UM10360

LPC17xx User manual

Rev. 01 — 4 January 2010 **User manual**

Document information

Info	Content
Keywords	LPC1769, LPC1768, LPC1767, LPC1766, LPC1765, LPC1764, LPC1759, LPC1758, LPC1756, LPC1754, LPC1752, LPC1751, ARM, ARM Cortex-M3, 32-bit, USB, Ethernet, CAN, I2S, Microcontroller
Abstract	LPC17xx user manual

founded by Philips

Revision history

Rev	Date	Description
1	20100104	LPC17xx user manual revision.
		Modifications: • “Draft” status removed. • Editorial updates and typographical corrections throughout the user manual. • LPC1758, LPC1767, and LPC1768 have been added to the keywords list on the front cover, the ordering information in section Section 1–4, and the part identification number table in Section 32–7.11. • The note about DMA operation in Sleep mode was removed from Section 4–8.1. • In Table 8–81, the CLKOUT function was removed from the description of P1.25. • In section the Ethernet chapter, in Section 10–16.1 and Section 10–17.2, it has been noted that the external PHY must be initialized and PHY clocks received by the Ethernet block prior to further initialization of the Ethernet block. Also, in Section 10–17.1, under the heading "Ownership of descriptors", the sentence about AHB arbitration was removed. A general and more correct discussion of the subject was added in Section 2–5. • The UART fractional baud rate generator is disabled in auto baud mode (see Section 14–14.4.10.1 and Section 15–4.14). • In section Section 14–4.12 and Section 15–4.16, the description of the value of the DLL register has been is corrected to read "the value of the DLL register must be greater than 2". • *** Motor Control PWM *** • The description of RPM calculation in the QEI chapter (see Section 26–4.3), definitions for the formula values are added, and the description improved. The description of the position and index compare registers incorrectly indicated that the less than,equal to, and greater than compare could be selected. It is changed to only indicate “equal to”. • A description of Flash signature generation has been added in Section 32–10.

Contact information

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

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

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 1: LPC17xx Introductory information

Rev. 01 — 4 January 2010 **User manual**

1. Introduction

The LPC17xx is an ARM Cortex-M3 based microcontroller for embedded applications requiring a high level of integration and low power dissipation. The ARM Cortex-M3 is a next generation core that offers system enhancements such as modernized debug features and a higher level of support block integration.

High speed versions (LPC1769 and LPC1759) operate at up to a 120 MHz CPU frequency. Other versions operate at up to an 100 MHz CPU frequency. The ARM Cortex-M3 CPU incorporates a 3-stage pipeline and uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals. The ARM Cortex-M3 CPU also includes an internal prefetch unit that supports speculative branches.

The peripheral complement of the LPC17xx includes up to 512 kB of flash memory, up to 64 kB of data memory, Ethernet MAC, a USB interface that can be configured as either Host, Device, or OTG, 8 channel general purpose DMA controller, 4 UARTs, 2 CAN channels, 2 SSP controllers, SPI interface, 3 I^2C interfaces, 2-input plus 2-output I^2S interface, 8 channel 12-bit ADC, 10-bit DAC, motor control PWM, Quadrature Encoder interface, 4 general purpose timers, 6-output general purpose PWM, ultra-low power RTC with separate battery supply, and up to 70 general purpose I/O pins.

© NXP B.V. 2010. All rights reserved.

2. Features

Refer to Section 1–4.1 for details of features on specific part numbers.

- ARM Cortex-M3 processor, running at frequencies of up to 120 MHz on high speed versions (LPC1769 and LPC1759), up to 100 MHz on other versions. A Memory Protection Unit (MPU) supporting eight regions is included.
- ARM Cortex-M3 built-in Nested Vectored Interrupt Controller (NVIC).
- Up to 512 kB on-chip flash program memory with In-System Programming (ISP) and In-Application Programming (IAP) capabilities. The combination of an enhanced flash memory accelerator and location of the flash memory on the CPU local code/data bus provides high code performance from flash.
- Up to 64 kB on-chip SRAM includes:
  - Up to 32 kB of SRAM on the CPU with local code/data bus for high-performance CPU access.
  - Up to two 16 kB SRAM blocks with separate access paths for higher throughput. These SRAM blocks may be used for Ethernet, USB, and DMA memory, as well as for general purpose instruction and data storage.
- Eight channel General Purpose DMA controller (GPDMA) on the AHB multilayer matrix that can be used with the SSP, I^2S, UART, the Analog-to-Digital and Digital-to-Analog converter peripherals, timer match signals, GPIO, and for memory-to-memory transfers.
- Multilayer AHB matrix interconnect provides a separate bus for each AHB master. AHB masters include the CPU, General Purpose DMA controller, Ethernet MAC, and the USB interface. This interconnect provides communication with no arbitration delays unless two masters attempt to access the same slave at the same time.
- Split APB bus allows for higher throughput with fewer stalls between the CPU and DMA. A single level of write buffering allows the CPU to continue without waiting for completion of APB writes if the APB was not already busy.
- Serial interfaces:
  - Ethernet MAC with RMII interface and dedicated DMA controller.
  - USB 2.0 full-speed controller that can be configured for either device, Host, or OTG operation with an on-chip PHY for device and Host functions and a dedicated DMA controller.
  - Four UARTs with fractional baud rate generation, internal FIFO, IrDA, and DMA support. One UART has modem control I/O and RS-485/EIA-485 support.
  - Two-channel CAN controller.
  - Two SSP controllers with FIFO and multi-protocol capabilities. The SSP interfaces can be used with the GPDMA controller.
  - SPI controller with synchronous, serial, full duplex communication and programmable data length. SPI is included as a legacy peripheral and can be used instead of SSP0.
  - Three enhanced I^2C-bus interfaces, one with an open-drain output supporting the full I^2C specification and Fast mode plus with data rates of 1Mbit/s, two with standard port pins. Enhancements include multiple address recognition and monitor mode.

© NXP B.V. 2010. All rights reserved.

- I^2S (Inter-IC Sound) interface for digital audio input or output, with fractional rate control. The I^2S interface can be used with the GPDMA. The I^2S interface supports 3-wire data transmit and receive or 4-wire combined transmit and receive connections, as well as master clock output.

- Other peripherals:
  - 70 (100 pin package) or 52 (80-pin package) General Purpose I/O (GPIO) pins with configurable pull-up/down resistors, open drain mode, and repeater mode. All GPIOs are located on an AHB bus for fast access, and support Cortex-M3 bit-banding. GPIOs can be accessed by the General Purpose DMA Controller. Any pin of ports 0 and 2 can be used to generate an interrupt.
  - 12-bit Analog-to-Digital Converter (ADC) with input multiplexing among eight pins, conversion rates up to 200 kHz, and multiple result registers. The 12-bit ADC can be used with the GPDMA controller.
  - 10-bit Digital-to-Analog Converter (DAC) with dedicated conversion timer and DMA support.
  - Four general purpose timers/counters, with a total of eight capture inputs and ten compare outputs. Each timer block has an external count input. Specific timer events can be selected to generate DMA requests.
  - One motor control PWM with support for three-phase motor control.
  - Quadrature encoder interface that can monitor one external quadrature encoder.
  - One standard PWM/timer block with external count input.
  - Real-Time Clock (RTC) with a separate power domain. The RTC is clocked by a dedicated RTC oscillator. The RTC block includes 20 bytes of battery-powered backup registers, allowing system status to be stored when the rest of the chip is powered off. Battery power can be supplied from a standard 3 V Lithium button cell. The RTC will continue working when the battery voltage drops to as low as 2.1 V. An RTC interrupt can wake up the CPU from any reduced power mode.
  - Watchdog Timer (WDT). The WDT can be clocked from the internal RC oscillator, the RTC oscillator, or the APB clock.
  - Cortex-M3 system tick timer, including an external clock input option.
  - Repetitive interrupt timer provides programmable and repeating timed interrupts.
- Standard JTAG test/debug interface as well as Serial Wire Debug and Serial Wire Trace Port options.
- Emulation trace module supports real-time trace.
- Four reduced power modes: Sleep, Deep-sleep, Power-down, and Deep power-down.
- Single 3.3 V power supply (2.4 V to 3.6 V). Temperature range of -40 °C to 85 °C.
- Four external interrupt inputs configurable as edge/level sensitive. All pins on PORT0 and PORT2 can be used as edge sensitive interrupt sources.
- Non-maskable Interrupt (NMI) input.
- Clock output function that can reflect the main oscillator clock, IRC clock, RTC clock, CPU clock, or the USB clock.
- The Wakeup Interrupt Controller (WIC) allows the CPU to automatically wake up from any priority interrupt that can occur while the clocks are stopped in deep sleep, Power-down, and Deep power-down modes.

© NXP B.V. 2010. All rights reserved.

- Processor wake-up from Power-down mode via any interrupt able to operate during Power-down mode (includes external interrupts, RTC interrupt, USB activity, Ethernet wake-up interrupt, CAN bus activity, PORT0/2 pin interrupt, and NMI).
- Each peripheral has its own clock divider for further power savings.
- Brownout detect with separate threshold for interrupt and forced reset.
- On-chip Power-On Reset (POR).
- On-chip crystal oscillator with an operating range of 1 MHz to 25 MHz.
- 4 MHz internal RC oscillator trimmed to 1% accuracy that can optionally be used as a system clock.
- An on-chip PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency crystal. May be run from the main oscillator, the internal RC oscillator, or the RTC oscillator.
- A second, dedicated PLL may be used for the USB interface in order to allow added flexibility for the Main PLL settings.
- Versatile pin function selection feature allows many possibilities for using on-chip peripheral functions.
- Available as 100-pin LQFP (14 x 14 x 1.4 mm) and 80-pin LQFP (12 x 12 x 1.4 mm) packages.

3. Applications

- eMetering
- Lighting
- Industrial networking
- Alarm systems
- White goods
- Motor control

© NXP B.V. 2010. All rights reserved.

4. Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC1769FBD100	LQFP100	plastic low profile quad flat package; 100 leads; body 14 × 14 × 1.4 mm	SOT407-1
LPC1768FBD100			
LPC1767FBD100			
LPC1766FBD100			
LPC1765FBD100			
LPC1764FBD100			
LPC1759FBD80	LQFP80	plastic low profile quad flat package; 80 leads; body 12 × 12 × 1.4 mm	SOT315-1
LPC1758FBD80			
LPC1756FBD80			
LPC1754FBD80			
LPC1752FBD80			
LPC1751FBD80			

4.1 Part options summary

Table 2. Ordering options for LPC17xx parts

Type number	Max. CPU speed	Flash	Total SRAM	Ethernet	USB	CAN	I^2S	DAC	Package
LPC1769FBD100	120 MHz	512 kB	64 kB	yes	Device/Host/OTG	2	yes	yes	100 pin
LPC1768FBD100	100 MHz	512 kB	64 kB	yes	Device/Host/OTG	2	yes	yes	100 pin
LPC1767FBD100	100 MHz	512 kB	64 kB	yes	no	no	yes	yes	100 pin
LPC1766FBD100	100 MHz	256 kB	64 kB	yes	Device/Host/OTG	2	yes	yes	100 pin
LPC1765FBD100	100 MHz	256 kB	64 kB	no	Device/Host/OTG	2	yes	yes	100 pin
LPC1764FBD100	100 MHz	128 kB	32 kB	yes	Device	2	no	no	100 pin
LPC1759FBD80	120 MHz	512 kB	64 kB	no	Device/Host/OTG	2	yes	yes	80 pin
LPC1758FBD80	100 MHz	512 kB	64 kB	yes	Device/Host/OTG	2	yes	yes	80 pin
LPC1756FBD80	100 MHz	256 kB	32 kB	no	Device/Host/OTG	2	yes	yes	80 pin
LPC1754FBD80	100 MHz	128 kB	32 kB	no	Device/Host/OTG	1	no	yes	80 pin
LPC1752FBD80	100 MHz	64 kB	16 kB	no	Device	1	no	no	80 pin
LPC1751FBD80	100 MHz	32 kB	8 kB	no	Device	1	no	no	80 pin

© NXP B.V. 2010. All rights reserved.

5. Simplified block diagram

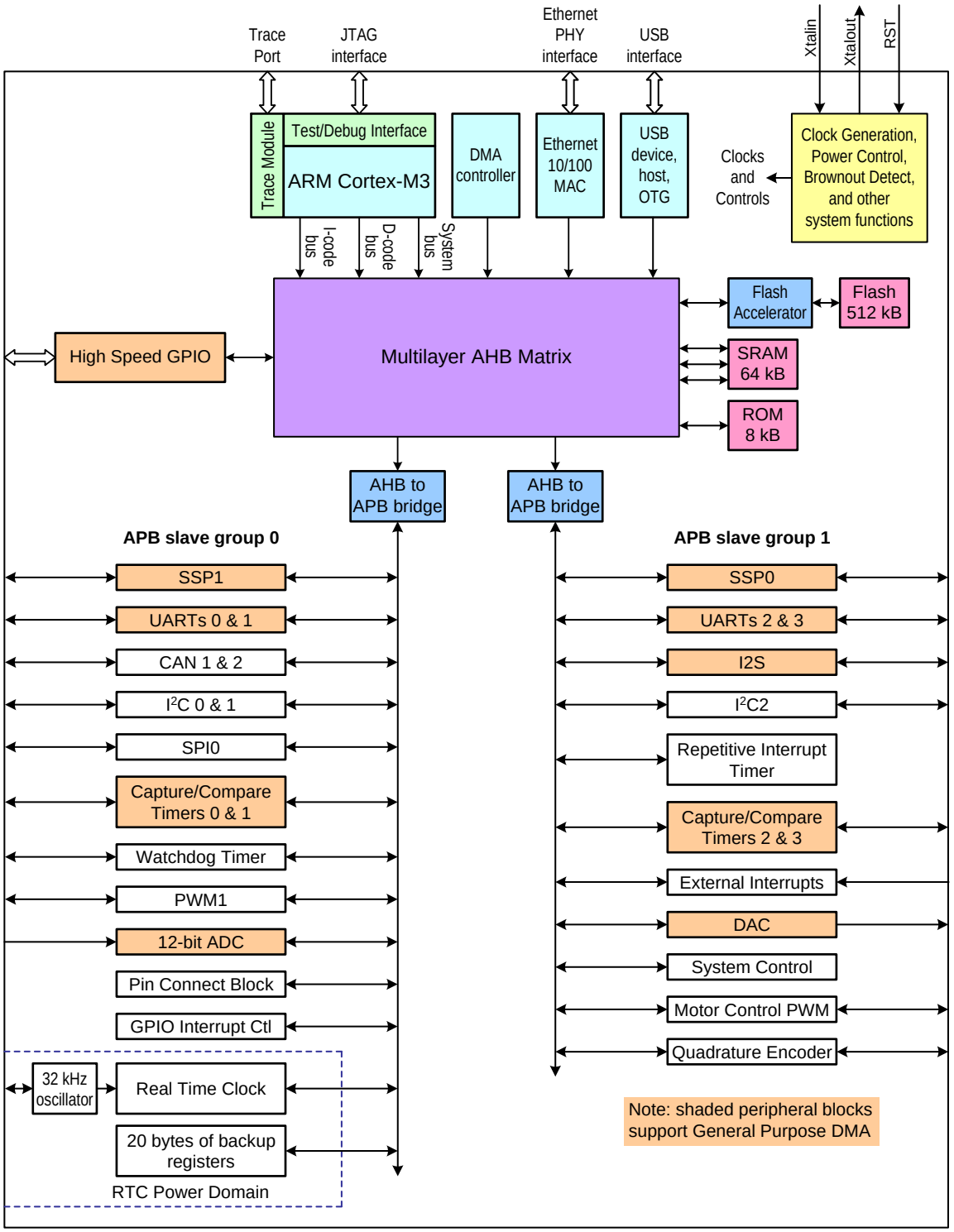


Fig 1. LPC1768 simplified block diagram

© NXP B.V. 2010. All rights reserved.

6. Architectural overview

The ARM Cortex-M3 includes three AHB-Lite buses, one system bus and the I-code and D-code buses which are faster and are used similarly to TCM interfaces: one bus dedicated for instruction fetch (I-code) and one bus for data access (D-code). The use of two core buses allows for simultaneous operations if concurrent operations target different devices.

The LPC17xx uses a multi-layer AHB matrix to connect the Cortex-M3 buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters. Details of the multilayer matrix connections are shown in Figure 1–2.

APB peripherals are connected to the CPU via two APB busses using separate slave ports from the multilayer AHB matrix. This allows for better performance by reducing collisions between the CPU and the DMA controller. The APB bus bridges are configured to buffer writes so that the CPU or DMA controller can write to APB devices without always waiting for APB write completion.

7. ARM Cortex-M3 processor

The ARM Cortex-M3 is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The Cortex-M3 offers many new features, including a Thumb-2 instruction set, low interrupt latency, hardware divide, interruptible/continuable multiple load and store instructions, automatic state save and restore for interrupts, tightly integrated interrupt controller with Wakeup Interrupt Controller, and multiple core buses capable of simultaneous accesses.

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

The ARM Cortex-M3 processor is described in detail in the Cortex-M3 User Guide that is appended to this manual.

7.1 Cortex-M3 Configuration Options

The LPC17xx uses the r2p0 version of the Cortex-M3 CPU, which includes a number of configurable options, as noted below.

System options:

- The Nested Vectored Interrupt Controller (NVIC) is included. The NVIC includes the SYSTICK timer.
- The Wakeup Interrupt Controller (WIC) is included. The WIC allows more powerful options for waking up the CPU from reduced power modes.
- A Memory Protection Unit (MPU) is included.
- A ROM Table in included. The ROM Table provides addresses of debug components to external debug systems.

Debug related options:

© NXP B.V. 2010. All rights reserved.

- A JTAG debug interface is included.
- Serial Wire Debug is included. Serial Wire Debug allows debug operations using only 2 wires, simple trace functions can be added with a third wire.
- The Embedded Trace Macrocell (ETM) is included. The ETM provides instruction trace capabilities.
- The Data Watchpoint and Trace (DWT) unit is included. The DWT allows data address or data value matches to be trace information or trigger other events. The DWT includes 4 comparators and counters for certain internal events.
- An Instrumentation Trace Macrocell (ITM) is included. Software can write to the ITM in order to send messages to the trace port.
- The Trace Port Interface Unit (TPIU) is included. The TPIU encodes and provides trace information to the outside world. This can be on the Serial Wire Viewer pin or the 4-bit parallel trace port.
- A Flash Patch and Breakpoint (FPB) is included. The FPB can generate hardware breakpoints and remap specific addresses in code space to SRAM as a temporary method of altering non-volatile code. The FPB include 2 literal comparators and 6 instruction comparators.

8. On-chip flash memory system

The LPC17xx contains up to 512 kB of on-chip flash memory. A new two-port flash memory accelerator maximizes performance for use with the two fast AHB-Lite buses. This memory may be used for both code and data storage. Programming of the flash memory may be accomplished in several ways. It may be programmed In System via the serial port. The application program may also erase and/or program the flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc.

9. On-chip Static RAM

The LPC17xx contains up to 64 kB of on-chip static RAM memory. Up to 32 kB of SRAM, accessible by the CPU and all three DMA controllers are on a higher-speed bus. Devices containing more than 32 kB SRAM have two additional 16 kB SRAM blocks, each situated on separate slave ports on the AHB multilayer matrix.

This architecture allows the possibility for CPU and DMA accesses to be separated in such a way that there are few or no delays for the bus masters.

© NXP B.V. 2010. All rights reserved.

10. Block diagram

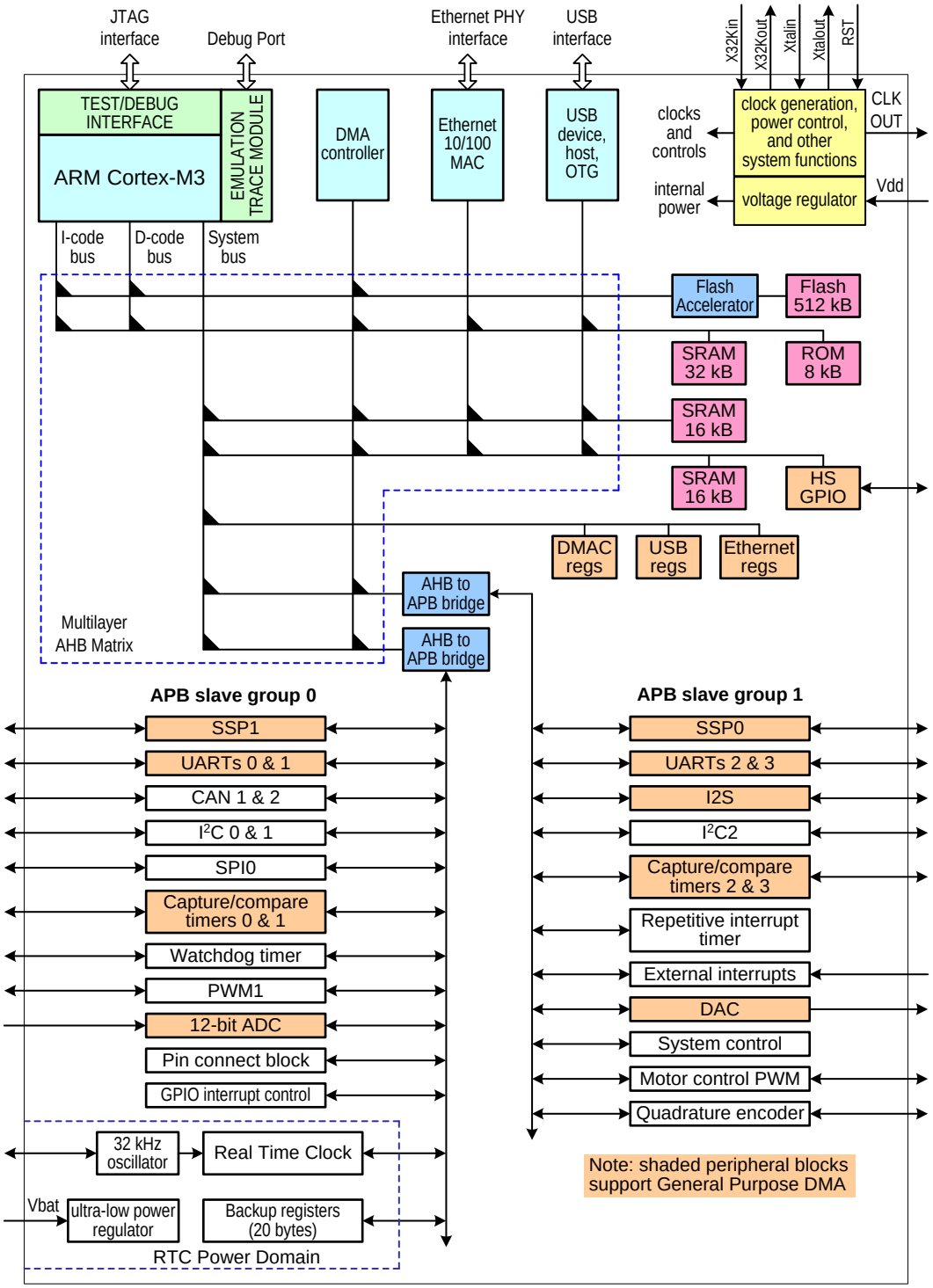


Fig 2. LPC1768 block diagram, CPU and buses

© NXP B.V. 2010. All rights reserved.

1. Memory map and peripheral addressing

The ARM Cortex-M3 processor has a single 4 GB address space. The following table shows how this space is used on the LPC17xx.

Table 3. LPC17xx memory usage and details

Address range	General Use	Address range details and description	
0x0000 0000 to 0x1FFF FFFF	On-chip non-volatile memory	0x0000 0000 - 0x0007 FFFF	For devices with 512 kB of flash memory.
		0x0000 0000 - 0x0003 FFFF	For devices with 256 kB of flash memory.
		0x0000 0000 - 0x0001 FFFF	For devices with 128 kB of flash memory.
		0x0000 0000 - 0x0000 FFFF	For devices with 64 kB of flash memory.
		0x0000 0000 - 0x0000 7FFF	For devices with 32 kB of flash memory.
	On-chip SRAM	0x1000 0000 - 0x1000 7FFF	For devices with 32 kB of local SRAM.
		0x1000 0000 - 0x1000 3FFF	For devices with 16 kB of local SRAM.
		0x1000 0000 - 0x1000 1FFF	For devices with 8 kB of local SRAM.
	Boot ROM	0x1FFF 0000 - 0x1FFF 1FFF	8 kB Boot ROM with flash services.
0x2000 0000 to 0x3FFF FFFF	On-chip SRAM (typically used for peripheral data)	0x2007 C000 - 0x2007 FFFF	AHB SRAM - bank 0 (16 kB), present on devices with 32 kB or 64 kB of total SRAM.
		0x2008 0000 - 0x2008 3FFF	AHB SRAM - bank 1 (16 kB), present on devices with 64 kB of total SRAM.
	GPIO	0x2009 C000 - 0x2009 FFFF	GPIO.
0x4000 0000 to 0x5FFF FFFF	APB Peripherals	0x4000 0000 - 0x4007 FFFF	APB0 Peripherals, up to 32 peripheral blocks, 16 kB each.
		0x4008 0000 - 0x400F FFFF	APB1 Peripherals, up to 32 peripheral blocks, 16 kB each.
	AHB peripherals	0x5000 0000 - 0x501F FFFF	DMA Controller, Ethernet interface, and USB interface.
0xE000 0000 to 0xE00F FFFF	Cortex-M3 Private Peripheral Bus	0xE000 0000 - 0xE00F FFFF	Cortex-M3 related functions, includes the NVIC and System Tick Timer.

