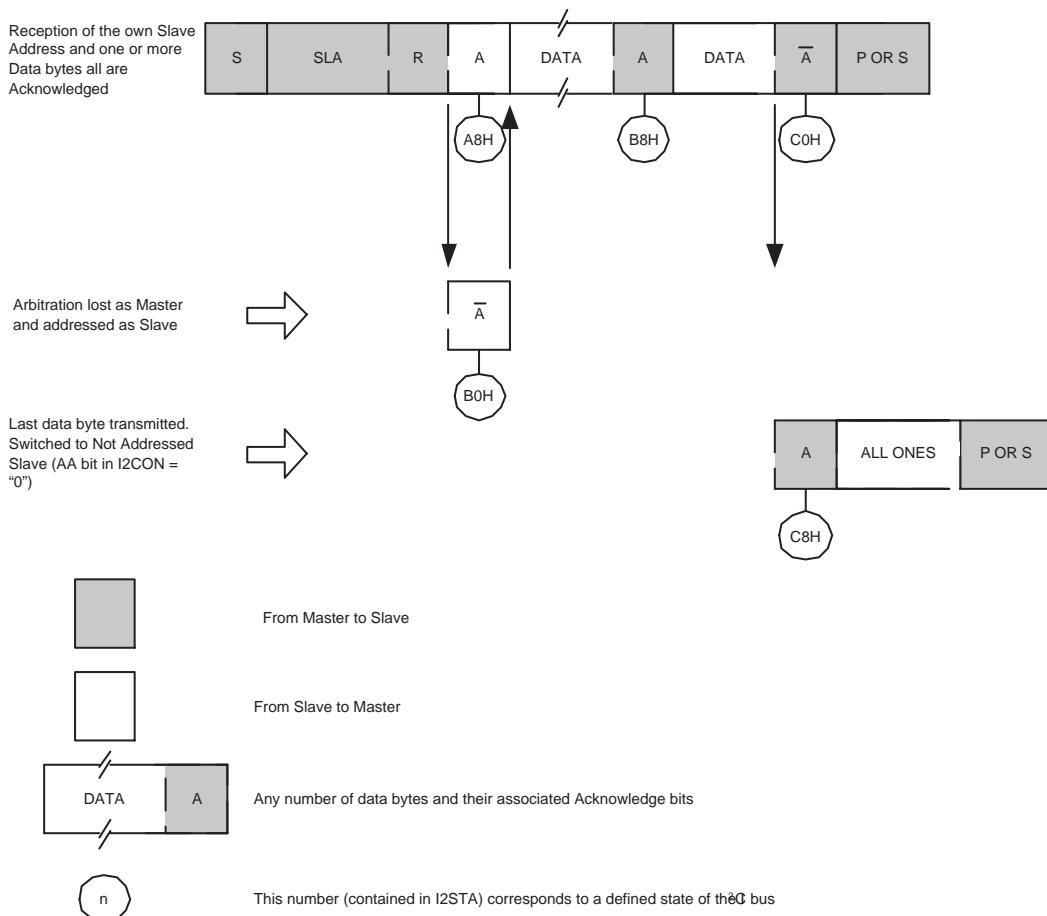


Fig 36. Format and States in the Slave Transmitter mode

11.8.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see [Figure 36](#)). 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 151](#). 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 148: Master Transmitter mode

| Status Code (I2CSTAT) | Status of the I ² C-bus and hardware | Application software response | | | | Next action taken by I ² C 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 Load SLA+R | X | 0 | 0 | X | As above. |
| 0x18 | SLA+W has been transmitted; ACK has been received. | Load data byte or No I2DAT action or No I2DAT action or No I2DAT action | 0 1 1 | 0 1 1 | 0 0 0 | X X X | Data byte will be transmitted; ACK bit will be received. Repeated START will be transmitted. STOP condition will be transmitted; STO flag will be reset. 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 No I2DAT action or No I2DAT action or No I2DAT action | 0 1 1 | 0 1 1 | 0 0 0 | X X X | Data byte will be transmitted; ACK bit will be received. Repeated START will be transmitted. STOP condition will be transmitted; STO flag will be reset. 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 No I2DAT action or No I2DAT action or No I2DAT action | 0 1 1 | 0 1 1 | 0 0 0 | X X X | Data byte will be transmitted; ACK bit will be received. Repeated START will be transmitted. STOP condition will be transmitted; STO flag will be reset. 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 No I2DAT action or No I2DAT action or No I2DAT action | 0 1 1 | 0 1 1 | 0 0 0 | X X X | Data byte will be transmitted; ACK bit will be received. Repeated START will be transmitted. STOP condition will be transmitted; STO flag will be reset. 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 No I2DAT action | 0 1 | 0 0 | 0 0 | X X | I ² C-bus will be released; not addressed slave will be entered. A START condition will be transmitted when the bus becomes free. |

Table 149: Master Receiver mode

| Status Code (I2CSTAT) | Status of the I ² C-bus and hardware | Application software response | | | | Next action taken by I ² C 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 Load SLA+W | X | 0 | 0 | X | As above. |
| 0x38 | Arbitration lost in NOT ACK bit. | No I2DAT action or No I2DAT action | 0 | 0 | 0 | X | I ² C-bus will be released; the I ² C block will enter a slave mode. |
| | | | 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 No I2DAT action | 0 | 0 | 0 | 0 | Data byte will be received; NOT ACK bit will be returned. |
| | | | 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 No I2DAT action | 1 | 0 | 0 | X | Repeated START condition will be transmitted. |
| | | | 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 Read data byte | 0 | 0 | 0 | 0 | Data byte will be received; NOT ACK bit will be returned. |
| | | | 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 Read data byte | 1 | 0 | 0 | X | Repeated START condition will be transmitted. |
| | | | 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 150: Slave Receiver mode

| Status Code (I2CSTAT) | Status of the I ² C-bus and hardware | Application software response | | | | Next action taken by I ² C 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 150: Slave Receiver mode

| Status Code (I ² CSTAT) | Status of the I ² C-bus and hardware | Application software response | | | | Next action taken by I ² C hardware | |
|------------------------------------|---|-------------------------------|-----|-----------------------|----|--|--|
| | | To/From I ² DAT | | To I ² CON | | | |
| | | 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 I ² ADR[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 I ² ADR[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 I ² ADR[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 I ² ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free. |

Table 151: Slave Transmitter mode

| Status Code (I2CSTAT) | Status of the I ² C-bus and hardware | Application software response | | | | Next action taken by I ² C hardware | |
|--------------------------|--|-------------------------------|-----|----------|----|--|---|
| | | To/From I2DAT | | To I2CON | | | |
| | | STA | STO | SI | AA | | |
| 0xA8 | Own SLA+R has been received; ACK has been returned. | Load data byte or | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK bit will be received. |
| | | Load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK will be received. |
| 0xB0 | Arbitration lost in SLA+R/W as master; Own SLA+R has been received, ACK has been returned. | Load data byte or | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK bit will be received. |
| | | Load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK bit will be received. |
| 0xB8 | Data byte in I2DAT has been transmitted; ACK has been received. | Load data byte or | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK bit will be received. |
| | | Load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK bit will be received. |
| 0xC0 | Data byte in I2DAT has been transmitted; NOT ACK has been received. | No I2DAT action or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. |
| | | No I2DAT action or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. |
| | | No I2DAT action or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free. |
| | | No I2DAT action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free. |
| 0xC8 | Last data byte in I2DAT has been transmitted (AA = 0); ACK has been received. | No I2DAT action or | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. |
| | | No I2DAT action or | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. |
| | | No I2DAT action or | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free. |
| | | No I2DAT action | 1 | 0 | 0 | 01 | Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free. |

11.8.5 Miscellaneous States

There are two I2STAT codes that do not correspond to a defined I²C hardware state (see [Table 152](#)). These are discussed below.

11.8.6 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.

11.8.7 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 152: Miscellaneous States

| Status Code (I2CSTAT) | Status of the I ² C-bus and hardware | Application software response | | | | Next action taken by I ² C hardware | | | |
|--------------------------|---|-------------------------------|---|-----------------|---|--|-----|----|---|
| | | To/From I2DAT | | To I2CON | | STA | STO | SI | AA |
| 0xF8 | No relevant state information available; SI = 0. | No I2DAT action | | No I2CON action | | | | | Wait or proceed current transfer. |
| 0x00 | Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 0x00 can also occur when interference causes the I ² C block to enter an undefined state. | No I2DAT action | 0 | 1 | 0 | X | | | Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the I ² C block is switched to the not addressed SLV mode. STO is reset. |

11.8.8 Some special cases

The I²C hardware has facilities to handle the following special cases that may occur during a serial transfer:

11.8.9 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 37](#)). 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.

11.8.10 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see [Figure 31](#)). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see [Figure 33](#) and [Figure 34](#)).

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.

11.8.11 Forced access to the I²C-bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I²C-bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I²C-bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I²C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 34).

11.8.12 I²C-bus obstructed by a low level on SCL or SDA

An I²C-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 I²C 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 39](#)). The I²C 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 I²C-bus is considered free. The I²C 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 I²C 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.

11.8.13 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 I²C 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 I²C 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 152](#).

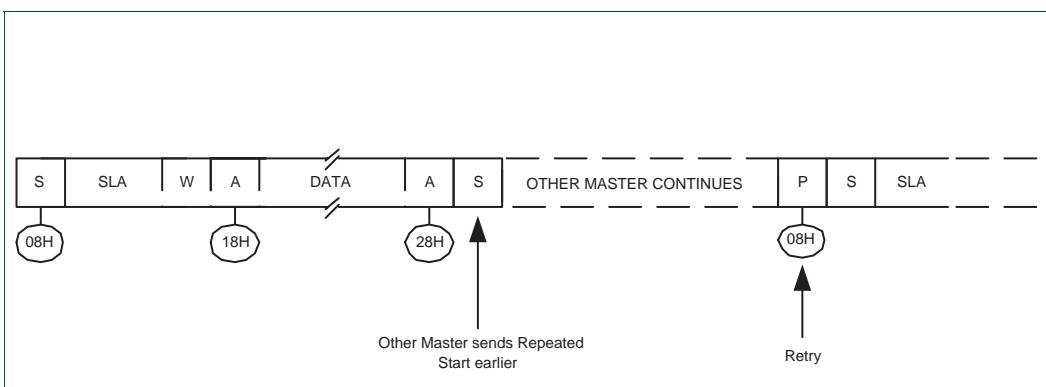
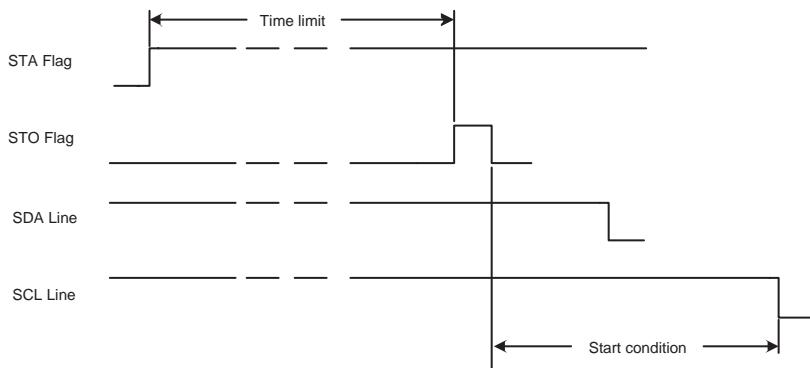
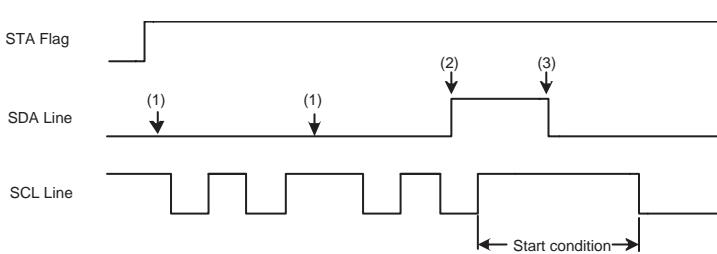


Fig 37. Simultaneous repeated START conditions from two masters

Fig 38. Forced access to a busy I²C-bus

1. Unsuccessful attempt to send a Start condition.
2. SDA Line released.
3. Successful attempt to send a Start condition; state 08H is entered.

Fig 39. Recovering from a bus obstruction caused by a low level on SDA

11.8.14 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.

11.8.15 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.

11.8.16 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.

11.8.17 The State service routines

Each state routine is part of the I²C interrupt routine and handles one of the 26 states.

11.8.18 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 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.

11.9 Software example

11.9.1 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.

11.9.2 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

11.9.3 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

11.9.4 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.

11.9.5 Non mode specific States

11.9.6 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

11.9.7 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.

11.9.8 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

11.9.9 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

11.9.10 Master Transmitter States

11.9.11 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

11.9.12 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

11.9.13 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

11.9.14 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

11.9.15 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

11.9.16 Master Receive States

11.9.17 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

11.9.18 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

11.9.19 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

11.9.20 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

11.9.21 Slave Receiver States

11.9.22 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

11.9.23 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.

11.9.24 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

11.9.25 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

11.9.26 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

11.9.27 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

11.9.28 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

11.9.29 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

11.9.30 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

11.9.31 Slave Transmitter States

11.9.32 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

11.9.33 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

11.9.34 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

11.9.35 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.

11.9.36 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



12.1 Features

- Single complete and independent SPI controller.
- Compliant with Serial Peripheral Interface (SPI) specification.
- Synchronous, Serial, Full Duplex Communication.
- Combined SPI master and slave.
- Maximum data bit rate of one eighth of the input clock rate.
- 8 to 16 bits per transfer

12.2 Description

12.2.1 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.

12.2.2 SPI data transfers

[Figure 40](#) 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 = 1, the SSEL signal will always go inactive between data transfers. This is not guaranteed when CPHA = 0 (the signal can remain active).

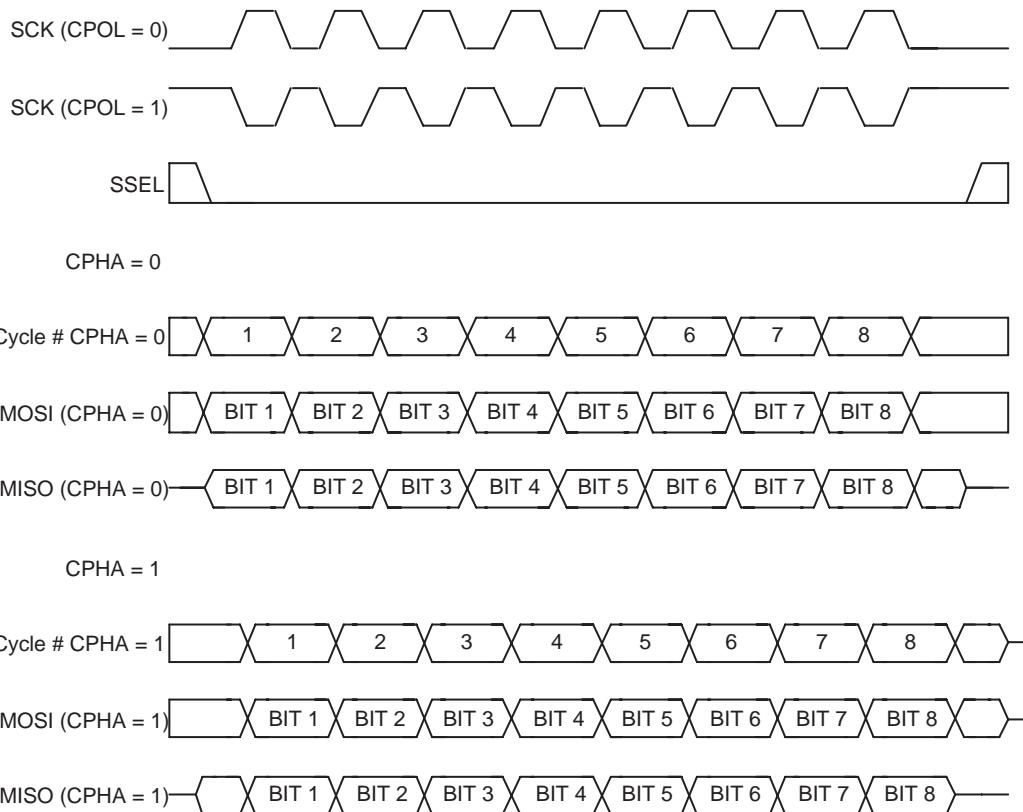


Fig 40. SPI data transfer format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in [Table 153](#). 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 153: 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.

12.2.3 General information

There are four registers that control the SPI peripheral. They are described in detail in [Section 12.4 “Register description” on page 175](#).

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 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.

12.2.4 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 that 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.

12.2.5 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 that 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.

12.2.6 Exception conditions

12.2.7 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.

12.2.8 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.

12.2.9 Mode Fault

The SSEL signal must always be inactive when the SPI block is a master. 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.

12.2.10 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.

12.3 Pin description

Table 154: SPI pin description

| Pin Name | Type | Pin Description |
|----------|--------------|--|
| SCK0 | 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. |
| SSEL0 | 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 LPC2141/2/4/6/8 (unlike earlier Philips ARM devices) the SSEL0 pin can be used for a different function when the SPI0 interface is only used in Master mode. For example, pin hosting the SSEL0 function can be configured as an output digital GPIO pin and used to select one of the SPI0 slaves. |
| MISO0 | 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. |
| MOSI0 | 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. |

12.4 Register description

The SPI contains 5 registers as shown in [Table 155](#). All registers are byte, half word and word accessible.

Table 155: 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 |
| 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.

12.4.1 SPI Control Register (S0SPCR - 0xE002 0000)

The S0SPCR register controls the operation of the SPI0 as per the configuration bits setting.

Table 156: 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. | |
| 4 | CPOL | 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. | |
| | | 0 | Clock polarity control. | 0 |
| 5 | MSTR | 0 | SCK is active high. | |
| | | 1 | SCK is active low. | |
| 0 | | | Master mode select. | 0 |
| | | 1 | The SPI operates in Slave mode. | |
| 1 | | | The SPI operates in Master mode. | |

Table 156: SPI Control Register (S0SPCR - address 0xE002 0000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 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. | |
| 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 | |
| 15:12 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

12.4.2 SPI Status Register (S0SPSR - 0xE002 0004)

The S0SPSR register controls the operation of the SPI0 as per the configuration bits setting.

Table 157: 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 |

Table 157: SPI Status Register (S0SPSR - address 0xE002 0004) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 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 |

12.4.3 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 158: 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 |

12.4.4 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 159: 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 /VPB divider rate as determined by the VPBDIV register contents.

12.4.5 SPI Interrupt register (S0SPINT - 0xE002 001C)

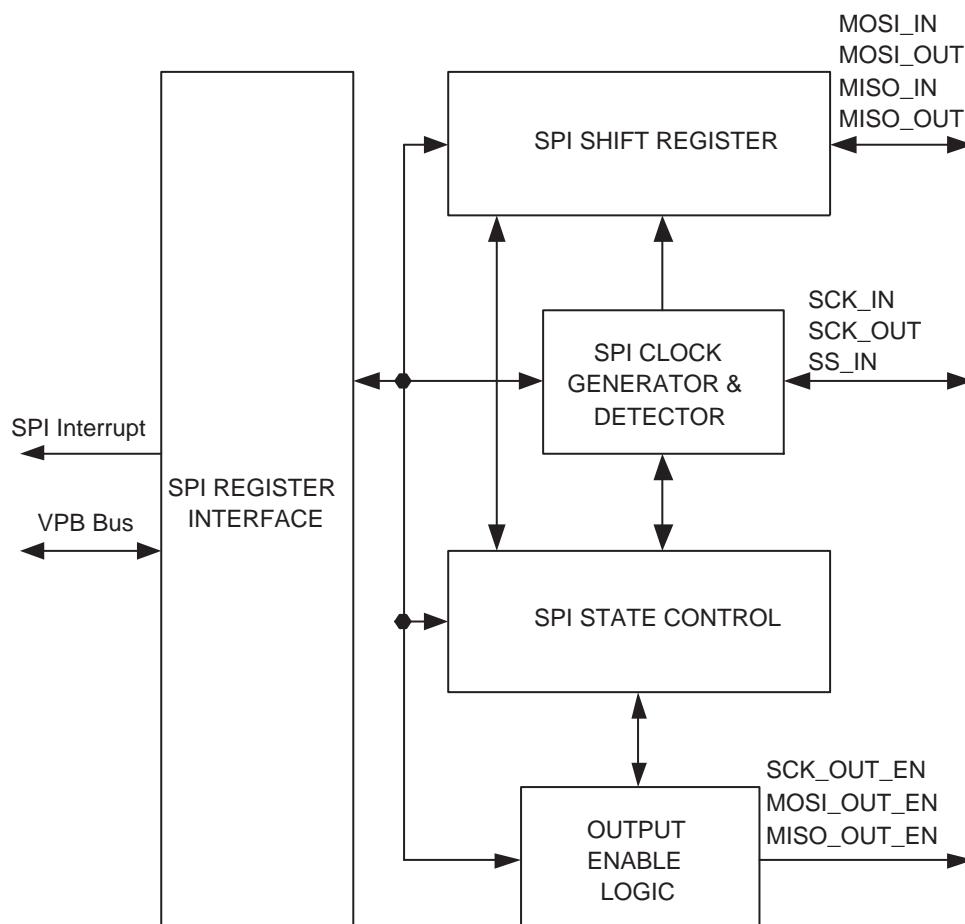
This register contains the interrupt flag for the SPI0 interface.

Table 160: 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 |

12.5 Architecture

The block diagram of the SPI solution implemented in SPI0 interface is shown in the [Figure 41](#).

**Fig 41. SPI block diagram**



13.1 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

13.2 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.

Table 161: SSP pin descriptions

| Pin Name | Type | Interface pin name/function | | | Pin Description |
|----------|------|-----------------------------|-----|-----------|---|
| | | SPI | SSI | Microwire | |
| SCK1 | 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 SSP either holds it in its inactive state, or does not drive it (leaves it in high impedance state). |

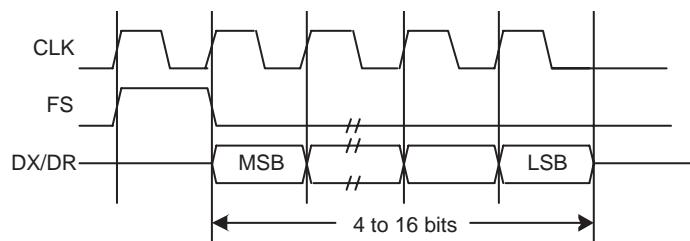
Table 161: SSP pin descriptions

| Pin Name | Type | Interface pin name/function | | | Pin Description |
|----------|------|-----------------------------|----------------|----------------|---|
| | | SPI | SSI | Microwire | |
| SSEL1 | I/O | SSEL | FS | CS | Slave Select/Frame Sync/Chip Select. When the SSP 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 SSP 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. |
| MISO1 | 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 SSP is a slave, serial data is output on this signal. When the SSP is a master, it clocks in serial data from this signal. When the SSP is a slave and is not selected by SSEL, it does not drive this signal (leaves it in high impedance state). |
| MOSI1 | 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 SSP is a master, it outputs serial data on this signal. When the SSP is a slave, it clocks in serial data from this signal. |

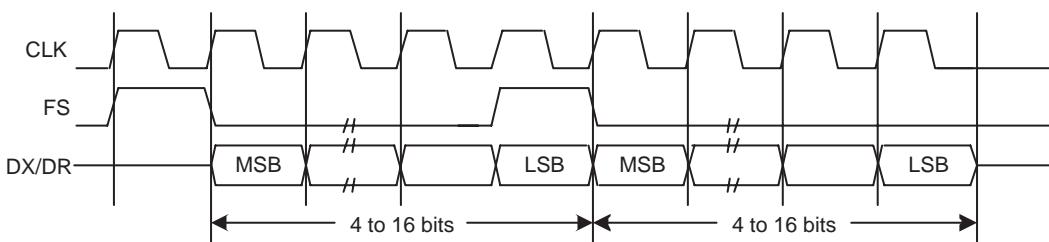
13.3 Bus description

13.3.1 Texas Instruments Synchronous Serial (SSI) frame format

[Figure 42](#) 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 42. 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.

13.3.2 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.

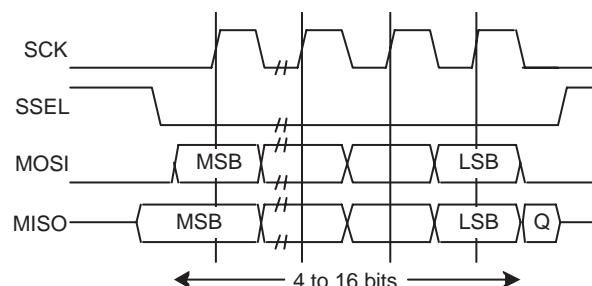
13.3.3 Clock Polarity (CPOL) and Clock Phase (CPHA) control

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.

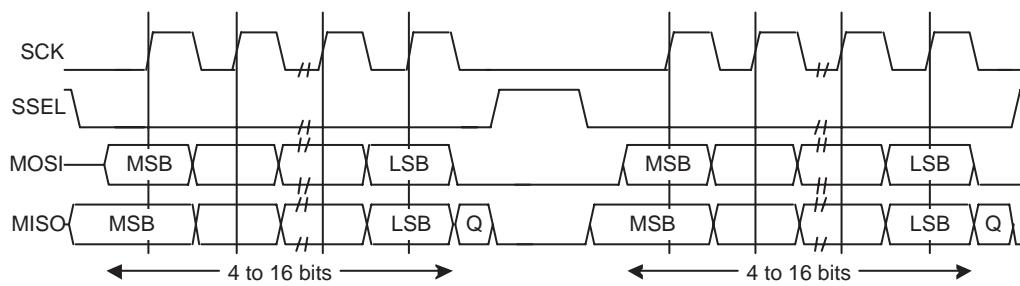
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is LOW, data is captured on the first clock edge transition. If the CPHA clock phase control bit is HIGH, data is captured on the second clock edge transition.

13.3.4 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 43](#).



a) Motorola SPI frame format (single transfer) with CPOL=0 and CPHA=0



b) Motorola SPI frame format (continuous transfer) with CPOL=0 and CPHA=0

Fig 43. 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.

13.3.5 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in [Figure 44](#), which covers both single and continuous transfers.

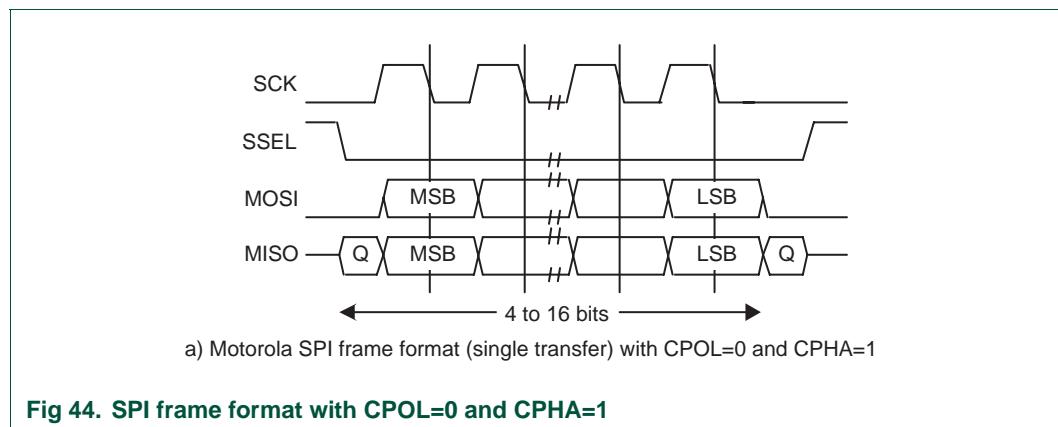


Fig 44. SPI frame format with CPOL=0 and CPHA=1

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.

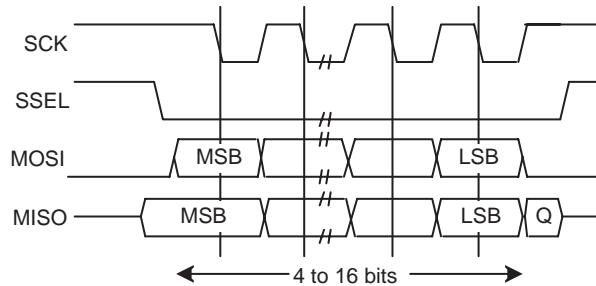
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

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.

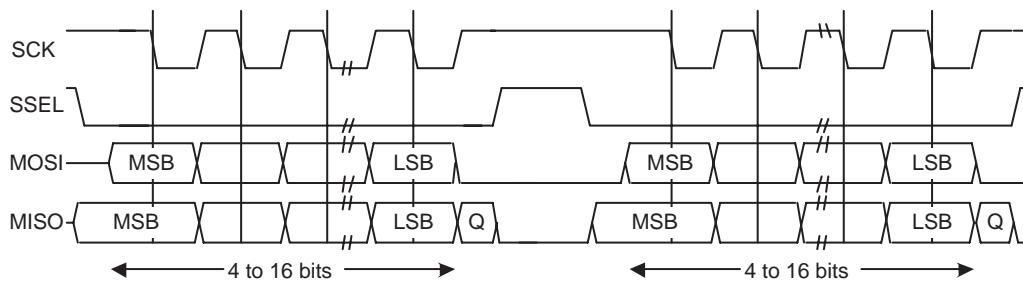
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

13.3.6 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 45](#).



a) Motorola SPI frame format (single transfer) with CPOL=1 and CPHA=0



b) Motorola SPI frame format (continuous transfer) with CPOL=1 and CPHA=0

Fig 45. 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.

13.3.7 SPI format with CPOL = 1, CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in [Figure 46](#), which covers both single and continuous transfers.

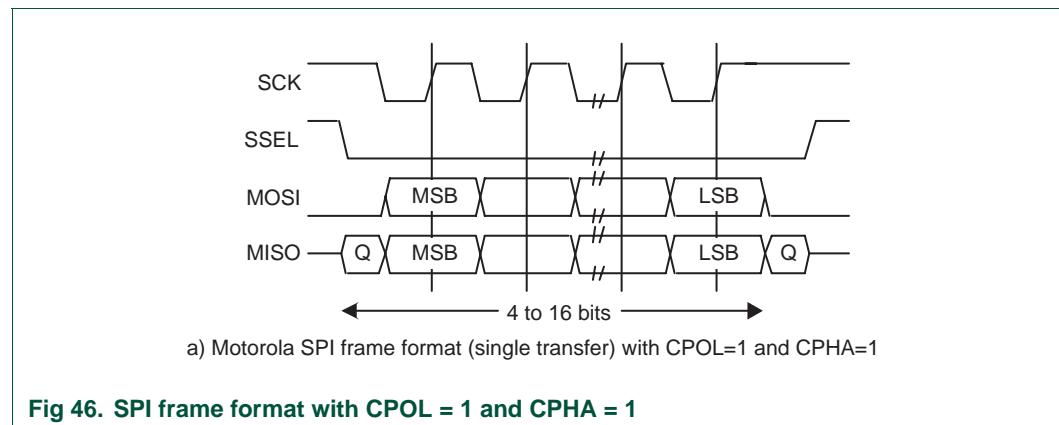


Fig 46. SPI frame format with CPOL = 1 and CPHA = 1

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.

13.3.8 Semiconductor Microwire frame format

[Figure 47](#) shows the Microwire frame format for a single frame. Figure 44 shows the same format when back-to-back frames are transmitted.

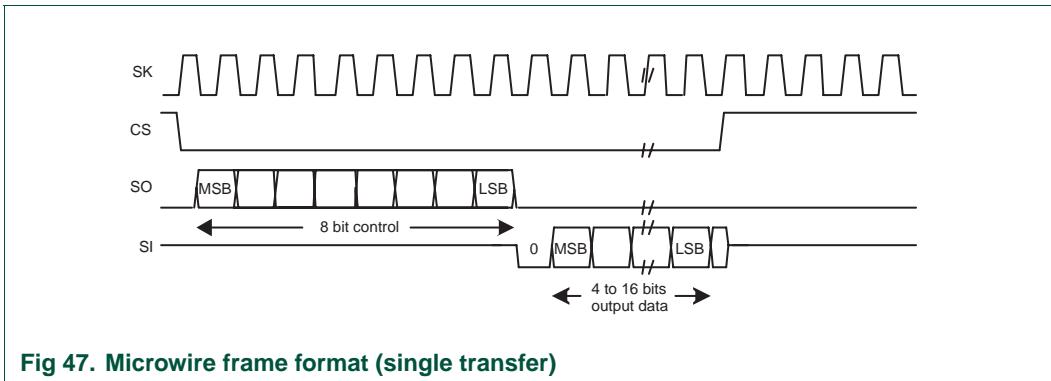


Fig 47. Microwire frame format (single transfer)

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 shifter, 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.

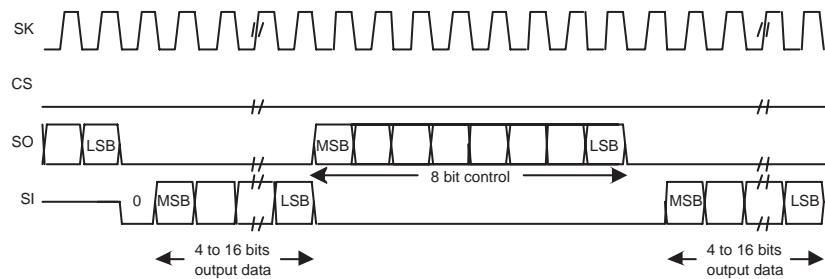


Fig 48. Microwire frame format (continuous transfers)

13.3.9 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 49](#) 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.

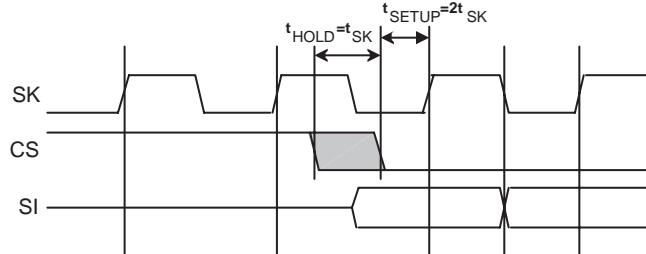


Fig 49. Microwire frame format (continuous transfers) - details

13.4 Register description

The SSP contains 9 registers as shown in [Table 162](#). All registers are byte, half word and word accessible.

Table 162: SSP register map

| Name | Description | Access | Reset value ^[1] | Address |
|---------|---|--------|----------------------------|-------------|
| SSPCR0 | Control Register 0. Selects the serial clock rate, bus type, and data size. | R/W | 0x0000 | 0xE006 8000 |
| SSPCR1 | Control Register 1. Selects master/slave and other modes. | R/W | 0x00 | 0xE006 8004 |
| SSPDR | Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO. | R/W | 0x0000 | 0xE006 8008 |
| SSPSR | Status Register | RO | 0x03 | 0xE006 800C |
| SSPCPSR | Clock Prescale Register | R/W | 0x00 | 0xE006 8010 |
| SSPIMSC | Interrupt Mask Set and Clear Register | R/W | 0x00 | 0xE006 8014 |
| SSPRIS | Raw Interrupt Status Register | R/W | 0x04 | 0xE006 8018 |
| SSPMIS | Masked Interrupt Status Register | RO | 0x00 | 0xE006 801C |
| SSPICR | SSPICR Interrupt Clear Register | WO | NA | 0xE006 8020 |

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

13.4.1 SSP Control Register 0 (SSPCR0 - 0xE006 8000)

This register controls the basic operation of the SSP controller.

Table 163: SSP Control Register 0 (SSPCR0 - address 0xE006 8000) 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 | SSI | |
| | | 10 | Microwire | |
| | | 11 | This combination is not supported and should not be used. | |

Table 163: SSP Control Register 0 (SSPCR0 - address 0xE006 8000) bit description

| Bit | Symbol | Value | Description | Reset value |
|--|--------|-------|---|-------------|
| 6 | CPOL | 0 | Clock Out Polarity. This bit is only used in SPI mode. | 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. | |
| 7 | CPHA | 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. | 0 |
| | | | Clock Out Phase. This bit is only used in SPI mode. | |
| 15:8 | SCR | 0 | SSP controller maintains the bus clock low between frames. | 0x00 |
| | | | SSP controller maintains the bus clock high between frames. | |
| 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 VPB clock PCLK clocks the prescaler, the bit frequency is $PCLK / (CPSDVR * [SCR+1])$. | | | | |

13.4.2 SSP Control Register 1 (SSPCR1 - 0xE006 8004)

This register controls certain aspects of the operation of the SSP controller.

Table 164: SSP Control Register 1 (SSPCR1 - address 0xE006 8004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--------|-------|--|-------------|
| 0 | LBM | 0 | Loop Back Mode. | 0 |
| | | | During normal operation. | |
| 1 | SSE | 1 | Serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively). | 0 |
| | | | SSP Enable. | |
| 2 | MS | 0 | The SSP controller is disabled. | 0 |
| | | | 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 | 1 | Master/Slave Mode. This bit can only be written when the SSE bit is 0. | 0 |
| | | | The SSP controller acts as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line. | |
| 3 | SOD | 1 | The SSP controller acts as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines. | 0 |
| | | | 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). | |
| 7:4 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

13.4.3 SSP Data Register (SSPDR - 0xE006 8008)

Software can write data to be transmitted to this register, and read data that has been received.

Table 165: SSP Data Register (SSPDR - address 0xE006 8008) 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 |

13.4.4 SSP Status Register (SSPSR - 0xE006 800C)

This read-only register reflects the current status of the SSP controller.

Table 166: SSP Status Register (SSPSR - address 0xE006 800C) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|---|-------------|
| 0 | TFE | Transmit FIFO Empty. This bit is 1 if 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 SSP 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 |

13.4.5 SSP Clock Prescale Register (SSPCPSR - 0xE006 8010)

This register controls the factor by which the Prescaler divides the VPB clock PCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in SSPCR0, to determine the bit clock.

Table 167: SSP Clock Prescale Register (SSPCPSR - address 0xE006 8010) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|---|-------------|
| 7:0 | CPSDVS | 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 SSPCPSR 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.

13.4.6 SSP Interrupt Mask Set/Clear register (SSPIMSC - 0xE006 8014)

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 168: SSP Interrupt Mask Set/Clear register (SSPIMSC - address 0xE006 8014) 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 new data has been received, nor has data been read from the FIFO, for 32 bit times. | 0 |
| 2 | RXIM | Software should set this bit to enable interrupt when the Rx FIFO is at least half full. | 0 |
| 3 | TXIM | Software should set this bit to enable interrupt when the Tx FIFO is at least half empty. | 0 |
| 7:4 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

13.4.7 SSP Raw Interrupt Status register (SSPRIS - 0xE006 8018)

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 SSPIMSC.

Table 169: SSP Raw Interrupt Status register (SSPRIS - address 0xE006 8018) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 0 | RORRIS | This bit is 1 if another frame was completely received while the Rx FIFO 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 there is a Receive Timeout condition. Note that a Receive Timeout can be negated if further data is received. | 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 |

13.4.8 SSP Masked Interrupt register (SSPMIS - 0xE006 801C)

This read-only register contains a 1 for each interrupt condition that is asserted and enabled in the SSPIMSC. When an SSP interrupt occurs, the interrupt service routine should read this register to determine the cause(s) of the interrupt.

Table 170: SSP Masked Interrupt Status register (SSPMIS -address 0xE006 801C) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 0 | RORMIS | This bit is 1 if another frame was completely received while the Rx FIFO was full, and this interrupt is enabled. | 0 |
| 1 | RTMIS | This bit is 1 when there is a Receive Timeout condition and this interrupt is enabled. Note that a Receive Timeout can be negated if further data is received. | 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 |
| 7:5 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

13.4.9 SSP Interrupt Clear Register (SSPICR - 0xE006 8020)

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 SSPIMSC.

Table 171: SSP interrupt Clear Register (SSPICR - address 0xE006 8020) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 0 | RORIC | Writing a 1 to this bit clears the “frame was received when Rx FIFO was full” interrupt. | NA |
| 1 | RTIC | Writing a 1 to this bit clears the Receive Timeout 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 |



14.1 Introduction

The USB is a 4 wire bus that supports communication between a host and a number (127 max.) of peripherals. The host controller allocates the USB bandwidth to attached devices through a token based protocol. The bus supports hot plugging, un-plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 1 ms frames. Each frame contains SoF marker and transactions that transfer data to/from device endpoints. Each device can have a maximum of 16 logical or 32 physical endpoints. There are 4 types of transfers defined on the endpoints. The control transfers are used to configure the device. The interrupt transfers are used for periodic data transfer. The bulk transfers are used when rate of transfer is not critical. The isochronous transfers have guaranteed delivery time but no error correction.