2. Memory maps

The LPC17xx incorporates several distinct memory regions, shown in the following figures. Figure 2–3 shows the overall map of the entire address space from the user program viewpoint following reset. The interrupt vector area supports address remapping, which is described later in this section.

© NXP B.V. 2010. All rights reserved.

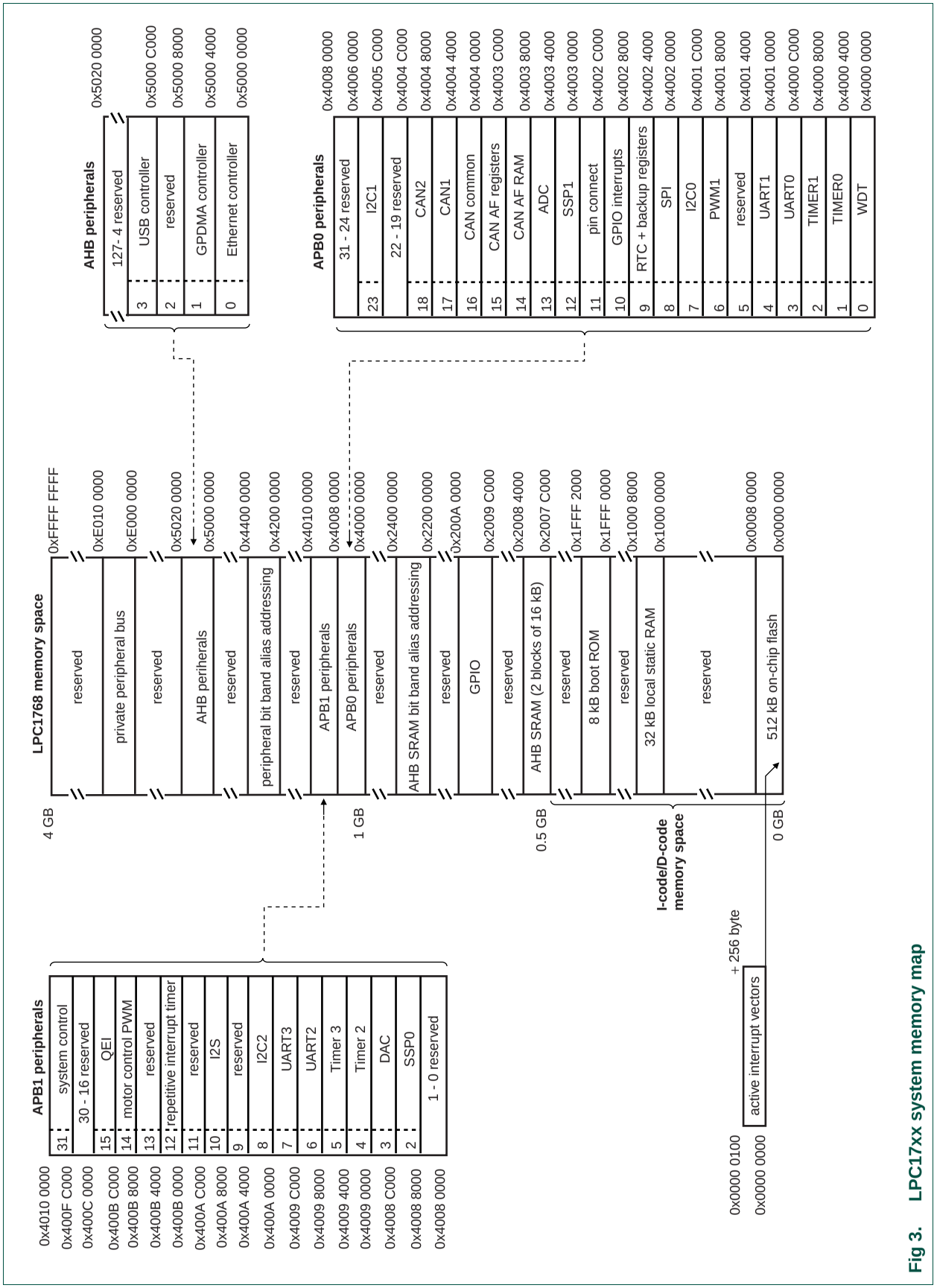


Fig 3. LPC17xx system memory map

© NXP B.V. 2010. All rights reserved.

Figure 2–3 and Table 2–4 show different views of the peripheral address space. The AHB peripheral area is 2 megabyte in size, and is divided to allow for up to 128 peripherals. The APB peripheral area is 1 megabyte in size and is divided to allow for up to 64 peripherals. Each peripheral of either type is allocated 16 kilobytes of space. This allows simplifying the address decoding for each peripheral.

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.

3. APB peripheral addresses

The following table shows the APB0/1 address maps. No APB peripheral uses all of the 16 kB space allocated to it. Typically each device's registers are "aliased" or repeated at multiple locations within each 16 kB range.

Table 4. APB0 peripherals and base addresses

APB0 peripheral	Base address	Peripheral name
0	0x4000 0000	Watchdog Timer
1	0x4000 4000	Timer 0
2	0x4000 8000	Timer 1
3	0x4000 C000	UART0
4	0x4001 0000	UART1
5	0x4001 4000	reserved
6	0x4001 8000	PWM1
7	0x4001 C000	I^2C0
8	0x4002 0000	SPI
9	0x4002 4000	RTC
10	0x4002 8000	GPIO interrupts
11	0x4002 C000	Pin Connect Block
12	0x4003 0000	SSP1
13	0x4003 4000	ADC
14	0x4003 8000	CAN Acceptance Filter RAM
15	0x4003 C000	CAN Acceptance Filter Registers
16	0x4004 0000	CAN Common Registers
17	0x4004 4000	CAN Controller 1
18	0x4004 8000	CAN Controller 2
19 to 22	0x4004 C000 to 0x4005 8000	reserved
23	0x4005 C000	I^2C1
24 to 31	0x4006 0000 to 0x4007 C000	reserved

© NXP B.V. 2010. All rights reserved.

Table 5. APB1 peripherals and base addresses

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

4. Memory re-mapping

The Cortex-M3 incorporates a mechanism that allows remapping the interrupt vector table to alternate locations in the memory map. This is controlled via the Vector Table Offset Register contained in the Cortex-M3. Refer to Section 6–4 and Section 34–4.3.5 of the Cortex-M3 User Guide appended to this manual for details of the Vector Table Offset feature.

Boot ROM re-mapping

Following a hardware reset, the Boot ROM is temporarily mapped to address 0. This is normally transparent to the user. However, if execution is halted immediately after reset by a debugger, it should correct the mapping for the user. See Section 33–6.

5. AHB arbitration

The Multilayer AHB Matrix arbitrates between several masters. By default, the Cortex-M3 D-code bus has the highest priority, followed by the I-Code bus. All other masters share a lower priority.

© NXP B.V. 2010. All rights reserved.

6. Bus fault exceptions

The LPC17xx generates Bus Fault 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 derivative. These include all spaces marked “reserved” in Figure 2–3.

For these areas, both attempted data access and instruction fetch generate an exception. In addition, a Bus Fault exception is generated for any instruction fetch that maps to an AHB or APB peripheral address.

Within the address space of an existing APB peripheral, an 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 0x4000 D000 (an undefined address within the UART0 space) may result in an access to the register defined at address 0x4000 C000. Details of such address aliasing within a peripheral space are not defined in the LPC17xx documentation and are not a supported feature.

If software executes a write directly to the flash memory, the flash accelerator will generate a Bus Fault exception. Flash programming must be accomplished by using the specified flash programming interface provided by the Boot Code.

Note that the Cortex-M3 core stores the exception flag along with the associated instruction in the pipeline and processes the exception only if an attempt is made to execute the instruction fetched from the disallowed address. This prevents accidental aborts that could be caused by prefetches that occur when code is executed very near a memory boundary.

© NXP B.V. 2010. All rights reserved.

1. Introduction

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

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

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

2. Pin description

Table 3–6 shows pins that are associated with System Control block functions.

Table 6. Pin summary

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

© NXP B.V. 2010. All rights reserved.

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 7. Summary of system control registers

Name	Description	Access	Reset value	Address
External Interrupts				
EXTINT	External Interrupt Flag Register	R/W	0	0x400F C140
EXTMODE	External Interrupt Mode register	R/W	0	0x400F C148
EXTPOLAR	External Interrupt Polarity Register	R/W	0	0x400F C14C
Reset				
RSID	Reset Source Identification Register	R/W	see Table 3–8	0x400F C180
Syscon Miscellaneous Registers				
SCS	System Control and Status	R/W	0	0x400F C1A0

4. Reset

Reset has 4 sources on the LPC17xx: the $\overline{\text{RESET}}$ pin, Watchdog Reset, Power On Reset (POR), and Brown Out Detect (BOD).

The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin. Assertion of chip Reset by any source, once the operating voltage attains a usable level, starts the wake-up timer (see description in Section 4–9 “Wake-up timer” in this chapter), causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, a fixed number of clocks have passed, and the flash controller has completed its initialization. The reset logic is shown in the following block diagram (see Figure 3–4).

© NXP B.V. 2010. All rights reserved.

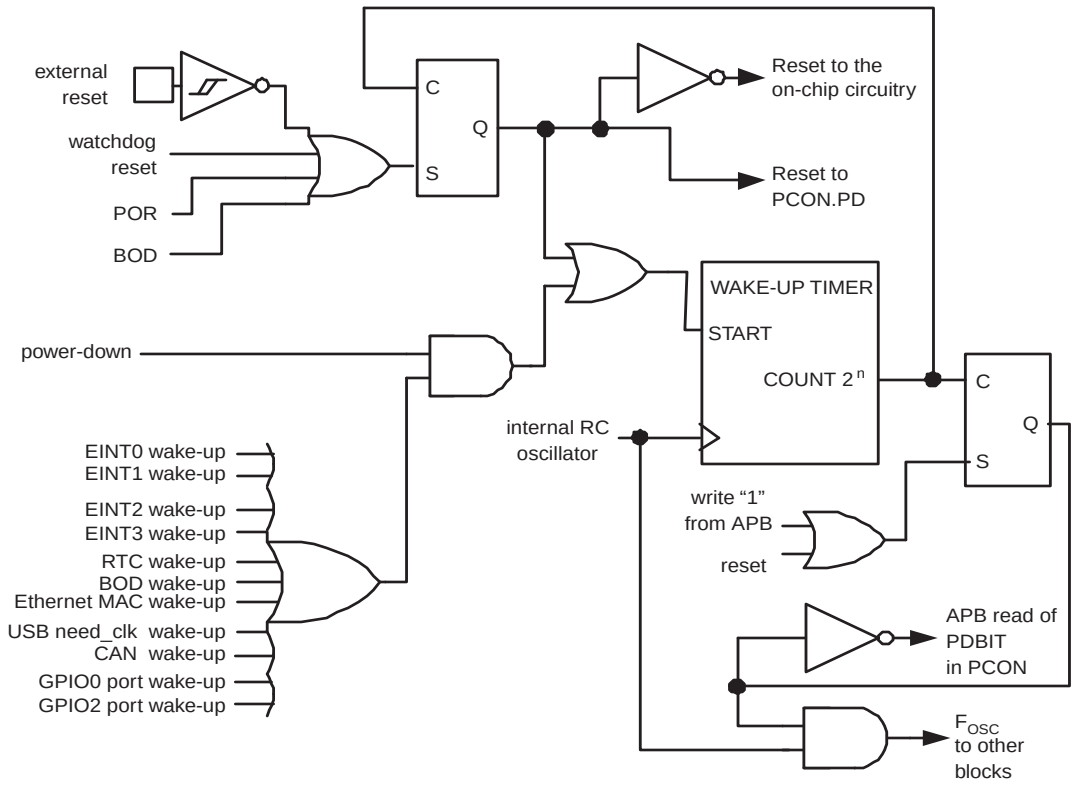


Fig 4. Reset block diagram including the wake-up timer

On the assertion of a reset source external to the Cortex-M3 CPU (POR, BOD reset, External reset, and Watchdog reset), the IRC starts up. After the IRC-start-up time (maximum of 60 μs on power-up) and after the IRC provides a stable clock output, the reset signal is latched and synchronized on the IRC clock. Then the following two sequences start simultaneously:

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

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

Figure 3–5 shows an example of the relationship between the $\overline{\text{RESET}}$, the IRC, and the processor status when the LPC17xx starts up after reset. See Section 4–3.2 "Main oscillator" for start-up of the main oscillator if selected by the user code.

© NXP B.V. 2010. All rights reserved.

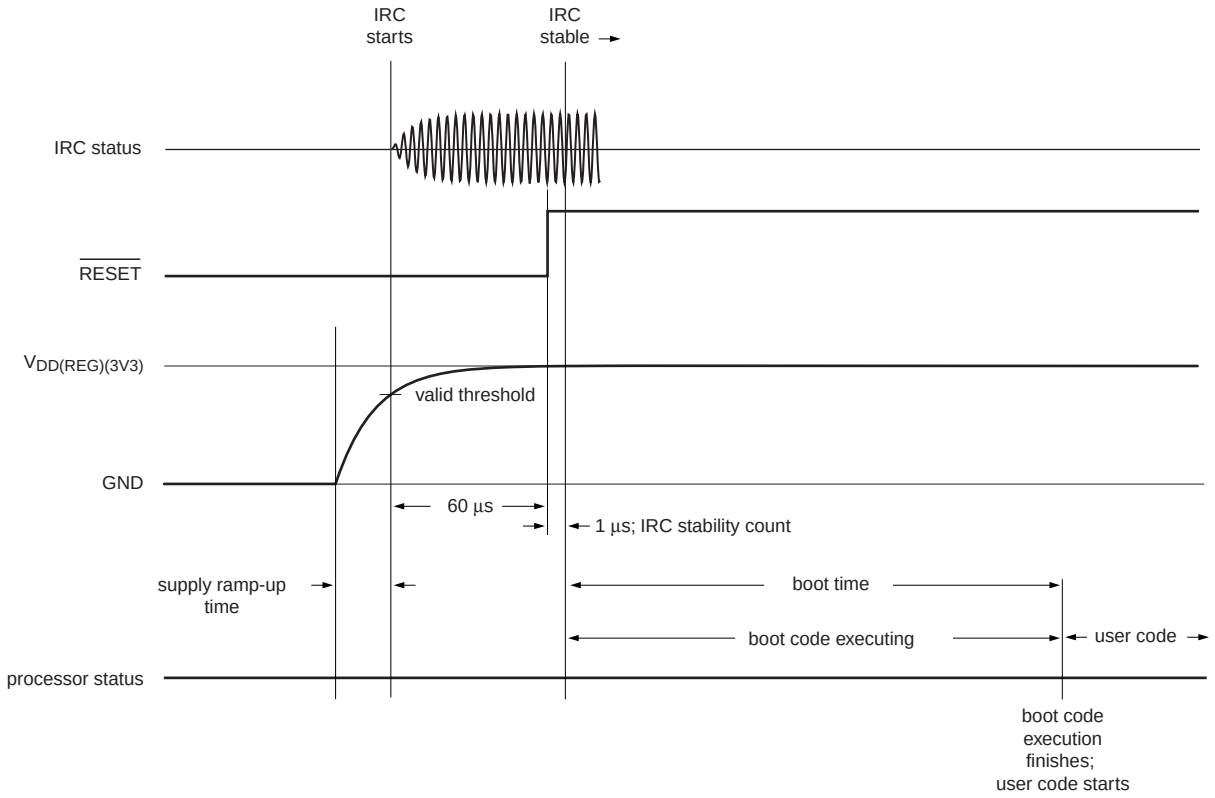


Fig 5. Example of start-up after reset

© NXP B.V. 2010. All rights reserved.

4.1 Reset Source Identification Register (RSID - 0x400F 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 8. Reset Source Identification register (RSID - address 0x400F C180) bit description

Bit	Symbol	Description	Reset value
0	POR	Assertion of the POR signal sets this bit, and clears all of the other bits in this register. But if another Reset signal (e.g., External Reset) remains asserted after the POR signal is negated, then its bit is set. This bit is not affected by any of the other sources of Reset.	See text
1	EXTR	Assertion of the $\overline{\text{RESET}}$ signal sets this bit. This bit is cleared by POR, but is not affected by WDT or BOD reset.	See text
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 $V_{DD(REG)(3V3)}$ voltage reaches a level below the BOD reset trip level (typically 1.85 V under nominal room temperature conditions). If the $V_{DD(REG)(3V3)}$ voltage dips from the normal operating range to below the BOD reset trip level and recovers, the BODR bit will be set to 1. If the $V_{DD(REG)(3V3)}$ voltage dips from the normal operating range to below the BOD reset trip level and continues to decline to the level at which POR is asserted (nominally 1 V), the BODR bit is cleared. If the $V_{DD(REG)(3V3)}$ voltage rises continuously from below 1 V to a level above the BOD reset trip level, the BODR will be set to 1. This bit is not affected by External Reset nor Watchdog Reset. **Note:** Only in the case where a reset occurs and the POR = 0, the BODR bit indicates if the $V_{DD(REG)(3V3)}$ voltage was below the BOD reset trip level or not.	See text
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

5. Brown-out detection

The LPC17xx includes a Brown-Out Detector (BOD) that provides 2-stage monitoring of the voltage on the $V_{DD(REG)(3V3)}$ pins. If this voltage falls below the BOD interrupt trip level (typically 2.2 V under nominal room temperature conditions), the BOD asserts an interrupt signal to the NVIC. This signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC in order to cause a CPU interrupt; if not, software can monitor the signal by reading the Raw Interrupt Status Register.

The second stage of low-voltage detection asserts Reset to inactivate the LPC17xx when the voltage on the $V_{DD(REG)(3V3)}$ pins falls below the BOD reset trip level (typically 1.85 V under nominal room temperature conditions). This Reset prevents alteration of the flash as operation of the various elements of the chip would otherwise become unreliable due to low voltage. The BOD circuit maintains this reset down below 1 V, at which point the Power-On Reset circuitry maintains the overall Reset.

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

But when Brown-Out Detection is enabled to bring the LPC17xx out of Power-down mode (which is itself not a guaranteed operation -- see Section 4–8.7 “Power Mode Control register (PCON - 0x400F C0C0)”), the supply voltage may recover from a transient before the wake-up timer has completed its delay. In this case, the net result of the transient BOD is that the part wakes up and continues operation after the instructions that set Power-down mode, without any interrupt occurring and with the BOD bit in the RSID being 0. Since all other wake-up conditions have latching flags (see Section 3–6.2 “External Interrupt flag register (EXTINT - 0x400F C140)” and Section 27–6.2), a wake-up of this type, without any apparent cause, can be assumed to be a Brown-Out that has gone away.

© NXP B.V. 2010. All rights reserved.

6. External interrupt inputs

TheLPC17xx includes four External Interrupt Inputs as selectable pin functions. The logic of an individual external interrupt is represented in Figure 3–6. In addition, external interrupts have the ability to wake up the CPU from Power-down mode. Refer to Section 4–8.8 “Wake-up from Reduced Power Modes” for details.

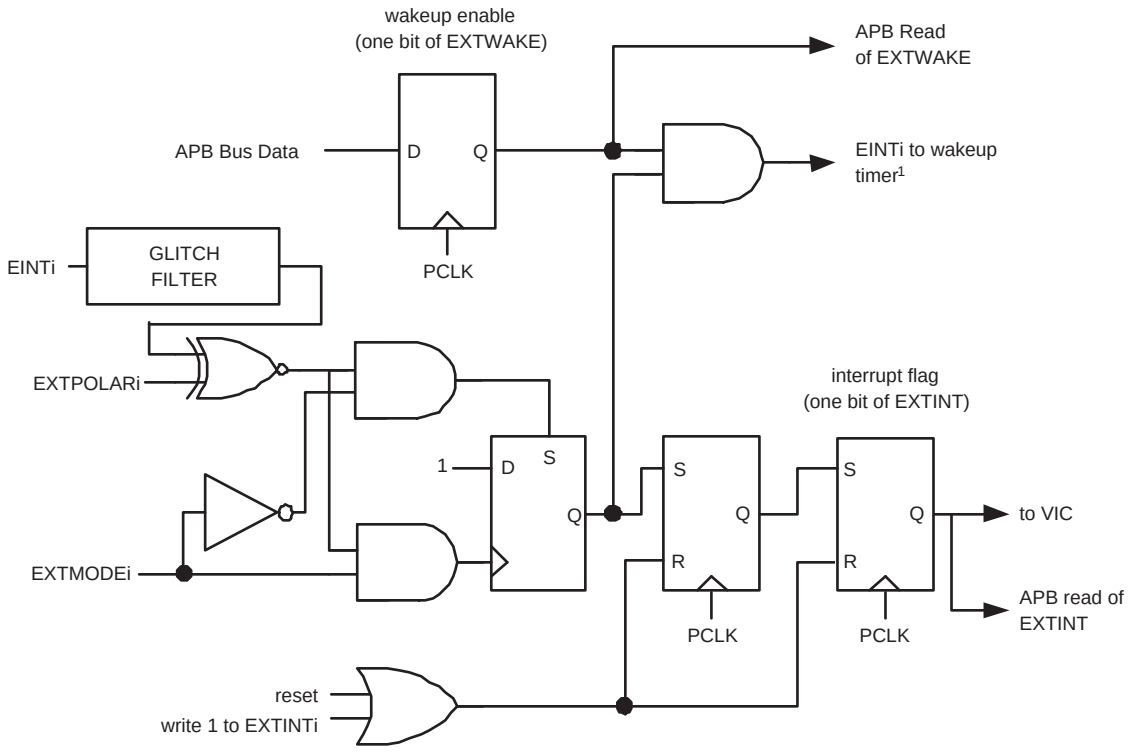


Fig 6. External interrupt logic

© NXP B.V. 2010. All rights reserved.

6.1 Register description

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

Table 9. External Interrupt registers

Name	Description	Access	Reset value[1]	Address
EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, EINT2 and EINT3. See Table 3–10.	R/W	0x00	0x400F C140
EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level-sensitive. See Table 3–11.	R/W	0x00	0x400F C148
EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt. See Table 3–12.	R/W	0x00	0x400F C14C

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

6.2 External Interrupt flag register (EXTINT - 0x400F 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 NVIC, which will cause an interrupt if interrupts from the pin are enabled.

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

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

Important: whenever a change of external interrupt operating mode (i.e. active level/edge) is performed (including the initialization of an external interrupt), the corresponding bit in the EXTINT register must be cleared! For details see Section 3–6.3 “External Interrupt Mode register (EXTMODE - 0x400F C148)” and Section 3–6.4 “External Interrupt Polarity register (EXTPOLAR - 0x400F C14C)”.

For example, if a system wakes up from Power-down using low level on external interrupt 0 pin, its post wake-up code must reset EINT0 bit in order to allow future entry into the Power-down mode. If EINT0 bit is left set to 1, 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.

© NXP B.V. 2010. All rights reserved.

Table 10. External Interrupt Flag register (EXTINT - address 0x400F C140) bit description

Bit	Symbol	Description	Reset value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.[1]	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

6.3 External Interrupt Mode register (EXTMODE - 0x400F 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 Section 8–5) and enabled in the appropriate NVIC register) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

© NXP B.V. 2010. All rights reserved.

Table 11. External Interrupt Mode register (EXTMODE - address 0x400F C148) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0	0	Level-sensitivity is selected for $\overline{\text{EINT0}}$.	0
		1	$\overline{\text{EINT0}}$ is edge sensitive.	
1	EXTMODE1	0	Level-sensitivity is selected for $\overline{\text{EINT1}}$.	0
		1	$\overline{\text{EINT1}}$ is edge sensitive.	
2	EXTMODE2	0	Level-sensitivity is selected for $\overline{\text{EINT2}}$.	0
		1	$\overline{\text{EINT2}}$ is edge sensitive.	
3	EXTMODE3	0	Level-sensitivity is selected for $\overline{\text{EINT3}}$.	0
		1	$\overline{\text{EINT3}}$ is edge sensitive.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.4 External Interrupt Polarity register (EXTPOLAR - 0x400F 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 Only pins that are selected for the EINT function (see Section 8–5) and enabled in the appropriate NVIC register) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

Table 12. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description

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

© NXP B.V. 2010. All rights reserved.

Table 12. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description

Bit	Symbol	Value	Description	Reset value
3	EXTPOLAR3	0	$\overline{\text{EINT3}}$ is low-active or falling-edge sensitive (depending on EXTMODE3).	0
		1	$\overline{\text{EINT3}}$ is high-active or rising-edge sensitive (depending on EXTMODE3).	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

7. Other system controls and status flags

Some aspects of controlling LPC17xx operation that do not fit into peripheral or other registers are grouped here.

7.1 System Controls and Status register (SCS - 0x400F C1A0)

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

Table 13. System Controls and Status register (SCS - address 0x400F C1A0) bit description

Bit	Symbol	Value	Description	Access	Reset value
3:0	-	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-	NA
4	OSCRANGE		Main oscillator range select.	R/W	0
		0	The frequency range of the main oscillator is 1 MHz to 20 MHz.		
		1	The frequency range of the main oscillator is 15 MHz to 25 MHz.		
5	OSCEN		Main oscillator enable.	R/W	0
		0	The main oscillator is disabled.		
		1	The main oscillator is enabled, and will start up if the correct external circuitry is connected to the XTAL1 and XTAL2 pins.		
6	OSCSTAT		Main oscillator status.	RO	0
		0	The main oscillator is not ready to be used as a clock source.		
		1	The main oscillator is ready to be used as a clock source. The main oscillator must be enabled via the OSCEN bit.		
31:7	-	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-	NA

© NXP B.V. 2010. All rights reserved.

1. Summary of clocking and power control functions

This section describes the generation of the various clocks needed by the LPC17xx and options of clock source selection, as well as power control and wake-up from reduced power modes. Functions described in the following subsections include:

- Oscillators
- Clock source selection
- PLLs
- Clock dividers
- APB dividers
- Power control
- Wake-up timer
- External clock output

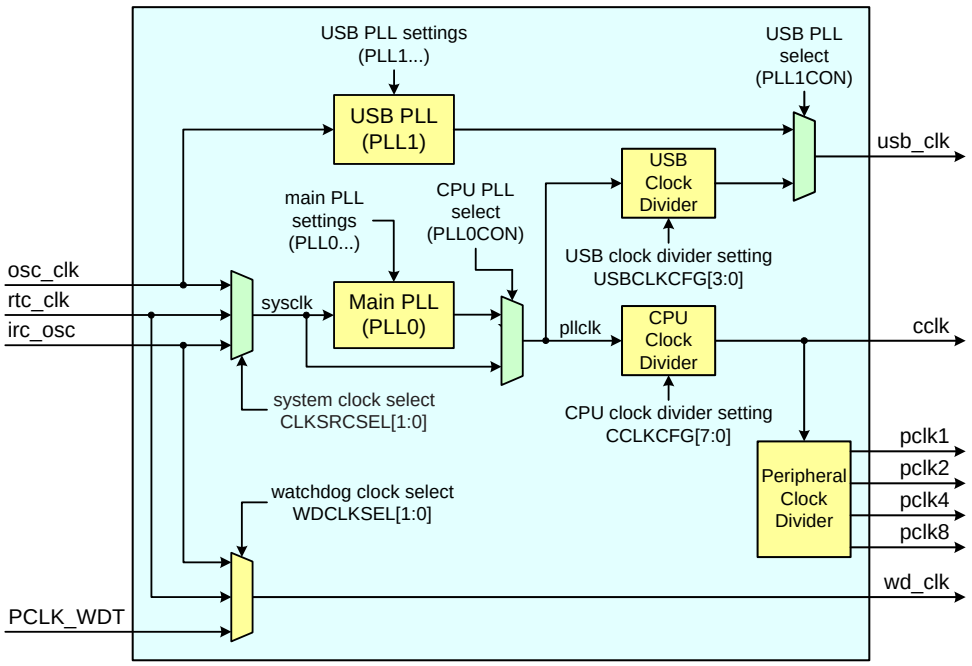


Fig 7. Clock generation for the LPC17xx

© NXP B.V. 2010. All rights reserved.

2. Register description

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

Table 14. Summary of system control registers

Name	Description	Access	Reset value	Address
Clock source selection				
CLKSRCSEL	Clock Source Select Register	R/W	0	0x400F C10C
Phase Locked Loop (PLL0, Main PLL)				
PLL0CON	PLL0 Control Register	R/W	0	0x400F C080
PLL0CFG	PLL0 Configuration Register	R/W	0	0x400F C084
PLL0STAT	PLL0 Status Register	RO	0	0x400F C088
PLL0FEED	PLL0 Feed Register	WO	NA	0x400F C08C
Phase Locked Loop (PLL1, USB PLL)				
PLL1CON	PLL1 Control Register	R/W	0	0x400F C0A0
PLL1CFG	PLL1 Configuration Register	R/W	0	0x400F C0A4
PLL1STAT	PLL1 Status Register	RO	0	0x400F C0A8
PLL1FEED	PLL1 Feed Register	WO	NA	0x400F C0AC
Clock dividers				
CCLKCFG	CPU Clock Configuration Register	R/W	0	0x400F C104
USBCLKCFG	USB Clock Configuration Register	R/W	0	0x400F C108
PCLKSEL0	Peripheral Clock Selection register 0.	R/W	0	0x400F C1A8
PCLKSEL1	Peripheral Clock Selection register 1.	R/W	0	0x400F C1AC
Power control				
PCON	Power Control Register	R/W	0	0x400F C0C0
PCONP	Power Control for Peripherals Register	R/W	0x03BE	0x400F C0C4
Utility				
CLKOUTCFG	Clock Output Configuration Register	R/W	0	0x400F C1C8

© NXP B.V. 2010. All rights reserved.

3. Oscillators

The LPC17xx includes three independent oscillators. These are the Main Oscillator, the Internal RC Oscillator, and the RTC oscillator. Each oscillator can be used for more than one purpose as required in a particular application. This can be seen in Figure 4–7.

Following Reset, the LPC17xx will operate from the Internal RC Oscillator until switched by software. This allows systems to operate without any external crystal, and allows the boot loader code to operate at a known frequency.

3.1 Internal RC oscillator

The Internal RC Oscillator (IRC) may be used as the clock source for the watchdog timer, and/or as the clock that drives PLL0 and subsequently the CPU. The precision of the IRC does not allow for use of the USB interface, which requires a much more precise time base in order to comply with the USB specification. Also, the IRC should not be used with the CAN1/2 block if the CAN baud rate is higher than 100 kbit/s.The nominal IRC frequency is 4 MHz.

Upon power-up or any chip reset, the LPC17xx uses the IRC as the clock source. Software may later switch to one of the other available clock sources.

3.2 Main oscillator

The main oscillator can be used as the clock source for the CPU, with or without using PLL0. The main oscillator operates at frequencies of 1 MHz to 25 MHz. This frequency can be boosted to a higher frequency, up to the maximum CPU operating frequency, by the Main PLL (PLL0). The oscillator output is called OSC_CLK. The clock selected as the PLL0 input is PLLCLKIN and the ARM processor clock frequency is referred to as CCLK for purposes of rate equations, etc. elsewhere in this document. The frequencies of PLLCLKIN and CCLK are the same value unless the PLL0 is active and connected. Refer to Section 4–5 “PLL0 (Phase Locked Loop 0)” for details.

The on-board oscillator in the LPC17xx 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 4–8, drawing a), with an amplitude between 200 mVrms and 1000 mVrms. This corresponds to a square wave signal with a signal swing of between 280 mV and 1.4 V. The XTAL2 pin in this configuration can be left unconnected.

External components and models used in oscillation mode are shown in Figure 4–8, drawings b and c, and in Table 4–15 and Table 4–16. 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 4–8, 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.

© NXP B.V. 2010. All rights reserved.

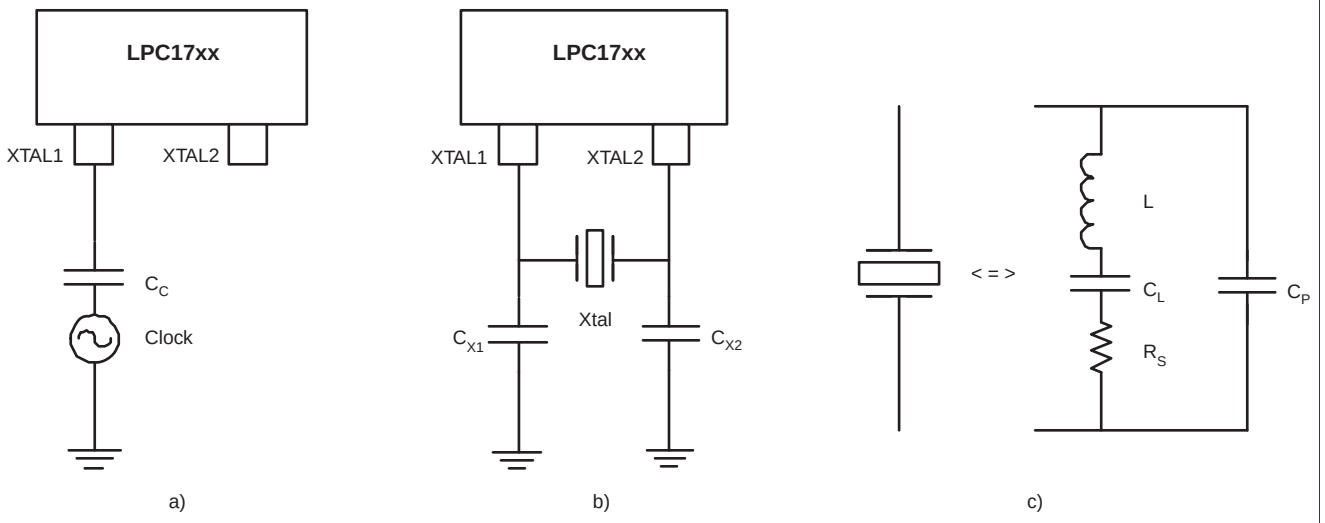


Fig 8. Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for $C_{X1}/_{X2}$ evaluation

Table 15. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) low frequency mode (OSCRANGE = 0, see Table 3–13)

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

Table 16. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) high frequency mode (OSCRANGE = 1, see Table 3–13)

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

Since chip operation always begins using the Internal RC Oscillator, and the main oscillator may not be used at all in some applications, it will only be started by software request. This is accomplished by setting the OSCEN bit in the SCS register, as described in Table 3–13. The main oscillator provides a status flag (the OSCSTAT bit in the SCS register) so that software can determine when the oscillator is running and stable. At that

© NXP B.V. 2010. All rights reserved.

point, software can control switching to the main oscillator as a clock source. Prior to starting the main oscillator, a frequency range must be selected by configuring the OSCRANGE bit in the SCS register.

3.3 RTC oscillator

The RTC oscillator provides a 1 Hz clock to the RTC and a 32 kHz clock output that can be used as the clock source for PLL0 and CPU and/or the watchdog timer.

© NXP B.V. 2010. All rights reserved.

4. Clock source selection multiplexer

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

The clock source selection can only be changed safely when PLL0 is not connected. For a detailed description of how to change the clock source in a system using PLL0 see Section 4–5.13 “PLL0 setup sequence”.

Note the following restrictions regarding the choice of clock sources:

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

4.1 Clock Source Select register (CLKSRCSEL - 0x400F C10C)

The CLKSRCSEL register contains the bits that select the clock source for PLL0.

Table 17. Clock Source Select register (CLKSRCSEL - address 0x400F C10C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CLKSRC		Selects the clock source for PLL0 as follows:	0
		00	Selects the Internal RC oscillator as the PLL0 clock source (default).	
		01	Selects the main oscillator as the PLL0 clock source.	
		10	Selects the RTC oscillator as the PLL0 clock source.	
		11	Reserved, do not use this setting.	
		Warning: Improper setting of this value, or an incorrect sequence of changing this value may result in incorrect operation of the device.		
31:2	-	0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

5. PLL0 (Phase Locked Loop 0)

PLL0 accepts an input clock frequency in the range of 32 kHz to 50 MHz. The clock source is selected in the CLKSRCSEL register (see Section 4–4). The input frequency is multiplied up to a high frequency, then divided down to provide the actual clock used by the CPU, peripherals, and optionally the USB subsystem. Note that the USB subsystem has its own dedicated PLL (see Section 4–6). PLL0 can produce a clock up to the maximum allowed for the CPU, which is 120 MHz on high speed versions (LPC1769 and LPC1759), and 100 MHz on other versions.

5.1 PLL0 operation

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

Following the PLL input divider is the PLL multiplier. This can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 6 through 512, plus additional values listed in Table 4–21. The resulting frequency must be in the range of 275 MHz to 550 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is used to adjust the CCO frequency.

There are additional dividers at the output of PLL0 to bring the frequency down to what is needed for the CPU, peripherals, and potentially the USB subsystem. PLL0 output dividers are described in the Clock Dividers section following the PLL0 description. A block diagram of PLL0 is shown in Figure 4–9

PLL activation is controlled via the PLL0CON register. PLL0 multiplier and divider values are controlled by the PLL0CFG register. These two registers are protected in order to prevent accidental alteration of PLL0 parameters or deactivation of the PLL. Since all chip operations, including the Watchdog Timer, could be dependent on PLL0 if so configured (for example when it is providing the chip clock), accidental changes to the PLL0 setup values could result in unexpected or fatal behavior of the microcontroller. The protection is accomplished by a feed sequence similar to that of the Watchdog Timer. Details are provided in the description of the PLL0FEED register.

PLL0 is turned off and bypassed following a chip Reset and by entering Power-down mode. PLL0 must be configured, enabled, and connected to the system by software.

It is important that the setup procedure described in Section 4–5.13 “PLL0 setup sequence” is followed or PLL0 might not operate at all!

5.1.1 PLL0 and startup/boot code interaction

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

© NXP B.V. 2010. All rights reserved.

5.2 PLL0 register description

PLL0 is controlled by the registers shown in Table 4–18. More detailed descriptions follow.

Warning: Improper setting of PLL0 values may result in incorrect operation of the device!

Table 18. PLL0 registers

Name	Description	Access	Reset value[1]	Address
PLL0CON	PLL0 Control Register. Holding register for updating PLL0 control bits. Values written to this register do not take effect until a valid PLL0 feed sequence has taken place.	R/W	0	0x400F C080
PLL0CFG	PLL0 Configuration Register. Holding register for updating PLL0 configuration values. Values written to this register do not take effect until a valid PLL0 feed sequence has taken place.	R/W	0	0x400F C084
PLL0STAT	PLL0 Status Register. Read-back register for PLL0 control and configuration information. If PLL0CON or PLL0CFG have been written to, but a PLL0 feed sequence has not yet occurred, they will not reflect the current PLL0 state. Reading this register provides the actual values controlling the PLL0, as well as the PLL0 status.	RO	0	0x400F C088
PLL0FEED	PLL0 Feed Register. This register enables loading of the PLL0 control and configuration information from the PLL0CON and PLL0CFG registers into the shadow registers that actually affect PLL0 operation.	WO	NA	0x400F C08C

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

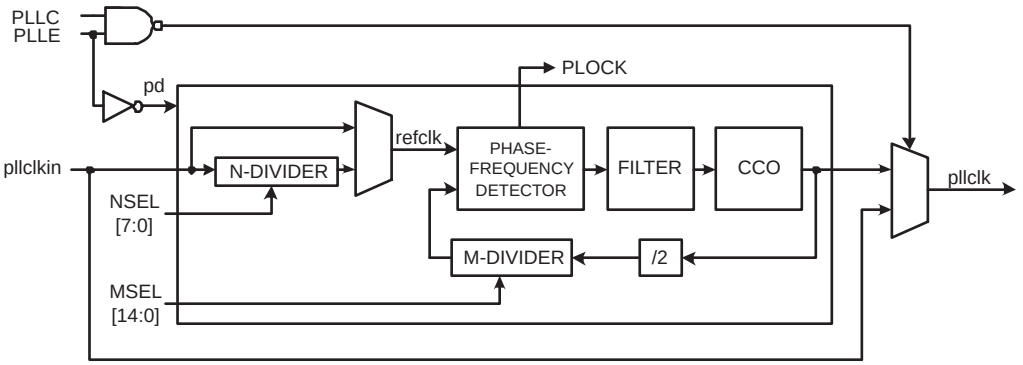


Fig 9. PLL0 block diagram

5.3 PLL0 Control register (PLL0CON - 0x400F C080)

The PLL0CON register contains the bits that enable and connect PLL0. Enabling PLL0 allows it to attempt to lock to the current settings of the multiplier and divider values. Connecting PLL0 causes the processor and most chip functions to run from the PLL0

© NXP B.V. 2010. All rights reserved.

output clock. Changes to the PLL0CON register do not take effect until a correct PLL0 feed sequence has been given (see Section 4–5.8 “PLL0 Feed register (PLL0FEED - 0x400F C08C)”).

Table 19. PLL Control register (PLL0CON - address 0x400F C080) bit description

Bit	Symbol	Description	Reset value
0	PLLE0	PLL0 Enable. When one, and after a valid PLL0 feed, this bit will activate PLL0 and allow it to lock to the requested frequency. See PLL0STAT register, Table 4–22.	0
1	PLLC0	PLL0 Connect. Setting PLLC0 to one after PLL0 has been enabled and locked, then followed by a valid PLL0 feed sequence causes PLL0 to become the clock source for the CPU, AHB peripherals, and used to derive the clocks for APB peripherals. The PLL0 output may potentially be used to clock the USB subsystem if the frequency is 48 MHz. See PLL0STAT register, Table 4–22.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL0 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 PLL0 output or vice versa, internal circuitry synchronizes the operation in order to ensure that glitches are not generated. Hardware does not insure that PLL0 is locked before it is connected or automatically disconnect PLL0 if lock is lost during operation. In the event of loss of lock on PLL0, it is likely that the oscillator clock has become unstable and disconnecting PLL0 will not remedy the situation.

5.4 PLL0 Configuration register (PLL0CFG - 0x400F C084)

The PLL0CFG register contains PLL0 multiplier and divider values. Changes to the PLL0CFG register do not take effect until a correct PLL feed sequence has been given (see Section 4–5.8 “PLL0 Feed register (PLL0FEED - 0x400F C08C)”). Calculations for the PLL frequency, and multiplier and divider values are found in the Section 4–5.10 “PLL0 frequency calculation”.

Table 20. PLL0 Configuration register (PLL0CFG - address 0x400F C084) bit description

Bit	Symbol	Description	Reset value
14:0	MSEL0	PLL0 Multiplier value. Supplies the value "M" in PLL0 frequency calculations. The value stored here is M - 1. Supported values for M are 6 through 512 and those listed in Table 4–21. **Note:** Not all values of M are needed, and therefore some are not supported by hardware. For details on selecting values for MSEL0 see Section 4–5.10 “PLL0 frequency calculation”.	0
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
23:16	NSEL0	PLL0 Pre-Divider value. Supplies the value "N" in PLL0 frequency calculations. The value stored here is N - 1. Supported values for N are 1 through 32. **Note:** For details on selecting the right value for NSEL0 see Section 4–5.10 “PLL0 frequency calculation”.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

Table 21. Multiplier values for PLL0 with a 32 kHz input

Multiplier (M)	Pre-divide (N)	F_{CCO}	Multiplier (M)	Pre-divide (N)	F_{CCO}
4272	1	279.9698	12085	2	396.0013
4395	1	288.0307	12207	2	399.9990
4578	1	300.0238	12817	2	419.9875
4725	1	309.6576	12817	3	279.9916
4807	1	315.0316	13184	2	432.0133
5127	1	336.0031	13184	3	288.0089
5188	1	340.0008	13672	2	448.0041
5400	1	353.8944	13733	2	450.0029
5493	1	359.9892	13733	3	300.0020
5859	1	383.9754	13916	2	455.9995
6042	1	395.9685	14099	2	461.9960
6075	1	398.1312	14420	3	315.0097
6104	1	400.0317	14648	2	479.9857
6409	1	420.0202	15381	2	504.0046
6592	1	432.0133	15381	3	336.0031
6750	1	442.3680	15564	3	340.0008
6836	1	448.0041	15625	2	512.0000
6866	1	449.9702	15869	2	519.9954
6958	1	455.9995	16113	2	527.9908
7050	1	462.0288	16479	3	359.9892
7324	1	479.9857	17578	3	383.9973
7425	1	486.6048	18127	3	395.9904
7690	1	503.9718	18311	3	400.0099
7813	1	512.0328	19226	3	419.9984
7935	1	520.0282	19775	3	431.9915
8057	1	528.0236	20508	3	448.0041
8100	1	530.8416	20599	3	449.9920
8545	2	280.0026	20874	3	455.9995
8789	2	287.9980	21149	3	462.0070
9155	2	299.9910	21973	3	480.0075
9613	2	314.9988	23071	3	503.9937
10254	2	336.0031	23438	3	512.0109
10376	2	340.0008	23804	3	520.0063
10986	2	359.9892	24170	3	528.0017
11719	2	384.0082			

© NXP B.V. 2010. All rights reserved.

5.5 PLL0 Status register (PLL0STAT - 0x400F C088)

The read-only PLL0STAT register provides the actual PLL0 parameters that are in effect at the time it is read, as well as PLL0 status. PLL0STAT may disagree with values found in PLL0CON and PLL0CFG because changes to those registers do not take effect until a proper PLL0 feed has occurred (see Section 4–5.8 “PLL0 Feed register (PLL0FEED - 0x400F C08C)”).

Table 22. PLL Status register (PLL0STAT - address 0x400F C088) bit description

Bit	Symbol	Description	Reset value
14:0	MSEL0	Read-back for the PLL0 Multiplier value. This is the value currently used by PLL0, and is one less than the actual multiplier.	0
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
23:16	NSEL0	Read-back for the PLL0 Pre-Divider value. This is the value currently used by PLL0, and is one less than the actual divider.	0
24	PLLE0_STAT	Read-back for the PLL0 Enable bit. This bit reflects the state of the PLEC0 bit in PLL0CON (see Table 4–19) after a valid PLL0 feed. When one, PLL0 is currently enabled. When zero, PLL0 is turned off. This bit is automatically cleared when Power-down mode is entered.	0
25	PLLC0_STAT	Read-back for the PLL0 Connect bit. This bit reflects the state of the PLLC0 bit in PLL0CON (see Table 4–19) after a valid PLL0 feed. When PLLC0 and PLLE0 are both one, PLL0 is connected as the clock source for the CPU. When either PLLC0 or PLLE0 is zero, PLL0 is bypassed. This bit is automatically cleared when Power-down mode is entered.	0
26	PLOCK0	Reflects the PLL0 Lock status. When zero, PLL0 is not locked. When one, PLL0 is locked onto the requested frequency. See text for details.	0
31:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.6 PLL0 Interrupt: PLOCK0

The PLOCK0 bit in the PLL0STAT register reflects the lock status of PLL0. When PLL0 is enabled, or parameters are changed, PLL0 requires some time to establish lock under the new conditions. PLOCK0 can be monitored to determine when PLL0 may be connected for use. The value of PLOCK0 may not be stable when the PLL reference frequency (F_{REF}, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 μs when FREF is greater than 400 kHz and 200 / FREF seconds when FREF is less than 400 kHz

PLOCK0 is connected to the interrupt controller. This allows for software to turn on PLL0 and continue with other functions without having to wait for PLL0 to achieve lock. When the interrupt occurs, PLL0 may be connected, and the interrupt disabled. PLOCK0 appears as interrupt 32 in Table 6–50. Note that PLOCK0 remains asserted whenever PLL0 is locked, so if the interrupt is used, the interrupt service routine must disable the PLOCK0 interrupt prior to exiting.

© NXP B.V. 2010. All rights reserved.

5.7 PLL0 Modes

The combinations of PLLE0 and PLLC0 are shown in Table 4–23.

Table 23. PLL control bit combinations

PLLC0	PLLE0	PLL Function
0	0	PLL0 is turned off and disconnected. PLL0 outputs the unmodified clock input.
0	1	PLL0 is active, but not yet connected. PLL0 can be connected after PLOCK0 is asserted.
1	0	Same as 00 combination. This prevents the possibility of PLL0 being connected without also being enabled.
1	1	PLL0 is active and has been connected as the system clock source.

5.8 PLL0 Feed register (PLL0FEED - 0x400F C08C)

A correct feed sequence must be written to the PLL0FEED register in order for changes to the PLL0CON and PLL0CFG registers to take effect. The feed sequence is:

1. Write the value 0xAA to PLL0FEED.
2. Write the value 0x55 to PLL0FEED.

The two writes must be in the correct sequence, and there must be no other register access in the same address space (0x400F C000 to 0x400F FFFF) between them. Because of this, it may be necessary to disable interrupts for the duration of the PLL0 feed operation, if there is a possibility that an interrupt service routine could write to another register in that space. If either of the feed values is incorrect, or one of the previously mentioned conditions is not met, any changes to the PLL0CON or PLL0CFG register will not become effective.

Table 24. PLL Feed register (PLL0FEED - address 0x400F C08C) bit description

Bit	Symbol	Description	Reset value
7:0	PLL0FEED	The PLL0 feed sequence must be written to this register in order for PLL0 configuration and control register changes to take effect.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.9 PLL0 and Power-down mode

Power-down mode automatically turns off and disconnects PLL0. Wake-up from Power-down mode does not automatically restore PLL0 settings, this must be done in software. Typically, a routine to activate PLL0, wait for lock, and then connect PLL0 can be called at the beginning of any interrupt service routine that might be called due to the wake-up. It is important not to attempt to restart PLL0 by simply feeding it when execution resumes after a wake-up from Power-down mode. This would enable and connect PLL0 at the same time, before PLL lock is established.

5.10 PLL0 frequency calculation

PLL0 equations use the following parameters:

© NXP B.V. 2010. All rights reserved.

Table 25. PLL frequency parameter

Parameter	Description
F_{IN}	the frequency of PLLCLKIN from the Clock Source Selection Multiplexer.
F_{CCO}	the frequency of the PLLCLK (output of the PLL Current Controlled Oscillator)
N	PLL0 Pre-divider value from the NSEL0 bits in the PLL0CFG register (PLL0CFG NSEL0 field + 1). N is an integer from 1 through 32.
M	PLL0 Multiplier value from the MSEL0 bits in the PLL0CFG register (PLL0CFG MSEL0 field + 1). Not all potential values are supported. See below.
F_{REF}	PLL internal reference frequency, F_{IN} divided by N.

The PLL0 output frequency (when PLL0 is both active and connected) is given by:

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

PLL inputs and settings must meet the following:

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

The equation can be solved for other PLL parameters:

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

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

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

Allowed values for M:

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

For lower frequencies, specifically when the RTC is used to clock PLL0, a set of 65 additional M values have been selected for supporting baud rate generation, CAN/USB operation, and obtaining integer MHz frequencies. These values are shown in Table 4–26.