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

Table 172: USB related acronyms, abbreviations and definitions used in this chapter

| Acronym/abbreviation | Description |
|----------------------|--|
| AHB | Advanced High-performance bus |
| ATLE | Auto Transfer Length Extraction |
| ATX | Analog Transceiver |
| DD | DMA Descriptor |
| DC | Device Core |
| DDP | DD Pointer |
| DMA | Direct Memory Access |
| EoP | End of Package |
| EP | End Point |
| FS | Full Speed |
| HREADY | When HIGH the HREADY signal indicates that a transfer has finished on the AHB bus. This signal may be driven LOW to extend a transfer. |
| LED | Light Emitting Diode |
| LS | Low Speed |
| MPS | Maximum Packet Size |
| PLL | Phase Locked Loop |
| RAM | Random Access Memory |
| SoF | Start of Frame |
| SIE | Serial Interface Engine |

Table 172: 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 |

14.2 Features

- Fully compliant with USB 2.0 Full Speed specification
- Supports 32 physical (16 logical) endpoints
- Supports Control, Bulk, Interrupt and Isochronous endpoints
- Scalable realization of endpoints at run time
- Endpoint Maximum packet size selection (up to USB maximum specification) by software at run time
- RAM message buffer size based on endpoint realization and maximum packet size
- Supports Soft Connect™ feature and Good Link™ LED indicator
- Supports bus-powered capability with low suspend current
- Support DMA transfer with the DMA RAM of 8 kB on all non-control endpoints (LPC2146/8 only)
- One Duplex DMA channel serves all endpoints (LPC2146/8 only)
- Allows dynamic switching between CPU controlled and DMA modes (available on LPC2146/8 only)
- Double buffer implementation for Bulk & Isochronous endpoints

14.3 Fixed Endpoint Configuration

Table 173: Pre-Fixed Endpoint Configuration

| Logical endpoint | Physical endpoint | Endpoint type | Direction | Packet size (bytes) | Double buffer |
|------------------|-------------------|---------------|-----------|---------------------|---------------|
| 0 | 0 | Control | Out | 8, 16, 32, 64 | No |
| 0 | 1 | Control | In | 8, 16, 32, 64 | No |
| 1 | 2 | Interrupt | Out | 1 to 64 | No |
| 1 | 3 | Interrupt | In | 1 to 64 | No |
| 2 | 4 | Bulk | Out | 8, 16, 32, 64 | Yes |
| 2 | 5 | Bulk | In | 8, 16, 32, 64 | Yes |
| 3 | 6 | Isochronous | Out | 1 to 1023 | Yes |
| 3 | 7 | Isochronous | In | 1 to 1023 | Yes |
| 4 | 8 | Interrupt | Out | 1 to 64 | No |
| 4 | 9 | Interrupt | In | 1 to 64 | No |
| 5 | 10 | Bulk | Out | 8, 16, 32, 64 | Yes |
| 5 | 11 | Bulk | In | 8, 16, 32, 64 | Yes |
| 6 | 12 | Isochronous | Out | 1 to 1023 | Yes |
| 6 | 13 | Isochronous | In | 1 to 1023 | Yes |

Table 173: Pre-Fixed Endpoint Configuration

| Logical endpoint | Physical endpoint | Endpoint type | Direction | Packet size (bytes) | Double buffer |
|------------------|-------------------|---------------|-----------|---------------------|---------------|
| 7 | 14 | Interrupt | Out | 1 to 64 | No |
| 7 | 15 | Interrupt | In | 1 to 64 | No |
| 8 | 16 | Bulk | Out | 8, 16, 32, 64 | Yes |
| 8 | 17 | Bulk | In | 8, 16, 32, 64 | Yes |
| 9 | 18 | Isochronous | Out | 1 to 1023 | Yes |
| 9 | 19 | Isochronous | In | 1 to 1023 | Yes |
| 10 | 20 | Interrupt | Out | 1 to 64 | No |
| 10 | 21 | Interrupt | In | 1 to 64 | No |
| 11 | 22 | Bulk | Out | 8, 16, 32, 64 | Yes |
| 11 | 23 | Bulk | In | 8, 16, 32, 64 | Yes |
| 12 | 24 | Isochronous | Out | 1 to 1023 | Yes |
| 12 | 25 | Isochronous | In | 1 to 1023 | Yes |
| 13 | 26 | Interrupt | Out | 1 to 64 | No |
| 13 | 27 | Interrupt | In | 1 to 64 | No |
| 14 | 28 | Bulk | Out | 8, 16, 32, 64 | Yes |
| 14 | 29 | Bulk | In | 8, 16, 32, 64 | Yes |
| 15 | 30 | Bulk | Out | 8, 16, 32, 64 | Yes |
| 15 | 31 | Bulk | In | 8, 16, 32, 64 | Yes |

14.4 Architecture

The architecture of the USB device controller is shown below in the block diagram.

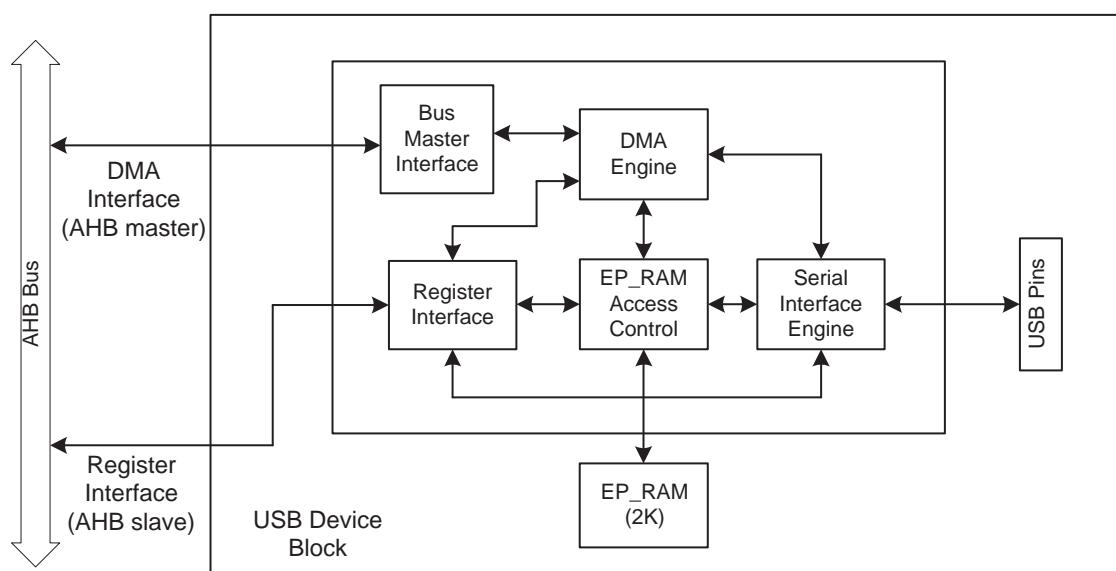


Fig 50. USB Device Controller Block Diagram

14.5 Data Flow

USB is a host controlled protocol, i.e., irrespective of whether the data transfer is from the host to the device or device to the host, transfer sequence is always initiated by the host. During data transfer from device to the host, the host sends an IN token to the device, following which the device responds with the data.

14.5.1 Data Flow from USB Host to the Device

The USB device protocol engine receives the serial data from the ATX and converts it into a parallel data stream. The parallel data is sent to the RAM interface which in turn transfers the data to the endpoint buffer. The endpoint buffer is implemented as an SRAM based FIFO. Each realized endpoint will have a reserved space in the RAM. So the total RAM space required depends on the number of realized endpoints, maximum packet size of the endpoint and whether the endpoint supports double buffering. Data is written to the buffers with the header showing how many bytes are valid in the buffer.

For non-isochronous endpoints, when a full data packet is received without any errors, the endpoint generates a request for data transfer from its FIFO by generating an interrupt to the system.

Isochronous endpoint will have one packet of data to be transferred in every frame. So the data transfer has to be synchronized to the USB frame rather than packet arrival. So, for every 1 ms there will be an interrupt to the system.

The data transfer follows the little endian format. The first byte received from the USB bus will be available in the least significant byte of the receive data register.

14.5.2 Data Flow from Device to the Host

For data transfer from an endpoint to the host, the host will send an IN token to that endpoint. If the FIFO corresponding to the endpoint is empty, the device will return a NAK and will raise an interrupt to the system. On this interrupt the CPU fills a packet of data in the endpoint FIFO. The next IN token that comes after filling this packet will transfer this packet to the host.

The data transfer follows the little endian format. The first byte sent on the USB bus will be the least significant byte of the transmit data register.

14.5.3 Slave Mode Transfer

Slave data transfer is done through the interrupt issued from the USB device to the CPU.

Reception of valid (error-free) data packet in any of the OUT non-isochronous endpoint buffer generates an interrupt. Upon receiving the interrupt, the software can read the data using receive length and data registers. When there is no empty buffer (for a given OUT non-isochronous endpoint), any data arrival generates an interrupt only if Interrupt on NAK feature for that endpoint type is enabled and the existing interrupt is cleared. For OUT isochronous endpoints, the data will always be written irrespective of the buffer status. There will be no interrupt generated specific to OUT isochronous endpoints other than the frame interrupt.

Similarly, when a packet is successfully transferred to the host from any of IN non-isochronous endpoint buffer, an interrupt is generated. When there is no data available in any of the buffers (for a given IN non-isochronous endpoint), a data request generates an interrupt only if Interrupt on NAK feature for that endpoint type is enabled and existing interrupt is cleared. Upon receiving the interrupt, the software can load any data to be sent using transmit length and data registers. For IN isochronous endpoints, the data available in the buffer will be sent only if the buffer is validated; otherwise, an empty packet will be sent. Like OUT isochronous endpoints, there will be no interrupt generated specific to IN isochronous endpoints other than the frame interrupt.

14.5.4 DMA Mode Transfer (LPC2146/8 only)

Under DMA mode operation the USB device will act as a master on the AHB bus and transfers the data directly from the memory to the endpoint buffer and vice versa. A duplex channel DMA acts as a AHB master on the bus.

The endpoint 0 of USB (default control endpoint) will receive the setup packet. It will not be efficient to transfer this data to the USB RAM since the CPU has to decode this command and respond back to the host. So, this transfer will happen in the slave mode only.

For each Isochronous endpoint, one packet transfer happens every frame. Hence, the DMA transfer has to be synchronized to the frame interrupt.

The DMA engine also support Auto Transfer Length Extraction (ATLE) mode for bulk transfers. In this mode the DMA engine recovers the transfer size from the incoming packet stream.

14.6 Interfaces

14.6.1 Software Interface

The software interface of the USB device block consists of a register view and the format definitions for the endpoint descriptors. These two aspects are addressed in the following sections.

14.6.2 Register Map

The following registers are located in the AHB clock domain. The minimum AHB clock frequency should be 18 MHz. They can be accessed directly by the CPU. All registers are 32 bit wide and aligned in the word address boundaries.

USB slave mode registers are located in the address region 0xE009 0000 to 0xE009 004C. All unused address in this region reads “DEADABBA”.

DMA related registers are located in the address region 0xE009 0050 to 0xE009 00FC. All unused address in this region reads invalid data.

Table 174: USB device register map

| Name | Description | Access | Reset value ^[1] | Address |
|---------------------------------------|--------------------------------------|--------|----------------------------|-------------|
| Device interrupt registers | | | | |
| USBIntSt | USB Interrupt Status | R/W | 0x8000 0000 | 0xE01F C1C0 |
| USBDevIntSt | USB Device Interrupt Status | RO | 0x0000 0010 | 0xE009 0000 |
| USBDevIntEn | USB Device Interrupt Enable | R/W | 0x0000 0000 | 0xE009 0004 |
| USBDevIntClr | USB Device Interrupt Clear | WO | 0x0000 0000 | 0xE009 0008 |
| USBDevIntSet | USB Device Interrupt Set | WO | 0x0000 0000 | 0xE009 000C |
| USBDevIntPri | USB Device Interrupt Priority | WO | 0x00 | 0xE009 002C |
| Endpoint interrupt registers | | | | |
| USBEpIntSt | USB Endpoint Interrupt Status | RO | 0x0000 0000 | 0xE009 0030 |
| USBEpIntEn | USB Endpoint Interrupt Enable | R/W | 0x0000 0000 | 0xE009 0034 |
| USBEpIntClr | USB Endpoint Interrupt Clear | WO | 0x0000 0000 | 0xE009 0038 |
| USBEpIntSet | USB Endpoint Interrupt Set | WO | 0x0000 0000 | 0xE009 003C |
| USBEpIntPri | USB Endpoint Priority | WO | 0x0000 0000 | 0xE009 0040 |
| Endpoint realization registers | | | | |
| USBReEp | USB Realize Endpoint | R/W | 0x0000 0003 | 0xE009 0044 |
| USBEplnd | USB Endpoint Index | WO | 0x0000 0000 | 0xE009 0048 |
| USBMaxPSize | USB MaxPacketSize | R/W | 0x0000 0008 | 0xE009 004C |
| USB transfer registers | | | | |
| USBRxData | USB Receive Data | RO | 0x0000 0000 | 0xE009 0018 |
| USBRxPLen | USB Receive Packet Length | RO | 0x0000 0000 | 0xE009 0020 |
| USBTxData | USB Transmit Data | WO | 0x0000 0000 | 0xE009 001C |
| USBTxPLen | USB Transmit Packet Length | WO | 0x0000 0000 | 0xE009 0024 |
| USBCtrl | USB Control | R/W | 0x0000 0000 | 0xE009 0028 |
| Command registers | | | | |
| USBCmdCode | USB Command Code | WO | 0x0000 0000 | 0xE009 0010 |
| USBCmdData | USB Command Data | RO | 0x0000 0000 | 0xE009 0014 |
| DMA registers (LPC2146/8 only) | | | | |
| USBDMARSt | USB DMA Request Status | RO | 0x0000 0000 | 0xE009 0050 |
| USBDMARClr | USB DMA Request Clear | WO | 0x0000 0000 | 0xE009 0054 |
| USBDMARSet | USB DMA Request Set | WO | 0x0000 0000 | 0xE009 0058 |
| USBUDCAH | USB UDCA Head | R/W | 0x0000 0000 | 0xE009 0080 |
| USBEpDMASt | USB Endpoint DMA Status | RO | 0x0000 0000 | 0xE009 0084 |
| USBEpDMAEn | USB Endpoint DMA Enable | WO | 0x0000 0000 | 0xE009 0088 |
| USBEpDMADis | USB Endpoint DMA Disable | WO | 0x0000 0000 | 0xE009 008C |
| USBDMAIntSt | USB DMA Interrupt Status | RO | 0x0000 0000 | 0xE009 0090 |
| USBDMAIntEn | USB DMA Interrupt Enable | R/W | 0x0000 0000 | 0xE009 0094 |
| USBEoTIntSt | USB End of Transfer Interrupt Status | RO | 0x0000 0000 | 0xE009 00A0 |
| USBEoTIntClr | USB End of Transfer Interrupt Clear | WO | 0x0000 0000 | 0xE009 00A4 |
| USBEoTIntSet | USB End of Transfer Interrupt Set | WO | 0x0000 0000 | 0xE009 00A8 |
| USBNDDRIntSt | USB New DD Request Interrupt Status | RO | 0x0000 0000 | 0xE009 00AC |
| USBNDDRIntClr | USB New DD Request Interrupt Clear | WO | 0x0000 0000 | 0xE009 00B0 |

Table 174: USB device register map

| Name | Description | Access | Reset value ^[1] | Address |
|-----------------|-----------------------------------|--------|----------------------------|-------------|
| USBNDDRIntSet | USB New DD Request Interrupt Set | WO | 0x0000 0000 | 0xE009 00B4 |
| USBSysErrIntSt | USB System Error Interrupt Status | RO | 0x0000 0000 | 0xE009 00B8 |
| USBSysErrIntClr | USB System Error Interrupt Clear | WO | 0x0000 0000 | 0xE009 00BC |
| USBSysErrIntSet | USB System Error Interrupt Set | WO | 0x0000 0000 | 0xE009 00C0 |

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

14.7 USB Device register definitions

14.7.1 USB Interrupt Status register (USBIntSt - 0xE01F C1C0)

The USB device has three interrupt output lines. The interrupts `usb_int_req_lp` and `usb_int_req_hp` facilitates transfer of data in slave mode. These two interrupt lines are provided to allow two different priority (high/low) levels in slave mode transfer. Each of the individual endpoint interrupts can be routed to either high priority or low priority levels using corresponding bits in the Endpoint Interrupt Priority register ([Section 14.7.11](#)). The interrupt level is triggered with active high polarity. The external interrupt generation takes place only if the necessary ‘enable’ bits are set in the Device Interrupt Enable register ([Section 14.7.3](#)). Otherwise, they will be registered only in the status registers. The `usb_int_req_dma` is raised when an `end_of_transfer` or a system error has occurred. DMA data transfer is not dependent on this interrupt.

The three interrupt output lines are ORed together to reduce the number of interrupt channels required for the USB device in the vectored interrupt controller. This register reflects the status of the each interrupt line. The `USBIntSt` is a read/write register.

Table 175: 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. (LPC2146/8 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 |

Table 175: USB Interrupt Status register (USBIntSt - address 0xE01F C1C0) bit description

| Bit | Symbol | Description | Reset value |
|------|----------------|---|-------------|
| 8 | USB_need_clock | USB need clock indicator. This bit is set to 1 when a USB activity/change of state on the USB data pins is detected, and it indicates that a USB PLL supplied clock of 48 MHz is needed. Once the USB_need_clock becomes one, it resets to zero 3 ms after the last frame has been received/sent. A change of this bit from 0 to 1 can wake up the microcontroller if an activity on the USB bus is selected to wake up the part from the Power-down mode (see Section 3.5.3 "Interrupt Wakeup register (INTWAKE - 0xE01F C144)" on page 22 for details). Also see Section 3.8.8 "PLL and Power-down mode" on page 32 and Section 3.9.2 "Power Control register (PCON - 0xE01F COCO)" on page 35 for considerations about the USB PLL and invoking the Power Down mode. | 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 ORed output of the USB interrupt lines is not seen by the Vectored Interrupt Controller. | 1 |

14.7.2 USB Device Interrupt Status register (USBDevIntSt - 0xE009 0000)

Interrupt status register holds the value of the interrupt. A 0 indicates no interrupt and 1 indicates the presence of the interrupt. The USBDevIntSt is a read only register.

Table 176: USB Device Interrupt Status register (USBDevIntSt - address 0xE009 0000) 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 | - | - | - | - | - | - | EPR_INT | EP_RLZED |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | TxDNPKT | RxDNPKT | CDFULL | CCEMTY | DEV_STAT | EP_SLOW | EP_FAST | FRAME |

Table 177: USB Device Interrupt Status register (USBDevIntSt - address 0xE009 0000) bit description

| Bit | Symbol | Description | Reset value |
|-----|----------|--|-------------|
| 0 | FRAME | The frame interrupt occurs every 1 ms. This is to be used in isochronous packet transfer. | 0 |
| 1 | EP_FAST | This is the fast interrupt transfer for the endpoint. If an Endpoint Interrupt Priority register bit is set, the endpoint interrupt will be routed to this bit. | 0 |
| 2 | EP_SLOW | This is the Slow interrupt transfer for the endpoint. If an Endpoint Interrupt Priority Register bit is not set, the endpoint interrupt will be routed to this bit. | 0 |
| 3 | DEV_STAT | Set when USB Bus reset, USB suspend change or Connect change event occurs. Refer to Section 14.9.6 "Set Device Status (Command: 0xFE, Data: write 1 byte)" on page 225 . | 0 |
| 4 | CCEMTY | The command code register is empty (New command can be written). | 1 |
| 5 | CDFULL | Command data register is full (Data can be read now). | 0 |

Table 177: USB Device Interrupt Status register (USBDevIntSt - address 0xE009 0000) bit description

| Bit | Symbol | Description | Reset value |
|-------|----------|--|-------------|
| 6 | RxENDPKT | The current packet in the FIFO is transferred to the CPU. | 0 |
| 7 | TxENDPKT | The number of data bytes transferred to the FIFO equals the number of bytes programmed in the TxPacket length register. | 0 |
| 8 | EP_RLZED | Endpoints realized. Set when Realize endpoint register or Maxpacket size register is updated. | 0 |
| 9 | ERR_INT | Error Interrupt. Any bus error interrupt from the USB device. Refer to Section 14.9.9 "Read Error Status (Command: 0xFB, Data: read 1 byte)" on page 227 | 0 |
| 31:10 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

14.7.3 USB Device Interrupt Enable register (USBDevIntEn - 0xE009 0004)

If the Interrupt Enable bit value is set, an interrupt is generated (on Fast or Slow Interrupt line) when the corresponding bit in the Device Interrupt Status register is set ([Section 14.7.2](#)). If it is not set, no external interrupt is generated but interrupt will still be held in the interrupt status register. All bits of this register are cleared after reset. The USBDevIntEn is a read/write register.

Table 178: USB Device Interrupt Enable register (USBDevIntEn - address 0xE009 0004) 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 | - | - | - | - | - | - | EPR_INT | EP_RLZED |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | TxENDPKT | RxENDPKT | CDFULL | CCEMTY | DEV_STAT | EP_SLOW | EP_FAST | FRAME |

Table 179: USB Device Interrupt Enable register (USBDevIntEn - address 0xE009 0004) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--|-------|-------------------------------------|-------------|
| 31:0 | See USBDevIntEn bit allocation table above | 0 | No external interrupt is generated. | 0 |

14.7.4 USB Device Interrupt Clear register (USBDevIntCir - 0xE009 0008)

Setting a particular bit to 1 in this register causes the clearing of the interrupt by resetting the corresponding bit in the interrupt status register. Writing a 0 will not have any influence. The USBDevIntCir is a write only register.

Table 180: USB Device Interrupt Clear register (USBDevIntCir - address 0xE009 0008) 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 | - | - | - | - | - | - | EPR_INT | EP_RLZED |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | TxENDPKT | Rx ENDPKT | CDFULL | CCEMTY | DEV_STAT | EP_SLOW | EP_FAST | FRAME |

Table 181: USB Device Interrupt Clear register (USBDevIntClr - address 0xE009 0008) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--|--------|---|-------------|
| 31:0 | See USBDevIntClr bit allocation table above | 0 1 | No effect. The corresponding bit in the Device Interrupt Status register (Section 14.7.2) is cleared. | 0 |

14.7.5 USB Device Interrupt Set register (USBDevIntSet - 0xE009 000C)

Setting a particular bit to 1 in this register will set the corresponding bit in the Interrupt Status register. Writing a 0 will not have any influence. The USBDevIntSet is a write only register.

Table 182: USB Device Interrupt Set register (USBDevIntSet - address 0xE009 000C) 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 | - | - | - | - | - | - | EPR_INT | EP_RLZED |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Symbol | TxENDPKT | Rx ENDPKT | CDFULL | CCEMTY | DEV_STAT | EP_SLOW | EP_FAST | FRAME |

Table 183: USB Device Interrupt Set register (USBDevIntSet - address 0xE009 000C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--|--------|---|-------------|
| 31:0 | See USBDevIntSet bit allocation table above | 0 1 | No effect. The corresponding bit in the Device Interrupt Status register (Section 14.7.2) is set. | 0 |

14.7.6 USB Device Interrupt Priority register (USBDevIntPri - 0xE009 002C)

By setting a particular bit to 1, the corresponding interrupt will be routed to the high priority interrupt line. If the bit is 0 the interrupt will be routed to the low priority interrupt line. Only one of the EP_FAST or FRAME can be routed to the high priority interrupt line. Setting both bits at the same time is not allowed. If the software attempts to set both bits to 1, none of them will be routed to the high priority interrupt line. All enabled endpoint

interrupts will be routed to the low priority interrupt line if the EP_FAST bit is set to 0, irrespective of the Endpoint Interrupt Priority register ([Section 14.7.11](#)) setting. The USBDevIntPri is a write only register.

Table 184: USB Device Interrupt Priority register (USBDevIntPri - address 0xE009 002C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------|-------|--|-------------|
| 0 | FRAME | 0 | FRAME interrupt is routed to the low priority interrupt line. | 0 |
| | | 1 | FRAME interrupt is routed to the high priority interrupt line. | |
| 1 | EP_FAST | 0 | EP_FAST interrupt is routed to the low priority interrupt line. | 0 |
| | | 1 | EP_FAST interrupt is routed to the high priority interrupt line. | |
| 7:2 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

14.7.7 USB Endpoint Interrupt Status register (USBEplntSt - 0xE009 0030)

Each physical non-isochronous endpoint is represented by one bit in this register to indicate that it has generated the interrupt. All non-isochronous OUT endpoints give an interrupt when they receive a packet without any error. All non-isochronous IN endpoints will give an interrupt when a packet is successfully transmitted or a NAK handshake is sent on the bus provided that the interrupt on NAK feature is enabled. Isochronous endpoint transfer takes place with respect to frame interrupt. The USBEplntSt is a read only register.

Table 185: USB Endpoint Interrupt Status register (USBEplntSt - address 0xE009 0030) 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 186: USB Endpoint Interrupt Status register (USBEplntSt - address 0xE009 0030) 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 |

Table 186: USB Endpoint Interrupt Status register (USBEPlntSt - address 0xE009 0030) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 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 |
| 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 |

14.7.8 USB Endpoint Interrupt Enable register (USBEPlntEn - 0xE009 0034)

Setting bits in this register will cause the corresponding bit in the interrupt status register to transfer its status to the device interrupt status register. Either the EP_FAST or EP_SLOW bit will be set depending on the value in the endpoint interrupt priority register. Setting this bit to 1 implies operating in the slave mode. The USBEPlntEn is a read/write register.

Table 187: USB Endpoint Interrupt Enable register (USBEPlntEn - address 0xE009 0034) 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 188: USB Endpoint Interrupt Enable register (USBEplntEn - address 0xE009 0034) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--|-------|-------------|-------------|
| 31:0 | See USBEplntEn bit allocation table above | 0 | No effect. | 0 |

14.7.9 USB Endpoint Interrupt Clear register (USBEplntCir - 0xE009 0038)

Writing a 1 to this bit clears the bit in the endpoint interrupt status register. Writing 0 will not have any impact. When the endpoint interrupt is cleared from this register, the hardware will clear the CDFULL bit in the Device Interrupt Status register. On completion of this action, the CDFULL bit will be set and the Command Data register will have the status of the endpoint. Endpoint Interrupt register and CDFULL bit of Device Interrupt status register are related through clearing of interrupts in USB clock domain. Whenever software attempts to clear a bit of Endpoint Interrupt register, hardware will clear CDFULL bit before it starts issuing "Select Endpoint/Clear Interrupt" command (refer to [Section 14.9.11 "Select Endpoint/Clear Interrupt \(Command: 0x40 - 0x5F, Data: read 1 byte\)" on page 229](#)) and sets the same bit when command data is available for reading. Software will have to wait for CDFULL bit to be set to '1' (whenever it expects data from hardware) before it can read Command Data register. Each physical endpoint has its own reserved bit in this register. The bit field definition is the same as the Endpoint Interrupt Status Register as shown in Table 172. The USBEplntCir is a write only register.

Table 189: USB Endpoint Interrupt Clear register (USBEplntCir - address 0xE009 0038) 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 190: USB Endpoint Interrupt Clear register (USBEplntCir - address 0xE009 0038) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---|-------|-------------|-------------|
| 31:0 | See USBEplntCir bit allocation table above | 0 | No effect. | 0 |

Software is allowed to issue clear operation on multiple endpoints as well. Let us take an example:

Assume bits 5 and 10 of Endpoint Interrupt Status register are to be cleared. The software can issue Clear operation by writing in Endpoint Interrupt Clear register (with corresponding bit positions set to '1'). Then hardware will do the following:

1. Clears CDFULL bit of Device Interrupt Status register.

2. Issues 'Select Endpoint/Interrupt Clear' command for endpoint 10.
3. Waits for command to get processed and CDFULL bit to get set.
4. Now, endpoint status (for endpoint 10) is available in Command Data register (note that hardware does not wait for the software to finish reading endpoint status in Command Data register for endpoint 10).
5. Clears CDFULL bit again.
6. Issues 'Select Endpoint/Interrupt Clear' command for endpoint 5.
7. Waits for command to get processed and CDFULL bit to get set.
8. Now, endpoint status (for endpoint 5) is available in Command Data register for the software to read.

14.7.10 USB Endpoint Interrupt Set register (USBEplnSet - 0xE009 003C)

Writing a 1 to a bit in this register sets the corresponding bit in the endpoint interrupt status register. Writing 0 will not have any impact. Each endpoint has its own bit in this register. The USBEplnSet is a write only register.

Table 191: USB Endpoint Interrupt Set register (USBEplnSet - address 0xE009 003C) 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 192: USB Endpoint Interrupt Set register (USBEplnSet - address 0xE009 003C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--|-------|-------------|-------------|
| 31:0 | See USBEplnSet bit allocation table above | 0 | No effect. | 0 |

14.7.11 USB Endpoint Interrupt Priority register (USBEplnPri - 0xE009 0040)

This register determines whether the interrupt has to be routed to the fast interrupt line (EP_FAST) or to the slow interrupt line (EP_SLOW). If set 1 the interrupt will be routed to the fast interrupt bit of the device status register. Otherwise it will be routed to the slow endpoint interrupt bit. Note that routing of multiple endpoints to EP_FAST or EP_SLOW is possible. The Device Interrupt Priority register may override this register setting. Refer to [Section 14.7.6 "USB Device Interrupt Priority register \(USBDevIntPri - 0xE009 002C\)" on page 203](#) for more details. The USBEplnPri is a write only register.

Table 193: USB Endpoint Interrupt Priority register (USBEplnPri - address 0xE009 0040) 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 194: USB Endpoint Interrupt Priority register (USBEplntPri - address 0xE009 0040) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---|-------|--|-------------|
| 31:0 | See USBEplntPri bit allocation table above | 0 | The corresponding interrupt will be routed to the slow endpoint interrupt bit in the Device Status register. | 0 |
| | | 1 | The corresponding interrupt will be routed to the fast endpoint interrupt bit in the Device Status register. | |

14.7.12 USB Realize Endpoint register (USBReEp - 0xE009 0044)

Though fixed-endpoint configuration implements 32 endpoints, it is not a must that all have to be used. If the endpoint has to be used, it should have buffer space in the EP_RAM. The EP_RAM space can be optimized by realizing a subset of endpoints. This is done through programming the Realize Endpoint register. Each physical endpoint has one bit as shown in [Table 196](#). The USBReEp is a read/write register.

Table 195: USB Realize Endpoint register (USBReEp - address 0xE009 0044) 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 196: USB Realize Endpoint register (USBReEp - address 0xE009 0044) 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. | |

At power on only default control endpoint is realized. Other endpoints if required have to be realized by programming the corresponding bit in the Realize Endpoint register. Realization of endpoints is a multi-cycle operation. The pseudo code of endpoint realization is shown below.

for every endpoint to be realized,

```

{
    /* OR with the existing value of the register */
    RealizeEndpointRegister |= (UInt32) ((0x1 << endpt));
    /* Load endpoint index Reg with physical endpoint no.*/
    EndpointIndexRegister = (UInt32) endpointnumber;

    /* load the max packet size Register */
    Endpoint MaxPacketSizeReg = PacketSize;

    /* check whether the EP_RLSED bit is set */
    while (!(DeviceInterruptStatusReg & PFL_HW_EP_RLSED_BIT))
    {
        /* wait till endpoint realization is complete */
    }
    /* Clear the EP_RLSED bit */
    Clear EP_RLSED bit in DeviceInterrupt Status Reg;
}

```

Device will not respond to any tokens to the un-realized endpoint. ‘Configure Device’ command can only enable all realized and enabled endpoints. For details see [Section 14.9.2 “Configure Device \(Command: 0xD8, Data: write 1 byte\)” on page 223](#).

14.8 EP_RAM requirements

The USB device controller uses dedicated RAM based FIFO (EP_RAM) as an endpoint buffer. Each endpoint has a reserved space in the EP_RAM. The EP_RAM size requirement for an endpoint depends on its Maxpacketsize and whether it is double buffered or not. 32 words of EP_RAM are used by the device for storing the buffer pointers. The EP_RAM is word aligned but the Maxpacketsize is defined in bytes hence the RAM depth has to be adjusted to the next word boundary. Also, each buffer has one word header showing the size of the packet length received.

EP_RAM size (in words) required for the physical endpoint can be expressed as

$$\text{EP_RAMsize} = ((\text{Maxpacketsize} + 3) / 4 + 1) \times \text{db_status}$$

where db_status = 1 for single buffered endpoint and 2 for double buffered endpoint.

Since all the realized endpoints occupy EP_RAM space, the total EP_RAM requirement is

$$\text{TotalEPRAMsize} = 32 + \sum_{n=0}^N \text{epramsize}(n)$$

where N is the number of realized endpoints. Total EP_RAM size should not exceed 2048 bytes (2 kB, 0.5 kwords).

EP_RAM can be accessed by 3 sources, which are SIE, DMA engine and CPU. Among them, CPU has the highest priority followed by the SIE and DMA engine. The DMA engine has got the lowest priority. Then again, under the above mentioned 3 request sources, write request has got higher priority than read request. Typically, CPU does single word

read or write accesses, the DMA logic can do 32-byte burst access. The CPU and DMA engine operates at a higher clock frequency as compared to the SIE engine. The CPU cycles are valuable and so the CPU is given the highest priority. The CPU clock frequency is higher than the SIE operating frequency (12 MHz). The SIE will take 32 clock cycles for a word transfer. In general, this time translates to more than 32 clock cycles of the CPU in which it can do easily several accesses to the memory.

14.8.1 USB Endpoint Index register (USBEPln - 0xE009 0048)

Each endpoint has a register carrying the Maxpacket size value for that endpoint. This is in fact a register array. Hence before writing, this register has to be ‘addressed’ through the Endpoint Index register. The USBEPln is a write only register.

The Endpoint Index register will hold the physical endpoint number. Writing into the Maxpacket size register will set the array element pointed by the Endpoint Index register.

Table 197: USB Endpoint Index register (USBEPln - address 0xE009 0048) bit description

| Bit | Symbol | Description | Reset value |
|------|--------------|--|-------------|
| 4:0 | Phy_endpoint | 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 |

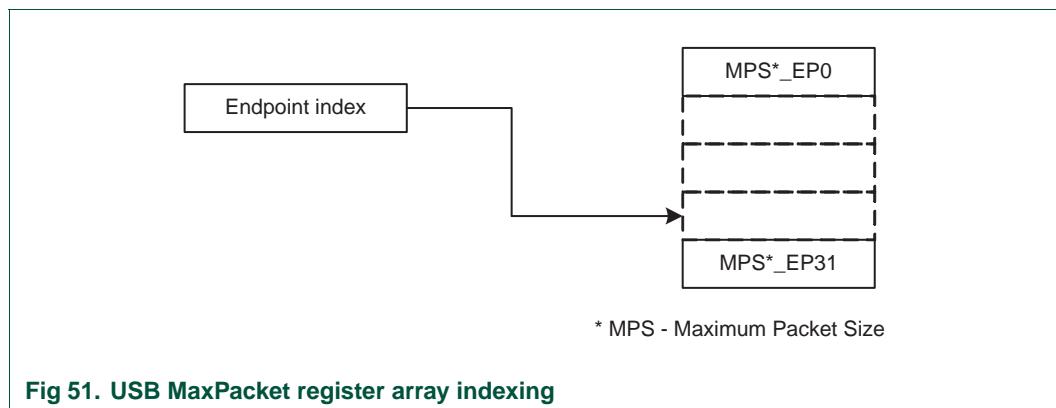
14.8.2 USB MaxPacketSize register (USBMaxPSize - 0xE009 004C)

At power on control endpoint is assigned the Maxpacketsize of 8 bytes. Other endpoints are assigned 0. Modifying MaxPacketSize register content will cause the buffer address of the internal RAM to be recalculated. This is essentially a multi-cycle process. At the end of it, the EP_RLZED bit will be set in the Device Interrupt Status register ([Section 14.7.2](#)).

The USB MaxPacket register array indexing is shown in [Figure 51](#). The USBMaxPSize is a read/write register.

Table 198: USB MaxPacketSize register (USBMaxPSize - address 0xE009 004C) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------------|--|-------------|
| 9:0 | MaxPacketSize | The maximum packet size value. | 0x008 |
| 31:10 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |



14.8.3 USB Receive Data register (USBRxData - 0xE009 0018)

For an OUT transaction, CPU reads the endpoint data from this register. Data from the endpoint RAM is fetched and filled in this register. There is no interrupt when the register is full. The USBRxData is a read only register.

Table 199: USB Receive Data register (USBRxData - address 0xE009 0018) bit description

| Bit | Symbol | Description | Reset value |
|------|-------------|----------------|-------------|
| 31:0 | ReceiveData | Data received. | 0x0000 0000 |

14.8.4 USB Receive Packet Length register (USBRxPLen - 0xE009 0020)

This register gives the number of bytes remaining in the EP_RAM for the current packet being transferred and whether the packet is valid or not. This register will get updated at every word that gets transferred to the system. Software can use this register to get the number of bytes to be transferred. When the number of bytes reaches zero, an end of packet interrupt is generated. The USBRxPLen is a read only register.

Table 200: USB Receive Packet Length register (USBRxPLen - address 0xE009 0020) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|---|-------------|
| 9:0 | PKT_LNGTH | - | The remaining amount of data in bytes still to be read from the RAM. | 0 |
| 10 | DV | | Non-isochronous end point will not raise an interrupt when an erroneous data packet is received. But invalid data packet can be produced with bus reset. For isochronous endpoint, data transfer will happen even if an erroneous packet is received. In this case DV bit will not be set for the packet. | 0 |
| | | 0 | Data is invalid. | |
| | | 1 | Data is valid. | |
| 11 | PKT_RDY | - | Packet length field in the register is valid and 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 |

14.8.5 USB Transmit Data register (USBTxDData - 0xE009 001C)

For an IN transaction the CPU writes the data into this register. This data will be transferred into the EP_RAM before the next writing occurs. There is no interrupt when the register is empty. The USBTxDData is a write only register.

Table 201: USB Transmit Data register (USBTxDData - address 0xE009 001C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------------|----------------|-------------|
| 31:0 | TransmitData | Transmit Data. | 0x0000 0000 |

14.8.6 USB Transmit Packet Length register (USBTxPLen - 0xE009 0024)

The software should first write the packet length (\leq Maximum Packet Size) in the Transmit Packet Length register followed by the data write(s) to the Transmit Data register. This register counts the number of bytes transferred from the CPU to the EP_RAM. The

software can read this register to determine the number of bytes it has transferred to the EP_RAM. After each write to the Transmit Data register the hardware will decrement the contents of the Transmit Packet Length register. For lengths larger than the Maximum Packet Size, the software should submit data in steps of Maximum Packet Size and the remaining extra bytes in the last packet. For example, if the Maximum Packet Size is 64 bytes and the data buffer to be transferred is of length 130 bytes, then the software submits 64 bytes packet twice followed by 2 bytes in the last packet. So, a total of 3 packets are sent on USB. The USBTxPLen is a write only register.

Table 202: USB Transmit Packet Length register (USBTxPLen - address 0xE009 0024) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 9:0 | PKT_LNGTH | - | The remaining amount of data in bytes to be written to the EP_RAM. | 0x000 |
| 31:10 | - | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

14.8.7 USB Control register (USBCtrl - 0xE009 0028)

This register controls the data transfer operation of the USB device. The USBCtrl is a read/write register.

Table 203: USB Control register (USBCtrl - address 0xE009 0028) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------------|-------|--|-------------|
| 0 | RD_EN | | Read mode control. | 0 |
| | | 0 | Read mode is disabled. | |
| | | 1 | Read mode is enabled. | |
| 1 | WR_EN | | Write mode control. | 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 |

14.8.8 Slave Mode data transfer

When the software wants to read the data from an endpoint buffer it should make the Read Enable bit high and should program the LOG_ENDPOINT in the USB control register. The control logic will first fetch the packet length to the receive packet length register. The PKT_RDY bit ([Table 200](#)) in the Packet Length Register is set along with this. Also the hardware fills the receive data register with the first word of the packet.