© NXP B.V. 2010. All rights reserved.

Table 26. Additional Multiplier Values for use with a Low Frequency Clock Input

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

5.11 Procedure for determining PLL0 settings

PLL0 parameter determination can be simplified by using a spreadsheet available from NXP. To determine PLL0 parameters by hand, the following general procedure may be used:

1. Determine if the application requires use of the USB interface, and whether it will be clocked from PLL0. The USB requires a 50% duty cycle clock of 48 MHz within a very small tolerance, which means that F_{CCO} must be an even integer multiple of 48 MHz (i.e. an integer multiple of 96 MHz), within a very small tolerance.
2. Choose the desired processor operating frequency (CCLK). This may be based on processor throughput requirements, need to support a specific set of UART baud rates, etc. Bear in mind that peripheral devices may be running from a lower clock frequency than that of the processor (see Section 4–7 “Clock dividers” on page 55 and Section 4–8 “Power control” on page 59). Find a value for F_{CCO} that is close to a multiple of the desired CCLK frequency, bearing in mind the requirement for USB support in [1] above, and that lower values of F_{CCO} result in lower power dissipation.
3. Choose a value for the PLL input frequency (F_{IN}). This can be a clock obtained from the main oscillator, the RTC oscillator, or the on-chip RC oscillator. For USB support, the main oscillator should be used. Bear in mind that if PLL1 rather than PLL0 is used to clock the USB subsystem, this affects the choice of the main oscillator frequency.
4. Calculate values for M and N to produce a sufficiently accurate F_{CCO} frequency. The desired M value -1 will be written to the MSEL0 field in PLL0CFG. The desired N value -1 will be written to the NSEL0 field in PLL0CFG.

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

© NXP B.V. 2010. All rights reserved.

5.12 Examples of PLL0 settings

The following table gives a summary of examples that illustrate selecting PLL0 values based on different system requirements.

Table 27. Summary of PLL0 examples

Example	Description
1	• The PLL0 clock source is 10 MHz. • PLL0 is not used as the USB clock source, or the USB interface is not used. • The desired CPU clock is 100 MHz.
2	• The PLL0 clock source is 4 MHz. • PLL0 is used as the USB clock source. • The desired CPU clock is 60 MHz.
3	• The PLL0 clock source is the 32.768 kHz RTC clock. • PLL0 is not used as the USB clock source, or the USB interface is not used. • The desired CPU clock is 72 MHz.

Example 1

Assumptions:

- The USB interface will not be used in the application, or will be clocked by PLL1.
- The desired CPU rate is 100 MHz.
- An external 10 MHz crystal or clock source will be used as the system clock source.

Calculations:

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

A smaller value for the PLL pre-divide (N) as well as a smaller value of the multiplier (M), both result in better PLL operational stability and lower output jitter. Lower values of F_{CCO} also save power. So, the process of determining PLL setup parameters involves looking for the smallest N and M values giving the lowest F_{CCO} value that will support the required CPU and/or USB clocks. It is usually easier to work backward from the desired output clock rate and determine a target F_{CCO} rate, then find a way to obtain that F_{CCO} rate from the available input clock.

Potential precise values of F_{CCO} are integer multiples of the desired CPU clock. In this example, it is clear that the smallest frequency for F_{CCO} that can produce the desired CPU clock rate and is within the PLL0 operating range of 275 to 550 MHz is 300 MHz (3×100 MHz).

Assuming that the PLL pre-divide is 1 (N = 1), the equation above gives $M = ((300 \times 10^6 \times 1) / (2 \times 10 \times 10^6) = 300 / 20 = 15$. Since the result is an integer, there is no need to look any further for a good set of PLL0 configuration values. The value written to PLL0CFG would be 0x0E (N - 1 = 0; M - 1 = 14 gives 0x0E).

The PLL output must be further divided in order to produce the CPU clock. This is accomplished using a separate divider that is described later in this chapter, see Section 4–7.1.

© NXP B.V. 2010. All rights reserved.

Example 2

Assumptions:

- The USB interface will be used in the application and will be clocked from PLL0.
- The desired CPU rate is 60 MHz.
- An external 4 MHz crystal or clock source will be used as the system clock source. This clock source could be the Internal RC oscillator (IRC).

Calculations:

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

Because supporting USB requires a precise 48 MHz clock with a 50% duty cycle, that need must be addressed first. Potential precise values of F_{CCO} are integer multiples of the 2 × the 48 MHz USB clock. The 2 × insures that the clock has a 50% duty cycle, which would not be the case for a division of the PLL output by an odd number.

The possibilities for the F_{CCO} rate when the USB is used are 288 MHz, 384 MHz, and 480 MHz. The smallest frequency for F_{CCO} that can produce a valid USB clock rate and is within the PLL0 operating range is 288 MHz ($3 \times 2 \times 48$ MHz).

Start by assuming N = 1, since this produces the smallest multiplier needed for PLL0. So, $M = ((288 \times 10^6) \times 1) / (2 \times (4 \times 10^{6)}) = 288 / 8 = 36$. The result is an integer, which is necessary to obtain a precise USB clock. The value written to PLL0CFG would be 0x23 (N - 1 = 0; M - 1 = 35 = 0x23).

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

If it is important to obtain exactly 60 MHz, an F_{CCO} rate must be found that can be divided down to both 48 MHz and 60 MHz. As previously noted, the possibilities for the F_{CCO} rate when the USB is used are 288 MHz, 384 MHz, and 480 MHz. Of these, only is 480 MHz is also evenly divisible by 60. Divided by 10, this gives the 48 MHz with a 50% duty cycle needed by the USB subsystem. Divided by 8, it gives 60 MHz for the CPU clock. PLL0 settings for 480 MHz are N = 1 and M = 60.

The PLL output must be further divided in order to produce both the CPU clock and the USB clock. This is accomplished using separate dividers that are described later in this chapter. See Section 4–7.1 and Section 4–7.2.

© NXP B.V. 2010. All rights reserved.

Example 3

Assumptions:

- The USB interface will not be used in the application, or will be clocked by PLL1.
- The desired CPU rate is 72 MHz
- The 32.768 kHz RTC clock source will be used as the system clock source

Calculations:

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

The smallest integer multiple of the desired CPU clock rate that is within the PLL0 operating range is 288 MHz (4 × 72 MHz).

Using the equation above and assuming that N = 1, M = $((288 \times 10^6) \times 1) / (2 \times 32{,}768)$ = 4,394.53125. This is not an integer, so the CPU frequency will not be exactly 72 MHz with this setting. Since this example is less obvious, it may be useful to make a table of possibilities for different values of N (see below).

Table 28. Potential values for PLL example

N	M	M Rounded	F_{REF} in Hz (F_{IN} / N)	F_{CCO} in MHz (F_{REF} x M)	CCLK in MHz (F_{CCO} / 4)	% Error (CCLK-72) / 72
1	4394.53125	4395	32768	288.0307	72.0077	0.0107
2	8789.0625	8789	16384	287.9980	71.9995	-0.0007
3	13183.59375	13184	10922.67	288.0089	72.0022	0.0031
4	17578.125	17578	8192	287.9980	71.9995	-0.0007
5	21972.65625	21973	6553.6	288.0045	72.0011	0.0016

Beyond N = 5, the value of M is out of range or not supported, so the table stops at that point. In the third column of the table, the calculated M value is rounded to the nearest integer. If this results in CCLK being above the maximum operating frequency, it is allowed if it is not more than 1/2 % above the maximum frequency.

In general, larger values of F_{REF} result in a more stable PLL when the input clock is a low frequency. Even the first table entry shows a very small error of just over 1 hundredth of a percent, or 107 parts per million (ppm). If that is not accurate enough in the application, the second case gives a much smaller error of 7 ppm. There are no allowed combinations that give a smaller error than that.

Remember that when a frequency below about 1 MHz is used as the PLL0 clock source, not all multiplier values are available. As it turns out, all of the rounded M values found in Table 4–28 of this example are supported, which may be confirmed in Table 4–26. If PLL0 calculations suggest use of unsupported multiplier values, those values must be disregarded and other values examined to find the best fit.

The value written to PLL0CFG for the second table entry would be 0x12254 (N - 1 = 1 = 0x1; M - 1 = 8788 = 0x2254).

The PLL output must be further divided in order to produce the CPU clock. This is accomplished using a separate divider that is described later in this chapter, see Section 4–7.1.

© NXP B.V. 2010. All rights reserved.

5.13 PLL0 setup sequence

The following sequence must be followed step by step in order to have PLL0 initialized and running:

1. Disconnect PLL0 with one feed sequence if PLL0 is already connected.
2. Disable PLL0 with one feed sequence.
3. Change the CPU Clock Divider setting to speed up operation without PLL0, if desired.
4. Write to the Clock Source Selection Control register to change the clock source if needed.
5. Write to the PLL0CFG and make it effective with one feed sequence. The PLL0CFG can only be updated when PLL0 is disabled.
6. Enable PLL0 with one feed sequence.
7. Change the CPU Clock Divider setting for the operation with PLL0. It is critical to do this before connecting PLL0.
8. Wait for PLL0 to achieve lock by monitoring the PLOCK0 bit in the PLL0STAT register, or using the PLOCK0 interrupt, or wait for a fixed time when the input clock to PLL0 is slow (i.e. 32 kHz). The value of PLOCK0 may not be stable when the PLL reference frequency (FREF, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 µs when FREF is greater than 400 kHz and 200 / FREF seconds when FREF is less than 400 kHz.
9. Connect PLL0 with one feed sequence.

It is very important not to merge any steps above. For example, do not update the PLL0CFG and enable PLL0 simultaneously with the same feed sequence.

© NXP B.V. 2010. All rights reserved.

6. PLL1 (Phase Locked Loop 1)

PLL1 receives its clock input from the main oscillator only and can be used to provide a fixed 48 MHz clock only to the USB subsystem. This is an option in addition to the possibility of generating the USB clock from PLL0.

PLL1 is disabled and powered off on reset. If PLL1 is left disabled, the USB clock can be supplied by PLL0 if everything is set up to provide 48 MHz through that route. If PLL1 is enabled and connected via the PLL1CON register (see Section 4–6.2), it is automatically selected to drive the USB subsystem (see Figure 4–7).

PLL1 activation is controlled via the PLL1CON register. PLL1 multiplier and divider values are controlled by the PLL1CFG register. These two registers are protected in order to prevent accidental alteration of PLL1 parameters or deactivation of PLL1. The protection is accomplished by a feed sequence similar to that of the Watchdog Timer. Details are provided in the description of the PLL1FEED register.

PLL1 accepts an input clock frequency in the range of 10 MHz to 25 MHz only. The input frequency is multiplied up to the range of 48 MHz for the USB clock using a Current Controlled Oscillator (CCO). The multiplier can be an integer value from 1 to 32 (for USB, the multiplier value cannot be higher than 4. 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 PLL1 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 output of PLL1 has a 50% duty cycle. A block diagram of PLL1 is shown in Figure 4–10.

6.1 PLL1 register description

PLL1 is controlled by the registers shown in Table 4–29. More detailed descriptions follow. Writes to any unused bits are ignored. A read of any unused bits will return a logic zero.

Warning: Improper setting of PLL1 values may result in incorrect operation of the USB subsystem!

Table 29. PLL1 registers

Name	Description	Access	Reset value[1]	Address
PLL1CON	PLL1 Control Register. Holding register for updating PLL1 control bits. Values written to this register do not take effect until a valid PLL1 feed sequence has taken place.	R/W	0	0x400F C0A0

© NXP B.V. 2010. All rights reserved.

Table 29. PLL1 registers

Name	Description	Access	Reset value[1]	Address
PLL1CFG	PLL1 Configuration Register. Holding register for updating PLL1 configuration values. Values written to this register do not take effect until a valid PLL1 feed sequence has taken place.	R/W	0	0x400F C0A4
PLL1STAT	PLL1 Status Register. Read-back register for PLL1 control and configuration information. If PLL1CON or PLL1CFG have been written to, but a PLL1 feed sequence has not yet occurred, they will not reflect the current PL1L state. Reading this register provides the actual values controlling PLL1, as well as PLL1 status.	RO	0	0x400F C0A8
PLL1FEED	PLL1 Feed Register. This register enables loading of PLL1 control and configuration information from the PLL1CON and PLL1CFG registers into the shadow registers that actually affect PLL1 operation.	WO	NA	0x400F C0AC

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

© NXP B.V. 2010. All rights reserved.

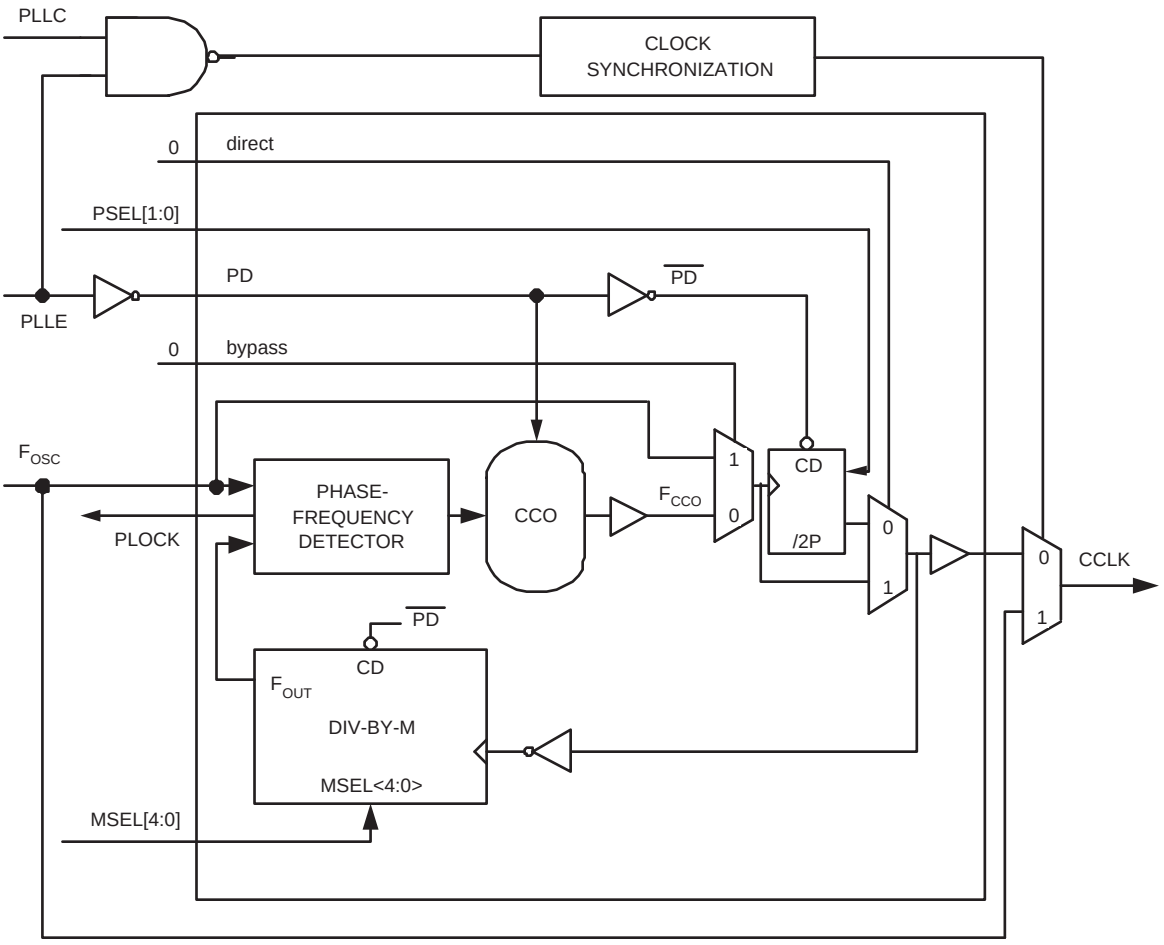


Fig 10. PLL1 block diagram

6.2 PLL1 Control register (PLL1CON - 0x400F C0A0)

The PLL1CON register contains the bits that enable and connect PLL1. Enabling PLL1 allows it to attempt to lock to the current settings of the multiplier and divider values. Connecting PLL1 causes the USB subsystem to run from the PLL1 output clock. Changes to the PLL1CON register do not take effect until a correct PLL feed sequence has been given (see Section 4–6.6 and Section 4–6.3).

© NXP B.V. 2010. All rights reserved.

Table 30. PLL1 Control register (PLL1CON - address 0x400F C0A0) bit description

Bit	Symbol	Description	Reset value
0	PLLE1	PLL1 Enable. When one, and after a valid PLL1 feed, this bit will activate PLL1 and allow it to lock to the requested frequency. See PLL1STAT register, Table 4–32.	0
1	PLLC1	PLL1 Connect. Setting PLLC to one after PLL1 has been enabled and locked, then followed by a valid PLL1 feed sequence causes PLL1 to become the clock source for the USB subsystem via the USB clock divider. See PLL1STAT register, Table 4–32.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL1 must be set up, enabled, and lock established before it may be used as a clock source for the USB subsystem. The hardware does not insure that the PLL is locked before it is connected nor does it automatically disconnect the PLL if lock is lost during operation.

6.3 PLL1 Configuration register (PLL1CFG - 0x400F C0A4)

The PLL1CFG register contains the PLL1 multiplier and divider values. Changes to the PLL1CFG register do not take effect until a correct PLL1 feed sequence has been given (see Section 4–6.6). Calculations for the PLL1 frequency, and multiplier and divider values are found in Section 4–6.9.

Table 31. PLL Configuration register (PLL1CFG - address 0x400F C0A4) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL1	PLL1 Multiplier value. Supplies the value "M" in the PLL1 frequency calculations. **Note:** For details on selecting the right value for MSEL1 see Section 4–5.10.	0
6:5	PSEL1	PLL1 Divider value. Supplies the value "P" in the PLL1 frequency calculations. **Note:** For details on selecting the right value for PSEL1 see Section 4–5.10.	0
31:7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.4 PLL1 Status register (PLL1STAT - 0x400F C0A8)

The read-only PLL1STAT register provides the actual PLL1 parameters that are in effect at the time it is read, as well as the PLL1 status. PLL1STAT may disagree with values found in PLL1CON and PLL1CFG because changes to those registers do not take effect until a proper PLL1 feed has occurred (see Section 4–6.6 “PLL1 Feed register (PLL1FEED - 0x400F C0AC)”).

© NXP B.V. 2010. All rights reserved.

Table 32. PLL1 Status register (PLL1STAT - address 0x400F C0A8) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL1	Read-back for the PLL1 Multiplier value. This is the value currently used by PLL1.	0
6:5	PSEL1	Read-back for the PLL1 Divider value. This is the value currently used by PLL1.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	PLLE1_STAT	Read-back for the PLL1 Enable bit. When one, PLL1 is currently activated. When zero, PLL1 is turned off. This bit is automatically cleared when Power-down mode is activated.	0
9	PLLC1_STAT	Read-back for the PLL1 Connect bit. When PLLC and PLLE are both one, PLL1 is connected as the clock source for the microcontroller. When either PLLC or PLLE is zero, PLL1 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	PLOCK1	Reflects the PLL1 Lock status. When zero, PLL1 is not locked. When one, PLL1 is locked onto the requested frequency.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.4.1 PLL1 modes

The combinations of PLLE1 and PLLC1 are shown in Table 4–33.

Table 33. PLL1 control bit combinations

PLLC1	PLLE1	PLL1 Function
0	0	PLL1 is turned off and disconnected.
0	1	PLL1 is active, but not yet connected. PLL1 can be connected after PLOCK1 is asserted.
1	0	Same as 00 combination. This prevents the possibility of PLL1 being connected without also being enabled.
1	1	PLL1 is active and has been connected. The clock for the USB subsystem is sourced from PLL1.

6.5 PLL1 Interrupt: PLOCK1

The PLOCK1 bit in the PLL1STAT register reflects the lock status of PLL1. When PLL1 is enabled, or parameters are changed, the PLL requires some time to establish lock under the new conditions. PLOCK1 can be monitored to determine when the PLL may be connected for use.

PLOCK1 is connected to the interrupt controller. This allows for software to turn on the PLL and continue with other functions without having to wait for the PLL to achieve lock. When the interrupt occurs, the PLL may be connected, and the interrupt disabled. PLOCK1 appears as interrupt 48 in Table 6–50. Note that PLOCK1 remains asserted whenever PLL1 is locked, so if the interrupt is used, the interrupt service routine must disable the PLOCK1 interrupt prior to exiting.

© NXP B.V. 2010. All rights reserved.

6.6 PLL1 Feed register (PLL1FEED - 0x400F C0AC)

A correct feed sequence must be written to the PLL1FEED register in order for changes to the PLL1CON and PLL1CFG registers to take effect. The feed sequence is:

1. Write the value 0xAA to PLL1FEED.
2. Write the value 0x55 to PLL1FEED.

The two writes must be in the correct sequence, and there must be no other register access in the same address space (0x400F C000 to 0x400F FFFF) between them. Because of this, it may be necessary to disable interrupts for the duration of the PLL feed operation, if there is a possibility that an interrupt service routine could write to another register in that space. If either of the feed values is incorrect, or one of the previously mentioned conditions is not met, any changes to the PLL1CON or PLL1CFG register will not become effective.

Table 34. PLL1 Feed register (PLL1FEED - address 0x400F C0AC) bit description

Bit	Symbol	Description	Reset value
7:0	PLL1FEED	The PLL1 feed sequence must be written to this register in order for PLL1 configuration and control register changes to take effect.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.7 PLL1 and Power-down mode

Power-down mode automatically turns off and disconnects activated PLL(s). Wake-up from Power-down mode does not automatically restore 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 wake-up. It is important not to attempt to restart a PLL by simply feeding it when execution resumes after a wake-up 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 the microcontroller from Power-down mode (see Section 4–8.8 for details of wake up from reduced modes), both the Main PLL (PLL0) and the USB PLL (PLL1) will be automatically be turned off and disconnected when Power-down mode is invoked, as described above. However, if the USB activity interrupt is enabled and USB_NEED_CLK = 1 (see Table 11–190 for a description of USB_NEED_CLK), it is not possible to go into Power-down mode and any attempt to set the PD bit will fail, leaving the PLLs in the current state.

© NXP B.V. 2010. All rights reserved.

6.8 PLL1 frequency calculation

The PLL1 equations use the following parameters:

Table 35. Elements determining PLL frequency

Element	Description
F_{OSC}	the frequency from the crystal oscillator
F_{CCO}	the frequency of the PLL1 current controlled oscillator
USBCLK	the PLL1 output frequency (48 MHz for USB)
M	PLL1 Multiplier value from the MSEL1 bits in the PLL1CFG register
P	PLL1 Divider value from the PSEL1 bits in the PLL1CFG register

The PLL1 output frequency (when the PLL is both active and connected) is given by:

$USBCLK = M \times F_{OSC}$ or $USBCLK = F_{CCO} / (2 \times P)$

The CCO frequency can be computed as:

$F_{CCO} = USBCLK \times 2 \times P$ or $F_{CCO} = F_{OSC} \times M \times 2 \times P$

The PLL1 inputs and settings must meet the following criteria:

- F_{OSC} is in the range of 10 MHz to 25 MHz.
- USBCLK is 48 MHz.
- F_{CCO} is in the range of 156 MHz to 320 MHz.

6.9 Procedure for determining PLL1 settings

The PLL1 configuration for USB may be determined as follows:

1. The desired PLL1 output frequency is USBCLK = 48 MHz.
2. Choose an oscillator frequency (F_{OSC}). USBCLK must be the whole (non-fractional) multiple of F_{OSC} meaning that the possible values for F_{OSC} are 12 MHz, 16 MHz, and 24 MHz.
3. Calculate the value of M to configure the MSEL1 bits. $M = USBCLK / F_{OSC}$. In this case, the possible values for M = 2, 3, or 4 (F_{OSC} = 24 MHz, 16 MHz, or 12 MHz). The value written to the MSEL1 bits in PLL1CFG is M − 1 (see Table 4–37).
4. Find a value for P to configure the PSEL1 bits, such that F_{CCO} is within its defined frequency limits of 156 MHz to 320 MHz. F_{CCO} is calculated using $F_{CCO} = USBCLK \times 2 \times P$. It follows that P = 2 is the only P value to yield F_{CCO} in the allowed range. The value written to the PSEL1 bits in PLL1CFG is '01' for P = 2 (see Table 4–36).

© NXP B.V. 2010. All rights reserved.

Table 36. PLL1 Divider values

Values allowed for using PLL1 with USB are highlighted.

PSEL1 Bits (PLL1CFG bits [6:5])	Value of P
00	1
01	**2**
10	4
11	8

Table 37. PLL1 Multiplier values

Values allowed for using PLL1 with USB are highlighted.

MSEL1 Bits (PLL1CFG bits [4:0])	Value of M
00000	1
00001	**2**
00010	**3**
00011	**4**
...	...
11110	31
11111	32

© NXP B.V. 2010. All rights reserved.

7. Clock dividers

The output of the PLL0 must be divided down for use by the CPU and the USB subsystem (if used with PLL0, see Section 4–6). Separate dividers are provided such that the CPU frequency can be determined independently from the USB subsystem, which always requires 48 MHz with a 50% duty cycle for proper operation.

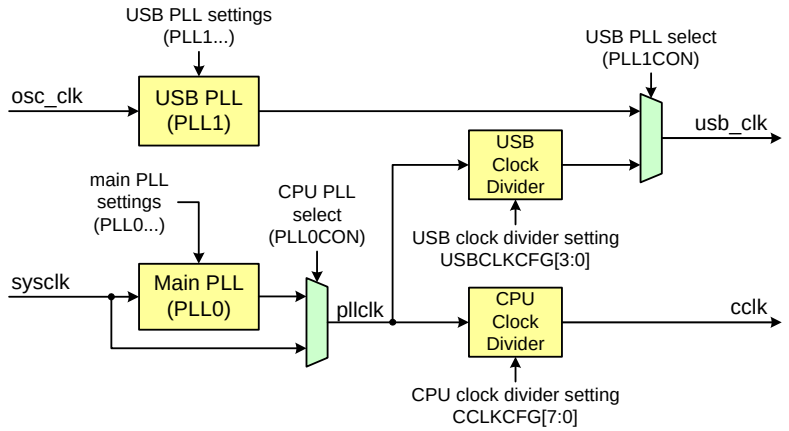


Fig 11. PLLs and clock dividers

7.1 CPU Clock Configuration register (CCLKCFG - 0x400F C104)

The CCLKCFG register controls the division of the PLL0 output before it is used by the CPU. When PLL0 is bypassed, the division may be by 1. When PLL0 is running, the output must be divided in order to bring the CPU clock frequency (CCLK) within operating limits. An 8-bit divider allows a range of options, including slowing CPU operation to a low rate for temporary power savings without turning off PLL0.

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

Table 38. CPU Clock Configuration register (CCLKCFG - address 0x400F C104) bit description

Bit	Symbol	Value	Description	Reset value
7:0	CCLKSEL		Selects the divide value for creating the CPU clock (CCLK) from the PLL0 output.	0x00
		0 to 1	Not allowed, the CPU clock will always be greater than 100 MHz.	
		2	PLL0 output is divided by 3 to produce the CPU clock.	
		3	PLL0 output is divided by 4 to produce the CPU clock.	
		4	PLL0 output is divided by 5 to produce the CPU clock.	
		:	:	
		255	PLL0 output is divided by 256 to produce the CPU clock.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

The CCLK is derived from the PLL0 output signal, divided by CCLKSEL + 1. Having CCLKSEL = 1 results in CCLK being one half the PLL0 output, CCLKSEL = 3 results in CCLK being one quarter of the PLL0 output, etc.

7.2 USB Clock Configuration register (USBCLKCFG - 0x400F C108)

This register is used only if the USB PLL (PLL1) is not connected (via the PLLC1 bit in PLL1CON). If PLL1 is connected, its output is automatically used as the USB clock source, and PLL1 must be configured to supply the correct 48 MHz clock to the USB subsystem. If PLL1 is not connected, the USB subsystem will be driven by PLL0 via the USB clock divider.

The USBCLKCFG register controls the division of the PLL0 output before it is used by the USB subsystem.The PLL0 output must be divided in order to bring the USB clock frequency to 48 MHz with a 50% duty cycle. A 4-bit divider allows obtaining the correct USB clock from any even multiple of 48 MHz (i.e. any multiple of 96 MHz) within the PLL operating range.

Remark: The Internal RC oscillator should not be used to drive PLL0 when the USB is using PLL0 as a clock source because a more precise clock is needed for USB specification compliance (see Table 4–17).

Table 39. USB Clock Configuration register (USBCLKCFG - address 0x400F C108) bit description

Bit	Symbol	Value	Description	Reset value
3:0	USBSEL		Selects the divide value for creating the USB clock from the PLL0 output. Only the values shown below can produce even number multiples of 48 MHz from the PLL0 output. **Warning:** Improper setting of this value will result in incorrect operation of the USB interface.	0
		5	PLL0 output is divided by 6. PLL0 output must be 288 MHz.	
		7	PLL0 output is divided by 8. PLL0 output must be 384 MHz.	
		9	PLL0 output is divided by 10. PLL0 output must be 480 MHz.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.3 Peripheral Clock Selection registers 0 and 1 (PCLKSEL0 - 0x400F C1A8 and PCLKSEL1 - 0x400F C1AC)

A pair of bits in a Peripheral Clock Selection register controls the rate of the clock signal that will be supplied to the corresponding peripheral as specified in Table 4–40, Table 4–41 and Table 4–42.

Remark: The peripheral clock for the RTC block is fixed at CCLK/8.

© NXP B.V. 2010. All rights reserved.

Table 40. Peripheral Clock Selection register 0 (PCLKSEL0 - address 0x400F C1A8) bit description

Bit	Symbol	Description	Reset value
1:0	PCLK_WDT	Peripheral clock selection for WDT.	00
3:2	PCLK_TIMER0	Peripheral clock selection for TIMER0.	00
5:4	PCLK_TIMER1	Peripheral clock selection for TIMER1.	00
7:6	PCLK_UART0	Peripheral clock selection for UART0.	00
9:8	PCLK_UART1	Peripheral clock selection for UART1.	00
11:10	-	Reserved.	NA
13:12	PCLK_PWM1	Peripheral clock selection for PWM1.	00
15:14	PCLK_I2C0	Peripheral clock selection for I^2C0.	00
17:16	PCLK_SPI	Peripheral clock selection for SPI.	00
19:18	-	Reserved.	NA
21:20	PCLK_SSP1	Peripheral clock selection for SSP1.	00
23:22	PCLK_DAC	Peripheral clock selection for DAC.	00
25:24	PCLK_ADC	Peripheral clock selection for ADC.	00
27:26	PCLK_CAN1	Peripheral clock selection for CAN1.[1]	00
29:28	PCLK_CAN2	Peripheral clock selection for CAN2.[1]	00
31:30	PCLK_ACF	Peripheral clock selection for CAN acceptance filtering.[1]	00

[1] PCLK_CAN1 and PCLK_CAN2 must have the same PCLK divide value when the CAN function is used.

Table 41. Peripheral Clock Selection register 1 (PCLKSEL1 - address 0x400F C1AC) bit description

Bit	Symbol	Description	Reset value
1:0	PCLK_QEI	Peripheral clock selection for the Quadrature Encoder Interface.	00
3:2	PCLK_GPIOINT	Peripheral clock selection for GPIO interrupts.	00
5:4	PCLK_PCB	Peripheral clock selection for the Pin Connect block.	00
7:6	PCLK_I2C1	Peripheral clock selection for I^2C1.	00
9:8	-	Reserved.	NA
11:10	PCLK_SSP0	Peripheral clock selection for SSP0.	00
13:12	PCLK_TIMER2	Peripheral clock selection for TIMER2.	00
15:14	PCLK_TIMER3	Peripheral clock selection for TIMER3.	00
17:16	PCLK_UART2	Peripheral clock selection for UART2.	00
19:18	PCLK_UART3	Peripheral clock selection for UART3.	00
21:20	PCLK_I2C2	Peripheral clock selection for I^2C2.	00
23:22	PCLK_I2S	Peripheral clock selection for I^2S.	00
25:24	-	Reserved.	NA
27:26	PCLK_RIT	Peripheral clock selection for Repetitive Interrupt Timer.	00
29:28	PCLK_SYSCON	Peripheral clock selection for the System Control block.	00
31:30	PCLK_MC	Peripheral clock selection for the Motor Control PWM.	00

© NXP B.V. 2010. All rights reserved.

Table 42. Peripheral Clock Selection register bit values

PCLKSEL0 and PCLKSEL1 individual peripheral's clock select options	Function	Reset value
00	PCLK_peripheral = CCLK/4	00
01	PCLK_peripheral = CCLK	
10	PCLK_peripheral = CCLK/2	
11	PCLK_peripheral = CCLK/8, except for CAN1, CAN2, and CAN filtering when "11" selects = CCLK/6.	

© NXP B.V. 2010. All rights reserved.

8. Power control

The LPC17xx supports a variety of power control features: Sleep mode, Deep Sleep mode, Power-down mode, and Deep Power-down mode. The CPU clock rate may also be controlled as needed by changing clock sources, re-configuring PLL values, and/or altering the CPU clock divider value. This allows a trade-off of power versus processing speed based on application requirements. In addition, Peripheral Power Control allows shutting down the clocks to individual on-chip peripherals, allowing fine tuning of power consumption by eliminating all dynamic power use in any peripherals that are not required for the application.

Entry to any reduced power mode begins with the execution of either a WFI (Wait For Interrupt) or WFE (Wait For Exception) instruction by the Cortex-M3. The Cortex-M3 internally supports two reduced power modes: Sleep and Deep Sleep. These are selected by the SLEEPDEEP bit in the cortex-M3 System Control Register. Power-down and Deep Power-down modes are selected by bits in the PCON register. See Table 4–44. The same register contains flags that indicate whether entry into each reduced power mode actually occurred.

The LPC17xx also implements a separate power domain in order to allow turning off power to the bulk of the device while maintaining operation of the Real Time Clock.

Reduced power modes have some limitation during debug, see Section 33–5 for more information.

8.1 Sleep mode

Note: Sleep mode on the LPC17xx corresponds to the Idle mode on LPC2xxx series devices. The name is changed because ARM has incorporated portions of reduced power mode control into the Cortex-M3. LPC17xx documentation uses the Cortex-M3 terminology where applicable.

When Sleep mode is entered, the clock to the core is stopped, and the SMFLAG bit in PCON is set, see Table 4–44.Resumption from the Sleep mode does not need any special sequence but re-enabling the clock to the ARM core.

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

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

8.2 Deep Sleep mode

Note: Deep Sleep mode on the LPC17xx corresponds to the Sleep mode on LPC23xx and LPC24xx series devices. The name is changed because ARM has incorporated portions of reduced power mode control into the Cortex-M3. LPC17xx documentation uses the Cortex-M3 terminology where applicable.

When the chip enters the Deep Sleep mode, the main oscillator is powered down, nearly all clocks are stopped, and the DSFLAG bit in PCON is set, see Table 4–44. The IRC remains running and can be configured to drive the Watchdog Timer, allowing the

© NXP B.V. 2010. All rights reserved.

Watchdog to wake up the CPU. The 32 kHz RTC oscillator is not stopped and RTC interrupts may be used as a wake-up source. The flash is left in the standby mode allowing a quick wake-up. The PLLs are automatically turned off and disconnected. The CCLK and USBCLK clock dividers automatically get reset to zero.

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

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

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

8.3 Power-down mode

Power-down mode does everything that Deep Sleep mode does, but also turns off the flash memory. Entry to Power-down mode causes the PDFLAG bit in PCON to be set, see Table 4–44. This saves more power, but requires waiting for resumption of flash operation before execution of code or data access in the flash memory can be accomplished.

When the chip enters Power-down mode, the IRC, the main oscillator, and all clocks are stopped. The RTC remains running if it has been enabled and RTC interrupts may be used to wake up the CPU. The flash is forced into Power-down mode. The PLLs are automatically turned off and disconnected. The CCLK and USBCLK clock dividers automatically get reset to zero.

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

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

© NXP B.V. 2010. All rights reserved.

8.4 Deep Power-down mode

In Deep Power-down mode, power is shut off to the entire chip with the exception of the Real-Time Clock, the RESET pin, the WIC, and the RTC backup registers. Entry to Deep Power-down mode causes the DPDFLAG bit in PCON to be set, see Table 4–44.

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

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

8.5 Peripheral power control

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

8.6 Register description

The Power Control function uses registers shown in Table 4–43. More detailed descriptions follow.

Table 43. Power Control registers

Name	Description	Access	Reset value[1]	Address
PCON	Power Control Register. This register contains control bits that enable some reduced power operating modes of the LPC17xx. See Table 4–44.	R/W	0x00	0x400F 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		0x400F C0C4

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

© NXP B.V. 2010. All rights reserved.

8.7 Power Mode Control register (PCON - 0x400F C0C0)

Controls for some reduced power modes and other power related controls are contained in the PCON register, as described in Table 4–44.

Table 44. Power Mode Control register (PCON - address 0x400F C0C0) bit description

Bit	Symbol	Description	Reset value
0	PM0	Power mode control bit 0. This bit controls entry to the Power-down mode. See Section 4–8.7.1 below for details.	0
1	PM1	Power mode control bit 1. This bit controls entry to the Deep Power-down mode. See Section 4–8.7.1 below for details.	0
2	BODRPM	Brown-Out Reduced Power Mode. When BODRPM is 1, the Brown-Out Detect circuitry will be turned off when chip Power-down mode or Deep Sleep mode is entered, resulting in a further reduction in power usage. However, the possibility of using Brown-Out Detect as a wake-up source from the reduced power mode will be lost. When 0, the Brown-Out Detect function remains active during Power-down and Deep Sleep modes. See the System Control Block chapter for details of Brown-Out detection.	0
3	BOGD	Brown-Out Global Disable. When BOGD is 1, the Brown-Out Detect circuitry is fully disabled at all times, and does not consume power. When 0, the Brown-Out Detect circuitry is enabled. See the System Control Block chapter for details of Brown-Out detection.	0
4	BORD	Brown-Out Reset Disable. When BORD is 1, the BOD will not reset the device when the $V_{DD(REG)(3V3)}$ voltage dips goes below the BOD reset trip level. The Brown-Out interrupt is not affected. When BORD is 0, the BOD reset is enabled. See the Section 3–5 for details of Brown-Out detection.	0
7:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	SMFLAG	Sleep Mode entry flag. Set when the Sleep mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][2]
9	DSFLAG	Deep Sleep entry flag. Set when the Deep Sleep mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][2]
10	PDFLAG	Power-down entry flag. Set when the Power-down mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][2]
11	DPDFLAG	Deep Power-down entry flag. Set when the Deep Power-down mode is successfully entered. Cleared by software writing a one to this bit.	0 [1][3]
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

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

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

© NXP B.V. 2010. All rights reserved.

8.7.1 Encoding of Reduced Power Modes

The PM1and PM0 bits in PCON allow entering reduced power modes as needed. The encoding of these bits allows backward compatibility with devices that previously only supported Sleep and Power-down modes. Table 4–45 below shows the encoding for the three reduced power modes supported by the LPC17xx.

Table 45. Encoding of reduced power modes

PM1, PM0	Description
00	Execution of WFI or WFE enters either Sleep or Deep Sleep mode as defined by the SLEEPDEEP bit in the Cortex-M3 System Control Register.
01	Execution of WFI or WFE enters Power-down mode if the SLEEPDEEP bit in the Cortex-M3 System Control Register is 1.
10	Reserved, this setting should not be used.
11	Execution of WFI or WFE enters Deep Power-down mode if the SLEEPDEEP bit in the Cortex-M3 System Control Register is 1.

8.8 Wake-up from Reduced Power Modes

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

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

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

8.9 Power Control for Peripherals register (PCONP - 0x400F C0C4)

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, the Pin Connect block, and the System Control block).

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

© NXP B.V. 2010. All rights reserved.

Each bit in PCONP controls one peripheral as shown in Table 4–46.

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

Important: valid 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 46. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved.	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	-	Reserved.	NA
6	PCPWM1	PWM1 power/clock control bit.	1
7	PCI2C0	The I^2C0 interface power/clock control bit.	1
8	PCSPI	The SPI interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSSP1	The SSP 1 interface power/clock control bit.	1
11	-	Reserved.	NA
12	PCADC	A/D converter (ADC) power/clock control bit. **Note:** Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	0
13	PCCAN1	CAN Controller 1 power/clock control bit.	0
14	PCCAN2	CAN Controller 2 power/clock control bit.	0
15	-	Reserved.	NA
16	PCRIT	Repetitive Interrupt Timer power/clock control bit.	0
17	PCMCPWM	Motor Control PWM	0
18	PCQEI	Quadrature Encoder Interface power/clock control bit.	0
19	PCI2C1	The I^2C1 interface power/clock control bit.	1
20	-	Reserved.	NA
21	PCSSP0	The SSP0 interface power/clock control bit.	1
22	PCTIM2	Timer 2 power/clock control bit.	0
23	PCTIM3	Timer 3 power/clock control bit.	0
24	PCUART2	UART 2 power/clock control bit.	0
25	PCUART3	UART 3 power/clock control bit.	0
26	PCI2C2	I^2C interface 2 power/clock control bit.	1
27	PCI2S	I^2S interface power/clock control bit.	0
28	-	Reserved.	NA

© NXP B.V. 2010. All rights reserved.

Table 46. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description

Bit	Symbol	Description	Reset value
29	PCGPDMA	GPDMA function power/clock control bit.	0
30	PCENET	Ethernet block power/clock control bit.	0
31	PCUSB	USB interface power/clock control bit.	0

Note that the DAC peripheral does not have a control bit in PCONP. To enable the DAC, its output must be selected to appear on the related pin, P0.26, by configuring the PINSEL1 register. See Section 8–5.2 “Pin Function Select Register 1 (PINSEL1 - 0x4002 C004)”.

8.10 Power control usage notes

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

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

8.11 Power domains

The LPC17xx provides two independent power domains that allow the bulk of the device to have power removed while maintaining operation of the Real Time Clock.

The VBAT pin supplies power only to the RTC domain. The RTC requires a minimum of power to operate, which can be supplied by an external battery. Whenever the device core power is present, that power is used to operate the RTC, causing no power drain from a battery when main power is available.

© NXP B.V. 2010. All rights reserved.

9. Wake-up timer

The LPC17xx begins operation at power-up and when awakened from Power-down mode by using the 4 MHz IRC oscillator as the clock source. This allows chip operation to begin quickly. If the main oscillator or one or both PLLs are needed by the application, software will need to enable these features and wait for them to stabilize before they are used as a clock source.

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

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

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

© NXP B.V. 2010. All rights reserved.

10. External clock output pin

For system test and development purposes, any one of several internal clocks may be brought out on the CLKOUT function available on the P1.27 pin, as shown in Figure 4–12.

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

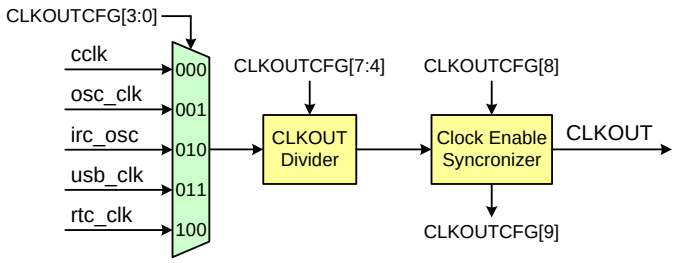


Fig 12. CLKOUT selection

10.1 Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8)

The CLKOUTCFG register controls the selection of the internal clock that appears on the CLKOUT pin and allows dividing the clock by an integer value up to 16. The divider can be used to produce a system clock that is related to one of the on-chip clocks. For most clock sources, the division may be by 1. When the CPU clock is selected and is higher than approximately 50 MHz, the output must be divided in order to bring the frequency within the ability of the pin to switch with reasonable logic levels.

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

Table 47. Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8) bit description

Bit	Symbol	Value	Description	Reset value
3:0	CLKOUTSEL		Selects the clock source for the CLKOUT function.	0
		0000	Selects the CPU clock as the CLKOUT source.	
		0001	Selects the main oscillator as the CLKOUT source.	
		0010	Selects the Internal RC oscillator as the CLKOUT source (default).	
		0011	Selects the USB clock as the CLKOUT source.	
		0100	Selects the RTC oscillator as the CLKOUT source.	
		others	Reserved, do not use these settings.	

© NXP B.V. 2010. All rights reserved.

Table 47. Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8) bit description

Bit	Symbol	Value	Description	Reset value
7:4	CLKOUTDIV		Integer value to divide the output clock by, minus one.	0
		0000	Clock is divided by 1.	
		0001	Clock is divided by 2.	
		0010	Clock is divided by 3.	
		...	...	
		1111	Clock is divided by 16.	
8	CLKOUT_EN		CLKOUT enable control, allows switching the CLKOUT source without glitches. Clear to stop CLKOUT on the next falling edge. Set to enable CLKOUT. Used in concert with the CLKOUT_EN bit below.	0
9	CLKOUT_ACT		CLKOUT activity indication. Reads as 1 when CLKOUT is enabled. Read as 0 when CLKOUT has been disabled via the CLKOUT_EN bit and the clock has completed being stopped.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

1. Introduction

The flash accelerator block in the LPC17xx allows maximization of the performance of the Cortex-M3 processor when it is running code from flash memory, while also saving power. The flash accelerator also provides speed and power improvements for data accesses to the flash memory.

2. Flash accelerator blocks

The flash accelerator is divided into several functional blocks:

- AHB-Lite bus interface, accessible by the Cortex-M3 I-code and D-code buses, as well as by the General Purpose DMA Controller
- An array of eight 128-bit buffers
- Flash accelerator control logic, including address compare and flash control
- A flash memory interface

Figure 5–13 shows a simplified diagram of the flash accelerator blocks and data paths.

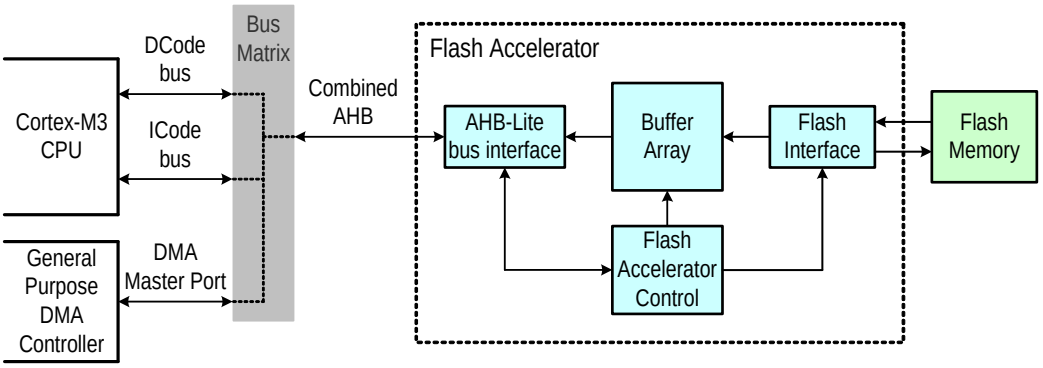


Fig 13. Simplified block diagram of the flash accelerator showing potential bus connections

In the following descriptions, the term “fetch” applies to an explicit flash read request from the CPU. “Prefetch” is used to denote a flash read of instructions beyond the current processor fetch address.

2.1 Flash memory bank

There is one bank of flash memory controlled by the LPC17xx flash accelerator.

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

© NXP B.V. 2010. All rights reserved.

2.2 Flash programming Issues

Since the flash memory does not allow accesses during programming and erase operations, it is necessary for the flash accelerator to force the CPU to wait if a memory access to a flash address is requested while the flash memory is busy with a programming operation. Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the flash memory.

In order to preclude the possibility of stale data being read from the flash memory, the LPC17xx flash accelerator buffers are automatically invalidated at the beginning of any flash programming or erase operation. Any subsequent read from a flash address will cause a new fetch to be initiated after the flash operation has completed.

3. Register description

The flash accelerator is controlled by the register shown in Table 5–48. More detailed descriptions follow.

Table 48. Summary of flash accelerator registers

Name	Description	Access	Reset value[1]	Address
FLASHCFG	Flash Accelerator Configuration Register. Controls flash access timing. See Table 5–49.	R/W	0x303A	0x400F C000

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

© NXP B.V. 2010. All rights reserved.

4. Flash Accelerator Configuration register (FLASHCFG - 0x400F C000)

Configuration bits select the flash access time, as shown in Table 5–49. **The lower bits of FLASHCFG control internal flash accelerator functions and should not be altered.**

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

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

Table 49. Flash Accelerator Configuration register (FLASHCFG - address 0x400F C000) bit description

Bit	Symbol	Value	Description	Reset value
11:0	-	-	Reserved, user software should not change these bits from the reset value.	0x03A
15:12	FLASHTIM		Flash access time. The value of this field plus 1 gives the number of CPU clocks used for a flash access. **Warning:** improper setting of this value may result in incorrect operation of the device. **Important Note: Frequency values shown below are estimates at this time.**	0x3
		0000	Flash accesses use 1 CPU clock. Use for up to 20 MHz CPU clock.	
		0001	Flash accesses use 2 CPU clocks. Use for up to 40 MHz CPU clock.	
		0010	Flash accesses use 3 CPU clocks. Use for up to 60 MHz CPU clock.	
		0011	Flash accesses use 4 CPU clocks. Use for up to 80 MHz CPU clock.	
		0100	Flash accesses use 5 CPU clocks. Use for up to 100 MHz CPU clock. Use for up to 120 Mhz for LPC1759 and LPC1769 only.	
		0101	Flash accesses use 6 CPU clocks. This “safe” setting will work under any conditions.	
		Other	Intended for potential future higher speed devices.	
31:16	-		Reserved. The value read from a reserved bit is not defined.	NA

5. Operation

Simply put, the flash accelerator attempts to have the next Cortex-M3 instruction that will be needed in its latches in time to prevent CPU fetch stalls. The LPC17xx uses one bank of flash memory. The flash accelerator includes an array of eight 128-bit buffers to store both instructions and data in a configurable manner. Each 128-bit buffer in the array can include four 32-bit instructions, eight 16-bit instructions or some combination of the two. During sequential code execution, a buffer typically contains the current instruction and the entire flash line that contains that instruction, or one flash line of data containing a previously requested address. Buffers are marked according to how they are used (as instruction or data buffers), and when they have been accessed. This information is used to carry out the buffer replacement strategy.

The Cortex-M3 provides a separate bus for instruction access (I-code) and data access (D-code) in the code memory space. These buses, plus the General Purpose DMA Controllers’s master port, are arbitrated by the AHB multilayer matrix. Any access to the flash memory’s address space is presented to the flash accelerator.

© NXP B.V. 2010. All rights reserved.