The software can now start reading the Receive Data register ([Section 14.8.3](#)). When the end of packet is reached the Read Enable bit (RD_EN in [Table 203](#)) will be disabled by the control logic and RxENDPkt bit is set in the Device Interrupt Status register. The software should issue a Clear Buffer (refer to [Section 14.9.13 “Clear Buffer \(Command: 0xF2, Data: read 1 byte \(optional\)\)” on page 230](#)) command. The endpoint is now ready to accept the next packet.

If the software makes the Read Enable bit low midway, the reading will be terminated. In this case the data will remain in the EP_RAM. When the Read Enable signal is made high again for this endpoint, data will be read from the beginning.

For writing data to an endpoint buffer, Write Enable bit (WR_EN in [Table 203](#)) should be made high and software should write to the Transmit Packet Length register ([Section 14.8.6](#)) the number of bytes it is going to send in the packet. It can then write data continuously in the Transmit Data register.

When the control logic receives the number of bytes programmed in the Transmit Packet Length register, it will reset the Write Enable bit. The TxENDPKT bit is set in the Device Interrupt Status register. The software should issue a Validate Buffer (refer to [Section 14.9.14 "Validate Buffer \(Command: 0xFA, Data: none\)" on page 230](#)) command. The endpoint is now ready to send the packet. If the software resets this bit midway, writing will start again from the beginning.

A synchronization mechanism is used to transfer data between the two clock domains i.e. AHB slave clock and the USB bit clock at 12 MHz. This synchronization process takes up to 5 clock cycles of the slow clock (i.e. 12 MHz) for reading/writing from/to a register before the next read/write can happen. The AHB HREADY output from the USB device is driven appropriately to take care of the timing.

Both Read Enable and Write Enable bits can be high at the same time for the same logical endpoint. The interleaved read and write operation is possible.

14.8.9 USB Command Code register (USBCmdCode - 0xE009 0010)

This register is used for writing the commands. The commands written here will get propagated to the Protocol Engine and will be executed there. After executing the command, the register will be empty, and the “CCEMTY” bit of the Interrupt Status register is set high. See [Section 14.9 “Protocol engine command description” on page 222](#) for details. The USBCmdCode is a write only register.

Table 204: USB Command Code register (USBCmdCode - address 0xE009 0010) bit description

| Bit | Symbol | 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 |
| 23:16 | CMD_CODE | The code for the command. | 0x00 |
| 31:24 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

14.8.10 USB Command Data register (USBCmdData - 0xE009 0014)

This is a read-only register which will carry the data retrieved after executing a command. When this register is ready, the “CD_FULL” bit of the Device Interrupt Status register is set. The CPU can poll this bit or enable an interrupt corresponding to this to sense the arrival of the data. The data is always one-byte wide. See [Section 14.9 “Protocol engine command description” on page 222](#) for details.

Table 205: USB Command Data register (USBCmdData - address 0xE009 0014) bit description

| Bit | Symbol | Description | Reset value |
|------|-------------|--|-------------|
| 7:0 | CommandData | Command 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 |

14.8.11 USB DMA Request Status register (USBDMARSt - 0xE009 0050)

This register is set by the hardware whenever a packet (OUT) or token (IN) is received on a realized endpoint. It serves as a flag for DMA engine to start the data transfer if the DMA is enabled for this particular endpoint. Each endpoint has one reserved bit in this register. Hardware sets this bit when a realized endpoint needs to be serviced through DMA. Software can read the register content. DMA cannot be enabled for control endpoints (EP0 and EP1). For easy readability the control endpoint is shown in the register contents. The USBDMARSt is a read only register.

Table 206: USB DMA Request Status register (USBDMARSt - address 0xE009 0050) 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 207: USB DMA Request Status register (USBDMARSt - address 0xE009 0050) 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 | 0 | Endpoint xx ($2 \leq xx \leq 31$) DMA request. | 0 |
| | | 1 | 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.

14.8.12 USB DMA Request Clear register (USBDMARClr - 0xE009 0054)

Writing 1 into the register will clear the corresponding interrupt from the DMA Request Status register. Writing 0 will not have any effect. Also, after a packet transfer, the hardware clears the particular bit in DMA Request Status register. The USBDMARClr is a write only register.

The USBDMARClr bit allocation is identical to the USBDMARSt register ([Table 206](#)).

Table 208: USB DMA Request Clear register (USBDMARClr - address 0xE009 0054) 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 ($2 \leq xx \leq 31$) DMA request. | 0 |
| | | 0 | No effect. | |
| | | 1 | Clear the corresponding interrupt from the DMA request register. | |

The software should not clear the DMA request clear bit while the DMA operation is in progress. But if at all the clearing happens, the behavior of DMA engine will depend on at what time the clearing is done. There can be more than one DMA requests pending at any given time. The DMA engine processes these requests serially (i.e starting from EP2 to EP31). If the DMA request for a particular endpoint is cleared before DMA operation has started for that request, then the DMA engine will never know about the request and no DMA operation on that endpoint will be done (till the next request appears). On the other hand, if the DMA request for a particular endpoint is cleared after the DMA operation corresponding to that request has begun, it does not matter even if the request is cleared, since the DMA engine has registered the endpoint number internally and will not sample the same request before finishing the current DMA operation.

14.8.13 USB DMA Request Set register (USBDMARSet - 0xE009 0058)

Writing 1 into the register will set the corresponding interrupt from the DMA request register. Writing 0 will not have any effect. The USBDMARSet is a write only register.

The USBDMARSet bit allocation is identical to the USBDMARSt register ([Table 206](#)).

Table 209: USB DMA Request Set register (USBDMARSet - address 0xE009 0058) 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 ($2 \leq xx \leq 31$) DMA request interrupt. | 0 |
| | | 0 | No effect. | |
| | | 1 | Set the corresponding interrupt from the DMA request register. | |

The DMA Request Set register is normally used for the test purpose. It is also useful in the normal operation mode to avoid a "lock" situation if the DMA is programmed after that the USB packets are already received. Normally the arrival of a packet generates an interrupt when it is completely received. This interrupt is used by the DMA to start working. This works fine as long as the DMA is programmed before the arrival of the packet (2 packets - if double buffered). If the DMA is programmed "too late", the interrupts were already generated in slave mode (but not handled because the intention was to use the DMA) and when the DMA is programmed no interrupts are generated to "activate" it. In this case the usage of the DMA Request Set register is useful to manually start the DMA transfer.

14.8.14 USB UDCA Head register (USBUDCAH - 0xE009 0080)

The UDCA (USB Device Communication Area) Head register maintains the address where UDCA is allocated in the USB RAM ([Figure 52](#)). The USB RAM is part of the system memory which is used for the USB purposes. It is located at address 0x7FD0 0000 and is 8 kB in size. Note, however, DMA on endpoint 0 is not feasible. The UDCA has to be aligned to 128 - byte boundary and should be of size 128 bytes (32 words that correspond to 32 physical endpoints). Each word can point to a DMA descriptor of a physical endpoint or can point to NULL (i.e. zero value) when the endpoint is not enabled for DMA operation. This implies that the DMA descriptors need to be created only for the DMA enabled endpoints. Gaps can be there while realizing the endpoints and there is no need to keep dummy DMA descriptors. The DMA engine will not process the descriptors of the DMA disabled endpoints. The reset value for this register is 0. Refer to [Section 14.10 "DMA descriptor" on page 230](#) and [Section 14.11 "DMA operation" on page 234](#) for more details on DMA descriptors. The USBUDCAH is a read/write register.

Table 210: USB UDCA Head register (USBUDCAH - address 0xE009 0080) bit description

| Bit | Symbol | Description | Reset value |
|------|-------------|--|-------------|
| 6:0 | - | UDCA header is aligned in 128-byte boundaries. | 0x00 |
| 31:7 | UDCA_Header | Start address of the UDCA Header. | 0 |

The DMA Request Set register is normally used for the test purpose. It is also useful in the normal operation mode to avoid a "lock" situation if the DMA is programmed after that the USB packets are already received. Normally the arrival of a packet generates an interrupt when it is completely received. This interrupt is used by the DMA to start working. This works fine as long as the DMA is programmed before the arrival of the packet (2 packets - if double buffered). If the DMA is programmed "too late", the interrupts were already generated in slave mode (but not handled because the intention was to use the DMA) and when the DMA is programmed no interrupts are generated to "activate" it. In this case the usage of the DMA Request Set register is useful to manually start the DMA transfer.

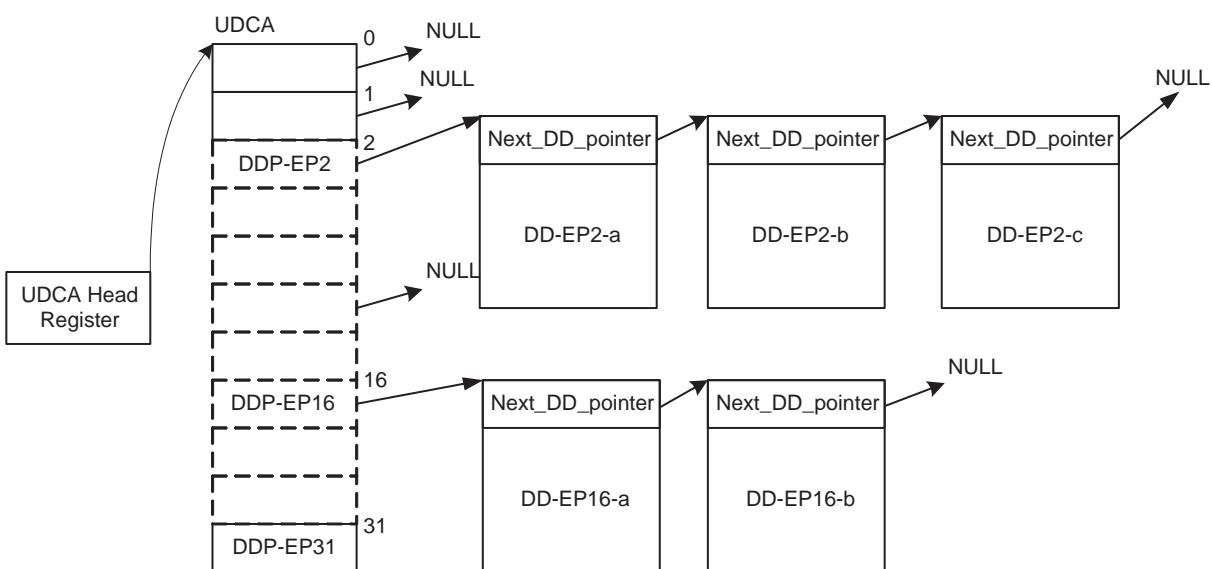


Fig 52. UDCA Head register and DMA descriptors

14.8.15 USB EP DMA Status register (USBEpDMASt - 0xE009 0084)

This register indicates whether the DMA for a particular endpoint is enabled or disabled. Each endpoint has one bit assigned in the EP DMA Status register. Bit 0 corresponds to endpoint 0 and Bit 31 to endpoint 15 IN). DMA transfer for a specific endpoint can start only if its bit is set in the USBEpDMASt register. Hence, it is referred as DMA_ENABLE bit. If the bit in the EP DMA Status register is made 0 (by writing into EP DMA Disable register) in between a packet transfer, the current packet transfer will still be completed. After the current packet, DMA gets disabled. In other words, the packet transfer when started will end unless an error condition occurs. When error condition is detected the bit will be reset by the hardware. The USBEpDMASt is a read only register.

Table 211: USB EP DMA Status register (USBEpDMASt - address 0xE009 0084) 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 ($2 \leq xx \leq 31$) DMA enabled bit. | 0 |
| | | 0 | The DMA for endpoint EPxx is disabled. | |
| | | 1 | The DMA for endpoint EPxx is enabled. | |

Software does not have direct write permission to this register. It has to set the bit through EP DMA Enable register. Resetting of the bit is done through EP DMA Disable register.

14.8.16 USB EP DMA Enable register (USBEpDMAEn - 0xE009 0088)

Writing 1 to this register will enable the DMA operation for the corresponding endpoint. Writing 0 will not have any effect. The USBEpDMAEn is a write only register.

Table 212: USB EP DMA Enable register (USBEpDMAEn - address 0xE009 0088) 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 ($2 \leq xx \leq 31$) DMA enable control bit. | 0 |
| | | 0 | No effect. | |
| | | 1 | Enable the DMA operation for endpoint EPxx. | |

14.8.17 USB EP DMA Disable register (USBEpDMADis - 0xE009 008C)

Writing 1 to this register will disable the DMA operation for the corresponding endpoint. Writing 0 will have the effect of resetting the DMA_PROCEED flag. The USBEpDMADis is a write only register.

Table 213: USB EP DMA Disable register (USBEpDMADis - address 0xE009 008C) 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 ≤ xx ≤ 31) DMA disable control bit. | 0 |
| | | 0 | No effect. | |
| | | 1 | Disable the DMA operation for endpoint EPxx. | |

14.8.18 USB DMA Interrupt Status register (USBDMALIntSt - 0xE009 0090)

Bit 0, End_of_Transfer_Interrupt, will be set by hardware if any of the 32 bits in the End Of Transfer Interrupt Status register is 1. The same logic applies for Bit 1 and 2 of the DMA Interrupt Status register. The hardware checks the 32 bits of New DD Request Interrupt Status register to set/clear the bit 1 of DMA Interrupt Status register and similarly the 32 bits of System Error Interrupt Status register to set/clear the bit 2 of DMA Interrupt Status register. The USBDMALIntSt is a read only register.

Table 214: USB DMA Interrupt Status register (USBDMALIntSt - address 0xE009 0090) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---------------------------|-------|--|-------------|
| 0 | End_of_Transfer_Interrupt | | 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 | New_DD_Request_Interrupt | | 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 | System_Error_Interrupt | | 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 |

14.8.19 USB DMA Interrupt Enable register (USBDMALIntEn - 0xE009 0094)

Setting the bit in this register will cause external interrupt to happen for the bits set in the interrupt status register. The USBDMALIntEn is a read/write register.

Table 215: USB DMA Interrupt Enable register (USBDMAIntEn - address 0xE009 0094) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|------------------------------|-------|--|-------------|
| 0 | End_of_Transfer_Interrupt_En | 0 | End of Transfer Interrupt enable bit. | 0 |
| | | | The End of Transfer Interrupt is disabled. | |
| | | | The End of Transfer Interrupt is enabled. | |
| 1 | New_DD_Request_Interrupt_En | 0 | New DD Request Interrupt enable bit. | 0 |
| | | | The New DD Request Interrupt is disabled. | |
| | | | The New DD Request Interrupt is enabled. | |
| 2 | System_Error_Interrupt_En | 0 | System Error Interrupt enable bit. | 0 |
| | | | The System Error Interrupt is disabled. | |
| | | | 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 |

14.8.20 USB End of Transfer Interrupt Status register (USBEoTIntSt - 0xE009 00A0)

When the transfer completes for the descriptor, either normally (descriptor is retired) or because of an error, this interrupt occurs. The cause of the interrupt generation will be recorded in the DD_Status field of the descriptor. The USBEoTIntSt is a read only register.

Table 216: USB End of Transfer Interrupt Status register (USBEoTIntSt - address 0xE009 00A0s) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 31:0 | EPxx | 0 | Endpoint xx ($0 \leq xx \leq 31$) End of Transfer Interrupt request. | 0 |
| | | | There is no End of Transfer interrupt request for endpoint xx. | |
| | | | There is an End of Transfer Interrupt request for endpoint xx. | |

14.8.21 USB End of Transfer Interrupt Clear register (USBEoTIntClr - 0xE009 00A4)

Writing 1 into the register will clear the corresponding interrupt from the End of Transfer Interrupt Status register. Writing 0 will not have any effect. The USBEoTIntClr is a write only register.

Table 217: USB End of Transfer Interrupt Clear register (USBEoTIntClr - address 0xE009 00A4) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | 0 | Clear endpoint xx ($0 \leq xx \leq 31$) End of Transfer Interrupt request. | 0 |
| | | | Ne effect. | |
| | | | Clear the EPxx End of Transfer Interrupt request in the USBEoTIntSt register. | |

14.8.22 USB End of Transfer Interrupt Set register (USBEoTIntSet - 0xE009 00A8)

Writing 1 into the register will set the corresponding interrupt from the End of Transfer Interrupt Status register. Writing 0 will not have any effect. The USBEoTIntSet is a write only register.

Table 218: USB End of Transfer Interrupt Set register (USBEoTIntSet - address 0xE009 00A8) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Set endpoint xx ($0 \leq xx \leq 31$) End of Transfer Interrupt request. | 0 |
| | | 0 | Ne effect. | |
| | | 1 | Set the EPxx End of Transfer Interrupt request in the USBEoTIntSt register. | |

14.8.23 USB New DD Request Interrupt Status register (USBNDDRIntSt - 0xE009 00AC)

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. The USBNDDRIntSt is a read only register.

Table 219: USB New DD Request Interrupt Status register (USBNDDRIntSt - address 0xE009 00AC) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Endpoint xx ($0 \leq xx \leq 31$) 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. | |

14.8.24 USB New DD Request Interrupt Clear register (USBNDDRIntClr - 0xE009 00B0)

Writing 1 into the register will clear the corresponding interrupt from the New DD Request Interrupt Status register. Writing 0 will not have any effect. The USBNDDRIntClr is a write only register.

Table 220: USB New DD Request Interrupt Clear register (USBNDDRIntClr - address 0xE009 00B0) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Clear endpoint xx ($0 \leq xx \leq 31$) new DD interrupt request. | 0 |
| | | 0 | Ne effect. | |
| | | 1 | Clear the EPxx new DD interrupt request in the USBNDDRIntSt register. | |

14.8.25 USB New DD Request Interrupt Set register (USBNDDRIntSet - 0xE009 00B4)

Writing 1 into the register will set the corresponding interrupt from the New DD Request Interrupt Status register. Writing 0 will not have any effect. The USBNDDRIntSet is a write only register.

Table 221: USB New DD Request Interrupt Set register (USBNDDRIntSet - address 0xE009 00B4) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Set endpoint xx ($0 \leq xx \leq 31$) new DD interrupt request. | 0 |
| | | 0 | Ne effect. | |
| | | 1 | Set the EPxx new DD interrupt request in the USBNDDRIntSt register. | |

14.8.26 USB System Error Interrupt Status register (USBSysErrIntSt - 0xE009 00B8)

If a system error (AHB bus error) occurs when transferring the data or when fetching or updating the DD this interrupt bit is set. The USBSysErrIntSt is a read only register.

Table 222: USB System Error Interrupt Status register (USBSysErrIntSt - address 0xE009 00B8) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Endpoint xx ($0 \leq xx \leq 31$) 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. | |

14.8.27 USB System Error Interrupt Clear register (USBSysErrIntClr - 0xE009 00BC)

Writing 1 into the register will clear the corresponding interrupt from the System Error Interrupt Status register. Writing 0 will not have any effect. The USBSysErrIntClr is a write only register.

Table 223: USB System Error Interrupt Clear register (USBSysErrIntClr - address 0xE009 00BC) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Clear endpoint xx ($0 \leq xx \leq 31$) System Error Interrupt request. | 0 |
| | | 0 | Ne effect. | |
| | | 1 | Clear the EPxx System Error Interrupt request in the USBSysErrIntSt register. | |

14.8.28 USB System Error Interrupt Set register (USBSysErrIntSet - 0xE009 00C0)

Writing 1 into the register will set the corresponding interrupt from the System Error Interrupt Status register. Writing 0 will not have any effect. The USBSysErrIntSet is a write only register.

Table 224: USB System Error Interrupt Set register (USBSysErrIntSet - address 0xE009 00C0) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 31:0 | EPxx | | Set endpoint xx (0 ≤ xx ≤ 31) System Error Interrupt request. | 0 |
| | | 0 | No effect. | |
| | | 1 | Set the EPxx System Error Interrupt request in the USBSysErrIntSt register. | |

14.9 Protocol engine command description

The protocol engine operates based on the commands issued from the CPU.

These commands have to be written into the Command Code register ([Section 14.8.9](#)). The read data when present will be available in the Command Data register ([Section 14.8.10](#)) after the successful execution of the command. [Table 225](#) lists all protocol engine commands.

Here is an example of the Read Current Frame Number command (reading 2 bytes):

```
USBDevIntClr = 0x30;           // Clear cmd_code_empty & cmd_data_full int. bits
USBCmdCode = 0x00F50500;
while (!(USBDevIntSt & 0x10)); // Wait cmd_code_empty
USBDevIntClr = 0x10;           // Clear cmd_code_empty interrupt bit
USBCmdCode = 0x00F50200;
while (!(USBDevIntSt & 0x20)); // Wait for cmd_data_full
CurFrameNum = USBCmdData;      // Read Frame number LSB byte
USBDevIntClr = 0x10;           // Clear cmd_code_empty interrupt bit
USBCmdCode = 0x00F50200;
while (!(USBDevIntSt & 0x20)); // Wait for cmd_data_full
Temp = USBCmdData;            // Read Frame number MSB byte
CurFrameNum = CurFrameNum | (Temp << 8);
```

Table 225: Protocol engine command code table

| Command name | Recipient | Command | Data phase (coding) |
|---------------------------|-----------|-------------|---------------------------------|
| Device commands | | | |
| Set Address | Device | 00 D0 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| Configure Device | Device | 00 D8 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| Set Mode | Device | 00 F3 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| Read Current Frame Number | Device | 00 F5 05 00 | Read 1 or 2 bytes - 00 F5 02 00 |
| Read Test Register | Device | 00 FD 05 00 | Read 2 bytes - 00 FD 02 00 |
| Set Device Status | Device | 00 FE 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| Get Device Status | Device | 00 FE 05 00 | Read 1 byte - 00 FE 02 00 |
| Get Error Code | Device | 00 FF 05 00 | Read 1 byte - 00 FF 02 00 |
| Read Error Status | Device | 00 FB 05 00 | Read 1 byte - 00 FB 02 00 |
| Endpoint Commands | | | |

Table 225: Protocol engine command code table

| Command name | Recipient | Command | Data phase (coding) |
|---------------------------------|-------------------|-------------|--|
| Select Endpoint | Endpoint 0 | 00 00 05 00 | Read 1 byte (optional) - 00 00 02 00 |
| | Endpoint 1 | 00 01 05 00 | Read 1 byte (optional) - 00 01 02 00 |
| | Endpoint 2 | 00 02 05 00 | Read 1 byte (optional) - 00 02 02 00 |
| | Endpoint xx | 00 xx 05 00 | Read 1 byte (optional) - 00 xx 02 00 xx - physical endpoint number |
| | Endpoint 31 | 00 1F 05 00 | Read 1 byte (optional) - 00 1F 02 00 |
| Select Endpoint/Clear Interrupt | Endpoint 0 | 00 40 05 00 | Read 1 byte - 00 40 02 00 |
| | Endpoint 1 | 00 41 05 00 | Read 1 byte - 00 41 02 00 |
| | Endpoint 2 | 00 42 05 00 | Read 1 byte - 00 42 02 00 |
| | Endpoint xx | 00 xx 05 00 | Read 1 byte - 00 xx 02 00 xx - (physical endpoint number + 0x40) |
| | Endpoint 31 | 00 5F 05 00 | Read 1 byte - 00 5F 02 00 |
| Set Endpoint Status | Endpoint 0 | 00 40 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| | Endpoint 1 | 00 41 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| | Endpoint 2 | 00 42 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| | Endpoint xx | 00 xx 05 00 | Write 1 byte - 00 <Byte> 01 00 xx - (physical endpoint number + 0x40) |
| | Endpoint 31 | 00 5F 05 00 | Write 1 byte - 00 <Byte> 01 00 |
| Clear Buffer | Selected Endpoint | 00 F2 05 00 | Read 1 byte (optional) - 00 F2 02 00 |
| Validate Buffer | Selected Endpoint | 00 FA 05 00 | None |

14.9.1 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 phase of the setup token. (Alternately, issuing the Set Address command twice will set the address in the device). At power on reset, the DEV_EN is set to 0. After bus reset, the address is reset to 0x00. The enable bit is set to 1. The device will respond on packets for function address 0x00, endpoint 0 (default endpoint).

Table 226: Device Set Address Register bit description

| Bit | Symbol | Description | Reset value |
|-----|----------|-------------------------------------|-------------|
| 6:0 | DEV_ADDR | Device address set by the software. | 0x00 |
| 7 | DEV_EN | Device Enable. | 0 |

14.9.2 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 227: Configure Device Register bit description

| Bit | Symbol | Description | Reset value |
|-----|-------------|---|-------------|
| 0 | CONF_DEVICE | Device is configured. This bit is set after the set configuration command is executed. Good link LED signal is asserted when configuration is done. | 0 |
| 7:1 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

14.9.3 Set Mode (Command: 0xF3, Data: write 1 byte)

Table 228: Set Mode Register bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------|---|--|-------------|
| 0 | AP_CLK | Always PLL Clock. | | 0 |
| | | 0 | usb_needclk is functional; 48 Mhz clock can be stopped when the device enters suspend state. | |
| | | 1 | usb_needclk always have the value 1. 48 Mhz clock cannot be stopped in case when the device enters suspend state. | |
| 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.

14.9.4 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.

14.9.5 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 (48 Mhz and hclk) are fine.

14.9.6 Set Device Status (Command: 0xFE, Data: write 1 byte)

The Set Device Status command sets bits in the Device Status Register.

Table 229: Set Device Status Register bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--------|-------|---|-------------|
| 0 | CON | 0 | The Connect bit indicates the current connect status of the device. It controls the SoftConnect_N output pin, used for SoftConnect. Reading the connect bit returns the current connect status. | 0 |
| | | | Writing a 0 will make SoftConnect_N inactive. | |
| | | | Writing a 1 will make SoftConnect_N active. | |
| 1 | CON_CH | 0 | Connect Change. | 0 |
| | | 1 | This bit is set when the device's pull-up resistor is disconnected because VBus disappeared. DEV_STAT interrupt is generated when this bit is 1. | |
| 2 | SUS | 0 | 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 into this register has never an effect. | 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 229: 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: <ul style="list-style-type: none">• The device goes into the suspended state.• The device is disconnected.• The device receives resume signalling on its upstream port.• The Suspend Change bit is reset after the register has been read. | 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: <ul style="list-style-type: none">• Device is unconfigured.• Will respond to address 0.• Control endpoint will be in the Stalled state.• All endpoints are enabled.• Data toggling is reset for all endpoints.• All buffers are cleared.• There is no change to the endpoint interrupt status.• DEV_STAT interrupt is generated. | 0 |
| 0 | | | This bit is cleared when read. | |
| 1 | | | This bit is set when the device receives a bus reset. | |
| 7:5 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

14.9.7 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 229](#).

14.9.8 Get Error Code (Command: 0xFF, Data: read 1 byte)

Different error conditions can arise inside the protocol engine. The Get Error Code command returns the error code which last occurred. The 4 least significant bits form the error code.

Table 230: 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 |

14.9.9 Read Error Status (Command: 0xFB, Data: read 1 byte)

This command reads the 8 bit Error register from the USB device. If any of these bits is set, there will be an interrupt to the CPU. The error bits are reset after reading the register.

Table 231: 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 |

14.9.10 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 in the buffer. The command code of 'select endpoint' is equal to the physical endpoint number. In the case of single buffer, B_2_FULL bit is not valid.

Table 232: Select Endpoint Register bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|---|-------------|
| 0 | F/E | 0 | The F/E bit gives the ORed result of B_1_FULL and B_2_FULL bits. | 0 |
| | | | For IN endpoint if the next write buffer is empty this bit is 0. | |
| | | 1 | For OUT endpoint if the next read buffers is full this bit is 1. | |
| 1 | ST | 0 | Stalled endpoint indicator. | 0 |
| | | | The selected endpoint is not stalled. | |
| | | 1 | The selected endpoint is stalled. | |
| 2 | STP | 0 | 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 |
| | | | 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 | 0 | Packet over-written bit. | 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 | 0 | 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 to an empty IN buffer, the device returns NAK. | 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 | 0 | The buffer 1 status. | 0 |
| | | | Buffer 1 is empty. | |
| | | 1 | Buffer 1 is full. | |
| 6 | B_2_FULL | 0 | The buffer 2 status. | 0 |
| | | | Buffer 2 is empty. | |
| | | 1 | Buffer 2 is full. | |
| 7 | - | - | Reserved, user software should not write ones to reserved bits. NA The value read from a reserved bit is not defined. | |

14.9.11 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 associated interrupt in the USB clock domain only.
- In case of a control out endpoint, they clear the setup and over-written bits
- Reading one byte is obligatory.

14.9.12 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 value. Not all bits can be set for all types of endpoints.

Table 233: 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 un-stall it. 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 on the interrupt status of the endpoint. Even when unstalled, setting the stalled bit to 0 initializes the endpoint. When an endpoint is stalled by the Set Endpoint Status command it is also re-initialized. The command to set the conditional stall bit will be ignored if the 'Setup Packet' bit is set (the EP will not be reset and no status bits will change). | 0 |
| | | 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 |
| 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 Setup Packet bit is set. It is defined only for control OUT endpoints. | |

14.9.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional))

When a packet sent by the host has been received successfully, an internal Endpoint Buffer Full flag is set. All subsequent packets will be refused by returning a NAK. When the CPU has read the data, it should free the buffer by the Clear Buffer command. When the buffer is cleared, new packets will be accepted.

When bit 0 of the optional data byte is 1, the previously received packet was over-written by a SETUP packet. The Packet overwritten bit is used only in control transfers. According to the USB specification, SETUP packet should be accepted irrespective of the buffer status. The software should always check the status of the PO bit after reading the SETUP data. If it is set then it should discard the previously read data, clear the PO bit by issuing a Select Endpoint/Clear Interrupt command, read the new SETUP data and again check the status of the PO bit.

Table 234: 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 NA value read from a reserved bit is not defined. | |

14.9.14 Validate Buffer (Command: 0xFA, Data: none)

When the CPU has written data into an IN buffer, it should set the buffer full flag by the Validate Buffer command. This indicates that the data in the buffer is valid and can be sent to the host when the next IN token is received.

A control IN buffer cannot be validated when the Packet Over-written bit of its corresponding OUT buffer is set or when the Set up packet is pending in the buffer. For the control endpoint the validated buffer will be invalidated when a Setup packet is received.

14.10 DMA descriptor

A DMA transfer can be characterized by a structure describing these parameters. This structure is called the DMA Descriptor (DD).

The DMA descriptors are placed in the USB RAM. These descriptors can be located anywhere in the USB RAM in the wordaligned boundaries. USB RAM is part of the system memory which is used for the USB purposes. It is located at address 0x7FD0 0000 and is 8192 bytes (8 kB) in size.

DD for non-isochronous endpoints are four-word long and isochronous endpoints are five-word long.

Total USB RAM required for DD is:

$$\text{Total_USBDRAM} = (\text{No.of_non-ISOendpoints} \times 4 + \text{No.of_ISOendpoints} \times 5)$$

There are certain parameters associated with a DMA transfer. These are:

- The start address of the DMA buffer in the USB RAM.
- The length of the DMA Buffer in the USB RAM.
- The start address of the next DMA buffer.
- Control information.
- DMA count information (Number of bytes transferred).
- DMA status information.

[Table 235](#) lists the DMA descriptor fields.

Table 235: 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 in bytes. |
| 2 | R/W | R/W | 31:0 | DMA_buffer_start_addr. |
| 3 | R/W | R/I | 0 | DD_retired (To be initialized to 0). |
| | W | R/I | 4:1 | DD_status (To be initialized to 0): 0000 - Not serviced. 0001 - Being serviced. 0010 - Normal completion. 0011 - Data under run (short packet). 1000 - Data over run. 1001 - System error. |
| | 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. |

[Legend: R - Read; W - Write; I - Initialize](#)

[1] Write only in ATLE mode

14.10.1 Next_DD_pointer

Pointer to the memory location from where the next DMA descriptor has to be fetched.

14.10.2 DMA_mode

Defines in which mode the DMA has to operate. Two modes have been defined, Normal and ATLE. In the normal mode the DMA engine will not split a packet into two different DMA buffers. In the ATLE mode splitting of the packet into two buffers can happen. This is because two transfers can be concatenated in the packet to improve the bandwidth. See [Section 14.13 “Concatenated transfer \(ATLE\) mode operation” on page 236](#) for more details.

14.10.3 Next_DD_valid

This bit indicates whether the software has prepared the next DMA descriptor. If it is valid, the DMA engine once finished with the current descriptor will load the new descriptor.

14.10.4 Isochronous_endpoint

The descriptor belongs to an isochronous endpoint. Hence, 5 words have to be read.

14.10.5 Max_packet_size

The maximum packet size of the endpoint. This parameter has to be used while transferring the data for IN endpoints from the memory. It is used for OUT endpoints to detect the short packet. This is applicable to non-isochronous endpoints only. The max_packet_size field should be the same as the value set in the MaxPacketSize register for the endpoint.

14.10.6 DMA_buffer_length

This indicates the depth of the DMA buffer allocated for transferring the data. The DMA engine will stop using this descriptor when this limit is reached and will look for the next descriptor. This will be set by the software in the normal mode operation for both IN and OUT endpoints. In the ATLE mode operation the buffer_length is set by software for IN endpoints. For OUT endpoints this is set by the hardware from the extracted length of the data stream. In case of the Isochronous endpoints the DMA_buffer_length is specified in terms of number of packets.

14.10.7 DMA_buffer_start_addr

The address from where the data has to be picked up or to be stored. This field is updated packet-wise by DMA engine.

14.10.8 DD_retired

This bit is set when the DMA engine finishes the current descriptor. This will happen when the end of the buffer is reached or a short packet is transferred (no isochronous endpoints) or an error condition is detected.

14.10.9 DD_status

The status of the DMA transfer is encoded in this field. The following status are defined:

- **Not serviced** - No packet has been transferred yet. DD is in the initial position itself.
- **Being serviced** - This status indicates that at least one packet is transferred.

- **Normal completion** - The DD is retired because the end of the buffer is reached and there were no errors. DD_retired bit also is set.
- **Data under run** - Before reaching the end of the buffer, transfer is terminated because a short packet is received. DD_retired bit also is set.
- **Data over run** - End of the DMA buffer is reached in the middle of a packet transfer. This is an error situation. DD_retired bit will be set. The DMA count will show the value of DMA buffer length. The packet has to be re-transmitted from the FIFO. DMA_ENABLE bit is reset.
- **System error** - Transfer is terminated because of an error in the system bus. DD_retired bit is not set in this case. DMA_ENABLE bit is reset. Since system error can happen while updating the DD, the DD fields in the USB RAM may not be very reliable.

14.10.10 Packet_valid

This bit indicates whether the last packet transferred to the memory is received with errors or not. This bit will be set if the packet is valid, i.e., it was received without errors. Since non-isochronous endpoint will not generate DMA request for packet with errors, this field will not make much sense to them as it will be set for all packets transferred. But for isochronous endpoints this information is useful. See [Section 14.14 “Isochronous Endpoint Operation” on page 240](#) for isochronous endpoint operation.

14.10.11 LS_byte_extracted

Applicable only in the ATLE mode. This bit set indicates that the Least Significant Byte (LSB) of the transfer length has been already extracted. The extracted size will be reflected in the ‘dma_buffer_length’ field in the bits 23:16.

14.10.12 MS_byte_extracted

Applicable only in the ATLE mode. This bit set indicates that the Most Significant Byte (MSB) of the transfer size has been already extracted. The size extracted will be reflected in the ‘dma_buffer_length’ field at 31:24. Extraction stops when ‘LS_Byte_extracted’ and ‘MS_byte_extracted’ fields are set.

14.10.13 Present_DMA_count

The number of bytes transferred by the DMA engine at any point of time. This is updated packet-wise by the DMA engine when it updates the descriptor. In case of the Isochronous endpoints the Present_DMA_count is specified in terms of number of packets transferred.

14.10.14 Message_length_position

This applies only in the ATLE mode. This field gives the offset of the message length position embedded in the packet. This is applicable only for OUT endpoints. Offset 0 indicates that the message length starts from the first byte of the packet onwards.

14.10.15 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 55](#). This is applicable to isochronous endpoints only.

14.11 DMA operation

14.11.1 Triggering the DMA engine

An endpoint will raise a DMA request when the slave mode transfer is disabled by setting the corresponding bit in Endpoint Interrupt Enable register to 0 ([Section 14.7.8](#)).

The DMA transfer for an OUT endpoint is triggered when it receives a packet without any errors (i.e., the buffer is full) and the DMA_ENABLE ([Section 14.8.15 "USB EP DMA Status register \(USBEpDMASt - 0xE009 0084\)"](#)) bit is set for this endpoint.

Transfer for an IN endpoint is triggered when the host requests for a packet of data and the DMA_ENABLE bit is set for this endpoint.

In DMA mode, the bits corresponding to Interrupt on NAK for Bulk OUT and Interrupt OUT endpoints (bit INAK_BO and INAK_IO) in Set Mode register ([Section 14.9.3 "Set Mode \(Command: 0xF3, Data: write 1 byte\)"](#)) should be reset to 0.

14.11.2 Arbitration between endpoints

If more than one endpoint is requests for data transfer at the same time the endpoint with lower physical endpoint number value gets the priority.

14.12 Non Isochronous Endpoints - Normal Mode operation

14.12.1 Setting up DMA transfer

The software prepares the DDs for the physical endpoints that need DMA transfer. These DDs are present in the USB RAM. Also, the start address of the first DD is programmed into the DDP location for the corresponding endpoint. The software will then set the DMA_ENABLE bit for this endpoint in the EP DMA Status register ([Section 14.8.15](#)). The 'dma_mode' bits in the descriptor has to be set to '00' for normal mode operation. It should also initialize all the bits in the DD as given in the table.

14.12.2 Finding DMA Descriptor

When there is a trigger for a DMA transfer for an endpoint, DMA engine will first determine whether a new descriptor has to be fetched or not. A new descriptor need not have to be fetched if the last transfer was also made for the same endpoint and the DD is not yet in the 'retired' state. A flag called 'DMA_PROCEED' is used to identify this (see [Section 14.12.4 "Optimizing Descriptor Fetch" on page 235](#)).

If a new descriptor has to be read, the DMA engine will calculate the location of the DDP for this endpoint and will fetch the start address of DD from this location. A DD start address at location zero is considered invalid. In this case a 'new_dd_request' interrupt is raised. All other word boundaries are valid.

At any point of time if the DD is to be fetched, the status of DD (word 3) is read first and the status of the 'DD_retired' bit is checked. If this is not set, DDP points to a valid DD. If the 'DD_retired' bit is set, the DMA engine will read the 'control' field (word 1) of the DD.

If the bit 'next_DD_valid' bit is set, the DMA engine will fetch the 'next_dd_pointer' field (word 0) of the DD and load it to the DDP. The new DDP is written to the UDCA area.

The full DMA descriptor (4 words) will in turn be fetched from this address pointed by DDP. The DD will give the details of the transfer to be done. The DMA engine will load its hardware resources with the information fetched from the DD (start address, DMA count etc.).

If the 'next_dd_valid' is not set and the DD_retired bit is set the DMA engine will raise the 'NEW_DD_REQUEST' interrupt for this endpoint. It also disables the DMA_ENABLE bit.

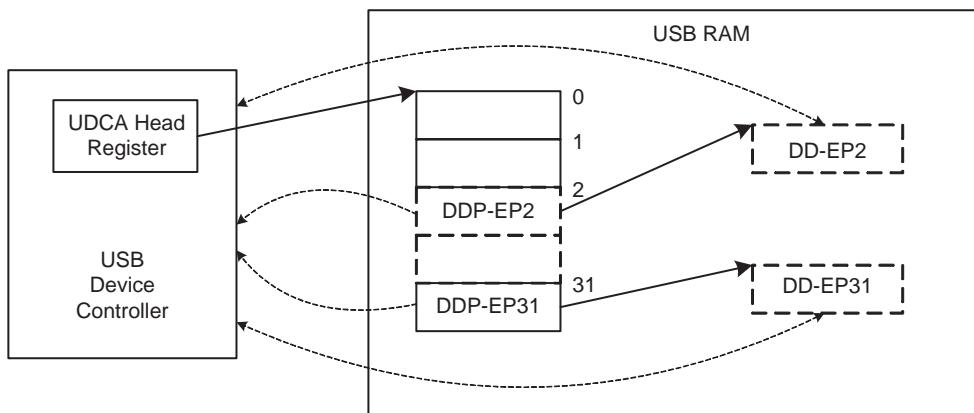


Fig 53. Finding the DMA descriptor

14.12.3 Transferring the Data

In case of OUT endpoints, the current packet will be read from the EP_RAM by the DMA Engine and will get transferred to the USB RAM memory locations starting from the address pointed by 'dma_buffer_start_addr'. In case of IN endpoints, the data will be fetched from the USB RAM and will be written to the EP_RAM. The 'dma_buffer_start_addr' and 'present_dma_count' will get updated while the transfer progresses.