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

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

If an attempt is made to write directly to the flash memory without using the normal flash programming interface (via Boot ROM function calls), the flash accelerator generates an error condition. The CPU treats this error as a data abort. The GPDMA handles error conditions as described in Section 31–4.1.6.3.

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

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

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

Some special cases include the possibility that the CPU will request a data access to an address already contained in an instruction buffer. In this case, the data will be read from the buffer as if it was a data buffer. The reverse case, if the CPU requests an instruction address that can be satisfied from an existing data buffer, causes the instruction to be supplied from the data buffer, and the buffer to be changed into an instruction buffer. This causes the buffer to be handled differently when the flash accelerator is determining which buffer is to be overwritten next.

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 6: LPC17xx Nested Vectored Interrupt Controller (NVIC)

Rev. 01 — 4 January 2010 User manual

1. Features

- Nested Vectored Interrupt Controller that is an integral part of the ARM Cortex-M3
- Tightly coupled interrupt controller provides low interrupt latency
- Controls system exceptions and peripheral interrupts
- In the LPC17xx, the NVIC supports 35 vectored interrupts
- 32 programmable interrupt priority levels, with hardware priority level masking
- Relocatable vector table
- Non-Maskable Interrupt
- Software interrupt generation

2. Description

The Nested Vectored Interrupt Controller (NVIC) is an integral part of the Cortex-M3. The tight coupling to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

Refer to the Cortex-M3 User Guide Section 34–4.2 for details of NVIC operation.

3. Interrupt sources

Table 6–50 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source, as noted.

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

In addition, the NVIC handles the Non-Maskable Interrupt (NMI). In order for NMI to operate from an external signal, the NMI function must be connected to the related device pin (P2.10 / EINT0n / NMI). When connected, a logic 1 on the pin will cause the NMI to be processed. For details, refer to the Cortex-M3 User Guide that is an appendix to this User Manual.

© NXP B.V. 2010. All rights reserved.

Table 50. Connection of interrupt sources to the Vectored Interrupt Controller

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

© NXP B.V. 2010. All rights reserved.

Table 50. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
14	30	0x78	SSP0	Tx FIFO half empty of SSP0 Rx FIFO half full of SSP0 Rx Timeout of SSP0 Rx Overrun of SSP0
15	31	0x7C	SSP 1	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
16	32	0x80	PLL0 (Main PLL)	PLL0 Lock (PLOCK0)
17	33	0x84	RTC	Counter Increment (RTCCIF) Alarm (RTCALF)
18	34	0x88	External Interrupt	External Interrupt 0 (EINT0)
19	35	0x8C	External Interrupt	External Interrupt 1 (EINT1)
20	36	0x90	External Interrupt	External Interrupt 2 (EINT2)
21	37	0x94	External Interrupt	External Interrupt 3 (EINT3). **Note:** EINT3 channel is shared with GPIO interrupts
22	38	0x98	ADC	A/D Converter end of conversion
23	39	0x9C	BOD	Brown Out detect
24	40	0xA0	USB	USB_INT_REQ_LP, USB_INT_REQ_HP, USB_INT_REQ_DMA
25	41	0xA4	CAN	CAN Common, CAN 0 Tx, CAN 0 Rx, CAN 1 Tx, CAN 1 Rx
26	42	0xA8	GPDMA	IntStatus of DMA channel 0, IntStatus of DMA channel 1
27	43	0xAC	I^2S	irq, dmareq1, dmareq2
28	44	0xB0	Ethernet	WakeupInt, SoftInt, TxDoneInt, TxFinishedInt, TxErrorInt, TxUnderrunInt, RxDoneInt, RxFinishedInt, RxErrorInt, RxOverrunInt.
29	45	0xB4	Repetitive Interrupt Timer	RITINT
30	46	0xB8	Motor Control PWM	IPER[2:0], IPW[2:0], ICAP[2:0], FES
31	47	0xBC	Quadrature Encoder	INX_Int, TIM_Int, VELC_Int, DIR_Int, ERR_Int, ENCLK_Int, POS0_Int, POS1_Int, POS2_Int, REV_Int, POS0REV_Int, POS1REV_Int, POS2REV_Int
32	48	0xC0	PLL1 (USB PLL)	PLL1 Lock (PLOCK1)
33	49	0xC4	USB Activity Interrupt	USB_NEED_CLK
34	50	0xC8	CAN Activity Interrupt	CAN1WAKE, CAN2WAKE

© NXP B.V. 2010. All rights reserved.

4. Vector table remapping

The Cortex-M3 incorporates a mechanism that allows remapping the interrupt vector table to alternate locations in the memory map. This is controlled via the Vector Table Offset Register (VTOR) contained in the Cortex-M3.

The vector table may be located anywhere within the bottom 1 GB of Cortex-M3 address space. The vector table should be located on a 256 word (1024 byte) boundary to insure alignment on LPC17xx family devices. Refer to Section 34–4.3.5 of the Cortex-M3 User Guide appended to this manual for details of the Vector Table Offset feature.

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

Examples:

To place the vector table at the beginning of the “local” static RAM, starting at address 0x1000 0000, place the value 0x1000 0000 in the VTOR register. This indicates address 0x1000 0000 in the code space, since bit 29 of the VTOR equals 0.

To place the vector table at the beginning of the AHB static RAM, starting at address 0x2007 C000, place the value 0x2007 C000 in the VTOR register. This indicates address 0x2007 C000 in the SRAM space, since bit 29 of the VTOR equals 1.

© NXP B.V. 2010. All rights reserved.

5. Register description

The following table summarizes the registers in the NVIC as implemented in the LPC17xx. The Cortex-M3 User Guide Section 34–4.2 provides a functional description of the NVIC.

Table 51. NVIC register map

Name	Description	Access	Reset value	Address
ISER0 to ISER1	Interrupt Set-Enable Registers. These 2 registers allow enabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ISER0 - 0xE000 E100 ISER1 - 0xE000 E104
ICER0 to ICER1	Interrupt Clear-Enable Registers. These 2 registers allow disabling interrupts and reading back the interrupt enables for specific peripheral functions.	RW	0	ICER0 - 0xE000 E180 ICER1 - 0xE000 E184
ISPR0 to ISPR1	Interrupt Set-Pending Registers. These 2 registers allow changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ISPR0 - 0xE000 E200 ISPR1 - 0xE000 E204
ICPR0 to ICPR1	Interrupt Clear-Pending Registers. These 2 registers allow changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions.	RW	0	ICPR0 - 0xE000 E280 ICPR1 - 0xE000 E284
IABR0 to IABR1	Interrupt Active Bit Registers. These 2 registers allow reading the current interrupt active state for specific peripheral functions.	RO	0	IABR0 - 0xE000 E300 IABR1 - 0xE000 E304
IPR0 to IPR8	Interrupt Priority Registers. These 9 registers allow assigning a priority to each interrupt. Each register contains the 5-bit priority fields for 4 interrupts.	RW	0	IPR0 - 0xE000 E400 IPR1 - 0xE000 E404 IPR2 - 0xE000 E408 IPR3 - 0xE000 E40C IPR4 - 0xE000 E410 IPR5 - 0xE000 E414 IPR6 - 0xE000 E418 IPR7 - 0xE000 E41C IPR8 - 0xE000 E420
STIR	Software Trigger Interrupt Register. This register allows software to generate an interrupt.	WO	0	STIR - 0xE000 EF00

© NXP B.V. 2010. All rights reserved.

5.1 Interrupt Set-Enable Register 0 register (ISER0 - 0xE000 E100)

The ISER0 register allows enabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are enabled via the ISER1 register (Section 6–5.2). Disabling interrupts is done through the ICER0 and ICER1 registers (Section 6–5.3 and Section 6–5.4).

Table 52. Interrupt Set-Enable Register 0 register (ISER0 - 0xE000 E100)

Bit	Name	Function
0	ISE_WDT	Watchdog Timer Interrupt Enable. Write: writing 0 has no effect, writing 1 enables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.
1	ISE_TIMER0	Timer 0 Interrupt Enable. See functional description for bit 0.
2	ISE_TIMER1	Timer 1. Interrupt Enable. See functional description for bit 0.
3	ISE_TIMER2	Timer 2 Interrupt Enable. See functional description for bit 0.
4	ISE_TIMER3	Timer 3 Interrupt Enable. See functional description for bit 0.
5	ISE_UART0	UART0 Interrupt Enable. See functional description for bit 0.
6	ISE_UART1	UART1 Interrupt Enable. See functional description for bit 0.
7	ISE_UART2	UART2 Interrupt Enable. See functional description for bit 0.
8	ISE_UART3	UART3 Interrupt Enable. See functional description for bit 0.
9	ISE_PWM	PWM1 Interrupt Enable. See functional description for bit 0.
10	ISE_I2C0	I^2C0 Interrupt Enable. See functional description for bit 0.
11	ISE_I2C1	I^2C1 Interrupt Enable. See functional description for bit 0.
12	ISE_I2C2	I^2C2 Interrupt Enable. See functional description for bit 0.
13	ISE_SPI	SPI Interrupt Enable. See functional description for bit 0.
14	ISE_SSP0	SSP0 Interrupt Enable. See functional description for bit 0.
15	ISE_SSP1	SSP1 Interrupt Enable. See functional description for bit 0.
16	ISE_PLL0	PLL0 (Main PLL) Interrupt Enable. See functional description for bit 0.
17	ISE_RTC	Real Time Clock (RTC) Interrupt Enable. See functional description for bit 0.
18	ISE_EINT0	External Interrupt 0 Interrupt Enable. See functional description for bit 0.
19	ISE_EINT1	External Interrupt 1 Interrupt Enable. See functional description for bit 0.
20	ISE_EINT2	External Interrupt 2 Interrupt Enable. See functional description for bit 0.
21	ISE_EINT3	External Interrupt 3 Interrupt Enable. See functional description for bit 0.
22	ISE_ADC	ADC Interrupt Enable. See functional description for bit 0.
23	ISE_BOD	BOD Interrupt Enable. See functional description for bit 0.
24	ISE_USB	USB Interrupt Enable. See functional description for bit 0.
25	ISE_CAN	CAN Interrupt Enable. See functional description for bit 0.
26	ISE_DMA	GPDMA Interrupt Enable. See functional description for bit 0.
27	ISE_I2S	I^2S Interrupt Enable. See functional description for bit 0.
28	ISE_ENET	Ethernet Interrupt Enable. See functional description for bit 0.
29	ISE_RIT	Repetitive Interrupt Timer Interrupt Enable. See functional description for bit 0.
30	ISE_MCPWM	Motor Control PWM Interrupt Enable. See functional description for bit 0.
31	ISE_QEI	Quadrature Encoder Interface Interrupt Enable. See functional description for bit 0.

© NXP B.V. 2010. All rights reserved.

5.2 Interrupt Set-Enable Register 1 register (ISER1 - 0xE000 E104)

The ISER1 register allows enabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER0 and ICER1 registers (Section 6–5.3 and Section 6–5.4).

Table 53. Interrupt Set-Enable Register 1 register (ISER1 - 0xE000 E104)

Bit	Name	Function
0	ISE_PLL1	PLL1 (USB PLL) Interrupt Enable. Write: writing 0 has no effect, writing 1 enables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.
1	ISE_USBACT	USB Activity Interrupt Enable. See functional description for bit 0.
2	ISE_CANACT	CAN Activity Interrupt Enable. See functional description for bit 0.
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.3 Interrupt Clear-Enable Register 0 (ICER0 - 0xE000 E180)

The ICER0 register allows disabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are disabled via the ICER1 register (Section 6–5.4). Enabling interrupts is done through the ISER0 and ISER1 registers (Section 6–5.1 and Section 6–5.2).

Table 54. Interrupt Clear-Enable Register 0 (ICER0 - 0xE000 E180)

Bit	Name	Function
0	ICE_WDT	Watchdog Timer Interrupt Disable. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.
1	ICE_TIMER0	Timer 0 Interrupt Disable. See functional description for bit 0.
2	ICE_TIMER1	Timer 1. Interrupt Disable. See functional description for bit 0.
3	ICE_TIMER2	Timer 2 Interrupt Disable. See functional description for bit 0.
4	ICE_TIMER3	Timer 3 Interrupt Disable. See functional description for bit 0.
5	ICE_UART0	UART0 Interrupt Disable. See functional description for bit 0.
6	ICE_UART1	UART1 Interrupt Disable. See functional description for bit 0.
7	ICE_UART2	UART2 Interrupt Disable. See functional description for bit 0.
8	ICE_UART3	UART3 Interrupt Disable. See functional description for bit 0.
9	ICE_PWM	PWM1 Interrupt Disable. See functional description for bit 0.
10	ICE_I2C0	I^2C0 Interrupt Disable. See functional description for bit 0.
11	ICE_I2C1	I^2C1 Interrupt Disable. See functional description for bit 0.
12	ICE_I2C2	I^2C2 Interrupt Disable. See functional description for bit 0.
13	ICE_SPI	SPI Interrupt Disable. See functional description for bit 0.
14	ICE_SSP0	SSP0 Interrupt Disable. See functional description for bit 0.
15	ICE_SSP1	SSP1 Interrupt Disable. See functional description for bit 0.
16	ICE_PLL0	PLL0 (Main PLL) Interrupt Disable. See functional description for bit 0.
17	ICE_RTC	Real Time Clock (RTC) Interrupt Disable. See functional description for bit 0.
18	ICE_EINT0	External Interrupt 0 Interrupt Disable. See functional description for bit 0.
19	ICE_EINT1	External Interrupt 1 Interrupt Disable. See functional description for bit 0.
20	ICE_EINT2	External Interrupt 2 Interrupt Disable. See functional description for bit 0.
21	ICE_EINT3	External Interrupt 3 Interrupt Disable. See functional description for bit 0.
22	ICE_ADC	ADC Interrupt Disable. See functional description for bit 0.
23	ICE_BOD	BOD Interrupt Disable. See functional description for bit 0.
24	ICE_USB	USB Interrupt Disable. See functional description for bit 0.
25	ICE_CAN	CAN Interrupt Disable. See functional description for bit 0.
26	ICE_DMA	GPDMA Interrupt Disable. See functional description for bit 0.
27	ICE_I2S	I^2S Interrupt Disable. See functional description for bit 0.
28	ICE_ENET	Ethernet Interrupt Disable. See functional description for bit 0.
29	ICE_RIT	Repetitive Interrupt Timer Interrupt Disable. See functional description for bit 0.
30	ICE_MCPWM	Motor Control PWM Interrupt Disable. See functional description for bit 0.
31	ICE_QEI	Quadrature Encoder Interface Interrupt Disable. See functional description for bit 0.

© NXP B.V. 2010. All rights reserved.

5.4 Interrupt Clear-Enable Register 1 register (ICER1 - 0xE000 E184)

The ICER1 register allows disabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER0 and ISER1 registers (Section 6–5.1 and Section 6–5.2).

Table 55. Interrupt Clear-Enable Register 1 register (ICER1 - 0xE000 E184)

Bit	Name	Function
0	ICE_PLL1	PLL1 (USB PLL) Interrupt Disable. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.
1	ICE_USBACT	USB Activity Interrupt Disable. See functional description for bit 0.
2	ICE_CANACT	CAN Activity Interrupt Disable. See functional description for bit 0.
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.5 Interrupt Set-Pending Register 0 register (ISPR0 - 0xE000 E200)

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state set via the ISPR1 register (Section 6–5.6). Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 6–5.7 and Section 6–5.8).

Table 56. Interrupt Set-Pending Register 0 register (ISPR0 - 0xE000 E200)

Bit	Name	Function
0	ISP_WDT	Watchdog Timer Interrupt Pending set. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.
1	ISP_TIMER0	Timer 0 Interrupt Pending set. See functional description for bit 0.
2	ISP_TIMER1	Timer 1. Interrupt Pending set. See functional description for bit 0.
3	ISP_TIMER2	Timer 2 Interrupt Pending set. See functional description for bit 0.
4	ISP_TIMER3	Timer 3 Interrupt Pending set. See functional description for bit 0.
5	ISP_UART0	UART0 Interrupt Pending set. See functional description for bit 0.
6	ISP_UART1	UART1 Interrupt Pending set. See functional description for bit 0.
7	ISP_UART2	UART2 Interrupt Pending set. See functional description for bit 0.
8	ISP_UART3	UART3 Interrupt Pending set. See functional description for bit 0.
9	ISP_PWM	PWM1 Interrupt Pending set. See functional description for bit 0.
10	ISP_I2C0	I^2C0 Interrupt Pending set. See functional description for bit 0.
11	ISP_I2C1	I^2C1 Interrupt Pending set. See functional description for bit 0.
12	ISP_I2C2	I^2C2 Interrupt Pending set. See functional description for bit 0.
13	ISP_SPI	SPI Interrupt Pending set. See functional description for bit 0.
14	ISP_SSP0	SSP0 Interrupt Pending set. See functional description for bit 0.
15	ISP_SSP1	SSP1 Interrupt Pending set. See functional description for bit 0.
16	ISP_PLL0	PLL0 (Main PLL) Interrupt Pending set. See functional description for bit 0.
17	ISP_RTC	Real Time Clock (RTC) Interrupt Pending set. See functional description for bit 0.
18	ISP_EINT0	External Interrupt 0 Interrupt Pending set. See functional description for bit 0.
19	ISP_EINT1	External Interrupt 1 Interrupt Pending set. See functional description for bit 0.
20	ISP_EINT2	External Interrupt 2 Interrupt Pending set. See functional description for bit 0.
21	ISP_EINT3	External Interrupt 3 Interrupt Pending set. See functional description for bit 0.
22	ISP_ADC	ADC Interrupt Pending set. See functional description for bit 0.
23	ISP_BOD	BOD Interrupt Pending set. See functional description for bit 0.
24	ISP_USB	USB Interrupt Pending set. See functional description for bit 0.
25	ISP_CAN	CAN Interrupt Pending set. See functional description for bit 0.
26	ISP_DMA	GPDMA Interrupt Pending set. See functional description for bit 0.
27	ISP_I2S	I^2S Interrupt Pending set. See functional description for bit 0.
28	ISP_ENET	Ethernet Interrupt Pending set. See functional description for bit 0.
29	ISP_RIT	Repetitive Interrupt Timer Interrupt Pending set. See functional description for bit 0.
30	ISP_MCPWM	Motor Control PWM Interrupt Pending set. See functional description for bit 0.
31	ISP_QEI	Quadrature Encoder Interface Interrupt Pending set. See functional description for bit 0.

© NXP B.V. 2010. All rights reserved.

5.6 Interrupt Set-Pending Register 1 register (ISPR1 - 0xE000 E204)

The ISPR1 register allows setting the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 6–5.7 and Section 6–5.8).

Table 57. Interrupt Set-Pending Register 1 register (ISPR1 - 0xE000 E204)

Bit	Name	Function
0	ISP_PLL1	PLL1 (USB PLL) Interrupt Pending set. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.
1	ISP_USBACT	USB Activity Interrupt Pending set. See functional description for bit 0.
2	ISP_CANACT	CAN Activity Interrupt Pending set. See functional description for bit 0.
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.7 Interrupt Clear-Pending Register 0 register (ICPR0 - 0xE000 E280)

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state cleared via the ICPR1 register (Section 6–5.8). Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 6–5.5 and Section 6–5.6).

Table 58. Interrupt Clear-Pending Register 0 register (ICPR0 - 0xE000 E280)

Bit	Name	Function
0	ICP_WDT	Watchdog Timer Interrupt Pending clear. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.
1	ICP_TIMER0	Timer 0 Interrupt Pending clear. See functional description for bit 0.
2	ICP_TIMER1	Timer 1. Interrupt Pending clear. See functional description for bit 0.
3	ICP_TIMER2	Timer 2 Interrupt Pending clear. See functional description for bit 0.
4	ICP_TIMER3	Timer 3 Interrupt Pending clear. See functional description for bit 0.
5	ICP_UART0	UART0 Interrupt Pending clear. See functional description for bit 0.
6	ICP_UART1	UART1 Interrupt Pending clear. See functional description for bit 0.
7	ICP_UART2	UART2 Interrupt Pending clear. See functional description for bit 0.
8	ICP_UART3	UART3 Interrupt Pending clear. See functional description for bit 0.
9	ICP_PWM	PWM1 Interrupt Pending clear. See functional description for bit 0.
10	ICP_I2C0	I^2C0 Interrupt Pending clear. See functional description for bit 0.
11	ICP_I2C1	I^2C1 Interrupt Pending clear. See functional description for bit 0.
12	ICP_I2C2	I^2C2 Interrupt Pending clear. See functional description for bit 0.
13	ICP_SPI	SPI Interrupt Pending clear. See functional description for bit 0.
14	ICP_SSP0	SSP0 Interrupt Pending clear. See functional description for bit 0.
15	ICP_SSP1	SSP1 Interrupt Pending clear. See functional description for bit 0.
16	ICP_PLL0	PLL0 (Main PLL) Interrupt Pending clear. See functional description for bit 0.
17	ICP_RTC	Real Time Clock (RTC) Interrupt Pending clear. See functional description for bit 0.
18	ICP_EINT0	External Interrupt 0 Interrupt Pending clear. See functional description for bit 0.
19	ICP_EINT1	External Interrupt 1 Interrupt Pending clear. See functional description for bit 0.
20	ICP_EINT2	External Interrupt 2 Interrupt Pending clear. See functional description for bit 0.
21	ICP_EINT3	External Interrupt 3 Interrupt Pending clear. See functional description for bit 0.
22	ICP_ADC	ADC Interrupt Pending clear. See functional description for bit 0.
23	ICP_BOD	BOD Interrupt Pending clear. See functional description for bit 0.
24	ICP_USB	USB Interrupt Pending clear. See functional description for bit 0.
25	ICP_CAN	CAN Interrupt Pending clear. See functional description for bit 0.
26	ICP_DMA	GPDMA Interrupt Pending clear. See functional description for bit 0.
27	ICP_I2S	I^2S Interrupt Pending clear. See functional description for bit 0.
28	ICP_ENET	Ethernet Interrupt Pending clear. See functional description for bit 0.
29	ICP_RIT	Repetitive Interrupt Timer Interrupt Pending clear. See functional description for bit 0.
30	ICP_MCPWM	Motor Control PWM Interrupt Pending clear. See functional description for bit 0.
31	ICP_QEI	Quadrature Encoder Interface Interrupt Pending clear. See functional description for bit 0.

© NXP B.V. 2010. All rights reserved.

5.8 Interrupt Clear-Pending Register 1 register (ICPR1 - 0xE000 E284)

The ICPR1 register allows clearing the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 6–5.5 and Section 6–5.6).

Table 59. Interrupt Set-Pending Register 1 register (ISPR1 - 0xE000 E204)

Bit	Name	Function
0	ICP_PLL1	PLL1 (USB PLL) Interrupt Pending clear. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.
1	ICP_USBACT	USB Activity Interrupt Pending clear. See functional description for bit 0.
2	ICP_CANACT	CAN Activity Interrupt Pending clear. See functional description for bit 0.
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.9 Interrupt Active Bit Register 0 (IABR0 - 0xE000 E300)

The IABR0 register is a read-only register that allows reading the active state of the first 32 peripheral interrupts. This allows determining which peripherals are asserting an interrupt to the NVIC, and may also be pending if there are enabled. The remaining interrupts can have their active state read via the IABR1 register (Section 6–5.10).

Table 60. Interrupt Active Bit Register 0 (IABR0 - 0xE000 E300)

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

© NXP B.V. 2010. All rights reserved.

5.10 Interrupt Active Bit Register 1 (IABR1 - 0xE000 E304)

The IABR1 register is a read-only register that allows reading the active state of the second group of peripheral interrupts. This allows determining which peripherals are asserting an interrupt to the NVIC, and may also be pending if there are enabled.

Table 61. Interrupt Active Bit Register 1 (IABR1 - 0xE000 E304)

Bit	Name	Function
0	IAB_PLL1	PLL1 (USB PLL) Interrupt Active. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.
1	IAB_USBACT	USB Activity Interrupt Active. See functional description for bit 0.
2	IAB_CANACT	CAN Activity Interrupt Active. See functional description for bit 0.
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.11 Interrupt Priority Register 0 (IPR0 - 0xE000 E400)

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

Table 62. Interrupt Priority Register 0 (IPR0 - 0xE000 E400)

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

5.12 Interrupt Priority Register 1 (IPR1 - 0xE000 E404)

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

Table 63. Interrupt Priority Register 1 (IPR1 - 0xE000 E404)

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

5.13 Interrupt Priority Register 2 (IPR2 - 0xE000 E408)

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

Table 64. Interrupt Priority Register 2 (IPR2 - 0xE000 E408)

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

© NXP B.V. 2010. All rights reserved.

5.14 Interrupt Priority Register 3 (IPR3 - 0xE000 E40C)

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

Table 65. Interrupt Priority Register 3 (IPR3 - 0xE000 E40C)

Bit	Name	Function
2:0	Unimplemented	These bits ignore writes, and read as 0.
7:3	IP_I2C2	I^2C2 Interrupt Priority. 0 = highest priority. 31 (0x1F) = lowest priority.
10:8	Unimplemented	These bits ignore writes, and read as 0.
15:11	IP_SPI	SPI Interrupt Priority. See functional description for bits 7-3.
18:16	Unimplemented	These bits ignore writes, and read as 0.
23:19	IP_SSP0	SSP0 Interrupt Priority. See functional description for bits 7-3.
26:24	Unimplemented	These bits ignore writes, and read as 0.
31:27	IP_SSP1	SSP1 Interrupt Priority. See functional description for bits 7-3.