14.12.4 Optimizing Descriptor Fetch

A DMA transfer normally involves multiple packet transfers. If a DD once fetched is equipped to do multiple transfers, the hardware will not fetch DD for all the succeeding packets. It will do the fetching only if the previous packet transferred on this channel does not belong to this endpoint. This is on the assumption that the current contents of the hardware resource and that of the descriptor to be fetched will be the same. In such a case DMA engine can proceed without fetching the new descriptor if it has not transferred enough data specified in the 'dma_buffer_length' field of the descriptor. To keep this information the hardware will have a flag set called 'DMA_PROCEED'.

This flag will be reset after the required number of bytes specified in the 'dma_buffer_length' field is transferred. It is also reset when the software writes into the EP DMA Disable register. This will give the software control over the reading of DD by the hardware. Hardware will be forced to read the DD for the next packet. Writing data 0x0 into the EP DMA Disable register will cause only resetting of the DMA_PROCEED flag without disabling DMA for any endpoint.

14.12.5 Ending the packet transfer

The DMA engine will write back the DD with an updated status to the same memory location from where it was read. The ‘dma_buffer_start_addr’, ‘present_dma_count’ and the status bits field in the DD gets updated. Only words 2 and 3 are updated by hardware in this mode.

A DD can have the following types of completion:

Normal completion - If the current packet is fully transferred and the ‘dma_count’ field equals the ‘dma_buffer_length’ defined in the descriptor, the DD has a normal completion. The DD will be written back to memory with ‘DD_retired’ bit set.

END_OF_TRANSFER interrupt is raised for this endpoint. DD_Status bits are updated for ‘normal_completion’ code.

Transfer end completion - If the current packet is fully transferred and its size is less than the ‘max_packet_size’ defined in the descriptor, and the end of the buffer is still not reached the transfer end completion occurs. The DD will be written back to the memory with ‘DD_retired’ bit set and DD_Status bits showing ‘data under run’ completion code. Also, the ‘END_OF_TRANSFER’ interrupt for this endpoint is raised.

Error completion - If the current packet is partially transferred i.e. end of the DMA buffer is reached in the middle of the packet transfer, an error situation occurs. The DD is written back with DD_status ‘data over run’ and ‘DD_retired’ bit is set. The DMA engine will raise the end of transfer interrupt and resets the corresponding bit for this endpoint in the ‘DMA_ENABLE’ register. This packet will be retransmitted to the memory fully when DMA_ENABLE bit is set again.

14.12.6 No_Packet DD

For IN transfers, it can happen that for a request, the system does not have any data to send for a long time. The system can suppress this request by programming a no_packet DD. This is done by setting the ‘Maxpacketsize’ and ‘dma_buffer_length’ in the DD control field to 0. No packets will be sent to the host in response to the no_packet DD.

14.13 Concatenated transfer (ATLE) mode operation

Some host drivers like ‘NDIS’ (Network Driver Interface Standard) are capable of concatenating small transfers (delta transfers) to form a single large transfer. The device hardware should be able to break up this single transfer back into delta transfers and transfer them to different DMA buffers. This is achieved in the ATLE mode operation. This is applicable only for Bulk endpoints.

In ATLE mode, the Host driver can concatenate various transfer lengths, which correspond to different DMA descriptors on Device side. And these transfers have to be done on USB without breaking the packet. This is the primary difference between the Normal Mode of DMA operation and ATLE mode, wherein one DMA transfer length ends with either a full USB packet or a short packet and next DMA transfer length starts with a new USB packet in Normal mode, but these two transfers may be concatenated in the last USB packet of the first DMA transfer in ATLE mode.

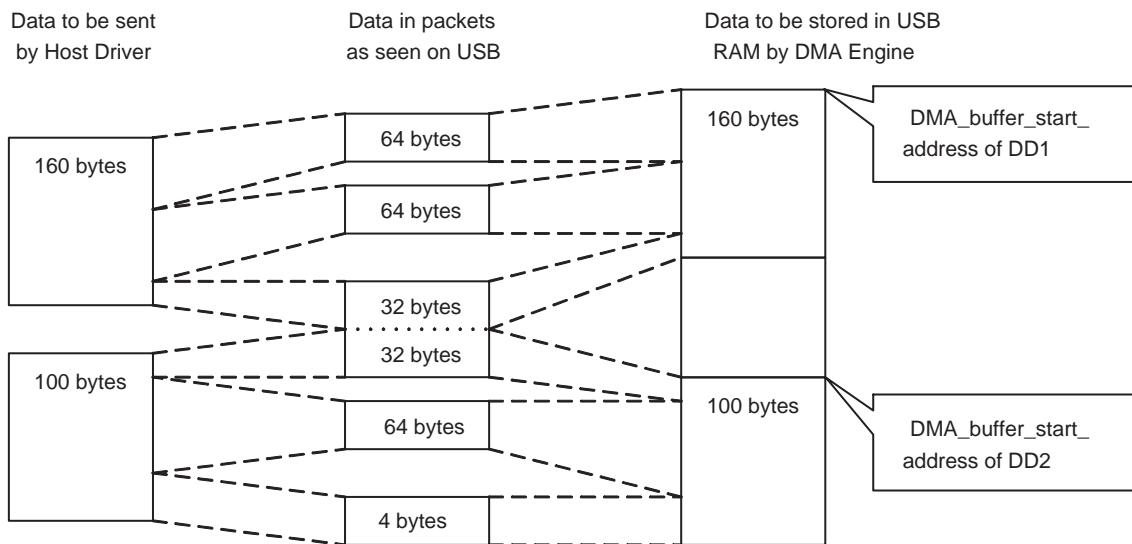


Fig 54. Data transfer in ATLE mode

[Figure 54](#) shows a typical OUT transfer, where the host concatenates two DMA transfer lengths of 160 bytes and 100 bytes respectively. As seen on USB, there would be four packets of 64 bytes ($MPS = 64$) and a short packet of 4 bytes in ATLE mode unlike Normal mode with five packets of 64, 64, 32, 64, 36 bytes in the given order.

It is now responsibility of the DMA engine to separate these two transfers and put them in proper memory locations as pointed by the "DMA_buffer_start_address" field of DMA Descriptor 1 (DD1) and DMA Descriptor 2 (DD2).

There are two things in OUT transfer of ATLE mode, which differentiate it from the OUT transfer in Normal mode of DMA operation. The first one is that the Device software does not know the "DMA_buffer_length" of the incoming transfer and hence this field in DD is programmed to 0. But by the NDIS protocol, device driver does know at which location in the incoming data transfer, will the transfer length be stored. This value is programmed in the field "Message_length_position" of the DD.

It is responsibility of the hardware to read the two byte wide "DMA_buffer_length" at the offset (from start of transfer) specified by "Message_length_position", from incoming data and write it in "DMA_buffer_length" field of the DD. Once this information is extracted from the incoming data and updated in the DD, the transfer continues as in Normal mode of operation.

It may happen that the message length position points to the last byte in the USB packet, which means that out of two bytes of buffer length, first (LS) byte is available in the current packet, and the second (MS) byte would follow in the next packet. To deal with such situations, the flags "LS_byte_extracted" and "MS_byte_extracted" are used by hardware. When the hardware reads the LS byte (which is the last byte of USB packet), it writes the contents of LS byte in position (23:16) of "DMA_buffer_length" field, sets the flag "LS_byte_extracted" to 1 and updates the DD in System memory (since the packet transfer is over).

On reception of the next packet, looking at "LS_byte_extracted" field 1 and "MS_byte_extracted" field 0, hardware knows that it has to read the first incoming byte as MS byte of buffer length, update the position (31:24) of "DMA_buffer_length" with the read contents and set the flag "MS_byte_extracted". After the extraction of MS byte of DMA buffer length, the transfer continues as in Normal mode of operation.

The second thing, which differentiates the ATLE mode OUT transfer from Normal mode OUT transfer, is the behavior in case when DD is retired in between a USB packet transfer.

As can be seen in the figure earlier, the first 32 bytes of the 3rd packet correspond to DD1 and the remaining 32 bytes correspond to DD2. In such a situation, on reception of first 32 bytes, the first DD (i.e. DD1) is retired and updated in the system memory, the new DD (pointed by "next_DD_pointer") is fetched and the remaining 32 bytes are transferred to the location in system memory pointed by "DMA_buffer_start_address" of new DD (i.e. DD2).

It should be noted that in ATLE mode, the software will always program the "LS_byte_extracted" and "MS_byte_extracted" fields to 0 while preparing a DD, and hence on fetching the DD2 in above situation, the Buffer Length Extraction process will start again as described earlier.

In case if the first DD is retired in between the packet transfer and the next DD is not programmed, i.e. "next_DD_valid" field in DD1 is 0, then the first DD is retired with the status "data over run" (DD_status = 1000), which has to be treated as an err or condition and the DMA channel for that particular endpoint is disabled by the hardware. Otherwise the first DD is retired with status "normal completion" (DD_status = 0010).

Please note that in this mode the last buffer length to be transferred would always end with a short packet or empty packet indicating that no more concatenated data is coming on the way. If the concatenated transfer lengths are such that the last transfer ends on a packet boundary, the (NDIS) host will send an empty packet to mark the End Of Transfer.

IN Transfer in ATLE mode

The operation in IN transfers is relatively simple than the OUT transfer in ATLE mode since device software knows the buffer length to be transferred and it is programmed in "DMA_buffer_length" field while preparing the DD, thus avoiding any transfer length extraction mechanism.

The only difference for IN transfers between ATLE mode and Normal mode of DMA operation is that the DDs can get retired in the middle of the USB packet transfer. In such a case, the hardware will update the first DD in system memory, fetch the new DD pointed by "next_DD_pointer" field of the first DD and fetch the remaining bytes from system memory pointed by "DMA_buffer_start_address" of second DD to complete the packet before sending it on USB.

In the above situation, if the next DD is not programmed, i.e. "next_DD_valid" field in DD is 0, and the buffer length for current DD has completed before the packet boundary, then the available bytes from current DD are sent as a short packet on USB, which marks the End Of Transfer for the Host.

In cases, where the intended buffer lengths are already transferred and the last buffer length has completed on the USB packet boundary, it is responsibility of Device software to program the next DD with "DMA_buffer_length" field 0, after which an empty packet is sent on USB by the hardware to mark the End Of Transfer for the Host.

14.13.1 Setting up the DMA transfer

There is an additional field in the descriptor called 'message_length_position' which has to be set for the OUT endpoints. This indicates the start location of the message length in the incoming data packet. Also the software will set the 'dma_buffer_length' field to '0' for OUT endpoints as this field has to be updated by hardware.

For IN endpoints, descriptors are to be set in the same way as the normal mode operation.

Since a single packet can have two transfers which has to be transferred or collected from different DMA buffers, the software should keep two buffers ready always, except for the last delta transfer which ends with a short packet.

14.13.2 Finding the DMA Descriptor

DMA descriptors are found in the same way as the normal mode operation.

14.13.3 Transferring the Data

For OUT end points if the 'LS_byte_extracted' or 'MS_byte_extracted' bit in the status field is not set, the hardware will extract the transfer length from the data stream. 'dma_buffer_length' field is derived from this information which is 2 bytes long. Once the extraction is complete both the 'LS_byte_extracted' and 'MS_byte_extracted' bits will be set.

For IN endpoints transfer proceeds like the normal mode and continues till the number of bytes transferred equals the 'dma_buffer_length'.

14.13.4 Ending the packet transfer

DMA engine proceeds with the transfer till the number of bytes specified in the field 'dma_buffer_length' gets transferred to or from the USB RAM. END_OF_TRANSFER interrupt will be generated. If this happens in the middle of the packet, the linked DD will get loaded and the remaining part of the packet gets transferred to or from the address pointed by the new DD.

For an OUT endpoint if the linked DD is not valid and the packet is partially transferred to memory, the DD ends with data_over_run status set and DMA will be disabled for this endpoint. Otherwise DD_status will be updated with 'normal completion'.

For an IN endpoint if the linked DD is not valid and the packet is partially transferred to USB, DD ends with 'normal completion' and the packet will be sent as a short packet (since this situation is the end of transfer). Also, when the linked DD is valid and buffer length is 0, a short packet will be sent.

14.14 Isochronous Endpoint Operation

In case of isochronous endpoint operation the packet size can vary on each and every packet. There will be one packet per isochronous endpoint at every frame.

14.14.1 Setting up of DMA transfer

For Isochronous DMA descriptor the DMA length is set in terms of the number of frames the transfer is to be made rather than the number of bytes. The DMA count is also updated in terms of the number of frames.

14.14.2 Finding the DMA Descriptor

Finding the descriptor is done in the same way as that for a non isochronous endpoint.

DMA descriptor has a bit field in the word 1 (isochronous_endpoint) to indicate that the descriptor belongs to an isochronous endpoint. Also, isochronous DD has a fifth word showing where the packet length for the frame has to be put (for OUT endpoint) or from where it has to be read.

DMA request will be placed for DMA enabled isochronous endpoints on every frame interrupt. For a DMA request the DMA engine will fetch the descriptor and if it identifies that the descriptor belongs to an Isochronous endpoint, it will fetch the fifth word of the DD which will give the location from where the packet length has to be placed or fetched.

14.14.3 Transferring the Data

The data is transferred to or from the memory location pointed by the `dma_buffer_start_addr`. After the end of the packet transfer the `dma_count` value is incremented by 1.

For an OUT transfer a word is formed by combining the frame number and the packet length such that the packet length appears at the least significant 2 bytes (15 to 0). Bit 16 shows whether the packet is valid or not (set when packet is valid i.e. it was received without any errors). The frame number appears in the most significant 2 bytes (bit 31 to 17). The frame number is available from the USB device. This word is then transferred to the address location pointed by the variable `Isochronous_packet_size_memory_address`. The `Isochronous_packet_size_memory_address` is incremented by 4 after receiving or transmitting an Isochronous data packet. The `Isochronous_packet_size` memory buffer should be big enough to hold information of all packets sent by the host.

For an IN endpoint only the bits from 15 to 0 are applicable. An Isochronous data packet of size specified by this field is transferred from the USB device to the Host in each frame. If the size programmed in this location is zero an empty packet will be sent by the USB device.

The Isochronous endpoint works only in the normal mode DMA operation.

An Isochronous endpoint can have only 'normal completion' since there is no short packet on Isochronous endpoint and the transfer continues infinitely till a system error occurs. Also, there is no `data_over_run` detection.

14.14.4 Isochronous OUT Endpoint Operation Example

For example assume that an isochronous endpoint is programmed for the transfer of 10 frames. After transferring four frames with packet size 10,15, 8 and 20 bytes; the descriptors and memory map looks as shown in [Figure 55](#). Assuming that the transfer starts when the internal frame number was 21.

The _total_number_of_bytes_transferred = 0x0A + 0x0F + 0x08 + 0x14 = 0x35.

The sixteenth bit for all the words in the packet length memory will be set to 1.

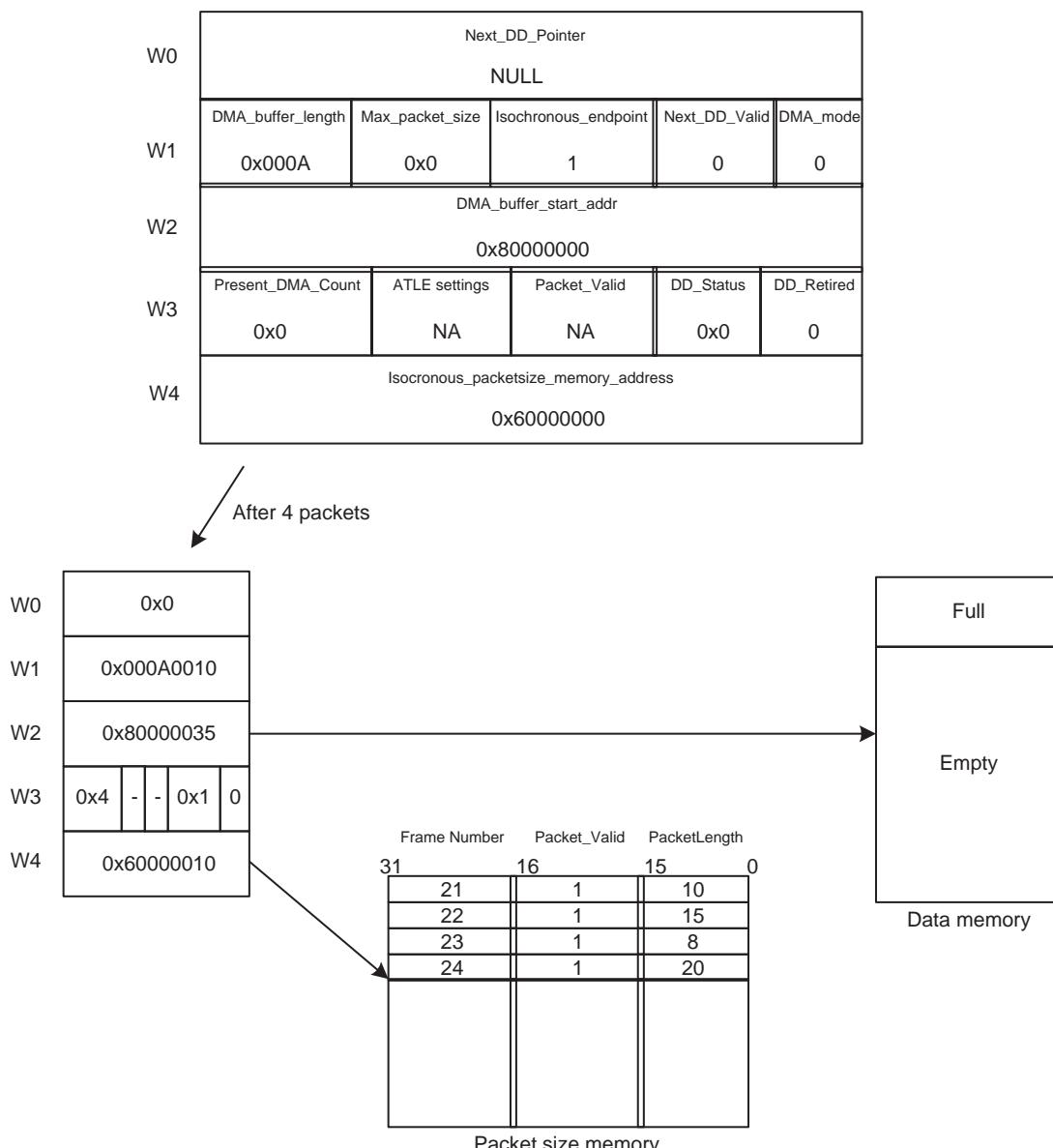


Fig 55. Isochronous OUT Endpoint operation example

Timer/Counter0 and Timer/Counter1 are functionally identical except for the peripheral base address.

15.1 Features

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Counter or Timer operation
- Up to four 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- Four 32-bit match registers that allow:
 - Continuous operation with optional interrupt generation on match.
 - Stop timer on match with optional interrupt generation.
 - Reset timer on match with optional interrupt generation.
- Up to four external outputs corresponding to match registers, with the following capabilities:
 - Set low on match.
 - Set high on match.
 - Toggle on match.
 - Do nothing on match.

15.2 Applications

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

15.3 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.

15.4 Pin description

[Table 236](#) gives a brief summary of each of the Timer/Counter related pins.

Table 236: Timer/Counter pin description

| Pin | Type | Description |
|------------------------|-------------|---|
| CAP0.3..0 CAP1.3..0 | Input | <p>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. If for example pins 30 (P0.6) and 46 (P0.16) are selected for CAP0.2, only pin 30 will be used by TIMER0 to perform CAP0.2 function.</p> <p>Here is the list of all CAPTURE signals, together with pins on where they can be selected:</p> <ul style="list-style-type: none"> • CAP0.0 (3 pins): P0.2, P0.22 and P0.30 • CAP0.1 (2 pins): P0.4 and P0.27 • CAP0.2 (3 pin): P0.6, P0.16 and P0.28 • CAP0.3 (1 pin): P0.29 • CAP1.0 (1 pin): P0.10 • CAP1.1 (1 pin): P0.11 • CAP1.2 (2 pins): P0.17 and P0.19 • CAP1.3 (2 pins): P0.18 and P0.21 <p>Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 15.5.3 "Count Control Register (CTCR, TIMER0: T0CTCR - 0xE000 4070 and TIMER1: T1TCR - 0xE000 8070)" on page 246.</p> |
| MAT0.3..0 MAT1.3..0 | Output | <p>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. It is also possible for example, to have 2 pins selected at the same time so that they provide MAT1.3 function in parallel.</p> <p>Here is the list of all MATCH signals, together with pins on where they can be selected:</p> <ul style="list-style-type: none"> • MAT0.0 (2 pins): P0.3 and P0.22 • MAT0.1 (2 pins): P0.5 and P0.27 • MAT0.2 (2 pin): P0.16 and P0.28 • MAT0.3 (1 pin): P0.29 • MAT1.0 (1 pin): P0.12 • MAT1.1 (1 pin): P0.13 • MAT1.2 (2 pins): P0.17 and P0.19 • MAT1.3 (2 pins): P0.18 and P0.20 |

15.5 Register description

Each Timer/Counter contains the registers shown in [Table 237](#). More detailed descriptions follow.

Table 237: TIMER/COUNTER0 and TIMER/COUNTER1 register map

| Generic Name | Description | Access | Reset value ^[1] | TIMER/COUNTER0 Address & Name | TIMER/COUNTER1 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 | 0xE000 4000 T0IR | 0xE000 8000 T1IR |
| 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 | 0xE000 4004 T0TCR | 0xE000 8004 T1TCR |
| 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 | 0xE000 4008 T0TC | 0xE000 8008 T1TC |
| 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 | 0xE000 400C T0PR | 0xE000 800C T1PR |
| 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 | 0xE000 4010 T0PC | 0xE000 8010 T1PC |
| 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 | 0xE0004014 T0MCR | 0xE000 8014 T1MCR |
| 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 | 0xE000 4018 T0MR0 | 0xE000 8018 T1MR0 |
| MR1 | Match Register 1. See MR0 description. | R/W | 0 | 0xE000 401C T0MR1 | 0xE000 801C T1MR1 |
| MR2 | Match Register 2. See MR0 description. | R/W | 0 | 0xE000 4020 T0MR2 | 0xE000 8020 T1MR2 |
| MR3 | Match Register 3. See MR0 description. | R/W | 0 | 0xE000 4024 T0MR3 | 0xE000 8024 T1MR3 |
| 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 | 0xE000 4028 T0CCR | 0xE000 8028 T1CCR |
| 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 | 0xE000 402C T0CR0 | 0xE000 802C T1CR0 |
| CR1 | Capture Register 1. See CR0 description. | RO | 0 | 0xE000 4030 T0CR1 | 0xE000 8030 T1CR1 |
| CR2 | Capture Register 2. See CR0 description. | RO | 0 | 0xE000 4034 T0CR2 | 0xE000 8034 T1CR2 |

Table 237: TIMER/COUNTER0 and TIMER/COUNTER1 register map

| Generic Name | Description | Access | Reset value ^[1] | TIMER/COUNTER0 Address & Name | TIMER/COUNTER1 Address & Name |
|--------------|---|--------|----------------------------|-------------------------------|-------------------------------|
| CR3 | Capture Register 3. See CR0 description. | RO | 0 | 0xE000 4038 T0CR3 | 0xE000 8038 T1CR3 |
| 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 | 0xE000 403C T0EMR | 0xE000 803C T1EMR |
| 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 | 0xE000 4070 T0CTCR | 0xE000 8070 T1CTCR |

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

15.5.1 Interrupt Register (IR, TIMER0: T0IR - 0xE000 4000 and TIMER1: T1IR - 0xE000 8000)

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 238: Interrupt Register (IR, TIMER0: T0IR - address 0xE000 4000 and TIMER1: T1IR - address 0xE000 8000) 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 |

15.5.2 Timer Control Register (TCR, TIMER0: T0TCR - 0xE000 4004 and TIMER1: T1TCR - 0xE000 8004)

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 239: Timer Control Register (TCR, TIMER0: T0TCR - address 0xE000 4004 and TIMER1: T1TCR - address 0xE000 8004) 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 |

15.5.3 Count Control Register (CTCR, TIMER0: T0CTCR - 0xE000 4070 and TIMER1: T1TCR - 0xE000 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.

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 half 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/PCLK.

Table 240: Count Control Register (CTCR, TIMER0: T0CTCR - address 0xE000 4070 and TIMER1: T1TCR - address 0xE000 8070) 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 | |
| | | 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 240: Count Control Register (CTCR, TIMER0: T0CTCR - address 0xE000 4070 and TIMER1: T1TCR - address 0xE000 8070) 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 (CAP0.0 for TIMER0 and CAP1.0 for TIMER1) | |
| | | 01 | CAPn.1 (CAP0.1 for TIMER0 and CAP1.1 for TIMER1) | |
| | | 10 | CAPn.2 (CAP0.2 for TIMER0 and CAP1.2 for TIMER1) | |
| | | 11 | CAPn.3 (CAP0.3 for TIMER0 and CAP1.3 for TIMER1) | |
| 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 |

15.5.4 Timer Counter (TC, TIMER0: T0TC - 0xE000 4008 and TIMER1: T1TC - 0xE000 8008)

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

15.5.5 Prescale Register (PR, TIMER0: T0PR - 0xE000 400C and TIMER1: T1PR - 0xE000 800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

15.5.6 Prescale Counter Register (PC, TIMER0: T0PC - 0xE000 4010 and TIMER1: T1PC - 0xE000 8010)

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1, etc.

15.5.7 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.

15.5.8 Match Control Register (MCR, TIMER0: T0MCR - 0xE000 4014 and TIMER1: T1MCR - 0xE000 8014)

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 241](#).

Table 241: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) 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. | |
| 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 |

15.5.9 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.

15.5.10 Capture Control Register (CCR, TIMER0: T0CCR - 0xE000 4028 and TIMER1: T1CCR - 0xE000 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 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.

Table 242: Capture Control Register (CCR, TIMER0: T0CCR - address 0xE000 4028 and TIMER1: T1CCR - address 0xE000 8028) 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. | |

Table 242: Capture Control Register (CCR, TIMER0: T0CCR - address 0xE000 4028 and TIMER1: T1CCR - address 0xE000 8028) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 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 |

15.5.11 External Match Register (EMR, TIMER0: T0EMR - 0xE000 403C; and TIMER1: T1EMR - 0xE000 803C)

The External Match Register provides both control and status of the external match pins MAT(0-3).

Table 243: External Match Register (EMR, TIMER0: T0EMR - address 0xE000 403C and TIMER1: T1EMR - address 0xE000 803C) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 0 | EM0 | External Match 0. This bit reflects the state of output MAT0.0/MAT1.0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[5:4] control the functionality of this output. | 0 |
| 1 | EM1 | External Match 1. This bit reflects the state of output MAT0.1/MAT1.1, whether or not this output is connected to its pin. When a match occurs between the TC and MR1, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[7:6] control the functionality of this output. | 0 |
| 2 | EM2 | External Match 2. This bit reflects the state of output MAT0.2/MAT1.2, whether or not this output is connected to its pin. When a match occurs between the TC and MR2, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[9:8] control the functionality of this output. | 0 |
| 3 | EM3 | External Match 3. This bit reflects the state of output MAT0.3/MAT1.3, whether or not this output is connected to its pin. When a match occurs between the TC and MR3, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[11:10] control the functionality of this output. | 0 |
| 5:4 | EMC0 | External Match Control 0. Determines the functionality of External Match 0. Table 244 shows the encoding of these bits. | 00 |
| 7:6 | EMC1 | External Match Control 1. Determines the functionality of External Match 1. Table 244 shows the encoding of these bits. | 00 |
| 9:8 | EMC2 | External Match Control 2. Determines the functionality of External Match 2. Table 244 shows the encoding of these bits. | 00 |
| 11:10 | EMC3 | External Match Control 3. Determines the functionality of External Match 3. Table 244 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 244: 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. |

15.6 Example timer operation

[Figure 56](#) 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 57](#) 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.

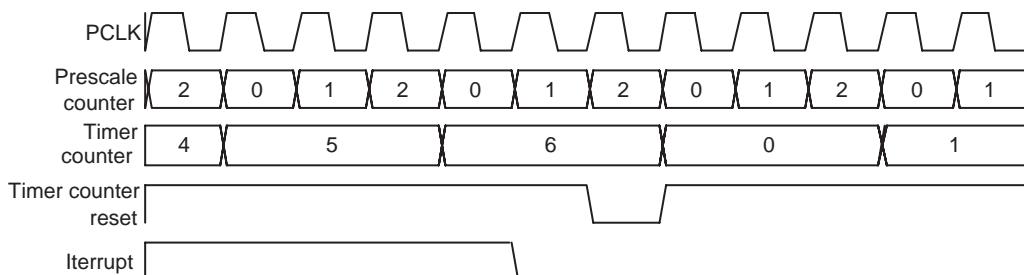


Fig 56. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

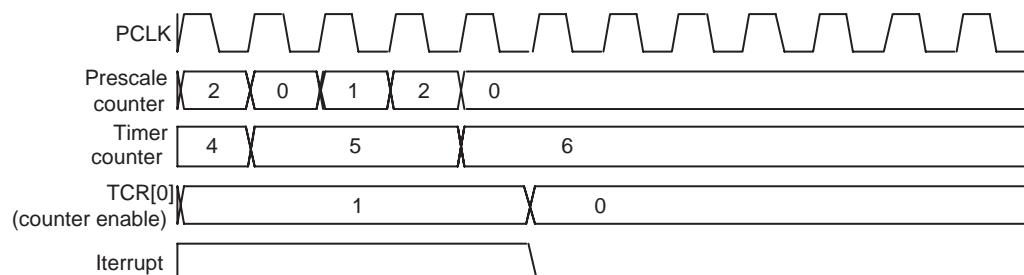
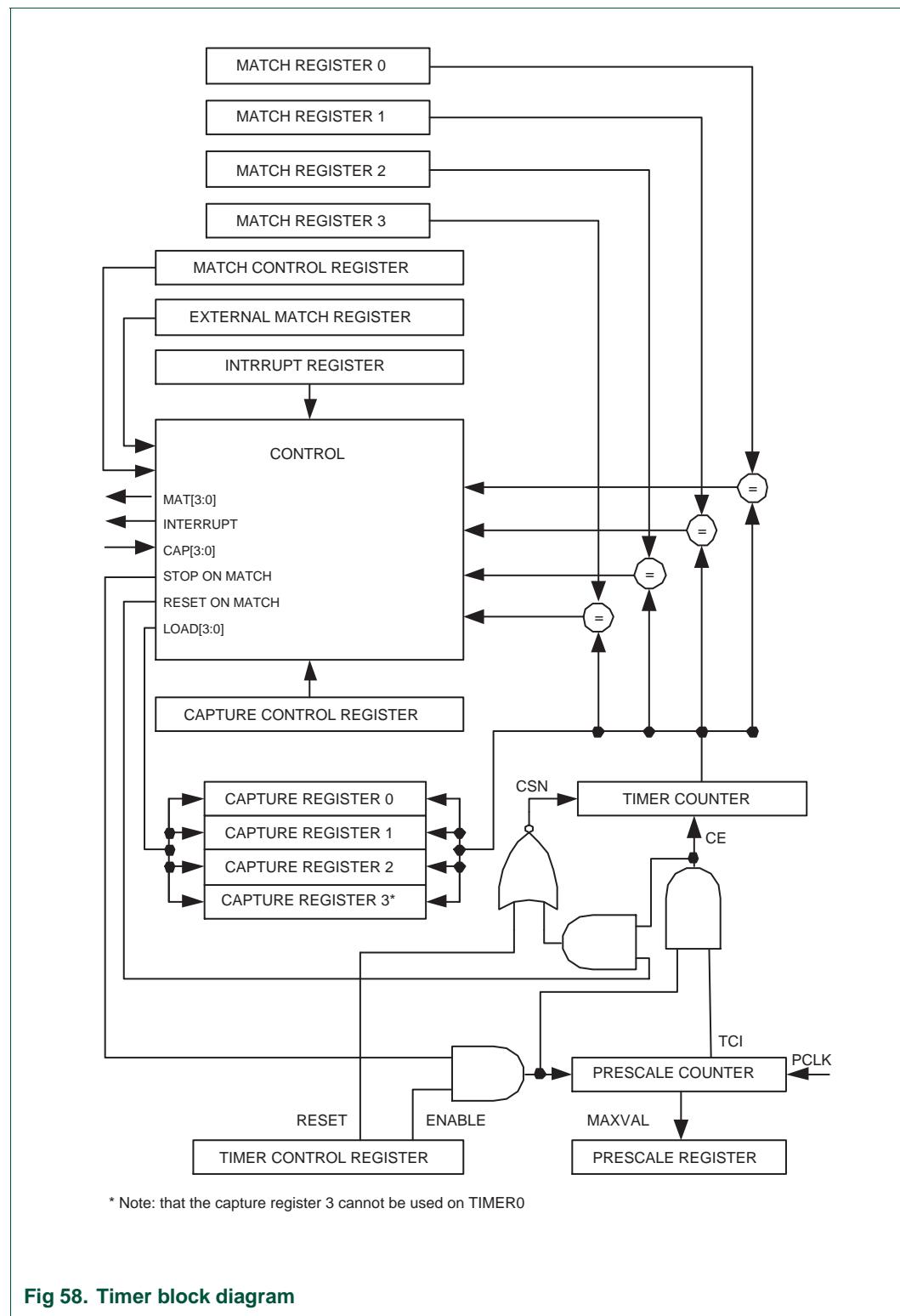


Fig 57. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

15.7 Architecture

The block diagram for TIMER/COUNTER0 and TIMER/COUNTER1 is shown in [Figure 58](#).



LPC2141/2/4/6/8 Pulse Width Modulator is based on standard Timer/Counter 0/1 described in the previous chapter. Application can choose among PWM and match functions available.

16.1 Features

- 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.
- An external output for each match register with the following capabilities:
 - Set low on match.
 - Set high on match.
 - Toggle on match.
 - Do nothing on match.
- 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.
- Four 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.

16.2 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 LPC2141/2/4/6/8. 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.

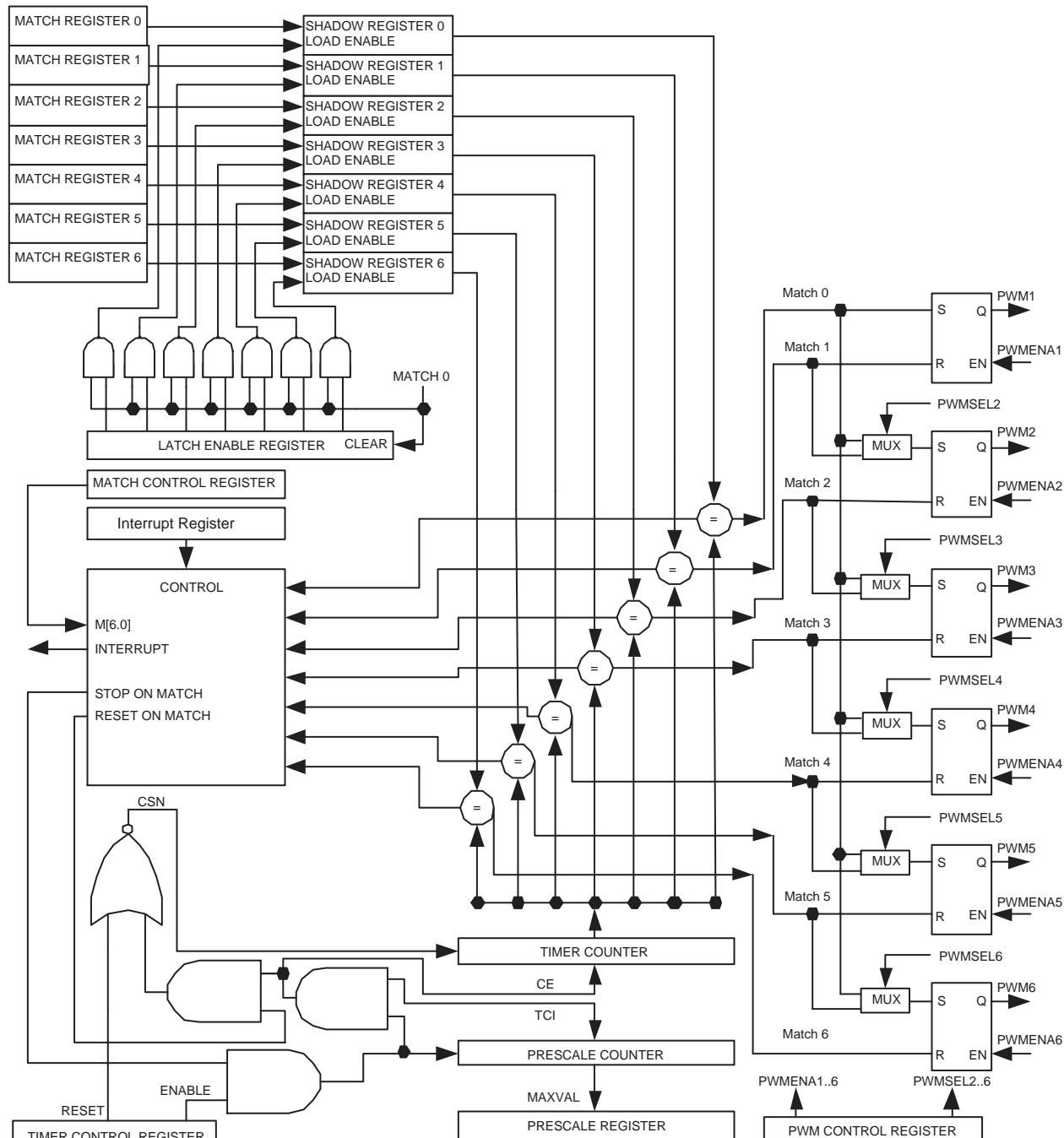
The ability to separately control rising and falling edge locations allows the PWM to be used for more applications. For instance, multi-phase motor control typically requires three non-overlapping PWM outputs with individual control of all three pulse widths and positions.

Two match registers can be used to provide a single edge controlled PWM output. One match register (PWMMR0) 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 PWMMR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the PWMMR0 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 59](#) 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.



Note: this diagram is intended to clarify the function of the PWM rather than to suggest a specific design implementation.

Fig 59. PWM block diagram

A sample of how PWM values relate to waveform outputs is shown in [Figure 60](#). PWM output logic is shown in [Figure 59](#) that allows selection of either single or double edge controlled PWM outputs via the muxes controlled by the PWMSEL_n bits. The match register selections for various PWM outputs is shown in [Table 245](#). This implementation supports up to N-1 single edge PWM outputs or (N-1)/2 double edge PWM outputs, where N is the number of match registers that are implemented. PWM types can be mixed if desired.

The waveforms below show a single PWM cycle and demonstrate PWM outputs under the following conditions:

The timer is configured for PWM mode.
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:
MRO = 100 (PWM rate)
MR1 = 41, MR2 = 78 (PWM2 output)
MR3 = 53, MR4 = 27 (PWM4 output)
MR5 = 65 (PWM5 output)

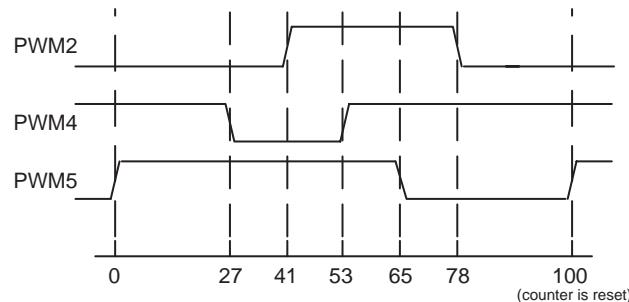


Fig 60. Sample PWM waveforms

Table 245: Set and reset inputs for PWM Flip-Flops

| PWM Channel | Single Edge PWM (PWMSEL _n = 0) | | Double Edge PWM (PWMSEL _n = 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.

16.2.1 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.

16.2.2 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.

16.3 Pin description

[Table 246](#) gives a brief summary of each of PWM related pins.

Table 246: 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. |

16.4 Register description

The PWM function adds new registers and registers bits as shown in [Table 247](#) below.

Table 247: Pulse Width Modulator (PWM) register map

| Name | Description | Access | Reset value ^[1] | Address |
|--------|---|--------|----------------------------|-------------|
| PWMIR | PWM Interrupt Register. The PWMIR can be written to clear interrupts. The PWMIR can be read to identify which of the possible interrupt sources are pending. | R/W | 0 | 0xE001 4000 |
| PWMTCR | PWM Timer Control Register. The PWMTCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the PWMTCR. | R/W | 0 | 0xE001 4004 |
| PWMTC | PWM Timer Counter. The 32-bit TC is incremented every PWMPR+1 cycles of PCLK. The PWMTC is controlled through the PWMTCR. | R/W | 0 | 0xE001 4008 |
| PWMPR | PWM Prescale Register. The PWMTC is incremented every PWMPR+1 cycles of PCLK. | R/W | 0 | 0xE001 400C |
| PWMPC | PWM Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PWMPR is reached, the PWMTC is incremented. The PWMPC is observable and controllable through the bus interface. | R/W | 0 | 0xE001 4010 |
| PWMMCR | PWM Match Control Register. The PWMMCR is used to control if an interrupt is generated and if the PWMTC is reset when a Match occurs. | R/W | 0 | 0xE001 4014 |
| PWMMR0 | PWM Match Register 0. PWMMR0 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR0 and the PWMTC sets all PWM outputs that are in single-edge mode, and sets PWM1 if it is in double-edge mode. | R/W | 0 | 0xE001 4018 |
| PWMMR1 | PWM Match Register 1. PWMMR1 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR1 and the PWMTC clears PWM1 in either single-edge mode or double-edge mode, and sets PWM2 if it is in double-edge mode. | R/W | 0 | 0xE001 401C |
| PWMMR2 | PWM Match Register 2. PWMMR2 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR2 and the PWMTC clears PWM2 in either single-edge mode or double-edge mode, and sets PWM3 if it is in double-edge mode. | R/W | 0 | 0xE001 4020 |
| PWMMR3 | PWM Match Register 3. PWMMR3 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR3 and the PWMTC clears PWM3 in either single-edge mode or double-edge mode, and sets PWM4 if it is in double-edge mode. | R/W | 0 | 0xE001 4024 |
| PWMMR4 | PWM Match Register 4. PWMMR4 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR4 and the PWMTC clears PWM4 in either single-edge mode or double-edge mode, and sets PWM5 if it is in double-edge mode. | R/W | 0 | 0xE001 4040 |
| PWMMR5 | PWM Match Register 5. PWMMR5 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR5 and the PWMTC clears PWM5 in either single-edge mode or double-edge mode, and sets PWM6 if it is in double-edge mode. | R/W | 0 | 0xE001 4044 |

Table 247: Pulse Width Modulator (PWM) register map

| Name | Description | Access | Reset value ^[1] | Address |
|--------|---|--------|----------------------------|-------------|
| PWMMR6 | PWM Match Register 6. PWMMR6 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR6 and the PWMTC clears PWM6 in either single-edge mode or double-edge mode. | R/W | 0 | 0xE001 4048 |
| PWMPCR | PWM Control Register. Enables PWM outputs and selects PWM channel types as either single edge or double edge controlled. | R/W | 0 | 0xE001 404C |
| PWMLER | PWM Latch Enable Register. Enables use of new PWM match values. | R/W | 0 | 0xE001 4050 |

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

16.4.1 PWM Interrupt Register (PWMIIR - 0xE001 4000)

The PWM Interrupt Register consists of eleven bits ([Table 248](#)), seven for the match interrupts and four reserved for the future use. If an interrupt is generated then the corresponding bit in the PWMIIR 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 248: PWM Interrupt Register (PWMIIR - address 0xE001 4000) 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 |
| 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 |

16.4.2 PWM Timer Control Register (PWMTCR - 0xE001 4004)

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 249](#).

Table 249: PWM Timer Control Register (PWMTCR - address 0xE001 4004) 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 TCR[1] 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 0) 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 |
| 7:4 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

16.4.3 PWM Timer Counter (PWMTC - 0xE001 4008)

The 32-bit PWM Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the PWMTC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

16.4.4 PWM Prescale Register (PWMPR - 0xE001 400C)

The 32-bit PWM Prescale Register specifies the maximum value for the PWM Prescale Counter.

16.4.5 PWM Prescale Counter register (PWMPC - 0xE001 4010)

The 32-bit PWM Prescale Counter controls division of PCLK by some constant value before it is applied to the PWM Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The PWM Prescale Counter is incremented on every PCLK. When it reaches the value stored in the PWM Prescale Register, the PWM Timer Counter is incremented and the PWM Prescale Counter is reset on the next PCLK. This causes the PWM TC to increment on every PCLK when PWMPR = 0, every 2 PCLKs when PWMPR = 1, etc.

16.4.6 PWM Match Registers (PWMMR0 - PWMMR6)

The 32-bit PWM Match register values are continuously compared to the PWM Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the PWM Timer Counter, or stop the timer. Actions are controlled by the settings in the PWMMCR register.

16.4.7 PWM Match Control Register (PWMMCR - 0xE001 4014)

The PWM Match Control Register is 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 250](#).

Table 250: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - 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[0] will be set to 0 if PWMMR0 matches the PWMTC. | 0 |
| | | 0 | This feature is disabled | |
| 3 | PWMMR1I | 1 | Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC. | 0 |
| | | 0 | This interrupt is disabled. | |
| 1 | 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[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 PWMTC 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[0] will be set to 0 if PWMMR2 matches the PWMTC. | 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 | |

Table 250: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|---------|-------|--|-------------|
| 11 | PWMMR3S | 1 | Stop on PWMMR3: The PWMTC and PWMPC will be stopped and PWMTCR[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. | |
| 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 |

16.4.8 PWM Control Register (PWMPCR - 0xE001 404C)

The PWM Control Register is used to enable and select the type of each PWM channel. The function of each of the bits are shown in [Table 251](#).

Table 251: PWM Control Register (PWMPCR - address 0xE001 404C) 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 | PWMSEL2 | 1 | Selects double edge controlled mode for the PWM2 output. | 0 |
| | | 0 | Selects single edge controlled mode for PWM2. | |
| 3 | PWMSEL3 | 1 | Selects double edge controlled mode for the PWM3 output. | 0 |
| | | 0 | Selects single edge controlled mode for PWM3. | |

Table 251: PWM Control Register (PWMPCR - address 0xE001 404C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------|-------|--|-------------|
| 4 | PWMSEL4 | 1 | Selects double edge controlled mode for the PWM4 output. | 0 |
| | | 0 | Selects single edge controlled mode for PWM4. | |
| 5 | PWMSEL5 | 1 | Selects double edge controlled mode for the PWM5 output. | 0 |
| | | 0 | Selects single edge controlled mode for PWM5. | |
| 6 | PWMSEL6 | 1 | Selects double edge controlled mode for the PWM6 output. | 0 |
| | | 0 | Selects single edge controlled mode for PWM6. | |
| 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 | 1 | The PWM1 output enabled. | 0 |
| | | 0 | The PWM1 output disabled. | |
| 10 | PWMENA2 | 1 | The PWM2 output enabled. | 0 |
| | | 0 | The PWM2 output disabled. | |
| 11 | PWMENA3 | 1 | The PWM3 output enabled. | 0 |
| | | 0 | The PWM3 output disabled. | |
| 12 | PWMENA4 | 1 | The PWM4 output enabled. | 0 |
| | | 0 | The PWM4 output disabled. | |
| 13 | PWMENA5 | 1 | The PWM5 output enabled. | 0 |
| | | 0 | The PWM5 output disabled. | |
| 14 | PWMENA6 | 1 | The PWM6 output enabled. | 0 |
| | | 0 | The PWM6 output disabled. | |
| 15 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

16.4.9 PWM Latch Enable Register (PWMLER - 0xE001 4050)

The PWM Latch Enable Register is 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 held in a shadow register. 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 252](#).

Table 252: PWM Latch Enable Register (PWMLER - address 0xE001 4050) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------------------------|---|-------------|
| 0 | Enable PWM Match 0 Latch | Writing a one to this bit allows the last value written to the PWM Match 0 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 1 | Enable PWM Match 1 Latch | Writing a one to this bit allows the last value written to the PWM Match 1 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 2 | Enable PWM Match 2 Latch | Writing a one to this bit allows the last value written to the PWM Match 2 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 3 | Enable PWM Match 3 Latch | Writing a one to this bit allows the last value written to the PWM Match 3 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 4 | Enable PWM Match 4 Latch | Writing a one to this bit allows the last value written to the PWM Match 4 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 5 | Enable PWM Match 5 Latch | Writing a one to this bit allows the last value written to the PWM Match 5 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 6 | Enable PWM Match 6 Latch | Writing a one to this bit allows the last value written to the PWM Match 6 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.4.7 "PWM Match Control Register (PWMMCR - 0xE001 4014)". | 0 |
| 7 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

17.1 Features

- 10 bit successive approximation analog to digital converter (one in LPC2141/2 and two in LPC2144/6/8).
- Input multiplexing among 6 or 8 pins (ADC0 and ADC1).
- Power-down mode.
- Measurement range 0 V to V_{REF} (typically 3 V; not to exceed V_{DDA} voltage level).
- 10 bit conversion time $\geq 2.44 \mu\text{s}$.
- Burst conversion mode for single or multiple inputs.
- Optional conversion on transition on input pin or Timer Match signal.
- Global Start command for both converters (LPC2144/6/8 only).

17.2 Description

Basic clocking for the A/D converters is provided by the VPB clock. 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.

17.3 Pin description

[Table 253](#) gives a brief summary of each of ADC related pins.

Table 253: ADC pin description

| Pin | Type | Description |
|--|-----------|--|
| AD0.7:6, AD0.4:1 & AD1.7:0 (LPC2144/6/8) | Input | <p>Analog Inputs. The ADC 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 function Select register assigns them to port pins. A simple self-test of the ADC can be done by driving these pins as port outputs.</p> <p>Note: if the ADC is used, signal levels on analog input pins must not be above the level of V_{3A} at any time. Otherwise, A/D converter readings will be invalid. If the A/D converter is not used in an application then the pins associated with A/D inputs can be used as 5 V tolerant digital IO pins.</p> <p>Warning: while the ADC pins are specified as 5 V tolerant (see Table 58 "Pin description" on page 69), the analog multiplexing in the ADC block is not. More than 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.</p> |
| V_{REF} | Reference | Voltage Reference. This pin is provides a voltage reference level for the A/D converter(s). |
| V_{DDA}, V_{SSA} | Power | Analog Power and Ground. These should be nominally the same voltages as V_{DD} and V_{SS} , but should be isolated to minimize noise and error. |

17.4 Register description

The A/D Converter registers are shown in [Table 254](#).

Table 254: ADC registers

| Generic Name | Description | Access | Reset value ^[1] | AD0 Address & Name | AD1 Address & Name |
|--------------|--|--------|----------------------------|-------------------------|-------------------------|
| ADCR | A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur. | R/W | 0x0000 0001 | 0xE003 4000 AD0CR | 0xE006 0000 AD1CR |
| ADGDR | A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion. | R/W | NA | 0xE003 4004 AD0GDR | 0xE006 0004 AD1GDR |
| ADSTAT | 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 | 0x0000 0000 | 0xE003 4030 AD0STAT | 0xE006 0030 AD1STAT |
| ADGSR | A/D Global Start Register. This address can be written (in the AD0 address range) to start conversions in both A/D converters simultaneously. | WO | 0x00 | | 0xE003 4008 ADGSR |
| ADINTEN | 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 AD0INTEN | 0xE006 000C AD1INTEN |
| ADDR0 | A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0. | RO | NA | 0xE003 4010 AD0DR0 | 0xE006 0010 AD1DR0 |
| ADDR1 | A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1. | RO | NA | 0xE003 4014 AD0DR1 | 0xE006 0014 AD1DR1 |
| ADDR2 | A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2. | RO | NA | 0xE003 4018 AD0DR2 | 0xE006 0018 AD1DR2 |
| ADDR3 | A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3. | RO | NA | 0xE003 401C AD0DR3 | 0xE006 001C AD1DR3 |
| ADDR4 | A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4. | RO | NA | 0xE003 4020 AD0DR4 | 0xE006 0020 AD1DR4 |
| ADDR5 | A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5. | RO | NA | 0xE003 4024 AD0DR5 | 0xE006 0024 AD1DR5 |
| ADDR6 | A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6. | RO | NA | 0xE003 4028 AD0DR6 | 0xE006 0028 AD1DR6 |
| ADDR7 | A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7. | RO | NA | 0xE003 402C AD0DR7 | 0xE006 002C AD1DR7 |

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

17.4.1 A/D Control Register (AD0CR - 0xE003 4000 and AD1CR - 0xE006 0000)

Table 255: A/D Control Register (AD0CR - address 0xE003 4000 and AD1CR - address 0xE006 0000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|---|-------------|
| 7:0 | SEL | | Selects which of the AD0.7:0/AD1.7:0 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 VPB 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 | 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. | 0 |
| | | 0 | Conversions are software controlled and require 11 clocks. | |
| 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 RESULT bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits). | 000 |
| | | 000 | 11 clocks / 10 bits | |
| | | 001 | 10 clocks / 9bits | |
| | | 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 |

Table 255: A/D Control Register (AD0CR - address 0xE003 4000 and AD1CR - address 0xE006 0000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 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 P0.16/EINT0/MAT0.2/CAP0.2 pin. | |
| | | 011 | Start conversion when the edge selected by bit 27 occurs on P0.22/TD3/CAP0.0/MAT0.0 pin. | |
| | | 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. | |
| 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 |

17.4.2 A/D Global Data Register (AD0GDR - 0xE003 4004 and AD1GDR - 0xE006 0004)

Table 256: A/D Global Data Register (AD0GDR - address 0xE003 4004 and AD1GDR - address 0xE006 0004) bit description

| Bit | Symbol | 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 | RESULT | 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 VDDA pin (V/V _{REF}). 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} . | NA |
| 23:16 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 26:24 | CHN | These bits contain the channel from which the RESULT bits were converted (e.g. 000 identifies channel 0, 001 channel 1...). | NA |
| 29:27 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 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 RESULT bits. This bit is cleared by reading this register. | 0 |
| 31 | DONE | This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started. | 0 |

17.4.3 A/D Global Start Register (ADGSR - 0xE003 4008)

Software can write this register to simultaneously initiate conversions on both A/D controllers. This register is available in LPC2144/6/8 devices only.

Table 257: A/D Global Start Register (ADGSR - address 0xE003 4008) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|---|-------------|
| 15:0 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 16 | BURST | 1 | The AD converters do repeated conversions at the rate selected by their CLKS fields, scanning (if necessary) through the pins selected by 1s in their 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. | 0 |
| | | 0 | Conversions are software controlled and require 11 clocks. | |
| 23:17 | - | | 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: 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 P0.16/EINT0/MAT0.2/CAP0.2 pin. 011 Start conversion when the edge selected by bit 27 occurs on P0.22/TD3/CAP0.0/MAT0.0 pin. 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. | 0 |
| 27 | EDGE | | This bit is significant only when the START field contains 010-111. In these cases: 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. | 0 |
| 31:28 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

17.4.4 A/D Status Register (ADSTAT, ADC0: AD0CR - 0xE003 4004 and ADC1: AD1CR - 0xE006 0004)

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 258: A/D Status Register (ADSTAT, ADC0: AD0STAT - address 0xE003 4004 and ADC1: AD1STAT - address 0xE006 0004) bit description

| Bit | Symbol | Description | Reset value |
|-------|----------|--|-------------|
| 0 | DONE0 | This bit mirrors the DONE status flag from the result register for A/D channel 0. | 0 |
| 1 | DONE1 | This bit mirrors the DONE status flag from the result register for A/D channel 1. | 0 |
| 2 | DONE2 | This bit mirrors the DONE status flag from the result register for A/D channel 2. | 0 |
| 3 | DONE3 | This bit mirrors the DONE status flag from the result register for A/D channel 3. | 0 |
| 4 | DONE4 | This bit mirrors the DONE status flag from the result register for A/D channel 4. | 0 |
| 5 | DONE5 | This bit mirrors the DONE status flag from the result register for A/D channel 5. | 0 |
| 6 | DONE6 | This bit mirrors the DONE status flag from the result register for A/D channel 6. | 0 |
| 7 | DONE7 | This bit mirrors the DONE status flag from the result register for A/D channel 7. | 0 |
| 8 | OVERRUN0 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 0. | 0 |
| 9 | OVERRUN1 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 1. | 0 |
| 10 | OVERRUN2 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 2. | 0 |
| 11 | OVERRUN3 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 3. | 0 |
| 12 | OVERRUN4 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 4. | 0 |
| 13 | OVERRUN5 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 5. | 0 |
| 14 | OVERRUN6 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 6. | 0 |
| 15 | OVERRUN7 | This bit mirrors the OVERRUN status flag from the result register for A/D channel 7. | 0 |
| 16 | ADINT | This bit is the A/D interrupt flag. It is one when any of the individual A/D channel Done flags is asserted and enabled to contribute to the A/D interrupt via the ADINTEN register. | 0 |
| 31:17 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

17.4.5 A/D Interrupt Enable Register (ADINTEN, ADC0: AD0INTEN - 0xE003 400C and ADC1: AD1INTEN - 0xE006 000C)

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 259: A/D Status Register (ADSTAT, ADC0: AD0STAT - address 0xE003 4004 and ADC1: AD1STAT - address 0xE006 0004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|---|-------------|
| 0 | ADINTENO | 0 | Completion of a conversion on ADC channel 0 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 0 will generate an interrupt. | |
| 1 | ADINTEN1 | 0 | Completion of a conversion on ADC channel 1 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 1 will generate an interrupt. | |
| 2 | ADINTEN2 | 0 | Completion of a conversion on ADC channel 2 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 2 will generate an interrupt. | |
| 3 | ADINTEN3 | 0 | Completion of a conversion on ADC channel 3 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 3 will generate an interrupt. | |

Table 259: A/D Status Register (ADSTAT, ADC0: AD0STAT - address 0xE003 4004 and ADC1: AD1STAT - address 0xE006 0004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|----------|-------|--|-------------|
| 4 | ADINTEN4 | 0 | Completion of a conversion on ADC channel 4 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 4 will generate an interrupt. | |
| 5 | ADINTEN5 | 0 | Completion of a conversion on ADC channel 5 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 5 will generate an interrupt. | |
| 6 | ADINTEN6 | 0 | Completion of a conversion on ADC channel 6 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 6 will generate an interrupt. | |
| 7 | ADINTEN1 | 0 | Completion of a conversion on ADC channel 7 will not generate an interrupt. | 0 |
| | | 1 | Completion of a conversion on ADC channel 7 will generate an interrupt. | |
| 8 | ADGINTEN | 0 | Only the individual ADC channels enabled by ADINTEN7:0 will generate interrupts. | 1 |
| | | 1 | Only the global DONE flag in ADDR is enabled to generate an interrupt. | |
| 31:17 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

17.4.6 A/D Data Registers (ADDR0 to ADDR7, ADC0: AD0DR0 to AD0DR7 - 0xE003 4010 to 0xE003 402C and ADC1: AD1DR0 to AD1DR7- 0xE006 0010 to 0xE006 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 260: A/D Data Registers (ADDR0 to ADDR7, ADC0: AD0DR0 to AD0DR7 - 0xE003 4010 to 0xE003 402C and ADC1: AD1DR0 to AD1DR7- 0xE006 0010 to 0xE006 402C) bit description

| Bit | Symbol | 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 | RESULT | When DONE is 1, this field contains a binary fraction representing the voltage on the AIN pin, divided by the voltage on the V _{REF} pin (V/V _{REF}). 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} . | NA |
| 29:16 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 30 | OVERRUN | This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register. | NA |
| 31 | DONE | This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read. | NA |

17.5 Operation

17.5.1 Hardware-triggered conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the ADC will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

17.5.2 Interrupts

An interrupt request is asserted to the Vectored Interrupt Controller (VIC) when the DONE bit is 1. Software can use the Interrupt Enable bit for the A/D Converter in the VIC to control whether this assertion results in an interrupt. DONE is negated when the ADDR is read.

17.5.3 Accuracy vs. digital receiver

The AIN function must be selected in corresponding Pin Select register (see "Pin Connect Block" on page 75) in order to get accurate voltage readings on the monitored pin. For pin hosting an ADC input, it is not possible to have a digital function selected and yet get valid ADC readings. An inside circuit disconnects ADC hardware from the associated pin whenever a digital function is selected on that pin.

This peripheral is available in LPC2142/4/6/8 devices.

18.1 Features

- 10 bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power-down mode
- Selectable speed vs. power

18.2 Pin description

[Table 261](#) gives a brief summary of each of DAC related pins.

Table 261: DAC 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 * V _{REF} . |
| V _{REF} | 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 ₃ and V _{SSD} , but should be isolated to minimize noise and error. |

18.3 DAC Register (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 262: DAC 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 AOUT pin (with respect to V _{SSA}) is VALUE/1024 * V _{REF} . | 0 |
| 16 | BIAS | 0 | The settling time of the DAC is 1 μ s max, and the maximum current is 700 μ A. | 0 |
| | | 1 | The settling time of the DAC is 2.5 μ s and the maximum current is 350 μ A. | |
| 31:17 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

18.4 Operation

Bits 19:18 of the PINSEL1 register ([Section 7.4.2 “Pin function Select register 1 \(PINSEL1 - 0xE002 C004\)” on page 77](#)) control whether the DAC is enabled and controlling the state of pin P0.25/AD0.4/AOUT. 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 A_{OUT} pin not exceeding 100 pF. A load impedance value greater than that value will cause settling time longer than the specified time.

19.1 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 VPB clock.
- Dedicated power supply pin can be connected to a battery or to the main 3.3 V.

19.2 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 LPC2141/2/4/6/8, the RTC can be clocked by a separate 32.768 KHz oscillator, or by a programmable prescale divider based on the VPB clock. Also, the RTC is powered by its own power supply pin, V_{BAT} , which can be connected to a battery or to the same 3.3 V supply used by the rest of the device.

19.3 Architecture

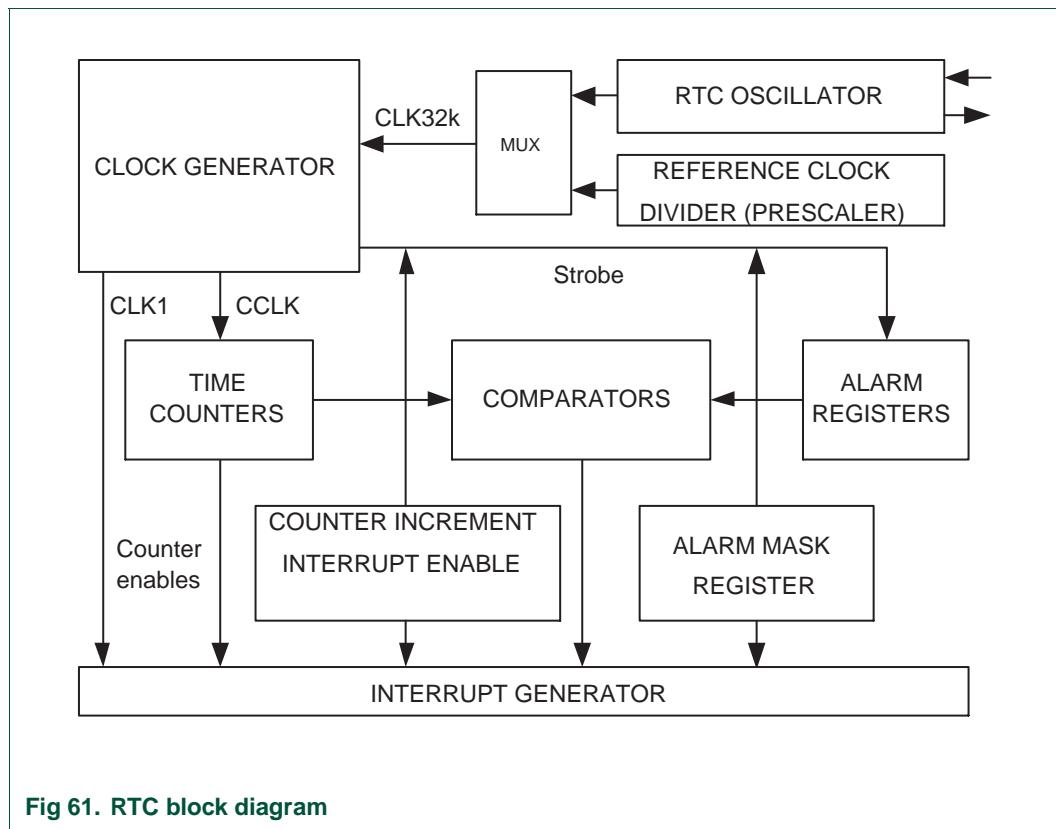


Fig 61. RTC block diagram

19.4 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 19.4.2](#)). The second set of eight locations are the Time Counter Group ([Section 19.4.12](#)). The third set of eight locations contain the Alarm Register Group ([Section 19.4.14](#)). The remaining registers control the Reference Clock Divider.

The Real Time Clock includes the register shown in [Table 263](#). Detailed descriptions of the registers follow.

Table 263: Real Time Clock (RTC) 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 |
| 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 |
| 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.

19.4.1 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, XTAL1/2 pins associated oscillator wakeup cycle is started. For details on the RTC based wakeup process see [Section 3.5.3 "Interrupt Wakeup register \(INTWAKE - 0xE01F C144\)" on page 22](#) and [Section 3.12 "Wakeup timer" on page 41](#).

19.4.2 Miscellaneous register group

[Table 264](#) summarizes the registers located from 0 to 7 of A[6:2]. More detailed descriptions follow.

Table 264: Miscellaneous registers

| Name | Size | Description | Access | Address |
|--------|------|--|--------|-------------|
| ILR | 2 | 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 |

19.4.3 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 265](#)). 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 265: 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. | NA |
| 1 | RTCALF | When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm 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 |

19.4.4 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 266: 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 19.6 "Reference clock divider (prescaler)" on page 282 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 |

19.4.5 Clock Control Register (CCR - 0xE002 4008)

The clock register is a 5-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in [Table 267](#).

Table 267: 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 Philips 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 19.7 "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 |

19.4.6 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 268: 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 |

19.4.7 Alarm Mask Register (AMR - 0xE002 4010)

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers.

[Table 269](#) 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 269: 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 |

19.4.8 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 270](#), [Table 271](#), and [Table 272](#). 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.

19.4.9 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 270: 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 |

19.4.10 Consolidated Time register 1 (CTIME1 - 0xE002 4018)

The Consolidate Time register 1 contains the Day of Month, Month, and Year values.

Table 271: 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 |

19.4.11 Consolidated Time register 2 (CTIME2 - 0xE002 401C)

The Consolidate Time register 2 contains just the Day of Year value.

Table 272: 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 |

19.4.12 Time counter group

The time value consists of the eight counters shown in [Table 273](#) and [Table 274](#). These counters can be read or written at the locations shown in [Table 274](#).

Table 273: Time counter relationships and values

| Counter | Size | Enabled by | Minimum value | Maximum value |
|--------------|------|---------------------------------------|---------------|----------------------------|
| Second | 6 | Clk1 (see Figure 61) | 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 274: 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 |
| 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.

19.4.13 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.

19.4.14 Alarm register group

The alarm registers are shown in [Table 275](#). The values in these registers are compared with the time counters. If all the unmasked (See [Section 19.4.7 "Alarm Mask Register \(AMR - 0xE002 4010\)" on page 279](#)) 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 275: 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 |

19.5 RTC usage notes

If the RTC is used, V_{BAT} must be connected to either pin V_3 or an independent power supply (external battery). Otherwise, V_{BAT} should be tied to the ground (V_{SS}). No provision is made in the LPC2141/2/4/6/8 to retain RTC status upon the V_{BAT} 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 VPB 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 VPB 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 VPB 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 3.9 "Power control" on page 35](#)).

19.6 Reference clock divider (prescaler)

The reference clock divider (hereafter referred to as the prescaler) allows generation of a 32.768 kHz reference clock from any peripheral clock 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 the peripheral clock (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 LPC2141/2/4/6/8, a 13-bit integer counter is required. This can be calculated as 160 MHz divided by 32,768 minus 1 = 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 ($32,768 \times 8192$).
2. The remainder value could be as large as 32,767, which requires 15 bits.

Table 276: 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 |

19.6.1 Prescaler Integer register (PREINT - 0xE002 4080)

This is the integer portion of the prescale value, calculated as:

$\text{PREINT} = \text{int}(\text{PCLK} / 32768) - 1$. The value of PREINT must be greater than or equal to 1.

Table 277: 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 |

19.6.2 Prescaler Fraction register (PREFRAC - 0xE002 4084)

This is the fractional portion of the prescale value, and may be calculated as:

$\text{PREFRAC} = \text{PCLK} - ([\text{PREINT} + 1] \times 32768)$.

Table 278: 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 |

19.6.3 Example of prescaler usage

In a simplistic case, the PCLK frequency is 65.537 kHz. So:

$$\begin{aligned}\text{PREINT} &= \text{int}(\text{PCLK} / 32768) - 1 = 1 \text{ and} \\ \text{PREFRAC} &= \text{PCLK} - ([\text{PREINT} + 1] \times 32768) = 1\end{aligned}$$

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,

$\text{PREINT} = \text{int}(\text{PCLK} / 32768) - 1 = 304$ and
 $\text{PREFRAC} = \text{PCLK} - ([\text{PREINT} + 1] \times 32768) = 5,760$.

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 19.4.4 "Clock Tick Counter Register \(CTCR - 0xE002 4004\)" on page 278](#)).

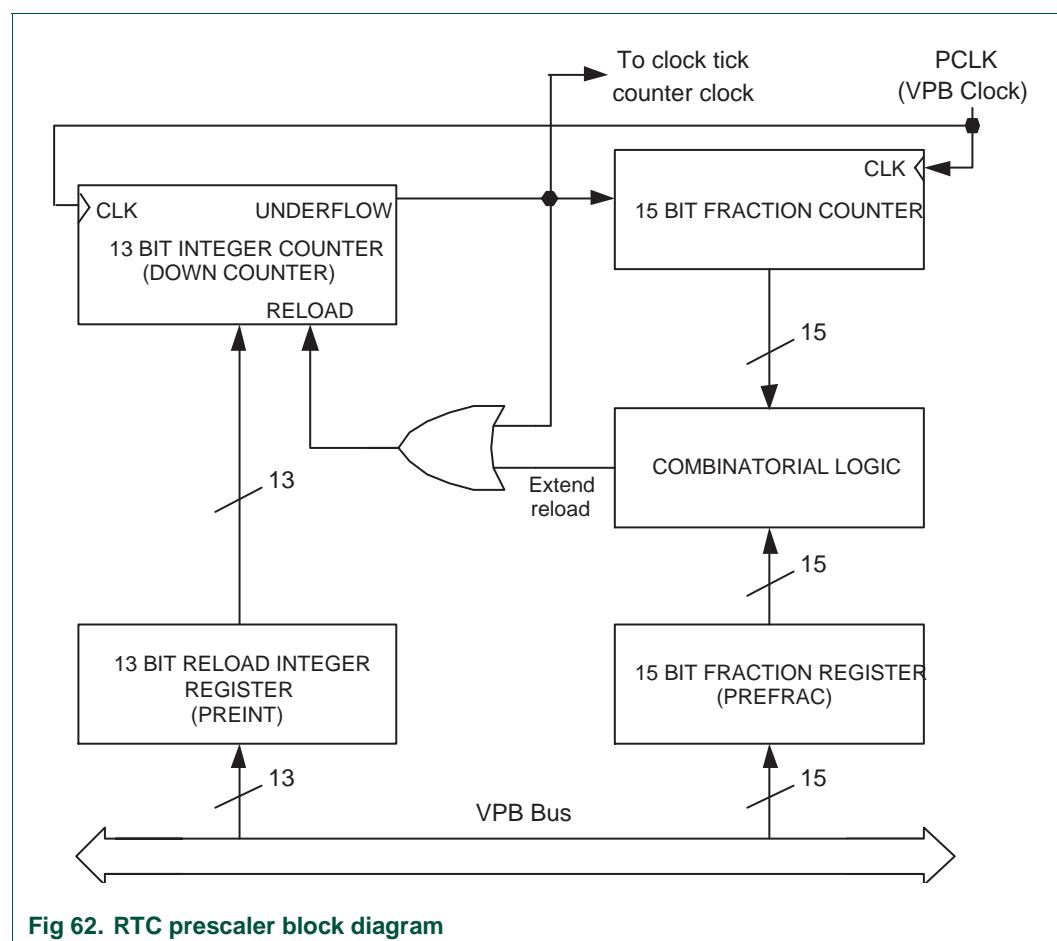


Fig 62. RTC prescaler block diagram

19.6.4 Prescaler operation

The Prescaler block labelled "Combination Logic" in [Figure 62](#) 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 279](#).

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 279: Prescaler cases where the Integer Counter reload value is incremented

19.7 RTC external 32 kHz oscillator component selection

The RTC external oscillator circuit is shown in [Figure 63](#). 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.

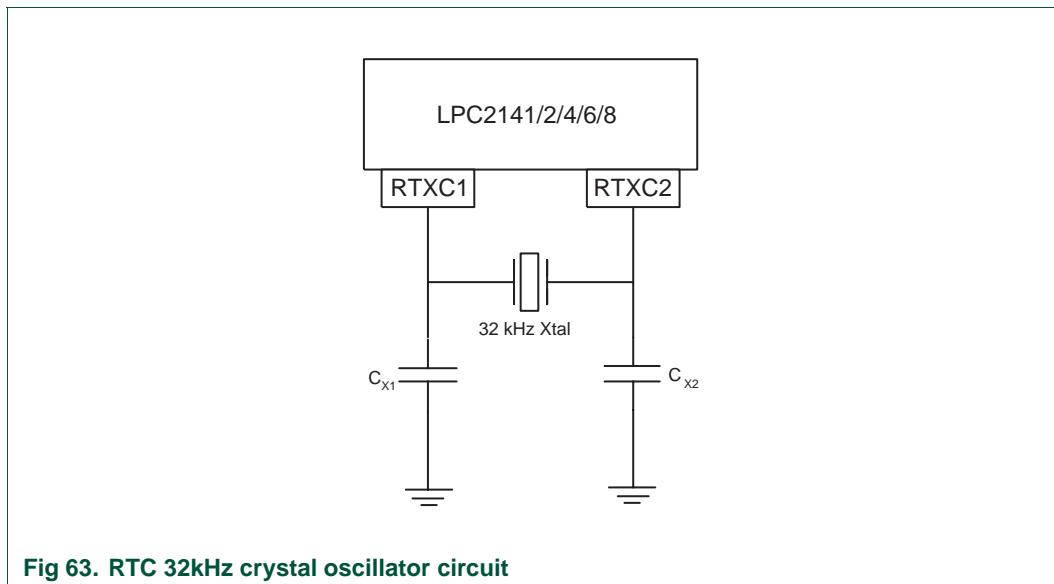


Fig 63. RTC 32kHz crystal oscillator circuit

[Table 280](#) 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 280](#) 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 280: 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 |



20.1 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 ($T_{PCLK} \times 256 \times 4$) to ($T_{PCLK} \times 2^{32} \times 4$) in multiples of $T_{PCLK} \times 4$.

20.2 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 3.10 "Reset" on page 38](#) of this document.

20.3 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 ($T_{PCLK} \times 256 \times 4$) and the maximum watchdog interval is ($T_{PCLK} \times 2^{32} \times 4$) in multiples of ($T_{PCLK} \times 4$). The watchdog should be used in the following manner:

- Set the watchdog timer constant reload value in WDTC register.
- Setup mode in WDMOD register.
- Start 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.

20.4 Register description

The watchdog contains 4 registers as shown in [Table 281](#) below.

Table 281: 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 to its preset value. | WO | NA | 0xE000 0008 |
| WDTV | Watchdog Timer Value register. This register reads out the current value of the Watchdog timer. | RO | 0xFF | 0xE000 000C |

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

20.4.1 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 282: 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. While 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 283: 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 |

20.4.2 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 0xFF to be loaded to the WDTC. Thus the minimum time-out interval is $T_{PCLK} \times 256 \times 4$.

Table 284: Watchdog Timer Constant register (WDTC - address 0xE000 0004) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|-----------------------------|-------------|
| 31:0 | Count | Watchdog time-out interval. | 0x0000 00FF |

20.4.3 Watchdog Feed register (WDFEED - 0xE000 0008)

Writing 0xAA followed by 0x55 to this register will reload the watchdog timer to 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 first be completed before the Watchdog is capable of generating an interrupt/reset. Until then, the watchdog will ignore feed errors. Once 0xAA is written to the WDFEED register the next operation in the Watchdog register space should be a **WRITE** (0x55) to the WDFFED register otherwise the watchdog is triggered. The interrupt/reset will be generated during the second PCLK following an incorrect access to a watchdog timer register during a feed sequence.

Table 285: 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 |

20.4.4 Watchdog Timer Value register (WDTV - 0xE000 000C)

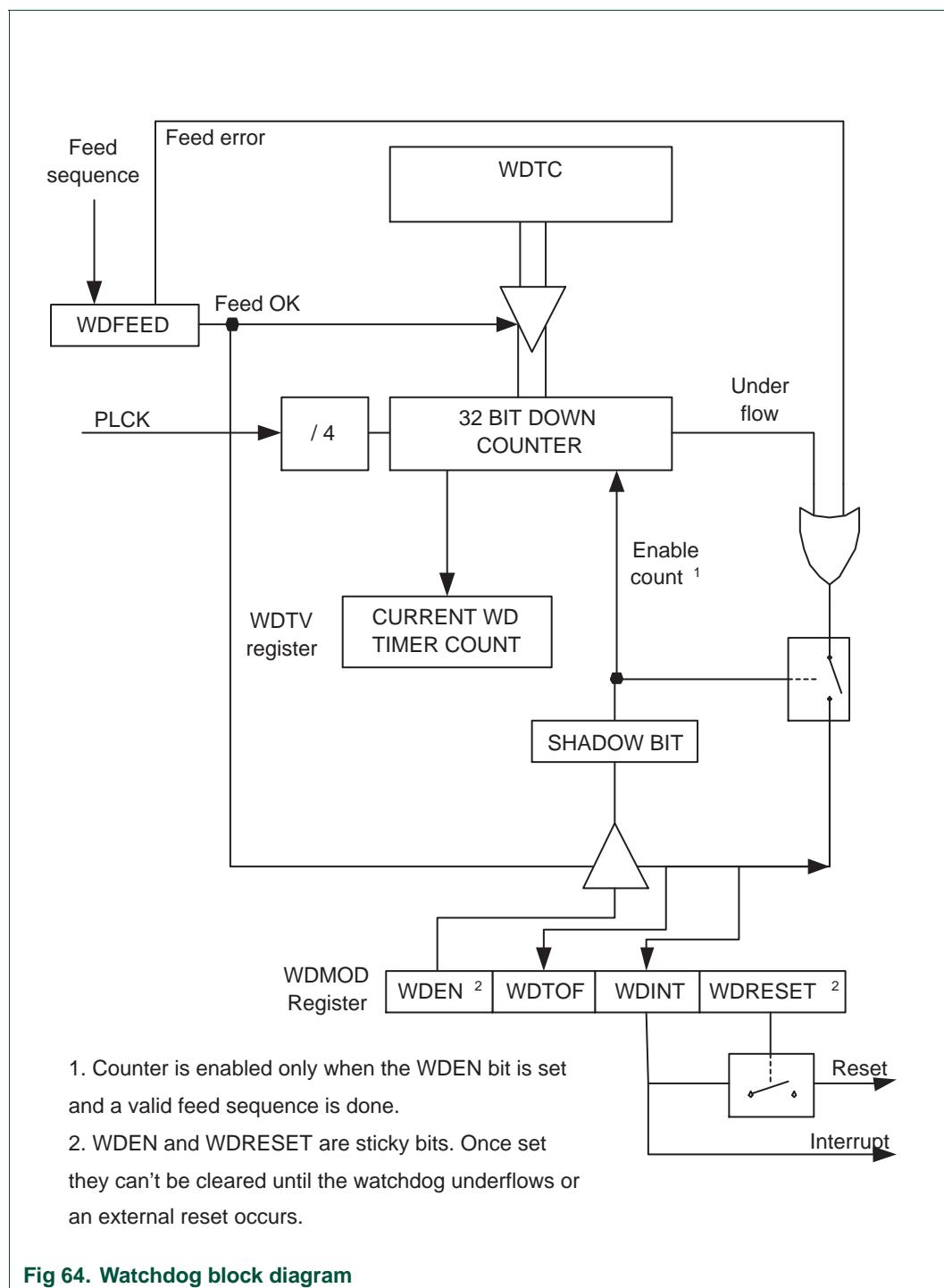
The WDTV register is used to read the current value of watchdog timer.