5.15 Interrupt Priority Register 4 (IPR4 - 0xE000 E410)

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

Table 66. Interrupt Priority Register 4 (IPR4 - 0xE000 E410)

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

5.16 Interrupt Priority Register 5 (IPR5 - 0xE000 E414)

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

Table 67. Interrupt Priority Register 5 (IPR5 - 0xE000 E414)

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

© NXP B.V. 2010. All rights reserved.

5.17 Interrupt Priority Register 6 (IPR6 - 0xE000 E418)

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

Table 68. Interrupt Priority Register 6 (IPR6 - 0xE000 E418)

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

5.18 Interrupt Priority Register 7 (IPR7 - 0xE000 E41C)

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

Table 69. Interrupt Priority Register 7 (IPR7 - 0xE000 E41C)

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

5.19 Interrupt Priority Register 8 (IPR8 - 0xE000 E420)

The IPR8 register controls the priority of the ninth and last group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 70. Interrupt Priority Register 8 (IPR8 - 0xE000 E420)

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

© NXP B.V. 2010. All rights reserved.

5.20 Software Trigger Interrupt Register (STIR - 0xE000 EF00)

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

By default, only privileged software can write to the STIR register. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the CCR register (see Section 34–4.3.8).

Table 71. Software Trigger Interrupt Register (STIR - 0xE000 EF00)

Bit	Name	Function
8:0	INTID	Writing a value to this field generates an interrupt for the specified the interrupt number (see Table 6–50). The range allowed for the LPC17xx is 0 to 111.
31:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

1. LPC17xx pin configuration

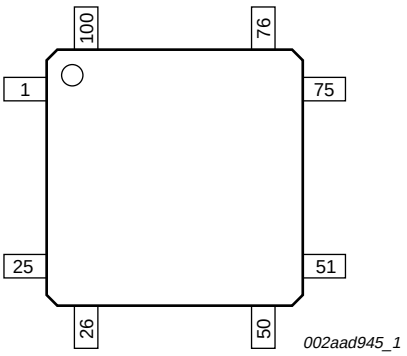


Fig 14. LPC176x LQFP100 pin configuration

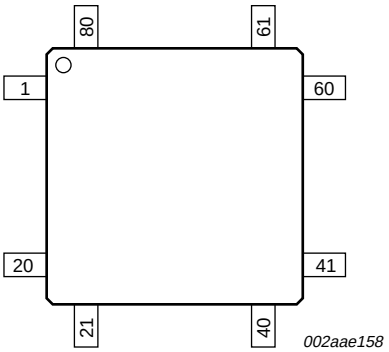


Fig 15. LPC175x LQFP80 pin configuration

1.1 LPC17xx pin description

I/O pins on the LPC17xx are 5V tolerant and have input hysteresis unless indicated in the table below. Crystal pins, power pins, and reference voltage pins are not 5V tolerant. In addition, when pins are selected to be A to D converter inputs, they are no longer 5V tolerant and must be limited to the voltage at the ADC positive reference pin (V_{REFP}).

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description

Symbol	LQFP 100	LQFP 80	Type	Description
P0[0] to P0[31]			I/O	**Port 0:** Port 0 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the pin connect block. Pins 12, 13, 14, and 31 of this port are not available.
P0[0] / RD1 / TXD3 / SDA1	46	37	I/O	**P0[0] —** General purpose digital input/output pin.
			I	**RD1 —** CAN1 receiver input.
			O	**TXD3 —** Transmitter output for UART3.
			I/O	**SDA1 —** I^2C1 data input/output (this pin is not fully compliant with the I^2C-bus specification, see Section 19–4 for details).
P0[1] / TD1 / RXD3 / SCL1	47	38	I/O	**P0[1] —** General purpose digital input/output pin.
			O	**TD1 —** CAN1 transmitter output.
			I	**RXD3 —** Receiver input for UART3.
			I/O	**SCL1 —** I^2C1 clock input/output (this pin is not fully compliant with the I^2C-bus specification, see Section 19–4 for details).
P0[2] / TXD0 / AD0[7]	98	79	I/O	**P0[2] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			O	**TXD0 —** Transmitter output for UART0.
			I	**AD0[7] —** A/D converter 0, input 7.
P0[3] / RXD0 / AD0[6]	99	80	I/O	**P0[3] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			I	**RXD0 —** Receiver input for UART0.
			I	**AD0[6] —** A/D converter 0, input 6.
P0[4] / I2SRX_CLK / RD2 / CAP2[0]	81	-	I/O	**P0[4] —** General purpose digital input/output pin.
			I/O	**I2SRX_CLK —** Receive Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S bus specification*.
			I	**RD2 —** CAN2 receiver input.
			I	**CAP2[0] —** Capture input for Timer 2, channel 0.
P0[5] / I2SRX_WS / TD2 / CAP2[1]	80	-	I/O	**P0[5] —** General purpose digital input/output pin.
			I/O	**I2SRX_WS —** Receive Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S bus specification*.
			O	**TD2 —** CAN2 transmitter output.
			I	**CAP2[1] —** Capture input for Timer 2, channel 1.
P0[6] / I2SRX_SDA / SSEL1 / MAT2[0]	79	64	I/O	**P0[6] —** General purpose digital input/output pin.
			I/O	**I2SRX_SDA —** Receive data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S bus specification*.
			I/O	**SSEL1 —** Slave Select for SSP1.
			O	**MAT2[0] —** Match output for Timer 2, channel 0.
P0[7] / I2STX_CLK / SCK1 / MAT2[1]	78	63	I/O	**P0[7] —** General purpose digital input/output pin.
			I/O	**I2STX_CLK —** Transmit Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S bus specification*.
			I/O	**SCK1 —** Serial Clock for SSP1.
			O	**MAT2[1] —** Match output for Timer 2, channel 1.

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P0[8] / I2STX_WS / MISO1 / MAT2[2]	77	62	I/O	**P0[8] —** General purpose digital input/output pin.
			I/O	**I2STX_WS —** Transmit Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S bus specification*.
			I/O	**MISO1 —** Master In Slave Out for SSP1.
			O	**MAT2[2] —** Match output for Timer 2, channel 2.
P0[9] / I2STX_SDA / MOSI1 / MAT2[3]	76	61	I/O	**P0[9] —** General purpose digital input/output pin.
			I/O	**I2STX_SDA —** Transmit data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S bus specification*.
			I/O	**MOSI1 —** Master Out Slave In for SSP1.
			O	**MAT2[3] —** Match output for Timer 2, channel 3.
P0[10] / TXD2 / SDA2 / MAT3[0]	48	39	I/O	**P0[10] —** General purpose digital input/output pin.
			O	**TXD2 —** Transmitter output for UART2.
			I/O	**SDA2 —** I^2C2 data input/output (this is not an open-drain pin).
			O	**MAT3[0] —** Match output for Timer 3, channel 0.
P0[11] / RXD2 / SCL2 / MAT3[1]	49	40	I/O	**P0[11] —** General purpose digital input/output pin.
			I	**RXD2 —** Receiver input for UART2.
			I/O	**SCL2 —** I^2C2 clock input/output (this is not an open-drain pin).
			O	**MAT3[1] —** Match output for Timer 3, channel 1.
P0[15] / TXD1 / SCK0 / SCK	62	47	I/O	**P0[15] —** General purpose digital input/output pin.
			O	**TXD1 —** Transmitter output for UART1.
			I/O	**SCK0 —** Serial clock for SSP0.
			I/O	**SCK —** Serial clock for SPI.
P0[16] / RXD1 / SSEL0 / SSEL	63	48	I/O	**P0[16] —** General purpose digital input/output pin.
			I	**RXD1 —** Receiver input for UART1.
			I/O	**SSEL0 —** Slave Select for SSP0.
			I/O	**SSEL —** Slave Select for SPI.
P0[17] / CTS1 / MISO0 / MISO	61	46	I/O	**P0[17] —** General purpose digital input/output pin.
			I	**CTS1 —** Clear to Send input for UART1.
			I/O	**MISO0 —** Master In Slave Out for SSP0.
			I/O	**MISO —** Master In Slave Out for SPI.
P0[18] / DCD1 / MOSI0 / MOSI	60	45	I/O	**P0[18] —** General purpose digital input/output pin.
			I	**DCD1 —** Data Carrier Detect input for UART1.
			I/O	**MOSI0 —** Master Out Slave In for SSP0.
			I/O	**MOSI —** Master Out Slave In for SPI.
P0[19] / DSR1 / SDA1	59	-	I/O	**P0[19] —** General purpose digital input/output pin.
			I	**DSR1 —** Data Set Ready input for UART1.
			I/O	**SDA1 —** I^2C1 data input/output (this pin is not fully compliant with the I^2C-bus specification, see Section 19–4 for details).

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P0[20] / DTR1 / SCL1	58	-	I/O	**P0[20] —** General purpose digital input/output pin.
			O	**DTR1 —** Data Terminal Ready output for UART1. Can also be configured to be an RS-485/EIA-485 output enable signal.
			I/O	**SCL1 —** I^2C1 clock input/output (this pin is not fully compliant with the I^2C-bus specification, see Section 19–4 for details).
P0[21] / RI1 / RD1	57	-	I/O	**P0[21] —** General purpose digital input/output pin.
			I	**RI1 —** Ring Indicator input for UART1.
			I	**RD1 —** CAN1 receiver input.
P0[22] / RTS1 / TD1	56	44	I/O	**P0[22] —** General purpose digital input/output pin.
			O	**RTS1 —** Request to Send output for UART1. Can also be configured to be an RS-485/EIA-485 output enable signal.
			O	**TD1 —** CAN1 transmitter output.
P0[23] / AD0[0] / I2SRX_CLK / CAP3[0]	9	-	I/O	**P0[23] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			I	**AD0[0] —** A/D converter 0, input 0.
			I/O	**I2SRX_CLK —** Receive Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S bus specification*.
			I	**CAP3[0] —** Capture input for Timer 3, channel 0.
P0[24] / AD0[1] / I2SRX_WS / CAP3[1]	8	-	I/O	**P0[24] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			I	**AD0[1] —** A/D converter 0, input 1.
			I/O	**I2SRX_WS —** Receive Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S bus specification*.
			I	**CAP3[1] —** Capture input for Timer 3, channel 1.
P0[25] / AD0[2] / I2SRX_SDA / TXD3	7	7	I/O	**P0[25] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			I	**AD0[2] —** A/D converter 0, input 2.
			I/O	**I2SRX_SDA —** Receive data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S bus specification*.
			O	**TXD3 —** Transmitter output for UART3.
P0[26] / AD0[3] / AOUT / RXD3	6	6	I/O	**P0[26] —** General purpose digital input/output pin. When configured as an ADC input or DAC output, the digital section of the pad is disabled.
			I	**AD0[3] —** A/D converter 0, input 3.
			O	**AOUT —** D/A converter output.
			I	**RXD3 —** Receiver input for UART3.
P0[27] / SDA0 / USB_SDA	25	-	I/O	**P0[27] —** General purpose digital input/output pin. Open-drain 5 V tolerant digital I/O pad, compatible with I^2C-bus specifications for 100 kHz standard mode, 400 kHz Fast Mode, and 1 MHz Fast Mode Plus. This pad requires an external pull-up to provide output functionality. When power is switched off, this pin connected to the I^2C-bus is floating and does not disturb the I^2C lines. Open-drain configuration applies to all functions on this pin.
			I/O	**SDA0 —** I^2C0 data input/output. Open-drain output (for I^2C-bus compliance).
			I/O	**USB_SDA —** USB port I^2C serial data (OTG transceiver).

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P0[28] / SCL0 / USB_SCL	24	-	I/O	**P0[28] —** General purpose digital input/output pin. Open-drain 5 V tolerant digital I/O pad, compatible with I^2C-bus specifications for 100 kHz standard mode, 400 kHz Fast Mode, and 1 MHz Fast Mode Plus. This pad requires an external pull-up to provide output functionality. When power is switched off, this pin connected to the I^2C-bus is floating and does not disturb the I^2C lines. Open-drain configuration applies to all functions on this pin.
			I/O	**SCL0 —** I^2C0 clock input/output. Open-drain output (for I^2C-bus compliance).
			I/O	**USB_SCL —** USB port I^2C serial clock (OTG transceiver).
P0[29] / USB_D+	29	22	I/O	**P0[29] —** General purpose digital input/output pin. Pad provides digital I/O and USB functions. It is designed in accordance with the USB specification, revision 2.0 (Full-speed and Low-speed mode only).
			I/O	**USB_D+ —** USB bidirectional D+ line.
P0[30] / USB_D−	30	23	I/O	**P0[30] —** General purpose digital input/output pin. Pad provides digital I/O and USB functions. It is designed in accordance with the USB specification, revision 2.0 (Full-speed and Low-speed mode only).
			I/O	**USB_D− —** USB bidirectional D− line.
P1[0] to P1[31]			I/O	**Port 1:** Port 1 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the pin connect block. Pins 2, 3, 5, 6, 7, 11, 12, and 13 of this port are not available.
P1[0] / ENET_TXD0	95	76	I/O	**P1[0] —** General purpose digital input/output pin.
			O	**ENET_TXD0 —** Ethernet transmit data 0.
P1[1] / ENET_TXD1	94	75	I/O	**P1[1] —** General purpose digital input/output pin.
			O	**ENET_TXD1 —** Ethernet transmit data 1.
P1[4] / ENET_TX_EN	93	74	I/O	**P1[4] —** General purpose digital input/output pin.
			O	**ENET_TX_EN —** Ethernet transmit data enable.
P1[8] / ENET_CRS	92	73	I/O	**P1[8] —** General purpose digital input/output pin.
			I	**ENET_CRS —** Ethernet carrier sense.
P1[9] / ENET_RXD0	91	72	I/O	**P1[9] —** General purpose digital input/output pin.
			I	**ENET_RXD0 —** Ethernet receive data.
P1[10] / ENET_RXD1	90	71	I/O	**P1[10] —** General purpose digital input/output pin.
			I	**ENET_RXD1 —** Ethernet receive data.
P1[14] / ENET_RX_ER	89	70	I/O	**P1[14] —** General purpose digital input/output pin.
			I	**ENET_RX_ER —** Ethernet receive error.
P1[15] / ENET_REF_CLK	88	69	I/O	**P1[15] —** General purpose digital input/output pin.
			I	**ENET_REF_CLK —** Ethernet reference clock.
P1[16] / ENET_MDC	87	-	I/O	**P1[16] —** General purpose digital input/output pin.
			O	**ENET_MDC —** Ethernet MIIM clock.
P1[17] / ENET_MDIO	86	-	I/O	**P1[17] —** General purpose digital input/output pin.
			I/O	**ENET_MDIO —** Ethernet MIIM data input and output.

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P1[18] / USB_UP_LED / PWM1[1] / CAP1[0]	32	25	I/O	**P1[18] —** General purpose digital input/output pin.
			O	**USB_UP_LED —** USB GoodLink LED indicator. It is LOW when device is configured (non-control endpoints enabled). It is HIGH when the device is not configured or during global suspend.
			O	**PWM1[1] —** Pulse Width Modulator 1, channel 1 output.
			I	**CAP1[0] —** Capture input for Timer 1, channel 0.
P1[19] / MCOA0 / $\overline{\text{USB\_PPWR}}$ / CAP1[1]	33	26	I/O	**P1[19] —** General purpose digital input/output pin.
			O	**MCOA0 —** Motor control PWM channel 0, output A.
			O	**$\overline{\text{USB\_PPWR}}$ —** Port Power enable signal for USB port.
			I	**CAP1[1] —** Capture input for Timer 1, channel 1.
P1[20] / MCI0 / PWM1[2] / SCK0	34	27	I/O	**P1[20] —** General purpose digital input/output pin.
			I	**MCI0 —** Motor control PWM channel 0 input. Also Quadrature Encoder Interface PHA input.
			O	**PWM1[2] —** Pulse Width Modulator 1, channel 2 output.
			I/O	**SCK0 —** Serial clock for SSP0.
P1[21] / $\overline{\text{MCABORT}}$ / PWM1[3] / SSEL0	35	-	I/O	**P1[21] —** General purpose digital input/output pin.
			O	**$\overline{\text{MCABORT}}$ —** Motor control PWM, active low fast abort.
			O	**PWM1[3] —** Pulse Width Modulator 1, channel 3 output.
			I/O	**SSEL0 —** Slave Select for SSP0.
P1[22] / MCOB0 / USB_PWRD / MAT1[0]	36	28	I/O	**P1[22] —** General purpose digital input/output pin.
			O	**MCOB0 —** Motor control PWM channel 0, output B.
			I	**USB_PWRD —** Power Status for USB port (host power switch).
			O	**MAT1[0] —** Match output for Timer 1, channel 0.
P1[23] / MCI1 / PWM1[4] / MISO0	37	29	I/O	**P1[23] —** General purpose digital input/output pin.
			I	**MCI1 —** Motor control PWM channel 1 input. Also Quadrature Encoder Interface PHB input.
			O	**PWM1[4] —** Pulse Width Modulator 1, channel 4 output.
			I/O	**MISO0 —** Master In Slave Out for SSP0.
P1[24] / MCI2 / PWM1[5] / MOSI0	38	30	I/O	**P1[24] —** General purpose digital input/output pin.
			I	**MCI2 —** Motor control PWM channel 2 input. Also Quadrature Encoder Interface INDEX input.
			O	**PWM1[5] —** Pulse Width Modulator 1, channel 5 output.
			I/O	**MOSI0 —** Master Out Slave in for SSP0.
P1[25] / MCOA1 / MAT1[1]	39	31	I/O	**P1[25] —** General purpose digital input/output pin.
			O	**MCOA1 —** Motor control PWM channel 1, output A.
			O	**MAT1[1] —** Match output for Timer 1, channel 1.
P1[26] / MCOB1 / PWM1[6] / CAP0[0]	40	32	I/O	**P1[26] —** General purpose digital input/output pin.
			O	**MCOB1 —** Motor control PWM channel 1, output B.
			O	**PWM1[6] —** Pulse Width Modulator 1, channel 6 output.
			I	**CAP0[0] —** Capture input for Timer 0, channel 0.

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P1[27] / CLKOUT / USB_OVRCR / CAP0[1]	43	-	I/O	**P1[27] —** General purpose digital input/output pin.
			O	**CLKOUT —** Clock output pin.
			I	**USB_OVRCR —** USB port Over-Current status.
			I	**CAP0[1] —** Capture input for Timer 0, channel 1.
P1[28] / MCOA2 / PCAP1[0] / MAT0[0]	44	35	I/O	**P1[28] —** General purpose digital input/output pin.
			O	**MCOA2 —** Motor control PWM channel 2, output A.
			I	**PCAP1[0] —** Capture input for PWM1, channel 0.
			O	**MAT0[0] —** Match output for Timer 0, channel 0.
P1[29] / MCOB2 / PCAP1[1] / MAT0[1]	45	36	I/O	**P1[29] —** General purpose digital input/output pin.
			O	**MCOB2 —** Motor control PWM channel 2, output B.
			I	**PCAP1[1] —** Capture input for PWM1, channel 1.
			O	**MAT0[1] —** Match output for Timer 0, channel 0.
P1[30] / V_{BUS} / AD0[4]	21	18	I/O	**P1[30] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			I	**V_{BUS} —** Monitors the presence of USB bus power. **Note:** This signal must be HIGH for USB reset to occur.
			I	**AD0[4] —** A/D converter 0, input 4.
P1[31] / SCK1 / AD0[5]	20	17	I/O	**P1[31] —** General purpose digital input/output pin. When configured as an ADC input, digital section of the pad is disabled.
			I/O	**SCK1 —** Serial Clock for SSP1.
			I	**AD0[5] —** A/D converter 0, input 5.
P2[0] to P2[31]			I/O	**Port 2:** Port 2 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 2 pins depends upon the pin function selected via the pin connect block. Pins 14 through 31 of this port are not available.
P2[0] / PWM1[1] / TXD1	75	60	I/O	**P2[0] —** General purpose digital input/output pin.
			O	**PWM1[1] —** Pulse Width Modulator 1, channel 1 output.
			O	**TXD1 —** Transmitter output for UART1.
P2[1] / PWM1[2] / RXD1	74	59	I/O	**P2[1] —** General purpose digital input/output pin.
			O	**PWM1[2] —** Pulse Width Modulator 1, channel 2 output.
			I	**RXD1 —** Receiver input for UART1.
P2[2] / PWM1[3] / CTS1 / TRACEDATA[3]	73	58	I/O	**P2[2] —** General purpose digital input/output pin.
			O	**PWM1[3] —** Pulse Width Modulator 1, channel 3 output.
			I	**CTS1 —** Clear to Send input for UART1.
			O	**TRACEDATA[3] —** Trace data, bit 3.
P2[3] / PWM1[4] / DCD1 / TRACEDATA[2]	70	55	I/O	**P2[3] —** General purpose digital input/output pin.
			O	**PWM1[4] —** Pulse Width Modulator 1, channel 4 output.
			I	**DCD1 —** Data Carrier Detect input for UART1.
			O	**TRACEDATA[2] —** Trace data, bit 2.

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P2[4] / PWM1[5] / DSR1 / TRACEDATA[1]	69	54	I/O	**P2[4] —** General purpose digital input/output pin.
			O	**PWM1[5] —** Pulse Width Modulator 1, channel 5 output.
			I	**DSR1 —** Data Set Ready input for UART1.
			O	**TRACEDATA[1] —** Trace data, bit 1.
P2[5] / PWM1[6] / DTR1 / TRACEDATA[0]	68	53	I/O	**P2[5] —** General purpose digital input/output pin.
			O	**PWM1[6] —** Pulse Width Modulator 1, channel 6 output.
			O	**DTR1 —** Data Terminal Ready output for UART1. Can also be configured to be an RS-485/EIA-485 output enable signal.
			O	**TRACEDATA[0] —** Trace data, bit 0.
P2[6] / PCAP1[0] / RI1 / TRACECLK	67	52	I/O	**P2[6] —** General purpose digital input/output pin.
			I	**PCAP1[0] —** Capture input for PWM1, channel 0.
			I	**RI1 —** Ring Indicator input for UART1.
			O	**TRACECLK —** Trace Clock.
P2[7] / RD2 / RTS1	66	51	I/O	**P2[7] —** General purpose digital input/output pin.
			I	**RD2 —** CAN2 receiver input.
			O	**RTS1 —** Request to Send output for UART1. Can also be configured to be an RS-485/EIA-485 output enable signal.
P2[8] / TD2 / TXD2 / ENET_MDC	65	50	I/O	**P2[8] —** General purpose digital input/output pin.
			O	**TD2 —** CAN2 transmitter output.
			O	**TXD2 —** Transmitter output for UART2.
			O	**ENET_MDC —** Ethernet MIIM clock.
P2[9] / USB_CONNECT / RXD2 / ENET_MDIO	64	49	I/O	**P2[9] —** General purpose digital input/output pin.
			O	**USB_CONNECT —** Signal used to switch an external 1.5 kΩ resistor under software control. Used with the SoftConnect USB feature.
			I	**RXD2 —** Receiver input for UART2.
			I/O	**ENET_MDIO —** Ethernet MIIM data input and output.
P2[10] / $\overline{\text{EINT0}}$ / NMI	53	41	I/O	**P2[10] —** General purpose digital input/output pin. 5 V tolerant pad with 5 ns glitch filter providing digital I/O functions with TTL levels and hysteresis. **Note:** A LOW on this pin while $\overline{\text{RESET}}$ is LOW forces the on-chip bootloader to take over control of the part after a reset and go into ISP mode. See Section 32–1.
			I	**$\overline{\text{EINT0}}$ —** External interrupt 0 input.
			I	**NMI —** Non-maskable interrupt input.
P2[11] / $\overline{\text{EINT1}}$ / I2STX_CLK	52	-	I/O	**P2[11] —** General purpose digital input/output pin. 5 V tolerant pad with 5 ns glitch filter providing digital I/O functions with TTL levels and hysteresis.
			I	**$\overline{\text{EINT1}}$ —** External interrupt 1 input.
			I/O	**I2STX_CLK —** Transmit Clock. It is driven by the master and received by the slave. Corresponds to the signal SCK in the *I^2S bus specification*.