Table 286: Watchdog Timer Value register (WDTV - address 0xE000 000C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|----------------------|-------------|
| 31:0 | Count | Counter timer value. | 0x0000 00FF |

20.5 Block diagram

The block diagram of the Watchdog is shown below in the [Figure 64](#).





21.1 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.

21.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and a 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.

21.3 Applications

The flash boot loader provides both In-System and In-Application programming interfaces for programming the on-chip flash memory.

21.4 Description

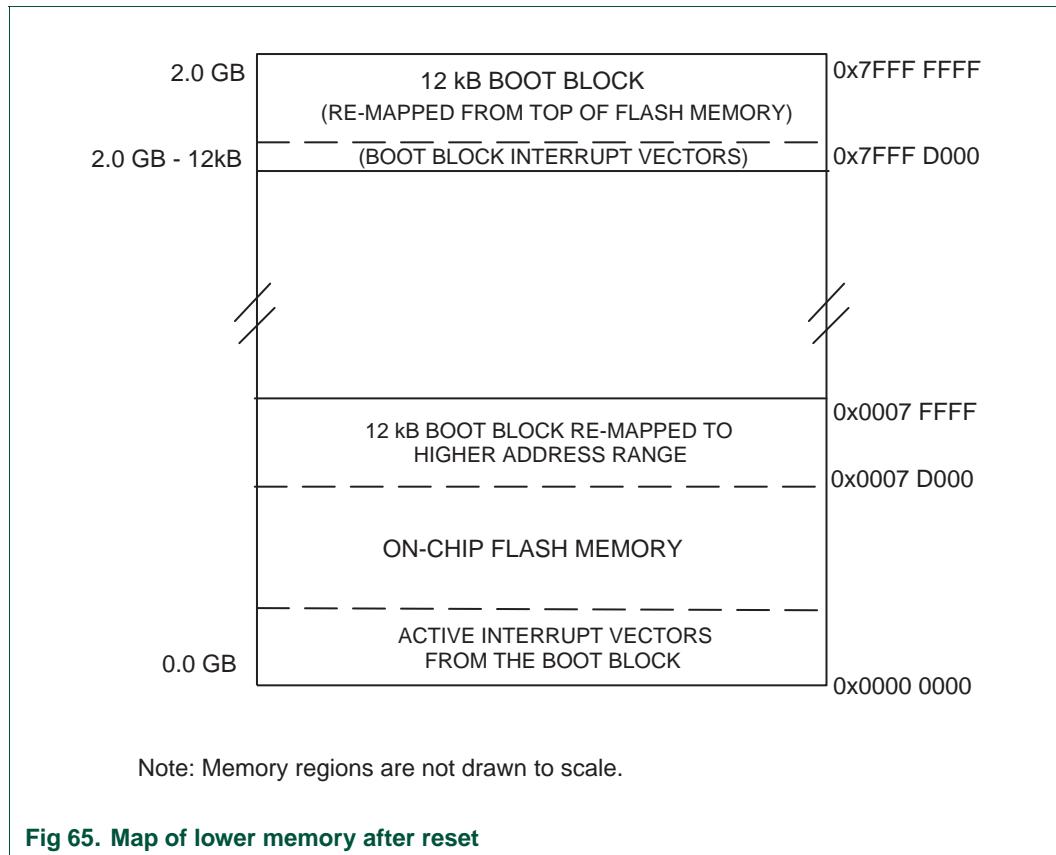
The flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A a LOW level after reset at the P0.14 pin is considered as an external hardware request to start the ISP command handler. Assuming that proper signal is present on X1 pin when the rising edge on RESET pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on whether to continue with user code or ISP handler is made. If P0.14 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 (P0.14 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 P0.14 that is used as hardware request for ISP requires special attention. Since P0.14 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.

21.4.1 Memory map after any reset

The boot block is 12 kB in size and resides in the top portion (starting from 0x0007 D000) 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 D000. 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.



21.4.2 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, the crystal frequency should be greater than or equal to 10 MHz. The on-chip PLL is not used by the boot code.

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 21.8 "ISP commands" on page 297](#).

21.4.3 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.

21.4.4 ISP command format

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

21.4.5 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)

21.4.6 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 <http://www.wotsit.org>.

21.4.7 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.

21.4.8 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.

21.4.9 Interrupts during ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

21.4.10 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.

21.4.11 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.

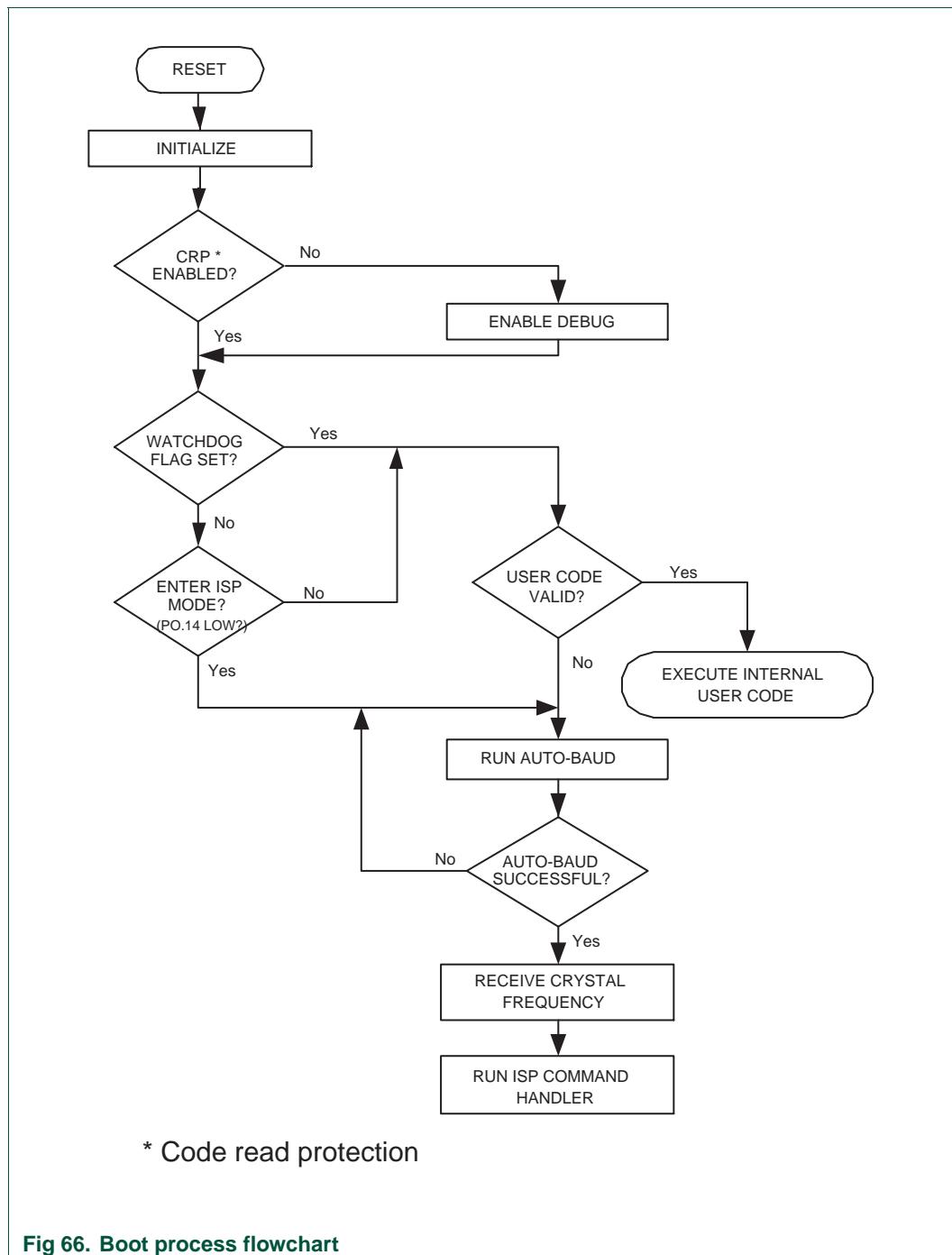
21.4.12 RAM used by IAP command handler

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.

21.4.13 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.

21.4.14 Boot process flowchart



21.5 Sector numbers

Some IAP and ISP commands operate on "sectors" and specify sector numbers. The following table indicate the correspondence between sector numbers and memory addresses for LPC2141/2/4/6/8 devices containing 32, 64, 128, 256 and 512K bytes of Flash respectively. IAP, ISP, and RealMonitor routines are located in the boot block. The

boot block is present at addresses 0x0007 D000 to 0x0007 FFFF in all devices. 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 500 K bytes in "512K" devices. On the other hand, in case of the LPC2141/2/4/6 microcontroller all 32/64/128/256 K of Flash are available for user's application.

Table 287: Flash sectors in LPC2141, LPC2142, LPC2144, LPC2146 and LPC2148

| Sector Number | Sector Size [kB] | Address Range | LPC2141 (32kB) | LPC2142 (64kB) | LPC2144 (128kB) | LPC2146 (256kB) | LPC2148 (512kB) |
|---------------|------------------|---------------------------|----------------|----------------|-----------------|-----------------|-----------------|
| 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 (0xF) | 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 | | | | | + |

21.6 Flash content protection mechanism

The LPC2141/2/4/6/8 is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user's code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 0003 are protected by the first ECC byte, Flash bytes from 0x0000 0004 to 0x0000 0007 are protected by the second ECC byte, etc.

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

When a sector of user's Flash memory is erased, corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the Flash memory in groups of 4 bytes (or multiples of 4), aligned as described above.

21.7 Code Read Protection (CRP)

Code read protection is enabled by programming the flash address location 0x1FC (User flash sector 0) with value 0x8765 4321 (2271560481 Decimal). Address 0x1FC is used to allow some room for the FIQ exception handler. When the code read protection is enabled the JTAG debug port, external memory boot and the following ISP commands are disabled:

- Read Memory
- Write to RAM
- Go
- Copy RAM to Flash

The ISP commands mentioned above terminate with return code CODE_READ_PROTECTION_ENABLED. The ISP erase command only allows erasure of all user sectors when the code read protection is enabled. This limitation does not exist if the code read protection is not enabled. IAP commands are not affected by the code read protection.

Important: CRP is active/inactive once the device has gone through a power cycle.

21.8 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 288: ISP command summary

| ISP Command | Usage | Described in |
|---------------------------------------|---|---------------------------|
| Unlock | U <Unlock Code> | Table 289 |
| Set Baud Rate | B <Baud Rate> <stop bit> | Table 290 |
| Echo | A <setting> | Table 292 |
| Write to RAM | W <start address> <number of bytes> | Table 293 |
| Read Memory | R <address> <number of bytes> | Table 294 |
| Prepare sector(s) for write operation | P <start sector number> <end sector number> | Table 295 |
| Copy RAM to Flash | C <Flash address> <RAM address> <number of bytes> | Table 296 |
| Go | G <address> <Mode> | Table 297 |
| Erase sector(s) | E <start sector number> <end sector number> | Table 298 |
| Blank check sector(s) | I <start sector number> <end sector number> | Table 299 |
| Read Part ID | J | Table 300 |
| Read Boot code version | K | Table 302 |
| Compare | M <address1> <address2> <number of bytes> | Table 303 |

21.8.1 Unlock <unlock code>

Table 289: 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. |

21.8.2 Set Baud Rate <baud rate> <stop bit>

Table 290: 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 291: Correlation between possible ISP baudrates and external crystal frequency (in MHz)

| ISP Baudrate .vs. External Crystal Frequency | 9600 | 19200 | 38400 | 57600 | 115200 | 230400 |
|---|------|-------|-------|-------|--------|--------|
| 10.0000 | + | + | + | | | |
| 11.0592 | + | + | | + | | |
| 12.2880 | + | + | + | | | |
| 14.7456 | + | + | + | + | + | + |
| 15.3600 | + | | | | | |
| 18.4320 | + | + | | + | | |
| 19.6608 | + | + | + | | | |
| 24.5760 | + | + | + | | | |
| 25.0000 | + | + | + | | | |

21.8.3 Echo <setting>

Table 292: 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. |

21.8.4 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 than 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 293: 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. |

21.8.5 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 than 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 294: 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. |

21.8.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 295: 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. |

21.8.7 Copy RAM to Flash <Flash address> <RAM address> <no of bytes>

Table 296: 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. |

21.8.8 Go <address> <mode>

Table 297: 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. |

21.8.9 Erase sector(s) <start sector number> <end sector number>

Table 298: 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. |

21.8.10 Blank check sector(s) <sector number> <end sector number>

Table 299: 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. |

21.8.11 Read Part Identification number

Table 300: ISP Read Part Identification number command

| Command | J |
|-------------|---|
| Input | None. |
| Return Code | CMD_SUCCESS followed by part identification number in ASCII (see Table 301). |
| Description | This command is used to read the part identification number. |

Table 301: LPC214x Part Identification numbers

| Device | ASCII/dec coding | Hex coding |
|---------|------------------|-------------|
| LPC2141 | 196353 | 0x0002 FF01 |
| LPC2142 | 196369 | 0x0002 FF11 |
| LPC2144 | 196370 | 0x0002 FF12 |
| LPC2146 | 196387 | 0x0002 FF23 |
| LPC2148 | 196389 | 0x0002 FF25 |

21.8.12 Read Boot code version number

Table 302: 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. |

21.8.13 Compare <address1> <address2> <no of bytes>

Table 303: 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. |

21.8.14 ISP Return codes

Table 304: 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. |
| 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. |

Table 304: ISP Return codes Summary

| Return Code | Mnemonic | Description |
|--------------------|------------------------------|---|
| 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. |

21.9 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 67](#). 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 0x7fffffff
```

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:

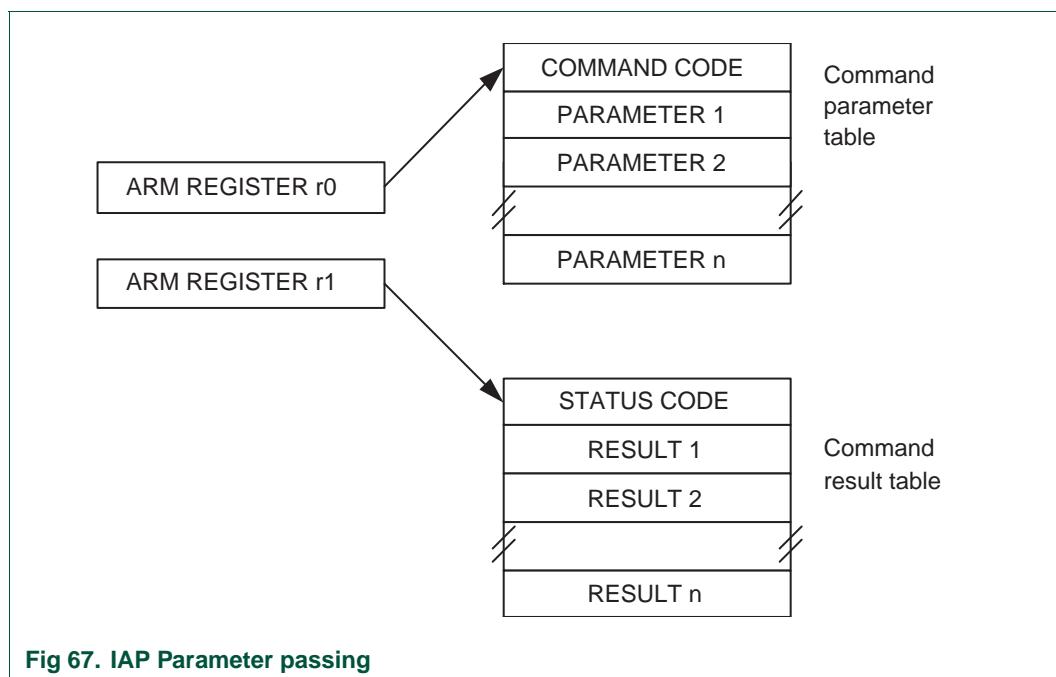
```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffffff90 T rm_init_entry
0x7fffffff90 A rm_undef_handler
0x7fffffff90 A rm_prefetchabort_handler
0x7fffffff90 A rm_dataabort_handler
0x7fffffff90 A rm_irghandler
0x7fffffff90 A rm_irghandler2
0x7fffffff90 T iap_entry
```

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 305: IAP Command Summary

| IAP Command | Command Code | Described in |
|---------------------------------------|------------------|---------------------------|
| Prepare sector(s) for write operation | 50 ₁₀ | Table 306 |
| Copy RAM to Flash | 51 ₁₀ | Table 307 |
| Erase sector(s) | 52 ₁₀ | Table 308 |
| Blank check sector(s) | 53 ₁₀ | Table 309 |
| Read Part ID | 54 ₁₀ | Table 310 |
| Read Boot code version | 55 ₁₀ | Table 311 |
| Compare | 56 ₁₀ | Table 312 |
| Reinvoke ISP | 57 ₁₀ | Table 313 |



21.9.1 Prepare sector(s) for write operation

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

Table 306: 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). |
| 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. |

21.9.2 Copy RAM to Flash

Table 307: 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. |

21.9.3 Erase sector(s)

Table 308: 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. |

21.9.4 Blank check sector(s)

Table 309: 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. |

21.9.5 Read Part Identification number

Table 310: 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 (see Table 301 "LPC214x Part Identification numbers" on page 303 for details) |
| Description | This command is used to read the part identification number. |

21.9.6 Read Boot code version number

Table 311: 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. |

21.9.7 Compare <address1> <address2> <no of bytes>

Table 312: 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. |

21.9.8 Reinvoke ISP

Table 313: Reinvoke ISP

| Command | Compare |
|-------------|--|
| Input | Command code: 57_{10} |
| Return Code | None |
| Result | None. |
| Description | This command is used to invoke the bootloader in ISP mode. This command maps boot vectors, configures P0.1 as an input and sets the VPB divider register to 0 before entering the ISP mode. This command may be used when a valid user program is present in the internal flash memory and the P0.14 pin is not accessible to force the ISP mode. This 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 PLL frequency after autobaud handshake. Another option is to disable the PLL before making this IAP call. Important: TIMER1 registers must be programmed with reset values before "Reinvoke ISP" command is used. |

21.9.9 IAP Status codes

Table 314: 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. |

Table 314: IAP Status codes Summary

| Status Code | Mnemonic | Description |
|-------------|---|---|
| 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. |

21.10 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.



22.1 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 into 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.

22.2 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.

22.3 Description

The ARM7TDMI-S Debug Architecture uses the existing JTAG¹ 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 data bus 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, data bus 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

1. 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.

22.4 Pin description

Table 315: 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 edge triggered clock with the TMS and TCK signals that define the internal state of the device. |
| TDI | Input | Test Data In. This is the serial data input for the shift register. |
| TDO | Output | Test Data Output. This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. |
| 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)". |

22.5 Reset state of multiplexed pins

On the LPC2141/2/4/6/8, the pins above are multiplexed with P1.31-26. To have them come up as a Debug port, connect a weak bias resistor (4.7-10 kΩ depending on the external JTAG circuitry) between V_{SS} and the P1.26/RTCK pin. To have them come up as GPIO pins, do not connect a bias resistor, and ensure that any external driver connected to P1.26/RTCK is either driving high, or is in high-impedance state, during Reset.

22.6 Register description

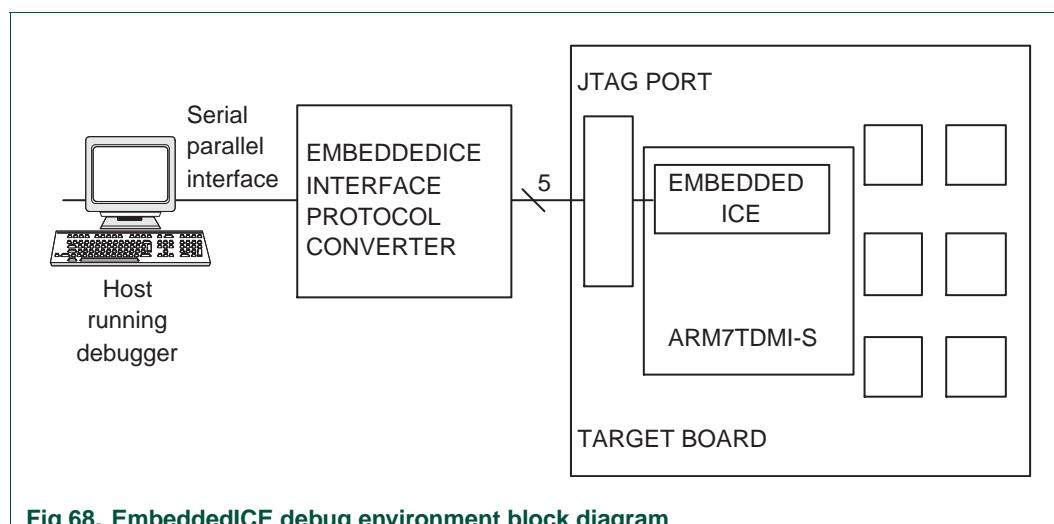
The EmbeddedICE logic contains 16 registers as shown in [Table 316](#) below. The ARM7TDMI-S debug architecture is described in detail in "ARM7TDMI-S (rev 4) Technical Reference Manual" (ARM DDI 0234A) published by ARM Limited and is available via Internet at <http://www.arm.com>.

Table 316: 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 |

22.7 Block diagram

The block diagram of the debug environment is shown below in [Figure 68](#).



23.1 Features

- Closely track the instructions that the ARM core is executing.
- 1 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

23.2 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.

23.3 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.

23.3.1 ETM configuration

The following standard configuration is selected for the ETM macrocell.

Table 317: 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 317: 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)".

23.4 Pin description

Table 318: 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-rate clocking 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 |

23.5 Reset state of multiplexed pins

On the LPC2141/2/4/6/8, the ETM pin functions are multiplexed with P1.25-16. To have these pins come as a Trace port, connect a weak bias resistor ($4.7\text{ k}\Omega$) between the P1.20/TRACESYNC pin and V_{SS} . To have them come up as port pins, do not connect a bias resistor to P1.20/TRACESYNC, and ensure that any external driver connected to P1.20/TRACESYNC is either driving high, or is in high-impedance state, during Reset.

23.6 Register description

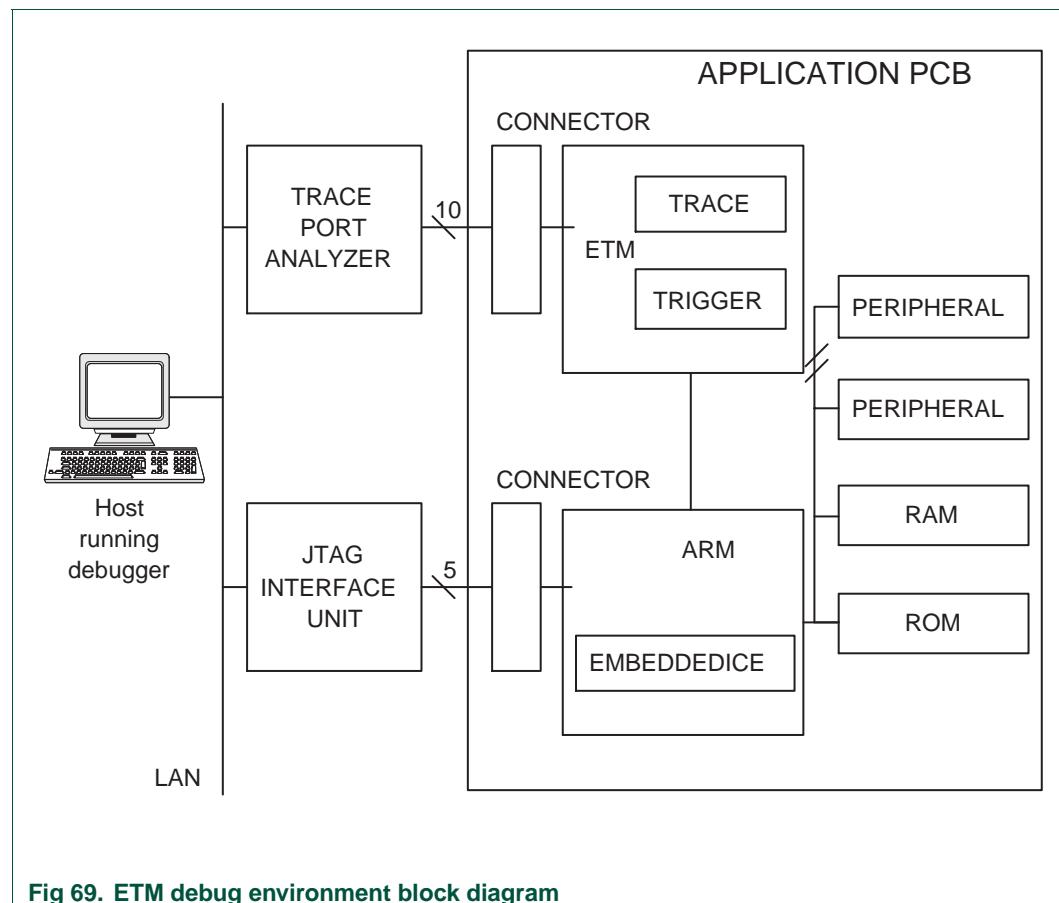
The ETM contains 29 registers as shown in [Table 319](#) below. They are described in detail in the ARM IHI 0014E document published by ARM Limited, which is available via the Internet at <http://www.arm.com>.

Table 319: 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 |

23.7 Block diagram

The block diagram of the ETM debug environment is shown below in [Figure 69](#).



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.

Refer to the white paper "Real Time Debug for System-on-Chip" available at
http://www.arm.com/support/White_Papers?OpenDocument for background information.

24.1 Features

- 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.

24.2 Applications

Real time debugging.

24.3 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 24.4 “How to enable Realmonitor” on page 322](#) for details.

24.3.1 RealMonitor components

As shown in [Figure 70](#), RealMonitor is split in to two functional components:

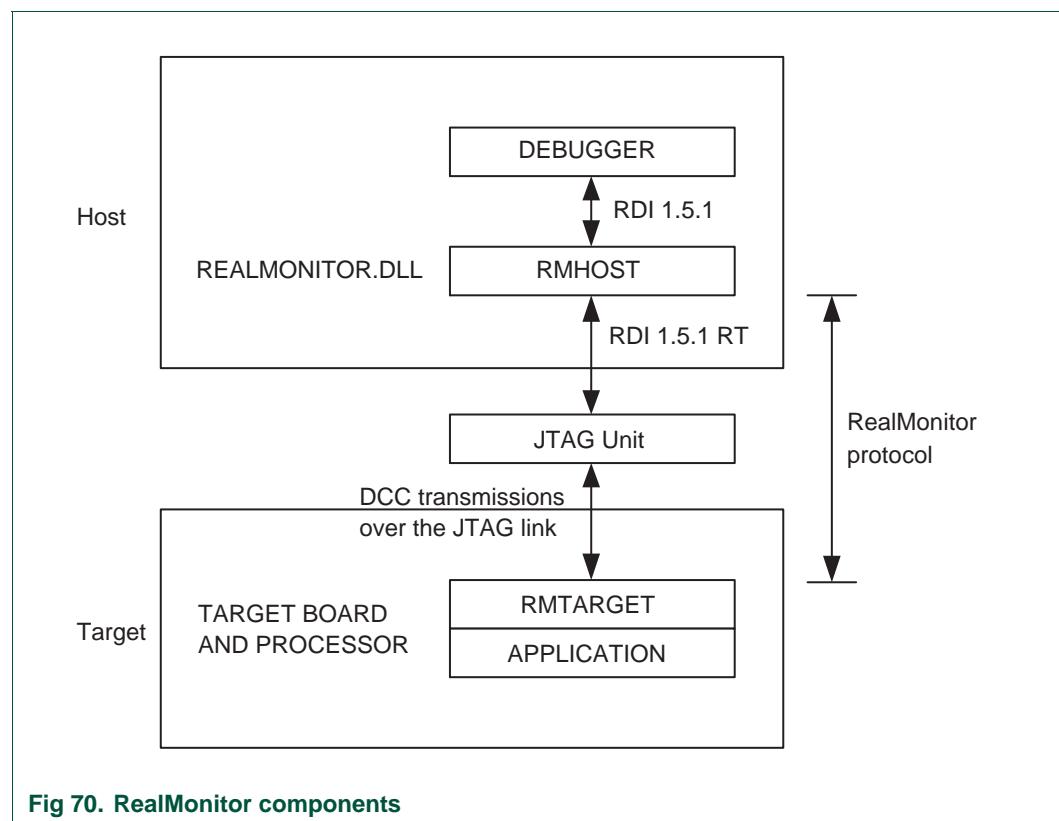


Fig 70. RealMonitor components

24.3.2 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).

24.3.3 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).

24.3.4 How RealMonitor works

In general terms, the RealMonitor operates as a state machine, as shown in [Figure 71](#). RealMonitor switches between running and stopped states, in response to packets received by the host, or due to asynchronous events on the target. RMTTarget 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.

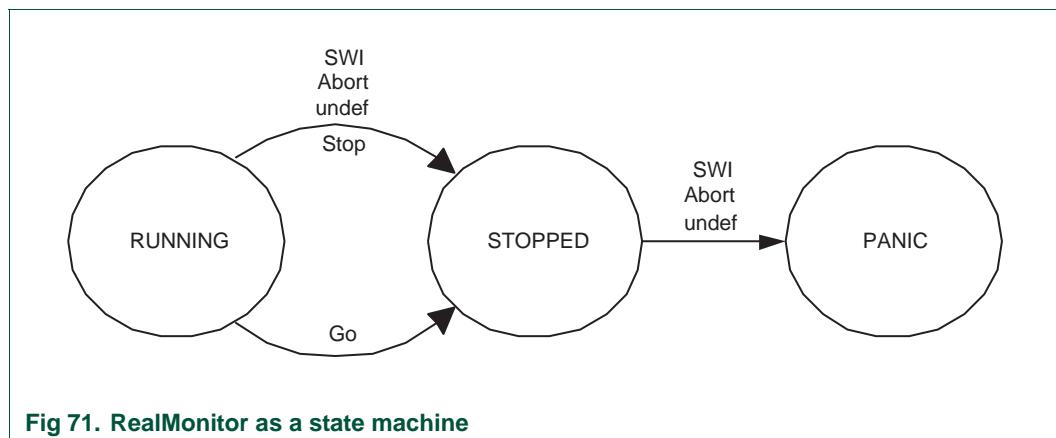


Fig 71. RealMonitor as a state machine

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 70](#).

The target component of RealMonitor, RMTTarget, 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, RMTTarget 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.

24.4 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.

24.4.1 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 320: RealMonitor stack requirement

| Processor Mode | RealMonitor Stack Usage (Bytes) |
|----------------|---------------------------------|
| Undef | 48 |
| Prefetch Abort | 16 |
| Data Abort | 16 |
| IRQ | 8 |

24.4.2 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.

24.4.3 Undef mode

A stack for this mode is always required. RealMonitor uses 12 words while processing an undefined instruction exception.

24.4.4 SVC mode

RealMonitor makes no use of this stack.

24.4.5 Prefetch Abort mode

RealMonitor uses four words on entry to its Prefetch abort interrupt handler.

24.4.6 Data Abort mode

RealMonitor uses four words on entry to its data abort interrupt handler.

24.4.7 User/System mode

RealMonitor makes no use of this stack.

24.4.8 FIQ mode

RealMonitor makes no use of this stack.

24.4.9 Handling exceptions

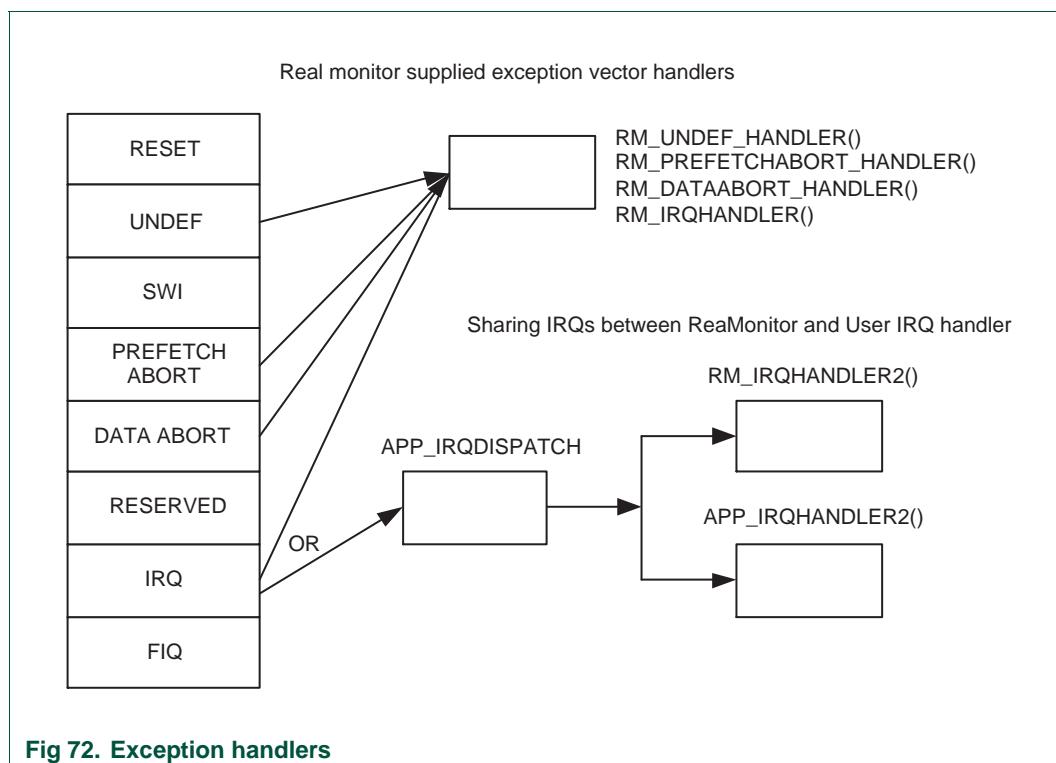
This section describes the importance of sharing exception handlers between RealMonitor and user application.

24.4.10 RealMonitor exception handling

To function properly, RealMonitor must be able to intercept certain interrupts and exceptions. [Figure 72](#) 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.



24.4.11 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().

24.4.12 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, #-0xFF0] ;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]
LDMFD sp!, {r12,r14,r0}      ;Acknowledge Non Vectored irq has finished
SUBS pc, r14, #4             ;Restore registers
                                ;Return to the interrupted instruction
END

```

24.5 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_EXECUTEPCODE=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_EXECUTEPCODE_SIZE=NA

"execute code" buffer size. Also refer to RM_OPT_EXECUTEPCODE 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).



25.1 Abbreviations

Table 321: Abbreviations

| Acronym | Description |
|---------|---|
| ADC | Analog-to-Digital Converter |
| BOD | Brown-Out Detection |
| CPU | Central Processing Unit |
| DAC | Digital-to-Analog Converter |
| DCC | Debug Communications Channel |
| FIFO | First In, First Out |
| GPIO | General Purpose Input/Output |
| NA | Not Applicable |
| PLL | Phase-Locked Loop |
| POR | Power-On Reset |
| PWM | Pulse Width Modulator |
| RAM | Random Access Memory |
| SRAM | Static Random Access Memory |
| UART | Universal Asynchronous Receiver/Transmitter |
| USB | Universal Serial Bus |
| VIC | Vector Interrupt Controller |
| VPB | VLSI Peripheral Bus |

25.2 Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these

products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

25.3 Trademarks

Notice — All referenced brands, product names, service names and trademarks are the property of their respective owners.

I²C-bus — wordmark and logo are trademarks of Koninklijke Philips Electronics N.V.

SoftConnect — is a trademark of Koninklijke Philips Electronics N.V.

GoodLink — is a trademark of Koninklijke Philips Electronics N.V.

25.4 Tables

| | | |
|-----------|--|----|
| Table 1: | LPC2141/2/4/6/8 device information | 4 |
| Table 2: | VPB peripheries and base addresses | 10 |
| Table 3: | ARM exception vector locations | 12 |
| Table 4: | LPC2141/2/4/6/8 memory mapping modes | 12 |
| Table 5: | Pin summary | 16 |
| Table 6: | Summary of system control registers | 17 |
| Table 7: | Recommended values for C _{X1/X2} in oscillation mode (crystal and external components parameters) | 19 |
| Table 8: | External interrupt registers | 20 |
| Table 9: | External Interrupt Flag register (EXTINT - address 0xE01F C140) bit description | 22 |
| Table 10: | Interrupt Wakeup register (INTWAKE - address 0xE01F C144) bit description | 23 |
| Table 11: | External Interrupt Mode register (EXTMODE - address 0xE01F C148) bit description | 23 |
| Table 12: | External Interrupt Polarity register (EXTPOLAR - address 0xE01F C14C) bit description | 24 |
| Table 13: | System Control and Status flags register (SCS - address 0xE01F C1A0) bit description | 26 |
| Table 14: | Memory Mapping control register (MEMMAP - address 0xE01F C040) bit description | 27 |
| Table 15: | PLL registers | 28 |
| Table 16: | PLL Control register (PLL0CON - address 0xE01F C080, PLL1CON - address 0xE01F C0A0) bit description | 30 |
| Table 17: | PLL Configuration register (PLL0CFG - address 0xE01F C084, PLL1CFG - address 0xE01F C0A4) bit description | 30 |
| Table 18: | PLL Status register (PLL0STAT - address 0xE01F C088, PLL1STAT - address 0xE01F C0A8) bit description | 31 |
| Table 19: | PLL Control bit combinations | 32 |
| Table 20: | PLL Feed register (PLL0FEED - address 0xE01F C08C, PLL1FEED - address 0xE01F C0AC) bit description | 32 |
| Table 21: | Elements determining PLL's frequency | 33 |
| Table 22: | PLL Divider values | 34 |
| Table 23: | PLL Multiplier values | 34 |
| Table 24: | Power control registers | 35 |
| Table 25: | Power Control register (PCON - address 0xE01F COCO) bit description | 36 |
| Table 26: | Power Control for Peripherals register (PCONP - address 0xE01F C0C4) bit description | 37 |
| Table 27: | Reset Source identification Register (RSIR - address 0xE01F C180) bit description | 39 |
| Table 28: | VPB divider register map | 40 |
| Table 29: | VPB Divider register (VPBDIV - address | |

| | | |
|--------------|---|----|
| 0xE01F C100) | bit description | 41 |
| Table 30: | MAM Responses to program accesses of various types | 47 |
| Table 31: | MAM responses to data and DMA accesses of various types | 47 |
| Table 32: | Summary of MAM registers | 48 |
| Table 33: | MAM Control Register (MAMCR - address 0xE01F C000) bit description | 48 |
| Table 34: | MAM Timing register (MAMTIM - address 0xE01F C004) bit description | 48 |
| Table 35: | VIC register map | 51 |
| Table 36: | Software Interrupt register (VICSoftInt - address 0xFFFF F018) bit allocation | 52 |
| Table 37: | Software Interrupt register (VICSoftInt - address 0xFFFF F018) bit description | 53 |
| Table 38: | Software Interrupt Clear register (VICSoftIntClear - address 0xFFFF F01C) bit allocation | 53 |
| Table 39: | Software Interrupt Clear register (VICSoftIntClear - address 0xFFFF F01C) bit description | 53 |
| Table 40: | Raw Interrupt status register (VICRawIntr - address 0xFFFF F008) bit allocation | 54 |
| Table 41: | Raw Interrupt status register (VICRawIntr - address 0xFFFF F008) bit description | 54 |
| Table 42: | Interrupt Enable register (VICIntEnable - address 0xFFFF F010) bit allocation | 54 |
| Table 43: | Interrupt Enable register (VICIntEnable - address 0xFFFF F010) bit description | 55 |
| Table 44: | Software Interrupt Clear register (VICIntEnClear - address 0xFFFF F014) bit allocation | 55 |
| Table 45: | Software Interrupt Clear register (VICIntEnClear - address 0xFFFF F014) bit description | 55 |
| Table 46: | Interrupt Select register (VICIntSelect - address 0xFFFF F00C) bit allocation | 55 |
| Table 47: | Interrupt Select register (VICIntSelect - address 0xFFFF F00C) bit description | 56 |
| Table 48: | IRQ Status register (VICIRQStatus - address 0xFFFF F000) bit allocation | 56 |
| Table 49: | IRQ Status register (VICIRQStatus - address 0xFFFF F000) bit description | 56 |
| Table 50: | FIQ Status register (VICFIQStatus - address 0xFFFF F004) bit allocation | 57 |
| Table 51: | FIQ Status register (VICFIQStatus - address 0xFFFF F004) bit description | 57 |
| Table 52: | Vector Control registers 0-15 (VICVectCntl0-15 - 0xFFFF F200-23C) bit description | 57 |
| Table 53: | Vector Address registers (VICVectAddr0-15 - addresses 0xFFFF F100-13C) bit description | 58 |
| Table 54: | Default Vector Address register (VICDefVectAddr | |

continued >>

| | |
|---|----|
| - address 0xFFFF F034) bit description | 58 |
| Table 55: Vector Address register (VICVectAddr - address 0xFFFF F030) bit description | 58 |
| Table 56: Protection Enable register (VICProtection - address 0xFFFF F020) bit description | 58 |
| Table 57: Connection of interrupt sources to the Vectored Interrupt Controller (VIC) | 59 |
| Table 58: Pin description | 69 |
| Table 59: Pin connect block register map. | 75 |
| Table 60: Pin function Select register 0 (PINSEL0 - address 0xE002 C000) bit description | 76 |
| Table 61: Pin function Select register 1 (PINSEL1 - address 0xE002 C004) bit description | 78 |
| Table 62: Pin function Select register 2 (PINSEL2 - 0xE002 C014) bit description | 80 |
| Table 63: Pin function select register bits. | 80 |
| Table 64: GPIO pin description | 81 |
| Table 65: GPIO register map (legacy VPB accessible registers). | 82 |
| Table 66: GPIO register map (local bus accessible registers - enhanced GPIO features). | 83 |
| Table 67: GPIO port 0 Direction register (IO0DIR - address 0xE002 8008) bit description | 83 |
| Table 68: GPIO port 1 Direction register (IO1DIR - address 0xE002 8018) bit description | 84 |
| Table 69: Fast GPIO port 0 Direction register (FIO0DIR - address 0x3FFF C000) bit description | 84 |
| Table 70: Fast GPIO port 1 Direction register (FIO1DIR - address 0x3FFF C020) bit description | 84 |
| Table 71: Fast GPIO port 0 Direction control byte and half-word accessible register description | 84 |
| Table 72: Fast GPIO port 1 Direction control byte and half-word accessible register description | 85 |
| Table 73: Fast GPIO port 0 Mask register (FIO0MASK - address 0x3FFF C010) bit description | 85 |
| Table 74: Fast GPIO port 1 Mask register (FIO1MASK - address 0x3FFF C030) bit description | 85 |
| Table 75: Fast GPIO port 0 Mask byte and half-word accessible register description | 86 |
| Table 76: Fast GPIO port 1 Mask byte and half-word accessible register description | 86 |
| Table 77: GPIO port 0 Pin value register (IO0PIN - address 0xE002 8000) bit description | 87 |
| Table 78: GPIO port 1 Pin value register (IO1PIN - address 0xE002 8010) bit description | 87 |
| Table 79: Fast GPIO port 0 Pin value register (FIO0PIN - address 0x3FFF C014) bit description | 87 |
| Table 80: Fast GPIO port 1 Pin value register (FIO1PIN - address 0x3FFF C034) bit description | 87 |
| Table 81: Fast GPIO port 0 Pin value byte and half-word accessible register description | 88 |

| | |
|--|-----|
| Table 82: Fast GPIO port 1 Pin value byte and half-word accessible register description. | 88 |
| Table 83: GPIO port 0 output Set register (IO0SET - address 0xE002 8004 bit description | 89 |
| Table 84: GPIO port 1 output Set register (IO1SET - address 0xE002 8014) bit description | 89 |
| Table 85: Fast GPIO port 0 output Set register (FIO0SET - address 0x3FFF C018) bit description | 89 |
| Table 86: Fast GPIO port 1 output Set register (FIO1SET - address 0x3FFF C038) bit description | 89 |
| Table 87: Fast GPIO port 0 output Set byte and half-word accessible register description. | 89 |
| Table 88: Fast GPIO port 1 output Set byte and half-word accessible register description. | 90 |
| Table 89: GPIO port 0 output Clear register 0 (IO0CLR - address 0xE002 800C) bit description | 90 |
| Table 90: GPIO port 1 output Clear register 1 (IO1CLR - address 0xE002 801C) bit description | 90 |
| Table 91: Fast GPIO port 0 output Clear register 0 (FIO0CLR - address 0x3FFF C01C) bit description | 90 |
| Table 92: Fast GPIO port 1 output Clear register 1 (FIO1CLR - address 0x3FFF C03C) bit description | 91 |
| Table 93: Fast GPIO port 0 output Clear byte and half-word accessible register description. | 91 |
| Table 94: Fast GPIO port 1 output Clear byte and half-word accessible register description. | 91 |
| Table 95: UART0 pin description | 95 |
| Table 96: UART0 register map | 96 |
| Table 97: UART0 Receiver Buffer Register (U0RBR - address 0xE000 C000, when DLAB = 0, Read Only) bit description | 97 |
| Table 98: UART0 Transmit Holding Register (U0THR - address 0xE000 C000, when DLAB = 0, Write Only) bit description | 97 |
| Table 99: UART0 Divisor Latch LSB register (U0DLL - address 0xE000 C000, when DLAB = 1) bit description | 98 |
| Table 100: UART0 Divisor Latch MSB register (U0DLM - address 0xE000 C004, when DLAB = 1) bit description | 98 |
| Table 101: UART0 Fractional Divider Register (U0FDR - address 0xE000 C028) bit description | 98 |
| Table 102: Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz). | 99 |
| Table 103: UART0 Interrupt Enable Register (U0IER - address 0xE000 C004, when DLAB = 0) bit description | 100 |
| Table 104: UART0 Interrupt Identification Register (U0IIR - address 0xE000 C008, read only) | |

continued >>

| | |
|---|-----|
| bit description | 101 |
| Table 105:UART0 interrupt handling | 102 |
| Table 106:UART0 FIFO Control Register (U0FCR - address 0xE000 C008) bit description | 103 |
| Table 107:UART0 Line Control Register (U0LCR - address 0xE000 C00C) bit description | 103 |
| Table 108:UART0 Line Status Register (U0LSR - address 0xE000 C014, read only) bit description | 104 |
| Table 109:UART0 Scratch pad register (U0SCR - address 0xE000 C01C) bit description | 105 |
| Table 110:Auto-baud Control Register (U0ACR - 0xE000 C020) bit description | 106 |
| Table 111:UART0 Transmit Enable Register (U0TER - address 0xE000 C030) bit description | 107 |
| Table 112:UART1 pin description | 112 |
| Table 113:UART1 register map | 114 |
| Table 114:UART1 Receiver Buffer Register (U1RBR - address 0xE001 0000, when DLAB = 0 Read Only) bit description | 115 |
| Table 115:UART1 Transmitter Holding Register (U1THR - address 0xE001 0000, when DLAB = 0 Write Only) bit description | 115 |
| Table 116:UART1 Divisor Latch LSB register (U1DLL - address 0xE001 0000, when DLAB = 1) bit description | 116 |
| Table 117:UART1 Divisor Latch MSB register (U1DLM - address 0xE001 0004, when DLAB = 1) bit description | 116 |
| Table 118:UART1 Fractional Divider Register (U1FDR - address 0xE001 0028) bit description | 116 |
| Table 119:Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz) | 117 |
| Table 120:UART1 Interrupt Enable Register (U1IER - address 0xE001 0004, when DLAB = 0) bit description | 118 |
| Table 121:UART1 Interrupt Identification Register (U1IIR - address 0xE001 0008, read only) bit description | 119 |
| Table 122:UART1 interrupt handling | 121 |
| Table 123:UART1 FIFO Control Register (U1FCR - address 0xE001 0008) bit description | 122 |
| Table 124:UART1 Line Control Register (U1LCR - address 0xE001 000C) bit description | 122 |
| Table 125:UART1 Modem Control Register (U1MCR - address 0xE001 0010), LPC2144/6/8 only bit description | 123 |
| Table 126:Modem status interrupt generation | 125 |
| Table 127:UART1 Line Status Register (U1LSR - address 0xE001 0014, read only) bit description | 125 |
| Table 128:UART1 Modem Status Register (U1MSR - address 0xE001 0018), LPC2144/6/8 only bit description | 125 |
| description | 127 |
| Table 129:UART1 Scratch pad register (U1SCR - address 0xE001 0014) bit description | 127 |
| Table 130:Auto-baud Control Register (U1ACR - 0xE001 0020) bit description | 128 |
| Table 131:UART1 Transmit Enable Register (U1TER - address 0xE001 0030) bit description | 131 |
| Table 132:I ² C Pin Description | 134 |
| Table 133:I ² C0CONSET and I ² C1CONSET used to configure Master mode | 135 |
| Table 134:I ² C0CONSET and I ² C1CONSET used to configure Slave mode | 136 |
| Table 135:I ² C register map | 142 |
| Table 136:I ² C Control Set register (I2CONSET: I2C0, I2C0CONSET - address 0xE001 C000 and I2C1, I2C1CONSET - address 0xE005 C000) bit description | 143 |
| Table 137:I ² C Control Set register (I2CONCLR: I2C0, I2C0CONCLR - address 0xE001 C018 and I2C1, I2C1CONCLR - address 0xE005 C018) bit description | 144 |
| Table 138:I ² C Status register (I2STAT: I2C0, I2C0STAT - address 0xE001 C004 and I2C1, I2C1STAT - address 0xE005 C004) bit description | 145 |
| Table 139:I ² C Data register (I2DAT: I2C0, I2C0DAT - address 0xE001 C008 and I2C1, I2C1DAT - address 0xE005 C008) bit description | 145 |
| Table 140:I ² C Slave Address register (I2ADR: I2C0, I2C0ADR - address 0xE001 C00C and I2C1, I2C1ADR - address 0xE005 C00C) bit description | 146 |
| Table 141:I ² C SCL High Duty Cycle register (I2SCLH: I2C0, I2C0SCLH - address 0xE001 C010 and I2C1, I2C1SCLH - address 0xE005 C010) bit description | 146 |
| Table 142:I ² C SCL Low Duty Cycle register (I2SCLL: I2C0, I2C0SCLL - address 0xE001 C014 and I2C1, I2C1SCLL - address 0xE005 C014) bit description | 146 |
| Table 143:Example I ² C clock rates | 147 |
| Table 144:Abbreviations used to describe an I ² C operation | 147 |
| Table 145:I ² C0CONSET used to initialize Master Transmitter mode | 148 |
| Table 146:I ² C0ADR and I ² C1ADR usage in Slave Receiver mode | 149 |
| Table 147:I ² C0CONSET and I ² C1CONSET used to initialize Slave Receiver mode | 149 |
| Table 148:Master Transmitter mode | 154 |
| Table 149:Master Receiver mode | 155 |
| Table 150:Slave Receiver mode | 156 |

continued >>

| | | | |
|---|-----|--|-----|
| Table 151: Slave Transmitter mode | 158 | (USBDevIntEn - address 0xE009 0004) bit description | 202 |
| Table 152: Miscellaneous States | 160 | Table 180: USB Device Interrupt Clear register (USBDevIntClr - address 0xE009 0008) bit allocation | 202 |
| Table 153: SPI data to clock phase relationship..... | 172 | Table 181: USB Device Interrupt Clear register (USBDevIntClr - address 0xE009 0008) bit description | 203 |
| Table 154: SPI pin description | 175 | Table 182: USB Device Interrupt Set register (USBDevIntSet - address 0xE009 000C) bit allocation | 203 |
| Table 155: SPI register map..... | 176 | Table 183: USB Device Interrupt Set register (USBDevIntSet - address 0xE009 000C) bit description..... | 203 |
| Table 156: SPI Control Register (S0SPCR - address 0xE002 0000) bit description | 176 | Table 184: USB Device Interrupt Priority register (USBDevIntPri - address 0xE009 002C) bit description | 204 |
| Table 157: SPI Status Register (S0SPSR - address 0xE002 0004) bit description | 177 | Table 185: USB Endpoint Interrupt Status register (USBEplnSt - address 0xE009 0030) bit allocation | 204 |
| Table 158: SPI Data Register (S0SPDR - address 0xE002 0008) bit description | 178 | Table 186: USB Endpoint Interrupt Status register (USBEplnSt - address 0xE009 0030) bit description | 204 |
| Table 159: SPI Clock Counter Register (S0SPCCR - address 0xE002 000C) bit description | 178 | Table 187: USB Endpoint Interrupt Enable register (USBEplnEn - address 0xE009 0034) bit allocation | 205 |
| Table 160: SPI Interrupt register (S0SPINT - address 0xE002 001C) bit description | 179 | Table 188: USB Endpoint Interrupt Enable register (USBEplnEn - address 0xE009 0034) bit description | 206 |
| Table 161: SSP pin descriptions | 180 | Table 189: USB Endpoint Interrupt Clear register (USBEplnClr - address 0xE009 0038) bit allocation | 206 |
| Table 162: SSP register map | 189 | Table 190: USB Endpoint Interrupt Clear register (USBEplnClr - address 0xE009 0038) bit description | 206 |
| Table 163: SSP Control Register 0 (SSPCR0 - address 0xE006 8000) bit description | 189 | Table 191: USB Endpoint Interrupt Set register (USBEplnSet - address 0xE009 003C) bit allocation | 207 |
| Table 164: SSP Control Register 1 (SSPCR1 - address 0xE006 8004) bit description | 190 | Table 192: USB Endpoint Interrupt Set register (USBEplnSet - address 0xE009 003C) bit description..... | 207 |
| Table 165: SSP Data Register (SSPDR - address 0xE006 8008) bit description | 191 | Table 193: USB Endpoint Interrupt Priority register (USBEplnPri - address 0xE009 0040) bit allocation | 207 |
| Table 166: SSP Status Register (SSPDR - address 0xE006 800C) bit description | 191 | Table 194: USB Endpoint Interrupt Priority register (USBEplnPri - address 0xE009 0040) bit description | 208 |
| Table 167: SSP Clock Prescale Register (SSPCPSR - address 0xE006 8010) bit description | 191 | Table 195: USB Realize Endpoint register (USBReEp - address 0xE009 0044) bit allocation | 208 |
| Table 168: SSP Interrupt Mask Set/Clear register (SSPIMSC - address 0xE006 8014) bit description | 192 | Table 196: USB Realize Endpoint register (USBReEp - address 0xE009 0044) bit description | 208 |
| Table 169: SSP Raw Interrupt Status register (SSPRIS - address 0xE006 8018) bit description | 192 | Table 197: USB Endpoint Index register (USBEpln - address 0xE009 0048) bit description | 210 |
| Table 170: SSP Masked Interrupt Status register (SSPMIS -address 0xE006 801C) bit description | 193 | Table 198: USB MaxPacketSize register (USBMaxPSIZE - address 0xE009 004C) bit description | 210 |
| Table 171: SSP interrupt Clear Register (SSPICR - address 0xE006 8020) bit description | 193 | | |
| Table 172: USB related acronyms, abbreviations and definitions used in this chapter | 194 | | |
| Table 173: Pre-Fixed Endpoint Configuration..... | 195 | | |
| Table 174: USB device register map | 199 | | |
| Table 175: USB Interrupt Status register (USBIntSt - address 0xE01F C1C0) bit description..... | 200 | | |
| Table 176: USB Device Interrupt Status register (USBDevIntSt - address 0xE009 0000) bit allocation | 201 | | |
| Table 177: USB Device Interrupt Status register (USBDevIntSt - address 0xE009 0000) bit description | 201 | | |
| Table 178: USB Device Interrupt Enable register (USBDevIntEn - address 0xE009 0004) bit allocation | 202 | | |
| Table 179: USB Device Interrupt Enable register | 202 | | |

continued >>

| | |
|--|-----|
| Table 199:USB Receive Data register (USBRxData - address 0xE009 0018) bit description | 211 |
| Table 200:USB Receive Packet Length register (USBRxPlen - address 0xE009 0020) bit description | 211 |
| Table 201:USB Transmit Data register (USBTxDData - address 0xE009 001C) bit description | 211 |
| Table 202:USB Transmit Packet Length register (USBTxPLen - address 0xE009 0024) bit description | 212 |
| Table 203:USB Control register (USBCtrl - address 0xE009 0028) bit description | 212 |
| Table 204:USB Command Code register (USBCmdCode - address 0xE009 0010) bit description | 213 |
| Table 205:USB Command Data register (USBCmdData - address 0xE009 0014) bit description | 214 |
| Table 206:USB DMA Request Status register (USBDMARSt - address 0xE009 0050) bit allocation | 214 |
| Table 207:USB DMA Request Status register (USBDMARSt - address 0xE009 0050) bit description | 214 |
| Table 208:USB DMA Request Clear register (USBDMARClr - address 0xE009 0054) bit description | 215 |
| Table 209:USB DMA Request Set register (USBDMARSet - address 0xE009 0058) bit description | 215 |
| Table 210:USB UDCA Head register (USBUDCAH - address 0xE009 0080) bit description | 216 |
| Table 211:USB EP DMA Status register (USBEpDMASt - address 0xE009 0084) bit description | 217 |
| Table 212:USB EP DMA Enable register (USBEpDMAEn - address 0xE009 0088) bit description | 217 |
| Table 213:USB EP DMA Disable register (USBEpDMADis - address 0xE009 008C) bit description | 218 |
| Table 214:USB DMA Interrupt Status register (USBDMAIntSt - address 0xE009 0090) bit description | 218 |
| Table 215:USB DMA Interrupt Enable register (USBDMAIntEn - address 0xE009 0094) bit description | 219 |
| Table 216:USB End of Transfer Interrupt Status register (USBEoTIntSt - address 0xE009 00A0s) bit description | 219 |
| Table 217:USB End of Transfer Interrupt Clear register (USBEoTIntClr - address 0xE009 00A4) bit description | 219 |
| Table 218:USB End of Transfer Interrupt Set register (USBEoTIntSet - address 0xE009 00A8) bit description | 220 |
| Table 219:USB New DD Request Interrupt Status register (USBNDDRIntSt - address 0xE009 00AC) bit description | 220 |
| Table 220:USB New DD Request Interrupt Clear register (USBNDDRIntClr - address 0xE009 00B0) bit description | 220 |
| Table 221:USB New DD Request Interrupt Set register (USBNDDRIntSet - address 0xE009 00B4) bit description | 221 |
| Table 222:USB System Error Interrupt Status register (USBSysErrIntSt - address 0xE009 00B8) bit description | 221 |
| Table 223:USB System Error Interrupt Clear register (USBSysErrIntClr - address 0xE009 00BC) bit description | 221 |
| Table 224:USB System Error Interrupt Set register (USBSysErrIntSet - address 0xE009 00C0) bit description | 222 |
| Table 225:Protocol engine command code table | 222 |
| Table 226:Device Set Address Register bit description | 223 |
| Table 227:Configure Device Register bit description | 224 |
| Table 228:Set Mode Register bit description | 224 |
| Table 229:Set Device Status Register bit description | 225 |
| Table 230:Get Error Code Register bit description | 227 |
| Table 231:Read Error Status Register bit description | 227 |
| Table 232:Select Endpoint Register bit description | 228 |
| Table 233:Set Endpoint Status Register bit description | 229 |
| Table 234:Clear Buffer Register bit description | 230 |
| Table 235:DMA descriptor | 231 |
| Table 236:Timer/Counter pin description | 243 |
| Table 237:TIMER/COUNTER0 and TIMER/COUNTER1 register map | 244 |
| Table 238:Interrupt Register (IR, TIMER0: T0IR - address 0xE000 4000 and TIMER1: T1IR - address 0xE000 8000) bit description | 245 |
| Table 239:Timer Control Register (TCR, TIMER0: T0TCR - address 0xE000 4004 and TIMER1: T1TCR - address 0xE000 8004) bit description | 246 |
| Table 240:Count Control Register (CTCR, TIMER0: T0CTCR - address 0xE000 4070 and TIMER1: T1CTCR - address 0xE000 8070) bit description | 246 |
| Table 241:Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description | 248 |
| Table 242:Capture Control Register (CCR, TIMER0: T0CCR - address 0xE000 4028 and TIMER1: T1CCR - address 0xE000 8028) bit description | 249 |
| Table 243:External Match Register (EMR, TIMER0: T0EMR - address 0xE000 403C and TIMER1: T1EMR - address 0xE000 803C) bit description | 250 |
| Table 244:External match control | 251 |
| Table 245:Set and reset inputs for PWM Flip-Flops | 256 |
| Table 246:Pin summary | 257 |
| Table 247:Pulse Width Modulator (PWM) register map | 258 |
| Table 248:PWM Interrupt Register (PWMIR - address | |

continued >>

| | |
|--|-----|
| 0xE001 4000) bit description | 259 |
| Table 249: PWM Timer Control Register (PWMTCR - address 0xE001 4004) bit description | 260 |
| Table 250: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description | 261 |
| Table 251: PWM Control Register (PWMPCR - address 0xE001 404C) bit description | 262 |
| Table 252: PWM Latch Enable Register (PWMLER - address 0xE001 4050) bit description | 264 |
| Table 253: ADC pin description | 265 |
| Table 254: ADC registers | 266 |
| Table 255: A/D Control Register (AD0CR - address 0xE003 4000 and AD1CR - address 0xE006 0000) bit description | 267 |
| Table 256: A/D Global Data Register (AD0GDR - address 0xE003 4004 and AD1GDR - address 0xE006 0004) bit description | 268 |
| Table 257: A/D Global Start Register (ADGSR - address 0xE003 4008) bit description | 269 |
| Table 258: A/D Status Register (ADSTAT, ADC0: AD0STAT - address 0xE003 4004 and ADC1: AD1STAT - address 0xE006 0004) bit description | 270 |
| Table 259: A/D Status Register (ADSTAT, ADC0: AD0STAT - address 0xE003 4004 and ADC1: AD1STAT - address 0xE006 0004) bit description | 270 |
| Table 260: A/D Data Registers (ADDR0 to ADDR7, ADC0: AD0DR0 to AD0DR7 - 0xE003 4010 to 0xE003 402C and ADC1: AD1DR0 to AD1DR7- 0xE006 0010 to 0xE006 402C) bit description | 271 |
| Table 261: DAC pin description | 273 |
| Table 262: DAC Register (DACR - address 0xE006 C000) bit description | 273 |
| Table 263: Real Time Clock (RTC) register map | 276 |
| Table 264: Miscellaneous registers | 277 |
| Table 265: Interrupt Location Register (ILR - address 0xE002 4000) bit description | 278 |
| Table 266: Clock Tick Counter Register (CTCR - address 0xE002 4004) bit description | 278 |
| Table 267: Clock Control Register (CCR - address 0xE002 4008) bit description | 278 |
| Table 268: Counter Increment Interrupt Register (CIIR - address 0xE002 400C) bit description | 279 |
| Table 269: Alarm Mask Register (AMR - address 0xE002 4010) bit description | 279 |
| Table 270: Consolidated Time register 0 (CTIME0 - address 0xE002 4014) bit description | 280 |
| Table 271: Consolidated Time register 1 (CTIME1 - address 0xE002 4018) bit description | 280 |
| Table 272: Consolidated Time register 2 (CTIME2 - address 0xE002 401C) bit description | 280 |
| Table 273: Time counter relationships and values | 281 |
| Table 274: Time counter registers | 281 |
| Table 275: Alarm registers | 282 |
| Table 276: Reference clock divider registers | 283 |
| Table 277: Prescaler Integer register (PREINT - address 0xE002 4080) bit description | 283 |
| Table 278: Prescaler Integer register (PREFRAC - address 0xE002 4084) bit description | 283 |
| Table 279: Prescaler cases where the Integer Counter reload value is incremented | 285 |
| Table 280: Recommended values for the RTC external 32 kHz oscillator C _{X1/X2} components | 286 |
| Table 281: Watchdog register map | 288 |
| Table 282: Watchdog operating modes selection | 288 |
| Table 283: Watchdog Mode register (WDMOD - address 0xE000 0000) bit description | 289 |
| Table 284: Watchdog Timer Constant register (WDTC - address 0xE000 0004) bit description | 289 |
| Table 285: Watchdog Feed register (WDFEED - address 0xE000 0008) bit description | 289 |
| Table 286: Watchdog Timer Value register (WDTV - address 0xE000 000C) bit description | 289 |
| Table 287: Flash sectors in LPC2141, LPC2142, LPC2144, LPC2146 and LPC2148 | 296 |
| Table 288: ISP command summary | 298 |
| Table 289: ISP Unlock command | 298 |
| Table 290: ISP Set Baud Rate command | 298 |
| Table 291: Correlation between possible ISP baudrates and external crystal frequency (in MHz) | 299 |
| Table 292: ISP Echo command | 299 |
| Table 293: ISP Write to RAM command | 300 |
| Table 294: ISP Read memory command | 300 |
| Table 295: ISP Prepare sector(s) for write operation command | 301 |
| Table 296: ISP Copy command | 301 |
| Table 297: ISP Go command | 302 |
| Table 298: ISP Erase sector command | 302 |
| Table 299: ISP Blank check sector command | 303 |
| Table 300: ISP Read Part Identification number command | 303 |
| Table 301: LPC214x Part Identification numbers | 303 |
| Table 302: ISP Read Boot code version number command | 303 |
| Table 303: ISP Compare command | 304 |
| Table 304: ISP Return codes Summary | 304 |
| Table 305: IAP Command Summary | 306 |
| Table 306: IAP Prepare sector(s) for write operation command | 307 |
| Table 307: IAP Copy RAM to Flash command | 308 |
| Table 308: IAP Erase sector(s) command | 308 |
| Table 309: IAP Blank check sector(s) command | 309 |

continued >>

| | |
|--|-----|
| Table 310:IAP Read Part Identification command..... | 309 |
| Table 311:IAP Read Boot code version number command | 309 |
| Table 312:IAP Compare command | 310 |
| Table 313:Reinvoke ISP | 310 |
| Table 314:IAP Status codes Summary | 310 |
| Table 315:EmbeddedICE pin description | 313 |
| Table 316:EmbeddedICE logic registers..... | 314 |
| Table 317:ETM configuration..... | 315 |
| Table 318:ETM pin description | 316 |
| Table 319:ETM registers | 317 |
| Table 320:RealMonitor stack requirement..... | 322 |
| Table 321:Abbreviations | 329 |

continued >>

25.5 Figures

| | | |
|---------|--|-----|
| Fig 1. | LPC2141/2/4/6/8 block diagram..... | 7 |
| Fig 2. | System memory map..... | 8 |
| Fig 3. | Peripheral memory map..... | 9 |
| Fig 4. | AHB peripheral map | 10 |
| Fig 5. | Map of lower memory is showing re-mapped and re-mappable areas (LPC2148 with 512 kB Flash) | 14 |
| Fig 6. | Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for C _{x1} /x ₂ evaluation | 19 |
| Fig 7. | F _{Osc} selection algorithm | 20 |
| Fig 8. | External interrupt logic..... | 25 |
| Fig 9. | PLL block diagram | 29 |
| Fig 10. | Reset block diagram including the wakeup timer. | 39 |
| Fig 11. | VPB divider connections | 41 |
| Fig 12. | Simplified block diagram of the Memory Accelerator Module (MAM) | 45 |
| Fig 13. | Block diagram of the Vectored Interrupt Controller (VIC) | 60 |
| Fig 14. | LPC2141 64-pin package | 66 |
| Fig 15. | LPC2142 64-pin package | 67 |
| Fig 16. | LPC2144/6/8 64-pin package | 68 |
| Fig 17. | Illustration of the fast and slow GPIO access and output showing 3.5 x increase of the pin output frequency | 94 |
| Fig 18. | Autobaud Mode 0 and Mode 1 waveform | 109 |
| Fig 19. | UART0 block diagram | 111 |
| Fig 20. | Auto-RTS functional timing | 124 |
| Fig 21. | Auto-CTS functional timing | 125 |
| Fig 22. | Autobaud Mode 0 and Mode 1 waveform | 130 |
| Fig 23. | UART1 block diagram | 132 |
| Fig 24. | I ² C-bus Configuration..... | 134 |
| Fig 25. | Format in the Master Transmitter mode..... | 135 |
| Fig 26. | Format of Master Receive mode | 136 |
| Fig 27. | A Master Receiver switches to Master Transmitter after sending Repeated START..... | 136 |
| Fig 28. | Format of Slave Receiver mode..... | 137 |
| Fig 29. | Format of Slave Transmitter mode..... | 137 |
| Fig 30. | I ² C serial interface block diagram | 139 |
| Fig 31. | Arbitration procedure | 140 |
| Fig 32. | Serial clock synchronization..... | 141 |
| Fig 33. | Format and States in the Master Transmitter mode | 150 |
| Fig 34. | Format and States in the Master Receiver mode | 151 |
| Fig 35. | Format and States in the Slave Receiver mode. | 152 |
| Fig 36. | Format and States in the Slave Transmitter mode | 153 |
| Fig 37. | Simultaneous repeated START conditions from two masters | 162 |
| Fig 38. | Forced access to a busy I ² C-bus | 162 |
| Fig 39. | Recovering from a bus obstruction caused by a low level on SDA | 162 |
| Fig 40. | SPI data transfer format (CPHA = 0 and CPHA = 1) | 172 |
| Fig 41. | SPI block diagram | 179 |
| Fig 42. | Texas Instruments synchronous serial frame format: a) single and b) continuous/back-to-back two frames transfer | 182 |
| Fig 43. | SPI frame format with CPOL=0 and CPHA=0 (a) single and b) continuous transfer)..... | 183 |
| Fig 44. | SPI frame format with CPOL=0 and CPHA=1.. | 184 |
| Fig 45. | SPI frame format with CPOL = 1 and CPHA = 0 (a) single and b) continuous transfer)..... | 185 |
| Fig 46. | SPI frame format with CPOL = 1 and CPHA = 1 | 186 |
| Fig 47. | Microwire frame format (single transfer) | 187 |
| Fig 48. | Microwire frame format (continuos transfers) | 188 |
| Fig 49. | Microwire frame format (continuos transfers) - details | 188 |
| Fig 50. | USB Device Controller Block Diagram | 196 |
| Fig 51. | USB MaxPacket register array indexing | 210 |
| Fig 52. | UDCA Head register and DMA descriptors. | 216 |
| Fig 53. | Finding the DMA descriptor..... | 235 |
| Fig 54. | Data transfer in ATLE mode | 237 |
| Fig 55. | Isochronous OUT Endpoint operation example | 241 |
| Fig 56. | A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled | 251 |
| Fig 57. | A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled | 251 |
| Fig 58. | Timer block diagram | 252 |
| Fig 59. | PWM block diagram | 255 |
| Fig 60. | Sample PWM waveforms | 256 |
| Fig 61. | RTC block diagram | 275 |
| Fig 62. | RTC prescaler block diagram | 284 |
| Fig 63. | RTC 32kHz crystal oscillator circuit..... | 286 |
| Fig 64. | Watchdog block diagram..... | 290 |
| Fig 65. | Map of lower memory after reset | 292 |
| Fig 66. | Boot process flowchart | 295 |
| Fig 67. | IAP Parameter passing | 307 |
| Fig 68. | EmbeddedICE debug environment block diagram | 314 |
| Fig 69. | ETM debug environment block diagram | 318 |
| Fig 70. | RealMonitor components | 320 |
| Fig 71. | RealMonitor as a state machine | 321 |
| Fig 72. | Exception handlers | 324 |

continued >>

25.6 Contents

Chapter 1: General information

| | | | | | |
|-----|------------------------------|---|-----|-----------------------------------|---|
| 1.1 | Introduction | 3 | 1.6 | ARM7TDMI-S processor | 5 |
| 1.2 | Features | 3 | 1.7 | On-chip Flash memory system | 6 |
| 1.3 | Applications | 4 | 1.8 | On-chip Static RAM (SRAM)..... | 6 |
| 1.4 | Device information..... | 4 | 1.9 | Block diagram | 7 |
| 1.5 | Architectural overview | 4 | | | |

Chapter 2: LPC2141/2/4/6/8 Memory Addressing

| | | | | | |
|-----|--|----|-------|---|----|
| 2.1 | Memory maps..... | 8 | 2.2.1 | Memory map concepts and operating modes | 11 |
| 2.2 | LPC2141/2142/2144/2146/2148 memory re-mapping and boot block..... | 11 | 2.2.2 | Memory re-mapping..... | 12 |
| | | | 2.3 | Prefetch abort and data abort exceptions .. | 15 |

Chapter 3: System Control Block

| | | | | | |
|-------|---|----|--------|---|----|
| 3.1 | Summary of system control block functions | 16 | 3.8.5 | PLL Interrupt | 31 |
| 3.2 | Pin description..... | 16 | 3.8.6 | PLL Modes..... | 31 |
| 3.3 | Register description | 17 | 3.8.7 | PLL Feed register (PLL0FEED - 0xE01F C08C, PLL1FEED - 0xE01F C0AC) | 32 |
| 3.4 | Crystal oscillator | 18 | 3.8.8 | PLL and Power-down mode..... | 32 |
| 3.5 | External interrupt inputs..... | 20 | 3.8.9 | PLL frequency calculation | 33 |
| 3.5.1 | Register description | 20 | 3.8.10 | Procedure for determining PLL settings..... | 33 |
| 3.5.2 | External Interrupt Flag register (EXTINT - 0xE01F C140) | 21 | 3.8.11 | PLL0 and PLL1 configuring examples | 34 |
| 3.5.3 | Interrupt Wakeup register (INTWAKE - 0xE01F C144) | 22 | 3.9 | Power control. | 35 |
| 3.5.4 | External Interrupt Mode register (EXTMODE - 0xE01F C148) | 23 | 3.9.1 | Register description | 35 |
| 3.5.5 | External Interrupt Polarity register (EXTPOLAR - 0xE01F C14C) | 24 | 3.9.2 | Power Control register (PCON - 0xE01F COCO)..... | 35 |
| 3.5.6 | Multiple external interrupt pins | 25 | 3.9.3 | Power Control for Peripherals register (PCONP - 0xE01F COC4) | 36 |
| 3.6 | Other system controls. | 26 | 3.9.4 | Power control usage notes..... | 38 |
| 3.6.1 | System Control and Status flags register (SCS - 0xE01F C1A0) | 26 | 3.10 | Reset | 38 |
| 3.7 | Memory mapping control | 26 | 3.10.1 | Reset Source Identification Register (RSIR - 0xE01F C180) | 39 |
| 3.7.1 | Memory Mapping control register (MEMMAP - 0xE01F C040) | 26 | 3.11 | VPB divider | 40 |
| 3.7.2 | Memory mapping control usage notes | 27 | 3.11.1 | Register description | 40 |
| 3.8 | Phase Locked Loop (PLL). | 27 | 3.11.2 | VPBDIV register (VPBDIV - 0xE01F C100) .. | 40 |
| 3.8.1 | Register description | 28 | 3.12 | Wakeup timer | 41 |
| 3.8.2 | PLL Control register (PLL0CON - 0xE01F C080, PLL1CON - 0xE01F C0A0) | 29 | 3.13 | Brown-out detection | 42 |
| 3.8.3 | PLL Configuration register (PLL0CFG - 0xE01F C084, PLL1CFG - 0xE01F C0A4) .. | 30 | 3.14 | Code security vs. debugging | 43 |
| 3.8.4 | PLL Status register (PLL0STAT - 0xE01F C088, PLL1STAT - 0xE01F C0A8) | 31 | | | |

continued >>

Chapter 4: Memory Acceleration Module (MAM)

| | | | | | |
|------------|--------------------------------------|-----------|------------|---|-----------|
| 4.1 | Introduction | 44 | 4.5 | MAM configuration | 47 |
| 4.2 | Operation | 44 | 4.6 | Register description | 47 |
| 4.3 | MAM blocks | 45 | 4.7 | MAM Control Register (MAMCR - 0xE01F C000) | 48 |
| 4.3.1 | Flash memory bank | 45 | 4.8 | MAM Timing register (MAMTIM - 0xE01F C004) | 48 |
| 4.3.2 | Instruction latches and data latches | 46 | | | |
| 4.3.3 | Flash programming Issues | 46 | | | |
| 4.4 | MAM operating modes | 46 | 4.9 | MAM usage notes | 49 |

Chapter 5: Vectored Interrupt Controller (VIC)

| | | | | | |
|------------|---|-----------|---------------|---|-----------|
| 5.1 | Features | 50 | 5.4.10 | Vector Address registers 0-15 (VICVectAddr0-15 - 0xFFFF F100-13C) | 58 |
| 5.2 | Description | 50 | 5.4.11 | Default Vector Address register (VICDefVectAddr - 0xFFFF F034) | 58 |
| 5.3 | Register description | 50 | 5.4.12 | Vector Address register (VICVectAddr - 0xFFFF F030) | 58 |
| 5.4 | VIC registers | 52 | 5.4.13 | Protection Enable register (VICProtection - 0xFFFF F020) | 58 |
| 5.4.1 | Software Interrupt register (VICSoftInt - 0xFFFF F018) | 52 | 5.5 | Interrupt sources | 59 |
| 5.4.2 | Software Interrupt Clear register (VICSoftIntClear - 0xFFFF F01C) | 53 | 5.6 | Spurious interrupts | 61 |
| 5.4.3 | Raw Interrupt status register (VICRawIntr - 0xFFFF F008) | 54 | 5.6.1 | Details and case studies on spurious interrupts | 61 |
| 5.4.4 | Interrupt Enable register (VICIntEnable - 0xFFFF F010) | 54 | 5.6.2 | Workaround | 62 |
| 5.4.5 | Interrupt Enable Clear register (VICIntEnClear - 0xFFFF F014) | 55 | 5.6.3 | Solution 1: test for an IRQ received during a write to disable IRQs | 62 |
| 5.4.6 | Interrupt Select register (VICIntSelect - 0xFFFF F00C) | 55 | 5.6.4 | Solution 2: disable IRQs and FIQs using separate writes to the CPSR | 63 |
| 5.4.7 | IRQ Status register (VICIRQStatus - 0xFFFF F000) | 56 | 5.6.5 | Solution 3: re-enable FIQs at the beginning of the IRQ handler | 63 |
| 5.4.8 | FIQ Status register (VICFIQStatus - 0xFFFF F004) | 57 | 5.7 | VIC usage notes | 63 |
| 5.4.9 | Vector Control registers 0-15 (VICVectCntl0-15 - 0xFFFF F200-23C) | 57 | | | |

Chapter 6: Pin configuration

| | | | | | |
|------------|---|-----------|------------|--|-----------|
| 6.1 | LPC2141/2142/2144/2146/2148 pinout | 66 | 6.2 | Pin description for LPC2141/2/4/6/8 | 68 |
|------------|---|-----------|------------|--|-----------|

Chapter 7: Pin Connect Block

| | | | | | |
|------------|--|-----------|--------------|--|----|
| 7.1 | Features | 75 | 7.4.3 | Pin function Select register 2 (PINSEL2 - 0xE002 C014) | 79 |
| 7.2 | Applications | 75 | 7.4.4 | Pin function select register values | 80 |
| 7.3 | Description | 75 | | | |
| 7.4 | Register description | 75 | | | |
| 7.4.1 | Pin Function Select Register 0 (PINSEL0 - 0xE002 C000) | 76 | | | |
| 7.4.2 | Pin function Select register 1 (PINSEL1 - 0xE002 C004) | 77 | | | |

continued >>

Chapter 8: General Purpose Input/Output ports (GPIO)

| | | | | |
|---|-----------|--------------|---|-----------|
| 8.1 Features | 81 | 8.4.4 | GPIO port output Set register (IOSET, Port 0: IO0SET - 0xE002 8004 and Port 1: IO1SET - 0xE002 8014; FIOSET, Port 0: FIO0SET - 0x3FFF C018 and Port 1: FIO1SET - 0x3FFF C038) | 88 |
| 8.2 Applications | 81 | 8.4.5 | GPIO port output Clear register (IOCLR, Port 0: IO0CLR - 0xE002 800C and Port 1: IO1CLR - 0xE002 801C; FIOCLR, Port 0: FIO0CLR - 0x3FFF C01C and Port 1: FIO1CLR - 0x3FFF C03C) | 90 |
| 8.3 Pin description..... | 81 | 8.5 | GPIO usage notes..... | 92 |
| 8.4 Register description | 81 | 8.5.1 | Example 1: sequential accesses to IOSET and IOCLR affecting the same GPIO pin/bit | 92 |
| 8.4.1 GPIO port Direction register (IODIR, Port 0: IO0DIR - 0xE002 8008 and Port 1: IO1DIR - 0xE002 8018; FIODIR, Port 0: FIO0DIR - 0x3FFF C000 and Port 1:FIO1DIR - 0x3FFF C020) | 83 | 8.5.2 | Example 2: an immediate output of 0s and 1s on a GPIO port | 92 |
| 8.4.2 Fast GPIO port Mask register (FIOMASK, Port 0: FIO0MASK - 0x3FFF C010 and Port 1:FIO1MASK - 0x3FFF C030) | 85 | 8.5.3 | Writing to IOSET/IOCLR .vs. IOPIN..... | 93 |
| 8.4.3 GPIO port Pin value register (IOPIN, Port 0: IO0PIN - 0xE002 8000 and Port 1: IO1PIN - 0xE002 8010; FIOPIN, Port 0: FIO0PIN - 0x3FFF C014 and Port 1: FIO1PIN - 0x3FFF C034) | 86 | 8.5.4 | Output signal frequency considerations when using the legacy and enhanced GPIO registers .. | 93 |

Chapter 9: Universal Asynchronous Receiver/Transmitter 0 (UART0)

| | | | | |
|---|------------|---------------|---|------------|
| 9.1 Features | 95 | 9.3.7 | UART0 Interrupt Identification Register (U0IIR - 0xE000 C008, Read Only)..... | 101 |
| 9.2 Pin description..... | 95 | 9.3.8 | UART0 FIFO Control Register (U0FCR - 0xE000 C008) | 103 |
| 9.3 Register description | 95 | 9.3.9 | UART0 Line Control Register (U0LCR - 0xE000 C00C) | 103 |
| 9.3.1 UART0 Receiver Buffer Register (U0RBR - 0xE000 C000, when DLAB = 0, Read Only) .. | 97 | 9.3.10 | UART0 Line Status Register (U0LSR - 0xE000 C014, Read Only)..... | 104 |
| 9.3.2 UART0 Transmit Holding Register (U0THR - 0xE000 C000, when DLAB = 0, Write Only) .. | 97 | 9.3.11 | UART0 Scratch pad register (U0SCR - 0xE000 C01C) | 105 |
| 9.3.3 UART0 Divisor Latch Registers (U0DLL - 0xE000 C000 and U0DLM - 0xE000 C004, when DLAB = 1) | 97 | 9.3.12 | UART0 Auto-baud Control Register (U0ACR - 0xE000 C020) | 106 |
| 9.3.4 UART0 Fractional Divider Register (U0FDR - 0xE000 C028)..... | 98 | 9.3.13 | Auto-baud | 106 |
| 9.3.5 UART0 baudrate calculation..... | 99 | 9.3.14 | UART0 Transmit Enable Register (U0TER - 0xE000 C030) | 107 |
| 9.3.6 UART0 Interrupt Enable Register (U0IER - 0xE000 C004, when DLAB = 0) | 100 | 9.3.15 | Auto-baud Modes..... | 108 |
| | | 9.4 | Architecture..... | 109 |

Chapter 10: Universal Asynchronous Receiver/Transmitter 1 (UART1)

| | | | | |
|--|------------|---------------|--|------------|
| 10.1 Features | 112 | 10.3.2 | UART1 Transmitter Holding Register (U1THR - 0xE001 0000, when DLAB = 0 Write Only) .. | 115 |
| 10.2 Pin description..... | 112 | 10.3.3 | UART1 Divisor Latch Registers 0 and 1 (U1DLL - 0xE001 0000 and U1DLM - 0xE001 0004, when DLAB = 1) | 115 |
| 10.3 Register description | 113 | | | |
| 10.3.1 UART1 Receiver Buffer Register (U1RBR - 0xE001 0000, when DLAB = 0 Read Only) .. | 115 | | | |

continued >>

| | | | | | |
|---------|--|-----|-------------|---|------------|
| 10.3.4 | UART1 Fractional Divider Register (U1FDR - 0xE001 0028) | 116 | 10.3.11 | UART1 Line Status Register (U1LSR - 0xE001 0014, Read Only) | 125 |
| 10.3.5 | UART1 baudrate calculation | 117 | 10.3.12 | UART1 Modem Status Register (U1MSR - 0xE001 0018), LPC2144/6/8 only | 127 |
| 10.3.6 | UART1 Interrupt Enable Register (U1IER - 0xE001 0004, when DLAB = 0) | 118 | 10.3.13 | UART1 Scratch pad register (U1SCR - 0xE001 001C) | 127 |
| 10.3.7 | UART1 Interrupt Identification Register (U1IIR - 0xE001 0008, Read Only) | 119 | 10.3.14 | UART1 Auto-baud Control Register (U1ACR - 0xE001 0020) | 127 |
| 10.3.8 | UART1 FIFO Control Register (U1FCR - 0xE001 0008) | 121 | 10.3.15 | Auto-baud | 128 |
| 10.3.9 | UART1 Line Control Register (U1LCR - 0xE001 000C) | 122 | 10.3.16 | Auto-baud Modes | 129 |
| 10.3.10 | UART1 Modem Control Register (U1MCR - 0xE001 0010), LPC2144/6/8 only | 123 | 10.3.17 | UART1 Transmit Enable Register (U1TER - 0xE001 0030) | 130 |
| | | | 10.4 | Architecture | 131 |

Chapter 11: I²C interfaces I²C0 and I²C1

| | | | | | |
|-------------|---|------------|---------------|--|------------|
| 11.1 | Features | 133 | 11.7.5 | I ² C Slave Address register (I2ADR: I2C0, I2C0ADR - 0xE001 C00C and I2C1, I2C1ADR - address 0xE005 C00C) | 146 |
| 11.2 | Applications | 133 | 11.7.6 | I ² C SCL High duty cycle register (I2SCLH: I2C0, I2C0SCLH - 0xE001 C010 and I2C1, I2C1SCLH - 0xE0015 C010) | 146 |
| 11.3 | Description | 133 | 11.7.7 | I ² C SCL Low duty cycle register (I2SCLL: I2C0 - I2C0SCLL: 0xE001 C014; I2C1 - I2C1SCLL: 0xE0015 C014) | 146 |
| 11.4 | Pin description | 134 | 11.7.8 | Selecting the appropriate I ² C data rate and duty cycle | 146 |
| 11.5 | I²C operating modes | 134 | 11.8 | Details of I²C operating modes | 147 |
| 11.5.1 | Master Transmitter mode | 134 | 11.8.1 | Master Transmitter mode | 148 |
| 11.5.2 | Master Receiver mode | 135 | 11.8.2 | Master Receiver mode | 148 |
| 11.5.3 | Slave Receiver mode | 136 | 11.8.3 | Slave Receiver mode | 149 |
| 11.5.4 | Slave Transmitter mode | 137 | 11.8.4 | Slave Transmitter mode | 153 |
| 11.6 | I²C Implementation and operation | 138 | 11.8.5 | Miscellaneous States | 159 |
| 11.6.1 | Input filters and output stages | 138 | 11.8.6 | I2STAT = 0xF8 | 159 |
| 11.6.2 | Address Register, I2ADDR | 140 | 11.8.7 | I2STAT = 0x00 | 159 |
| 11.6.3 | Comparator | 140 | 11.8.8 | Some special cases | 160 |
| 11.6.4 | Shift register, I2DAT | 140 | 11.8.9 | Simultaneous repeated START conditions from two masters | 160 |
| 11.6.5 | Arbitration and synchronization logic | 140 | 11.8.10 | Data transfer after loss of arbitration | 160 |
| 11.6.6 | Serial clock generator | 141 | 11.8.11 | Forced access to the I ² C-bus | 160 |
| 11.6.7 | Timing and control | 141 | 11.8.12 | I ² C-bus obstructed by a low level on SCL or SDA | 161 |
| 11.6.8 | Control register, I2CONSET and I2CONCLR | 141 | 11.8.13 | Bus error | 161 |
| 11.6.9 | Status decoder and Status register | 142 | 11.8.14 | I ² C State service routines | 162 |
| 11.7 | Register description | 142 | 11.8.15 | Initialization | 163 |
| 11.7.1 | I ² C Control Set register (I2CONSET: I2C0, I2C0CONSET - 0xE001 C000 and I2C1, I2C1CONSET - 0xE005 C000) | 143 | 11.8.16 | I ² C interrupt service | 163 |
| 11.7.2 | I ² C Control Clear register (I2CONCLR: I2C0, I2C0CONCLR - 0xE001 C018 and I2C1, I2C1CONCLR - 0xE005 C018) | 144 | 11.8.17 | The State service routines | 163 |
| 11.7.3 | I ² C Status register (I2STAT: I2C0, I2C0STAT - 0xE001 C004 and I2C1, I2C1STAT - 0xE005 C004) | 145 | 11.8.18 | Adapting State services to an application | 163 |
| 11.7.4 | I ² C Data register (I2DAT: I2C0, I2C0DAT - 0xE001 C008 and I2C1, I2C1DAT - 0xE005 C008) | 145 | 11.9 | Software example | 163 |
| | | | 11.9.1 | Initialization routine | 163 |

continued >>

| | | | | | |
|---------|--|-----|---------|--------------------------------|-----|
| 11.9.2 | Start Master Transmit function | 163 | 11.9.20 | State: 0x58..... | 167 |
| 11.9.3 | Start Master Receive function | 164 | 11.9.21 | Slave Receiver States | 167 |
| 11.9.4 | I ² C interrupt routine | 164 | 11.9.22 | State: 0x60..... | 167 |
| 11.9.5 | Non mode specific States..... | 164 | 11.9.23 | State: 0x68..... | 167 |
| 11.9.6 | State: 0x00 | 164 | 11.9.24 | State: 0x70..... | 168 |
| 11.9.7 | Master States | 164 | 11.9.25 | State: 0x78..... | 168 |
| 11.9.8 | State: 0x08 | 164 | 11.9.26 | State: 0x80..... | 168 |
| 11.9.9 | State: 0x10 | 165 | 11.9.27 | State: 0x88..... | 168 |
| 11.9.10 | Master Transmitter States..... | 165 | 11.9.28 | State: 0x90..... | 169 |
| 11.9.11 | State: 0x18 | 165 | 11.9.29 | State: 0x98..... | 169 |
| 11.9.12 | State: 0x20 | 165 | 11.9.30 | State: 0xA0 | 169 |
| 11.9.13 | State: 0x28 | 165 | 11.9.31 | Slave Transmitter States | 169 |
| 11.9.14 | State: 0x30 | 166 | 11.9.32 | State: 0xA8 | 169 |
| 11.9.15 | State: 0x38 | 166 | 11.9.33 | State: 0xB0 | 169 |
| 11.9.16 | Master Receive States | 166 | 11.9.34 | State: 0xB8 | 170 |
| 11.9.17 | State: 0x40 | 166 | 11.9.35 | State: 0xC0 | 170 |
| 11.9.18 | State: 0x48 | 166 | 11.9.36 | State: 0xC8 | 170 |
| 11.9.19 | State: 0x50 | 167 | | | |

Chapter 12: SPI Interface (SPI0)

| | | | | | |
|---------|---------------------------|-----|--------|---|-----|
| 12.1 | Features | 171 | 12.3 | Pin description | 175 |
| 12.2 | Description | 171 | 12.4 | Register description | 175 |
| 12.2.1 | SPI overview..... | 171 | 12.4.1 | SPI Control Register (S0SPCR - 0xE002 0000) | 176 |
| 12.2.2 | SPI data transfers..... | 171 | 12.4.2 | SPI Status Register (S0SPSR - 0xE002 0004) | 177 |
| 12.2.3 | General information | 173 | 12.4.3 | SPI Data Register (S0SPDR - 0xE002 0008) | 178 |
| 12.2.4 | Master operation..... | 173 | 12.4.4 | SPI Clock Counter Register (S0SPCCR - 0xE002 000C) | 178 |
| 12.2.5 | Slave operation..... | 174 | 12.4.5 | SPI Interrupt register (S0SPINT - 0xE002 001C). | 178 |
| 12.2.6 | Exception conditions..... | 174 | | | |
| 12.2.7 | Read Overrun | 174 | | | |
| 12.2.8 | Write Collision..... | 174 | | | |
| 12.2.9 | Mode Fault | 175 | | | |
| 12.2.10 | Slave Abort..... | 175 | 12.5 | Architecture | 179 |

Chapter 13: SSP Controller (SPI1)

| | | | | | |
|--------|--|-----|--------|--|-----|
| 13.1 | Features | 180 | 13.3.9 | Setup and hold time requirements on CS with respect to SK in Microwire mode | 188 |
| 13.2 | Description | 180 | 13.4 | Register description | 188 |
| 13.3 | Bus description | 181 | 13.4.1 | SSP Control Register 0 (SSPCR0 - 0xE006 8000) | 189 |
| 13.3.1 | Texas Instruments Synchronous Serial (SSI) frame format | 181 | 13.4.2 | SSP Control Register 1 (SSPCR1 - 0xE006 8004) | 190 |
| 13.3.2 | SPI frame format | 182 | 13.4.3 | SSP Data Register (SSPDR - 0xE006 8008) | 191 |
| 13.3.3 | Clock Polarity (CPOL) and Clock Phase (CPHA) control | 182 | 13.4.4 | SSP Status Register (SSPSR - 0xE006 800C) | 191 |
| 13.3.4 | SPI format with CPOL=0,CPHA=0 | 183 | 13.4.5 | SSP Clock Prescale Register (SSPCPSR - 0xE006 8010). | 191 |
| 13.3.5 | SPI format with CPOL=0,CPHA=1 | 184 | 13.4.6 | SSP Interrupt Mask Set/Clear register (SSPIMSC - 0xE006 8014) | 192 |
| 13.3.6 | SPI format with CPOL = 1,CPHA = 0 | 185 | | | |
| 13.3.7 | SPI format with CPOL = 1,CPHA = 1 | 186 | | | |
| 13.3.8 | Semiconductor Microwire frame format | 186 | | | |

continued >>

| | | | | | |
|--------|--|-----|--------|---|-----|
| 13.4.7 | SSP Raw Interrupt Status register (SSPRIS - 0xE006 8018) | 192 | 13.4.9 | SSP Interrupt Clear Register (SSPICR - 0xE006 8020) | 193 |
| 13.4.8 | SSP Masked Interrupt register (SSPMIS - 0xE006 801C) | 193 | | | |

Chapter 14: USB Device Controller

| | | | | | |
|-------------|--|------------|----------------|--|------------|
| 14.1 | Introduction | 194 | 14.8.3 | USB Receive Data register (USBRxData - 0xE009 0018) | 211 |
| 14.2 | Features | 195 | 14.8.4 | USB Receive Packet Length register (USBRxPLen - 0xE009 0020) | 211 |
| 14.3 | Fixed Endpoint Configuration | 195 | 14.8.5 | USB Transmit Data register (USBTxData - 0xE009 001C) | 211 |
| 14.4 | Architecture | 196 | 14.8.6 | USB Transmit Packet Length register (USBTxPLen - 0xE009 0024) | 211 |
| 14.5 | Data Flow | 197 | 14.8.7 | USB Control register (USBCtrl - 0xE009 0028) | 212 |
| 14.5.1 | Data Flow from USB Host to the Device | 197 | 14.8.8 | Slave Mode data transfer | 212 |
| 14.5.2 | Data Flow from Device to the Host | 197 | 14.8.9 | USB Command Code register (USBCmdCode - 0xE009 0010) | 213 |
| 14.5.3 | Slave Mode Transfer | 197 | 14.8.10 | USB Command Data register (USBCmdData - 0xE009 0014) | 213 |
| 14.5.4 | DMA Mode Transfer (LPC2146/8 only) | 198 | 14.8.11 | USB DMA Request Status register (USBDMARSt - 0xE009 0050) | 214 |
| 14.6 | Interfaces | 198 | 14.8.12 | USB DMA Request Clear register (USBDMARClr - 0xE009 0054) | 214 |
| 14.6.1 | Software Interface | 198 | 14.8.13 | USB DMA Request Set register (USBDMARSet - 0xE009 0058) | 215 |
| 14.6.2 | Register Map | 198 | 14.8.14 | USB UDCA Head register (USBUDCAH - 0xE009 0080) | 216 |
| 14.7 | USB Device register definitions | 200 | 14.8.15 | USB EP DMA Status register (USBEpDMASt - 0xE009 0084) | 217 |
| 14.7.1 | USB Interrupt Status register (USBIntSt - 0xE01F C1C0) | 200 | 14.8.16 | USB EP DMA Enable register (USBEpDMAEn - 0xE009 0088) | 217 |
| 14.7.2 | USB Device Interrupt Status register (USBDevIntSt - 0xE009 0000) | 201 | 14.8.17 | USB EP DMA Disable register (USBEpDMADis - 0xE009 008C) | 217 |
| 14.7.3 | USB Device Interrupt Enable register (USBDevIntEn - 0xE009 0004) | 202 | 14.8.18 | USB DMA Interrupt Status register (USBDMAIntSt - 0xE009 0090) | 218 |
| 14.7.4 | USB Device Interrupt Clear register (USBDevIntClr - 0xE009 0008) | 202 | 14.8.19 | USB DMA Interrupt Enable register (USBDMAIntEn - 0xE009 0094) | 218 |
| 14.7.5 | USB Device Interrupt Set register (USBDevIntSet - 0xE009 000C) | 203 | 14.8.20 | USB End of Transfer Interrupt Status register (USBEoTIntSt - 0xE009 00A0) | 219 |
| 14.7.6 | USB Device Interrupt Priority register (USBDevIntPri - 0xE009 002C) | 203 | 14.8.21 | USB End of Transfer Interrupt Clear register (USBEoTIntClr - 0xE009 00A4) | 219 |
| 14.7.7 | USB Endpoint Interrupt Status register (USBEpIntSt - 0xE009 0030) | 204 | 14.8.22 | USB End of Transfer Interrupt Set register (USBEoTIntSet - 0xE009 00A8) | 220 |
| 14.7.8 | USB Endpoint Interrupt Enable register (USBEpIntEn - 0xE009 0034) | 205 | 14.8.23 | USB New DD Request Interrupt Status register (USBNDDRIntSt - 0xE009 00AC) | 220 |
| 14.7.9 | USB Endpoint Interrupt Clear register (USBEpIntClr - 0xE009 0038) | 206 | 14.8.24 | USB New DD Request Interrupt Clear register (USBNDDRIntClr - 0xE009 00B0) | 220 |
| 14.7.10 | USB Endpoint Interrupt Set register (USBEpIntSet - 0xE009 003C) | 207 | | | |
| 14.7.11 | USB Endpoint Interrupt Priority register (USBEpIntPri - 0xE009 0040) | 207 | | | |
| 14.7.12 | USB Realize Endpoint register (USBReEp - 0xE009 0044) | 208 | | | |
| 14.8 | EP_RAM requirements | 209 | | | |
| 14.8.1 | USB Endpoint Index register (USBEpIn - 0xE009 0048) | 210 | | | |
| 14.8.2 | USB MaxPacketSize register (USBMaxPSize - 0xE009 004C) | 210 | | | |

continued >>

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|--|------------|--------------|--|------------|---------|--|-----|--------------|--|------------|--------------|--|------------|--------------|--|------------|--------------|----------------------------------|------------|---------|--|------------|---------|---------------------------------------|-----|---------|--------------------------------------|-----|---------|---------------------------------------|-----|---------|--|------------|---------|---|------------|---------|--|------------|---------|---------------------------------------|-----|---------|--------------------------------------|-----|---------|--|------------|---------|--------------------------------------|-----|---------|---|------------|---------|--------------------------------------|-----|---------|--|-----|---------|---------------------------------|-----|---------|--|-----|
| 14.8.25 | USB New DD Request Interrupt Set register (USBNDDRIntSet - 0xE009 00B4) | 220 | 14.10.1 | Next_DD_pointer | 231 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.8.26 | USB System Error Interrupt Status register (USBSysErrIntSt - 0xE009 00B8) | 221 | 14.10.2 | DMA_mode | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.8.27 | USB System Error Interrupt Clear register (USBSysErrIntClr - 0xE009 00BC) | 221 | 14.10.3 | Next_DD_valid | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.8.28 | USB System Error Interrupt Set register (USBSysErrIntSet - 0xE009 00C0) | 221 | 14.10.4 | Isochronous_endpoint | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9 | Protocol engine command description | 222 | 14.10.5 | Max_packet_size | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.1 | Set Address (Command: 0xD0, Data: write 1 byte) | 223 | 14.10.6 | DMA_buffer_length | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.2 | Configure Device (Command: 0xD8, Data: write 1 byte) | 223 | 14.10.7 | DMA_buffer_start_addr | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.3 | Set Mode (Command: 0xF3, Data: write 1 byte) | 224 | 14.10.8 | DD_retired | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.4 | Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes) | 225 | 14.10.9 | DD_status | 232 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.5 | Read Test Register (Command: 0xFD, Data: read 2 bytes) | 225 | 14.10.10 | Packet_valid | 233 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.6 | Set Device Status (Command: 0xFE, Data: write 1 byte) | 225 | 14.10.11 | LS_byte_extracted | 233 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.7 | Get Device Status (Command: 0xFE, Data: read 1 byte) | 226 | 14.10.12 | MS_byte_extracted | 233 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.8 | Get Error Code (Command: 0xFF, Data: read 1 byte) | 226 | 14.10.13 | Present_DMA_count | 233 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.9 | Read Error Status (Command: 0xFB, Data: read 1 byte) | 227 | 14.10.14 | Message_length_position | 233 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.10 | Select Endpoint (Command: 0x00 - 0x1F, Data: read 1 byte (optional)) | 228 | 14.10.15 | Isochronous_packetsize_memory_address | 233 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.11 | Select Endpoint/Clear Interrupt (Command: 0x40 - 0x5F, Data: read 1 byte) | 229 | 14.11 | DMA operation | 234 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.12 | Set Endpoint Status (Command: 0x40 - 0x55, Data: write 1 byte (optional)) | 229 | 14.11.1 | Triggering the DMA engine | 234 | 14.9.13 | Clear Buffer (Command: 0xF2, Data: read 1 byte (optional)) | 230 | 14.11.2 | Arbitration between endpoints | 234 | 14.9.14 | Validate Buffer (Command: 0xFA, Data: none) | 230 | 14.12 | Non Isochronous Endpoints - Normal Mode operation | 234 | 14.10 | DMA descriptor | 230 | 14.12.1 | Setting up DMA transfer | 234 | 14.12.2 | Finding DMA Descriptor | 234 | 14.12.3 | Transferring the Data | 235 | 14.12.4 | Optimizing Descriptor Fetch | 235 | 14.12.5 | Ending the packet transfer | 236 | 14.12.6 | No_Packet DD | 236 | 14.13 | Concatenated transfer (ATLE) mode operation | 236 | 14.13.1 | Setting up the DMA transfer | 239 | 14.13.2 | Finding the DMA Descriptor | 239 | 14.13.3 | Transferring the Data | 239 | 14.13.4 | Ending the packet transfer | 239 | 14.14 | Isochronous Endpoint Operation | 240 | 14.14.1 | Setting up of DMA transfer | 240 | 14.14.2 | Finding the DMA Descriptor | 240 | 14.14.3 | Transferring the Data | 240 | 14.14.4 | Isochronous OUT Endpoint Operation Example | 241 |
| 14.11.1 | Triggering the DMA engine | 234 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.13 | Clear Buffer (Command: 0xF2, Data: read 1 byte (optional)) | 230 | 14.11.2 | Arbitration between endpoints | 234 | 14.9.14 | Validate Buffer (Command: 0xFA, Data: none) | 230 | 14.12 | Non Isochronous Endpoints - Normal Mode operation | 234 | 14.10 | DMA descriptor | 230 | 14.12.1 | Setting up DMA transfer | 234 | 14.12.2 | Finding DMA Descriptor | 234 | 14.12.3 | Transferring the Data | 235 | 14.12.4 | Optimizing Descriptor Fetch | 235 | 14.12.5 | Ending the packet transfer | 236 | 14.12.6 | No_Packet DD | 236 | 14.13 | Concatenated transfer (ATLE) mode operation | 236 | 14.13.1 | Setting up the DMA transfer | 239 | 14.13.2 | Finding the DMA Descriptor | 239 | 14.13.3 | Transferring the Data | 239 | 14.13.4 | Ending the packet transfer | 239 | 14.14 | Isochronous Endpoint Operation | 240 | 14.14.1 | Setting up of DMA transfer | 240 | 14.14.2 | Finding the DMA Descriptor | 240 | 14.14.3 | Transferring the Data | 240 | 14.14.4 | Isochronous OUT Endpoint Operation Example | 241 | | | | | | |
| 14.11.2 | Arbitration between endpoints | 234 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.9.14 | Validate Buffer (Command: 0xFA, Data: none) | 230 | 14.12 | Non Isochronous Endpoints - Normal Mode operation | 234 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.10 | DMA descriptor | 230 | 14.12.1 | Setting up DMA transfer | 234 | 14.12.2 | Finding DMA Descriptor | 234 | 14.12.3 | Transferring the Data | 235 | 14.12.4 | Optimizing Descriptor Fetch | 235 | 14.12.5 | Ending the packet transfer | 236 | 14.12.6 | No_Packet DD | 236 | 14.13 | Concatenated transfer (ATLE) mode operation | 236 | 14.13.1 | Setting up the DMA transfer | 239 | 14.13.2 | Finding the DMA Descriptor | 239 | 14.13.3 | Transferring the Data | 239 | 14.13.4 | Ending the packet transfer | 239 | 14.14 | Isochronous Endpoint Operation | 240 | 14.14.1 | Setting up of DMA transfer | 240 | 14.14.2 | Finding the DMA Descriptor | 240 | 14.14.3 | Transferring the Data | 240 | 14.14.4 | Isochronous OUT Endpoint Operation Example | 241 | | | | | | | | | | | | | | | | | | |
| 14.12.1 | Setting up DMA transfer | 234 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.12.2 | Finding DMA Descriptor | 234 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.12.3 | Transferring the Data | 235 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.12.4 | Optimizing Descriptor Fetch | 235 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.12.5 | Ending the packet transfer | 236 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.12.6 | No_Packet DD | 236 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.13 | Concatenated transfer (ATLE) mode operation | 236 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.13.1 | Setting up the DMA transfer | 239 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.13.2 | Finding the DMA Descriptor | 239 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.13.3 | Transferring the Data | 239 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.13.4 | Ending the packet transfer | 239 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.14 | Isochronous Endpoint Operation | 240 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.14.1 | Setting up of DMA transfer | 240 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.14.2 | Finding the DMA Descriptor | 240 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.14.3 | Transferring the Data | 240 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14.14.4 | Isochronous OUT Endpoint Operation Example | 241 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Chapter 15: Timer/Counter TIMER0 and TIMER1

| | | | | | |
|-------------|--|------------|--------|---|-----|
| 15.1 | Features | 242 | 15.5.2 | Timer Control Register (TCR, TIMER0: T0TCR - 0xE000 4004 and TIMER1: T1TCR - 0xE000 8004) | 245 |
| 15.2 | Applications | 242 | 15.5.3 | Count Control Register (CTCR, TIMER0: T0CTCR - 0xE000 4070 and TIMER1: T1TCR - 0xE000 8070) | 246 |
| 15.3 | Description | 242 | 15.5.4 | Timer Counter (TC, TIMER0: T0TC - 0xE000 4008 and TIMER1: T1TC - 0xE000 8008) | 247 |
| 15.4 | Pin description | 242 | | | |
| 15.5 | Register description | 243 | | | |
| 15.5.1 | Interrupt Register (IR, TIMER0: T0IR - 0xE000 4000 and TIMER1: T1IR - 0xE000 8000) | 245 | | | |

continued >>

| | | | | | |
|---------|---|-----|---------|---|-----|
| 15.5.5 | Prescale Register (PR, TIMER0: T0PR - 0xE000 400C and TIMER1: T1PR - 0xE000 800C) | 247 | 15.5.9 | Capture Registers (CR0 - CR3) | 249 |
| 15.5.10 | | | 15.5.10 | Capture Control Register (CCR, TIMER0: T0CCR - 0xE000 4028 and TIMER1: T1CCR - 0xE000 8028) | 249 |
| 15.5.6 | Prescale Counter Register (PC, TIMER0: T0PC - 0xE000 4010 and TIMER1: T1PC - 0xE000 8010) | 247 | 15.5.11 | External Match Register (EMR, TIMER0: T0EMR - 0xE000 403C; and TIMER1: T1EMR - 0xE000 803C) | 250 |
| 15.5.7 | Match Registers (MR0 - MR3) | 247 | | | |
| 15.5.8 | Match Control Register (MCR, TIMER0: T0MCR - 0xE000 4014 and TIMER1: T1MCR - 0xE000 8014) | 248 | 15.6 | Example timer operation | 251 |
| | | | 15.7 | Architecture | 252 |

Chapter 16: Pulse Width Modulator (PWM)

| | | | | | |
|--------|---|-----|--------|---|-----|
| 16.1 | Features | 253 | 16.4.3 | PWM Timer Counter (PWMTTC - 0xE001 4008) | 260 |
| 16.2 | Description | 253 | 16.4.4 | PWM Prescale Register (PWMPR - 0xE001 400C) | 260 |
| 16.2.1 | Rules for single edge controlled PWM outputs | 256 | 16.4.5 | PWM Prescale Counter register (PWMPC - 0xE001 4010) | 260 |
| 16.2.2 | Rules for double edge controlled PWM outputs | 257 | 16.4.6 | PWM Match Registers (PWMMR0 - PWMMR6) | 261 |
| 16.3 | Pin description | 257 | 16.4.7 | PWM Match Control Register (PWMMCR - 0xE001 4014) | 261 |
| 16.4 | Register description | 257 | 16.4.8 | PWM Control Register (PWMPCR - 0xE001 404C) | 262 |
| 16.4.1 | PWM Interrupt Register (PWMIR - 0xE001 4000) | 259 | 16.4.9 | PWM Latch Enable Register (PWMLER - 0xE001 4050) | 263 |
| 16.4.2 | PWM Timer Control Register (PWMTCR - 0xE001 4004) | 259 | | | |

Chapter 17: Analog-to-Digital Converter (ADC)

| | | | | | |
|--------|---|-----|--------|--|-----|
| 17.1 | Features | 265 | 17.4.5 | A/D Interrupt Enable Register (ADINTEN, ADC0: AD0INTEN - 0xE003 400C and ADC1: AD1INTEN - 0xE006 000C) | 270 |
| 17.2 | Description | 265 | 17.4.6 | A/D Data Registers (ADDR0 to ADDR7, ADC0: AD0DR0 to AD0DR7 - 0xE003 4010 to 0xE003 402C and ADC1: AD1DR0 to AD1DR7 - 0xE006 0010 to 0xE006 402C) | 271 |
| 17.3 | Pin description | 265 | | | |
| 17.4 | Register description | 266 | 17.5 | Operation | 272 |
| 17.4.1 | A/D Control Register (AD0CR - 0xE003 4000 and AD1CR - 0xE006 0000) | 267 | 17.5.1 | Hardware-triggered conversion | 272 |
| 17.4.2 | A/D Global Data Register (AD0GDR - 0xE003 4004 and AD1GDR - 0xE006 0004) | 268 | 17.5.2 | Interrupts | 272 |
| 17.4.3 | A/D Global Start Register (ADGSR - 0xE003 4008) | 269 | 17.5.3 | Accuracy vs. digital receiver | 272 |
| 17.4.4 | A/D Status Register (ADSTAT, ADC0: AD0CR - 0xE003 4004 and ADC1: AD1CR - 0xE006 0004) | 269 | | | |

Chapter 18: Digital-to-Analog Converter (DAC)

| | | | | | |
|------|----------------------------------|-----|------|--|-----|
| 18.1 | Features | 273 | 18.3 | DAC Register (DACR - 0xE006 C000) | 273 |
| 18.2 | Pin description | 273 | 18.4 | Operation | 274 |

Chapter 19: Real Time Clock

continued >>

| | | | | | |
|-------------|---|------------|----------------|---|-----|
| 19.1 | Features | 275 | 19.4.10 | Consolidated Time register 1 (CTIME1 - 0xE002 4018) | 280 |
| 19.2 | Description | 275 | 19.4.11 | Consolidated Time register 2 (CTIME2 - 0xE002 401C) | 280 |
| 19.3 | Architecture | 275 | 19.4.12 | Time counter group | 280 |
| 19.4 | Register description | 276 | 19.4.13 | Leap year calculation | 281 |
| 19.4.1 | RTC interrupts | 277 | 19.4.14 | Alarm register group | 281 |
| 19.4.2 | Miscellaneous register group | 277 | 19.5 | RTC usage notes | 282 |
| 19.4.3 | Interrupt Location Register (ILR - 0xE002 4000) | 277 | 19.6 | Reference clock divider (prescaler) | 282 |
| 19.4.4 | Clock Tick Counter Register (CTCR - 0xE002 4004) | 278 | 19.6.1 | Prescaler Integer register (PREINT - 0xE002 4080) | 283 |
| 19.4.5 | Clock Control Register (CCR - 0xE002 4008) | 278 | 19.6.2 | Prescaler Fraction register (PREFRAC - 0xE002 4084) | 283 |
| 19.4.6 | Counter Increment Interrupt Register (CIIR - 0xE002 400C) | 278 | 19.6.3 | Example of prescaler usage | 283 |
| 19.4.7 | Alarm Mask Register (AMR - 0xE002 4010) | 279 | 19.6.4 | Prescaler operation | 284 |
| 19.4.8 | Consolidated time registers | 279 | 19.7 | RTC external 32 kHz oscillator component selection | 285 |
| 19.4.9 | Consolidated Time register 0 (CTIME0 - 0xE002 4014) | 279 | | | |

Chapter 20: Watchdog Timer

| | | | | | |
|-------------|--|------------|---------------|---|-----|
| 20.1 | Features | 287 | 20.4.2 | Watchdog Timer Constant register (WDTC - 0xE000 0004) | 289 |
| 20.2 | Applications | 287 | 20.4.3 | Watchdog Feed register (WDFEED - 0xE000 0008) | 289 |
| 20.3 | Description | 287 | 20.4.4 | Watchdog Timer Value register (WDTV - 0xE000 000C) | 289 |
| 20.4 | Register description | 288 | 20.5 | Block diagram | 289 |
| 20.4.1 | Watchdog Mode register (WDMOD - 0xE000 0000) | 288 | | | |

Chapter 21: Flash Memory System and Programming

| | | | | | |
|-------------|---|------------|-------------|---|-----|
| 21.1 | Flash Boot Loader | 291 | 21.6 | Flash content protection mechanism | 296 |
| 21.2 | Features | 291 | 21.7 | Code Read Protection (CRP) | 297 |
| 21.3 | Applications | 291 | 21.8 | ISP commands | 297 |
| 21.4 | Description | 291 | 21.8.1 | Unlock <unlock code> | 298 |
| 21.4.1 | Memory map after any reset | 291 | 21.8.2 | Set Baud Rate <baud rate> <stop bit> | 298 |
| 21.4.2 | Criterion for valid user code | 292 | 21.8.3 | Echo <setting> | 299 |
| 21.4.3 | Communication protocol | 293 | 21.8.4 | Write to RAM <start address> <number of bytes> | 299 |
| 21.4.4 | ISP command format | 293 | 21.8.5 | Read memory <address> <no. of bytes> | 300 |
| 21.4.5 | ISP response format | 293 | 21.8.6 | Prepare sector(s) for write operation <start sector number> <end sector number> | 300 |
| 21.4.6 | ISP data format | 293 | 21.8.7 | Copy RAM to Flash <Flash address> <RAM address> <no of bytes> | 301 |
| 21.4.7 | ISP flow control | 293 | 21.8.8 | Go <address> <mode> | 302 |
| 21.4.8 | ISP command abort | 294 | 21.8.9 | Erase sector(s) <start sector number> <end sector number> | 302 |
| 21.4.9 | Interrupts during ISP | 294 | 21.8.10 | Blank check sector(s) <sector number> <end sector number> | 303 |
| 21.4.10 | Interrupts during IAP | 294 | 21.8.11 | Read Part Identification number | 303 |
| 21.4.11 | RAM used by ISP command handler | 294 | 21.8.12 | Read Boot code version number | 303 |
| 21.4.12 | RAM used by IAP command handler | 294 | | | |
| 21.4.13 | RAM used by RealMonitor | 294 | | | |
| 21.4.14 | Boot process flowchart | 295 | | | |
| 21.5 | Sector numbers | 295 | | | |

continued >

| | | | | | |
|-------------|--|------------|--------------|--|------------|
| 21.8.13 | Compare <address1> <address2> <no of bytes> | 304 | 21.9.5 | Read Part Identification number | 309 |
| 21.8.14 | ISP Return codes | 304 | 21.9.6 | Read Boot code version number | 309 |
| 21.9 | IAP Commands | 305 | 21.9.7 | Compare <address1> <address2> <no of bytes> | 310 |
| 21.9.1 | Prepare sector(s) for write operation | 307 | 21.9.8 | Reinvoke ISP | 310 |
| 21.9.2 | Copy RAM to Flash. | 308 | 21.9.9 | IAP Status codes | 310 |
| 21.9.3 | Erase sector(s) | 308 | 21.10 | JTAG Flash programming interface. | 311 |
| 21.9.4 | Blank check sector(s) | 309 | | | |

Chapter 22: EmbeddedICE logic

| | | | | | |
|-------------|----------------------------------|------------|-------------|--|------------|
| 22.1 | Features | 312 | 22.5 | Reset state of multiplexed pins | 313 |
| 22.2 | Applications | 312 | 22.6 | Register description | 314 |
| 22.3 | Description | 312 | 22.7 | Block diagram | 314 |
| 22.4 | Pin description | 313 | | | |

Chapter 23: Embedded Trace Macrocell (ETM)

| | | | | | |
|-------------|-------------------------------|------------|-------------|--|------------|
| 23.1 | Features | 315 | 23.4 | Pin description | 316 |
| 23.2 | Applications | 315 | 23.5 | Reset state of multiplexed pins | 316 |
| 23.3 | Description | 315 | 23.6 | Register description | 317 |
| 23.3.1 | ETM configuration. | 315 | 23.7 | Block diagram | 318 |

Chapter 24: RealMonitor

| | | | | | |
|-------------|---|------------|----------------|--|------------|
| 24.1 | Features | 319 | 24.4.4 | SVC mode | 322 |
| 24.2 | Applications | 319 | 24.4.5 | Prefetch Abort mode | 323 |
| 24.3 | Description | 319 | 24.4.6 | Data Abort mode | 323 |
| 24.3.1 | RealMonitor components | 320 | 24.4.7 | User/System mode | 323 |
| 24.3.2 | RMHost. | 320 | 24.4.8 | FIQ mode | 323 |
| 24.3.3 | RMTarget | 320 | 24.4.9 | Handling exceptions. | 323 |
| 24.3.4 | How RealMonitor works | 321 | 24.4.10 | RealMonitor exception handling. | 323 |
| 24.4 | How to enable Realmonitor. | 322 | 24.4.11 | RMTarget initialization | 324 |
| 24.4.1 | Adding stacks | 322 | 24.4.12 | Code example | 324 |
| 24.4.2 | IRQ mode | 322 | 24.5 | RealMonitor build options | 327 |
| 24.4.3 | Undef mode | 322 | | | |

Chapter 25: Supplementary information

| | | |
|-------------|--------------------------------|------------|
| 25.1 | Abbreviations | 329 |
| 25.2 | Disclaimers | 330 |
| 25.3 | Trademarks | 330 |

© Koninklijke Philips Electronics N.V. 2005

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or other industrial or intellectual property rights.

Date of release: 15 August 2005



Published in The Netherlands