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
P2[12] / EINT2 / I2STX_WS	51	-	I/O	**P2[12] —** General purpose digital input/output pin. 5 V tolerant pad with 5 ns glitch filter providing digital I/O functions with TTL levels and hysteresis.
			I	**EINT2 —** External interrupt 2 input.
			I/O	**I2STX_WS —** Transmit Word Select. It is driven by the master and received by the slave. Corresponds to the signal WS in the *I^2S bus specification*.
P2[13] / EINT3 / I2STX_SDA	50	-	I/O	**P2[13] —** General purpose digital input/output pin. 5 V tolerant pad with 5 ns glitch filter providing digital I/O functions with TTL levels and hysteresis.
			I	**EINT3 —** External interrupt 3 input.
			I/O	**I2STX_SDA —** Transmit data. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the *I^2S bus specification*.
P3[0] to P3[31]			I/O	**Port 3:** Port 3 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 3 pins depends upon the pin function selected via the pin connect block. Pins 0 through 24, and 27 through 31 of this port are not available.
P3[25] / MAT0[0] / PWM1[2]	27	-	I/O	**P3[25] —** General purpose digital input/output pin.
			O	**MAT0[0] —** Match output for Timer 0, channel 0.
			O	**PWM1[2] —** Pulse Width Modulator 1, output 2.
P3[26] / STCLK / MAT0[1] / PWM1[3]	26	-	I/O	**P3[26] —** General purpose digital input/output pin.
			I	**STCLK —** System tick timer clock input.
			O	**MAT0[1] —** Match output for Timer 0, channel 1.
			O	**PWM1[3] —** Pulse Width Modulator 1, output 3.
P4[0] to P4[31]			I/O	**Port 4:** Port 4 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 4 pins depends upon the pin function selected via the pin connect block. Pins 0 through 27, 30, and 31 of this port are not available.
P4[28] / RX_MCLK / MAT2[0] / TXD3	82	65	I/O	**P4[28] —** General purpose digital input/output pin.
			I	**RX_MCLK —** I^2S receive master clock.
			O	**MAT2[0] —** Match output for Timer 2, channel 0.
			O	**TXD3 —** Transmitter output for UART3.
P4[29] TX_MCLK / MAT2[1] / RXD3	85	68	I/O	**P4[29] —** General purpose digital input/output pin.
			I	**TX_MCLK —** I^2S transmit master clock.
			O	**MAT2[1] —** Match output for Timer 2, channel 1.
			I	**RXD3 —** Receiver input for UART3.
TDO / SWO	1	1	O	**TDO —** Test Data out for JTAG interface.
			O	**SWO —** Serial wire trace output.
TDI	2	2	I	**TDI —** Test Data in for JTAG interface.
TMS / SWDIO	3	3	I	**TMS —** Test Mode Select for JTAG interface.
			I/O	**SWDIO —** Serial wire debug data input/output.
TRST	4	4	I	**TRST —** Test Reset for JTAG interface.
TCK / SWDCLK	5	5	I	**TCK —** Test Clock for JTAG interface.
			I	**SWDCLK —** Serial wire clock.
RTCK	100	-	I/O	**RTCK —** JTAG interface control signal.

© NXP B.V. 2010. All rights reserved.

Table 72. Pin description *…continued*

Symbol	LQFP 100	LQFP 80	Type	Description
$\overline{\text{RSTOUT}}$	14	11	O	**$\overline{\text{RSTOUT}}$ —** This is a 3.3 V pin. A LOW on this pin indicates that the LPC17xx is in a Reset state.
$\overline{\text{RESET}}$	17	14	I	**External reset input:** A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. This is a 5 V tolerant pad with a 20 ns glitch filter, TTL levels and hysteresis.
XTAL1	22[1]	19[1]	I	Input to the oscillator circuit and internal clock generator circuits.
XTAL2	23[1]	20[1]	O	Output from the oscillator amplifier.
RTCX1	16[1]	13[1]	I	Input to the RTC oscillator circuit.
RTCX2	18[1]	15[1]	O	Output from the RTC oscillator circuit.
V_{SS}	31, 41, 55, 72, 83, 97[1]	24, 33, 43, 57, 66, 78[1]	I	**ground:** 0 V reference.
V_{SSA}	11[1]	9[1]	I	**analog ground:** 0 V reference. This should be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{DD(3V3)}$	28, 54, 71, 96[1]	21, 42, 56, 77[1]	I	**3.3 V supply voltage:** This is the power supply voltage for I/O other than pins in the Vbat domain.
$V_{DD(REG)(3V3)}$	42, 84[1]	34, 67[1]	I	**3.3 V voltage regulator supply voltage:** This is the supply voltage for the on-chip voltage regulator only.
V_{DDA}	10[1]	8[1]	I	**analog 3.3 V pad supply voltage:** This can be connected to the same supply as $V_{DD(3V3)}$ but should be isolated to minimize noise and error. This voltage is used to power the ADC and DAC. **Note: this pin should be tied to 3.3v if the ADC and DAC are not used.**
V_{REFP}	12[1]	10[1]	I	**ADC positive reference voltage:** This should be nominally the same voltage as V_{DDA} but should be isolated to minimize noise and error. The voltage level on this pin is used as a reference for ADC and DAC. **Note: this pin should be tied to 3.3v if the ADC and DAC are not used.**
V_{REFN}	15[1]	12[1]	I	**ADC negative reference voltage:** This should be the same voltage as V_{SS} but should be isolated to minimize noise and error. Level on this pin is used as a reference for ADC and DAC.
V_{BAT}	19[1]	16[1]	I	**RTC domain power supply:** 3.3 V on this pin supplies the power to the RTC peripheral.
n.c.	13	-	-	not connected

[1] Pad provides special analog functionality.

© NXP B.V. 2010. All rights reserved.

1. How to read this chapter

Table 8–73 shows the functions of the PINSEL registers in the LPC17xx.

Table 73. Summary of PINSEL registers

Register	Controls	Table
PINSEL0	P0[15:0]	Table 8–78
PINSEL1	P0 [31:16]	Table 8–79
PINSEL2	P1 [15:0] (Ethernet)	Table 8–80
PINSEL3	P1 [31:16]	Table 8–81
PINSEL4	P2 [15:0]	Table 8–82
PINSEL5	P2 [31:16]	not used
PINSEL6	P3 [15:0]	not used
PINSEL7	P3 [31:16]	Table 8–83
PINSEL8	P4 [15:0]	not used
PINSEL9	P4 [31:16]	Table 8–84
PINSEL10	Trace port enable	Table 8–85

2. Description

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

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

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

3. 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 74. Pin function select register bits

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

© NXP B.V. 2010. All rights reserved.

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

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is in principle possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those pins. If a peripheral input function is configured to appear on more than one pin for some reason, the peripheral will receive its input from the lowest port number. For instance, any pin of port 0 will take precedence over any pin of a higher numbered port, and pin 0 of any port will take precedence over a higher numbered pin of the same port.

4. Pin mode select register values

The PINMODE registers control the input mode of all ports. This includes the use of the on-chip pull-up/pull-down resistor feature and a special open drain operating mode. The on-chip pull-up/pull-down resistor can be selected for every port pin regardless of the function on this pin with the exception of the I^2C pins for the I2C0 interface and the USB pins (see Section 8–5.10). Three bits are used to control the mode of a port pin, two in a PINMODE register, and an additional one in a PINMODE_OD register. Bits are reserved for unused pins as in the PINSEL registers.

Table 75. Pin Mode Select register Bits

PINMODE0 to PINMODE9 Values	Function	Value after Reset
00	Pin has an on-chip pull-up resistor enabled.	00
01	Repeater mode (see text below).	
10	Pin has neither pull-up nor pull-down resistor enabled.	
11	Pin has an on-chip pull-down resistor enabled.	

Repeater mode enables the pull-up resistor if the pin is at a logic high and enables the pull-down resistor if the pin is at a logic low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. The state retention is not applicable to the Deep Power-down mode. Repeater mode may typically be used to prevent a pin from floating (and potentially using significant power if it floats to an indeterminate state) if it is temporarily not driven.

The PINMODE_OD registers control the open drain mode for ports. The open drain mode causes the pin to be pulled low normally if it is configured as an output and the data value is 0. If the data value is 1, the output drive of the pin is turned off, equivalent to changing the pin direction. This combination simulates an open drain output.

© NXP B.V. 2010. All rights reserved.

Table 76. Open Drain Pin Mode Select register Bits

PINMODE_OD0 to PINMODE_OD4 Values	Function	Value after Reset
0	Pin is in the normal (not open drain) mode.	00
1	Pin is in the open drain mode.	

Function of PINMODE in open drain mode

Normally the value of PINMODE applies to a pin only when it is in the input mode. When a pin is in the open drain mode, caused by a 1 in the corresponding bit of one of the PINMODE_OD registers, the input mode still does not apply when the pin is outputting a 0. However, when the pin value is 1, PINMODE applies since this state turns off the pin's output driver. For example, this allows for the possibility of configuring a pin to be open drain with an on-chip pullup. A pullup in this case which is only on when the pin is not being pulled low by the pin's own output.

© NXP B.V. 2010. All rights reserved.

5. Register description

The Pin Control Module contains 11 registers as shown in Table 8–77 below.

Table 77. Pin Connect Block Register Map

Name	Description	Access	Reset Value[1]	Address
PINSEL0	Pin function select register 0.	R/W	0	0x4002 C000
PINSEL1	Pin function select register 1.	R/W	0	0x4002 C004
PINSEL2	Pin function select register 2.	R/W	0	0x4002 C008
PINSEL3	Pin function select register 3.	R/W	0	0x4002 C00C
PINSEL4	Pin function select register 4	R/W	0	0x4002 C010
PINSEL7	Pin function select register 7	R/W	0	0x4002 C01C
PINSEL8	Pin function select register 8	R/W	0	0x4002 C020
PINSEL9	Pin function select register 9	R/W	0	0x4002 C024
PINSEL10	Pin function select register 10	R/W	0	0x4002 C028
PINMODE0	Pin mode select register 0	R/W	0	0x4002 C040
PINMODE1	Pin mode select register 1	R/W	0	0x4002 C044
PINMODE2	Pin mode select register 2	R/W	0	0x4002 C048
PINMODE3	Pin mode select register 3.	R/W	0	0x4002 C04C
PINMODE4	Pin mode select register 4	R/W	0	0x4002 C050
PINMODE5	Pin mode select register 5	R/W	0	0x4002 C054
PINMODE6	Pin mode select register 6	R/W	0	0x4002 C058
PINMODE7	Pin mode select register 7	R/W	0	0x4002 C05C
PINMODE9	Pin mode select register 9	R/W	0	0x4002 C064
PINMODE_OD0	Open drain mode control register 0	R/W	0	0x4002 C068
PINMODE_OD1	Open drain mode control register 1	R/W	0	0x4002 C06C
PINMODE_OD2	Open drain mode control register 2	R/W	0	0x4002 C070
PINMODE_OD3	Open drain mode control register 3	R/W	0	0x4002 C074
PINMODE_OD4	Open drain mode control register 4	R/W	0	0x4002 C078
I2CPADCFG	I^2C Pin Configuration register	R/W	0	0x4002 C07C

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

Pin control module register reset values

On external reset, watchdog reset, power-on-reset (POR), and BOD reset, all registers in this module are reset to '0'.

© NXP B.V. 2010. All rights reserved.

5.1 Pin Function Select register 0 (PINSEL0 - 0x4002 C000)

The PINSEL0 register controls the functions of the lower half of Port 0. The direction control bit in FIO0DIR register is effective only when the GPIO function is selected for a pin. For other functions, the direction is controlled automatically.

Table 78. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

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

[1] Not available on 80-pin package.

5.2 Pin Function Select Register 1 (PINSEL1 - 0x4002 C004)

The PINSEL1 register controls the functions of the upper half of Port 0. The direction control bit in the FIO0DIR register is effective only when the GPIO function is selected for a pin. For other functions the direction is controlled automatically.

Table 79. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL	00
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO	00
5:4	P0.18	GPIO Port 0.18	DCD1	MOSI0	MOSI	00
7:6	P0.19[1]	GPIO Port 0.19	DSR1	Reserved	SDA1	00
9:8	P0.20[1]	GPIO Port 0.20	DTR1	Reserved	SCL1	00
11:10	P0.21[1]	GPIO Port 0.21	RI1	Reserved	RD1	00
13:12	P0.22	GPIO Port 0.22	RTS1	Reserved	TD1	00
15:14	P0.23[1]	GPIO Port 0.23	AD0.0	I2SRX_CLK	CAP3.0	00
17:16	P0.24[1]	GPIO Port 0.24	AD0.1	I2SRX_WS	CAP3.1	00
19:18	P0.25	GPIO Port 0.25	AD0.2	I2SRX_SDA	TXD3	00
21:20	P0.26	GPIO Port 0.26	AD0.3	AOUT	RXD3	00
23:22	P0.27[1][2]	GPIO Port 0.27	SDA0	USB_SDA	Reserved	00
25:24	P0.28[1][2]	GPIO Port 0.28	SCL0	USB_SCL	Reserved	00

© NXP B.V. 2010. All rights reserved.

Table 79. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
27:26	P0.29	GPIO Port 0.29	USB_D+	Reserved	Reserved	00
29:28	P0.30	GPIO Port 0.30	USB_D–	Reserved	Reserved	00
31:30	-	Reserved	Reserved	Reserved	Reserved	00

[1] Not available on 80-pin package.

[2] Pins P027] and P0[28] are open-drain for I^2C-bus compliance.

5.3 Pin Function Select register 2 (PINSEL2 - 0x4002 C008)

The PINSEL2 register controls the functions of the lower half of Port 1, which contains the Ethernet related pins. The direction control bit in the FIO1DIR register is effective only when the GPIO function is selected for a pin. For other functions, the direction is controlled automatically.

Table 80. Pin function select register 2 (PINSEL2 - address 0x4002 C008) bit description

PINSEL2	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P1.0	GPIO Port 1.0	ENET_TXD0	Reserved	Reserved	00
3:2	P1.1	GPIO Port 1.1	ENET_TXD1	Reserved	Reserved	00
7:4	-	Reserved	Reserved	Reserved	Reserved	0
9:8	P1.4	GPIO Port 1.4	ENET_TX_EN	Reserved	Reserved	00
15:10	-	Reserved	Reserved	Reserved	Reserved	0
17:16	P1.8	GPIO Port 1.8	ENET_CRS	Reserved	Reserved	00
19:18	P1.9	GPIO Port 1.9	ENET_RXD0	Reserved	Reserved	00
21:20	P1.10	GPIO Port 1.10	ENET_RXD1	Reserved	Reserved	00
27:22	-	Reserved	Reserved	Reserved	Reserved	0
29:28	P1.14	GPIO Port 1.14	ENET_RX_ER	Reserved	Reserved	00
31:30	P1.15	GPIO Port 1.15	ENET_REF_CLK	Reserved	Reserved	00

5.4 Pin Function Select Register 3 (PINSEL3 - 0x4002 C00C)

The PINSEL3 register controls the functions of the upper half of Port 1. The direction control bit in the FIO1DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 81. Pin function select register 3 (PINSEL3 - address 0x4002 C00C) bit description

PINSEL3	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P1.16[1]	GPIO Port 1.16	ENET_MDC	Reserved	Reserved	00
3:2	P1.17[1]	GPIO Port 1.17	ENET_MDIO	Reserved	Reserved	00
5:4	P1.18	GPIO Port 1.18	USB_UP_LED	PWM1.1	CAP1.0	00
7:6	P1.19	GPIO Port 1.19	MCOA0	$\overline{\text{USB\_PPWR}}$	CAP1.1	00
9:8	P1.20	GPIO Port 1.20	MCI0	PWM1.2	SCK0	00
11:10	P1.21[1]	GPIO Port 1.21	$\overline{\text{MCABORT}}$	PWM1.3	SSEL0	00
13:12	P1.22	GPIO Port 1.22	MCOB0	USB_PWRD	MAT1.0	00

© NXP B.V. 2010. All rights reserved.

Table 81. Pin function select register 3 (PINSEL3 - address 0x4002 C00C) bit description

PINSEL3	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
15:14	P1.23	GPIO Port 1.23	MCI1	PWM1.4	MISO0	00
17:16	P1.24	GPIO Port 1.24	MCI2	PWM1.5	MOSI0	00
19:18	P1.25	GPIO Port 1.25	MCOA1	Reserved	MAT1.1	00
21:20	P1.26	GPIO Port 1.26	MCOB1	PWM1.6	CAP0.0	00
23:22	P1.27[1]	GPIO Port 1.27	CLKOUT	USB_OVRCR	CAP0.1	00
25:24	P1.28	GPIO Port 1.28	MCOA2	PCAP1.0	MAT0.0	00
27:26	P1.29	GPIO Port 1.29	MCOB2	PCAP1.1	MAT0.1	00
29:28	P1.30	GPIO Port 1.30	Reserved	V_{BUS}	AD0.4	00
31:30	P1.31	GPIO Port 1.31	Reserved	SCK1	AD0.5	00

[1] Not available on 80-pin package.

5.5 Pin Function Select Register 4 (PINSEL4 - 0x4002 C010)

The PINSEL4 register controls the functions of the lower half of Port 2. The direction control bit in the FIO2DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 82. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved [2]	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved [2]	00
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved	00
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC	00
19:18	P2.9	GPIO Port 2.9	USB_CONNECT	RXD2	ENET_MDIO	00
21:20	P2.10	GPIO Port 2.10	EINT0	NMI	Reserved	00
23:22	P2.11[1]	GPIO Port 2.11	EINT1	Reserved	I2STX_CLK	00
25:24	P2.12[1]	GPIO Port 2.12	EINT2	Reserved	I2STX_WS	00
27:26	P2.13[1]	GPIO Port 2.13	EINT3	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

[1] Not available on 80-pin package.

[2] These pins support a debug trace function when selected via a development tool or by writing to the PINSEL10 register. See Section 8–5.8 “Pin Function Select Register 10 (PINSEL10 - 0x4002 C028)” for details.

© NXP B.V. 2010. All rights reserved.

5.6 Pin Function Select Register 7 (PINSEL7 - 0x4002 C01C)

The PINSEL7 register controls the functions of the upper half of Port 3. The direction control bit in the FIO3DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 83. Pin function select register 7 (PINSEL7 - address 0x4002 C01C) bit description

PINSEL7	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
17:0	-	Reserved	Reserved	Reserved	Reserved	0
19:18	P3.25[1]	GPIO Port 3.25	Reserved	MAT0.0	PWM1.2	00
21:20	P3.26[1]	GPIO Port 3.26	STCLK	MAT0.1	PWM1.3	00
31:22	-	Reserved	Reserved	Reserved	Reserved	0

[1] Not available on 80-pin package.

5.7 Pin Function Select Register 9 (PINSEL9 - 0x4002 C024)

The PINSEL9 register controls the functions of the upper half of Port 4. The direction control bit in the FIO4DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 84. Pin function select register 9 (PINSEL9 - address 0x4002 C024) bit description

PINSEL9	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
23:0	-	Reserved	Reserved	Reserved	Reserved	00
25:24	P4.28	GPIO Port 4.28	RX_MCLK	MAT2.0	TXD3	00
27:26	P4.29	GPIO Port 4.29	TX_MCLK	MAT2.1	RXD3	00
31:28	-	Reserved	Reserved	Reserved	Reserved	00

5.8 Pin Function Select Register 10 (PINSEL10 - 0x4002 C028)

Only bit 3 of this register is used to control the Trace function on pins P2.2 through P2.6.

Table 85. Pin function select register 10 (PINSEL10 - address 0x4002 C028) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-	-	Reserved. Software should not write 1 to these bits.	NA
3	GPIO/TRACE		TPIU interface pins control.	0
		0	TPIU interface is disabled.	
		1	TPIU interface is enabled. TPIU signals are available on the pins hosting them regardless of the PINSEL4 content.	
31:4	-	-	Reserved. Software should not write 1 to these bits.	NA

© NXP B.V. 2010. All rights reserved.

5.9 Pin Mode select register 0 (PINMODE0 - 0x4002 C040)

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

Table 86. Pin Mode select register 0 (PINMODE0 - address 0x4002 C040) bit description

PINMODE0	Symbol	Value	Description	Reset value
1:0	P0.00MODE		Port 0 pin 0 on-chip pull-up/down resistor control.	00
		00	P0.0 pin has a pull-up resistor enabled.	
		01	P0.0 pin has repeater mode enabled.	
		10	P0.0 pin has neither pull-up nor pull-down.	
		11	P0.0 has a pull-down resistor enabled.	
3:2	P0.01MODE		Port 0 pin 1 control, see P0.00MODE.	00
5:4	P0.02MODE		Port 0 pin 2 control, see P0.00MODE.	00
7:6	P0.03MODE		Port 0 pin 3 control, see P0.00MODE.	00
9:8	P0.04MODE[1]		Port 0 pin 4 control, see P0.00MODE.	00
11:10	P0.05MODE[1]		Port 0 pin 5 control, see P0.00MODE.	00
13:12	P0.06MODE		Port 0 pin 6 control, see P0.00MODE.	00
15:14	P0.07MODE		Port 0 pin 7 control, see P0.00MODE.	00
17:16	P0.08MODE		Port 0 pin 8 control, see P0.00MODE.	00
19:18	P0.09MODE		Port 0 pin 9control, see P0.00MODE.	00
21:20	P0.10MODE		Port 0 pin 10 control, see P0.00MODE.	00
23:22	P0.11MODE		Port 0 pin 11 control, see P0.00MODE.	00
29:24	-		Reserved.	NA
31:30	P0.15MODE		Port 0 pin 15 control, see P0.00MODE.	00

[1] Not available on 80-pin package.

5.10 Pin Mode select register 1 (PINMODE1 - 0x4002 C044)

This register controls pull-up/pull-down resistor configuration for Port 1 pins 16 to 26. For details see Section 8–4 “Pin mode select register values”.

Table 87. Pin Mode select register 1 (PINMODE1 - address 0x4002 C044) bit description

PINMODE1	Symbol	Description	Reset value
1:0	P0.16MODE	Port 1 pin 16 control, see P0.00MODE.	00
3:2	P0.17MODE	Port 1 pin 17 control, see P0.00MODE.	00
5:4	P0.18MODE	Port 1 pin 18 control, see P0.00MODE.	00
7:6	P0.19MODE[1]	Port 1 pin 19 control, see P0.00MODE.	00
9:8	P0.20MODE[1]	Port 1 pin 20control, see P0.00MODE.	00
11:10	P0.21MODE[1]	Port 1 pin 21 control, see P0.00MODE.	00
13:12	P0.22MODE	Port 1 pin 22 control, see P0.00MODE.	00
15:14	P0.23MODE[1]	Port 1 pin 23 control, see P0.00MODE.	00
17:16	P0.24MODE[1]	Port 1 pin 24 control, see P0.00MODE.	00
19:18	P0.25MODE	Port 1 pin 25 control, see P0.00MODE.	00

© NXP B.V. 2010. All rights reserved.

Table 87. Pin Mode select register 1 (PINMODE1 - address 0x4002 C044) bit description

PINMODE1	Symbol	Description	Reset value
21:20	P0.26MODE	Port 1 pin 26 control, see P0.00MODE.	00
29:22	-	Reserved. [2]	NA
31:30	-	Reserved.	NA

[1] Not available on 80-pin package.

[2] The pin mode cannot be selected for pins P0[27] to P0[30]. Pins P0[27] and P0[28] are dedicated I2C open-drain pins without pull-up/down. Pins P0[29] and P0[30] are USB specific pins without configurable pull-up or pull-down resistors. Pins P0[29] and P0[30] also must have the same direction since they operate as a unit for the USB function, see Section 9–5.1 “GPIO port Direction register FIOxDIR (FIO0DIR to FIO4DIR- 0x2009 C000 to 0x2009 C080)”.

5.11 Pin Mode select register 2 (PINMODE2 - 0x4002 C048)

This register controls pull-up/pull-down resistor configuration for Port 1 pins 0 to 15. For details see Section 8–4 “Pin mode select register values”.

Table 88. Pin Mode select register 2 (PINMODE2 - address 0x4002 C048) bit description

PINMODE2	Symbol	Description	Reset value
1:0	P1.00MODE	Port 1 pin 0 control, see P0.00MODE.	00
3:2	P1.01MODE	Port 1 pin 1 control, see P0.00MODE.	00
7:4	-	Reserved.	NA
9:8	P1.04MODE	Port 1 pin 4 control, see P0.00MODE.	00
15:10	-	Reserved.	NA
17:16	P1.08MODE	Port 1 pin 8 control, see P0.00MODE.	00
19:18	P1.09MODE	Port 1 pin 9 control, see P0.00MODE.	00
21:20	P1.10MODE	Port 1 pin 10 control, see P0.00MODE.	00
27:22	-	Reserved.	NA
29:28	P1.14MODE	Port 1 pin 14 control, see P0.00MODE.	00
31:30	P1.15MODE	Port 1 pin 15 control, see P0.00MODE.	00

5.12 Pin Mode select register 3 (PINMODE3 - 0x4002 C04C)

This register controls pull-up/pull-down resistor configuration for Port 1 pins 16 to 31. For details see Section 8–4 “Pin mode select register values”.

Table 89. Pin Mode select register 3 (PINMODE3 - address 0x4002 C04C) bit description

PINMODE3	Symbol	Description	Reset value
1:0	P1.16MODE[1]	Port 1 pin 16 control, see P0.00MODE.	00
3:2	P1.17MODE[1]	Port 1 pin 17 control, see P0.00MODE.	00
5:4	P1.18MODE	Port 1 pin 18 control, see P0.00MODE.	00
7:6	P1.19MODE	Port 1 pin 19 control, see P0.00MODE.	00
9:8	P1.20MODE	Port 1 pin 20 control, see P0.00MODE.	00
11:10	P1.21MODE[1]	Port 1 pin 21 control, see P0.00MODE.	00
13:12	P1.22MODE	Port 1 pin 22 control, see P0.00MODE.	00

© NXP B.V. 2010. All rights reserved.

Table 89. Pin Mode select register 3 (PINMODE3 - address 0x4002 C04C) bit description

PINMODE3	Symbol	Description	Reset value
15:14	P1.23MODE	Port 1 pin 23 control, see P0.00MODE.	00
17:16	P1.24MODE	Port 1 pin 24 control, see P0.00MODE.	00
19:18	P1.25MODE	Port 1 pin 25 control, see P0.00MODE.	00
21:20	P1.26MODE	Port 1 pin 26 control, see P0.00MODE.	00
23:22	P1.27MODE[1]	Port 1 pin 27 control, see P0.00MODE.	00
25:24	P1.28MODE	Port 1 pin 28 control, see P0.00MODE.	00
27:26	P1.29MODE	Port 1 pin 29 control, see P0.00MODE.	00
29:28	P1.30MODE	Port 1 pin 30 control, see P0.00MODE.	00
31:30	P1.31MODE	Port 1 pin 31 control, see P0.00MODE.	00

[1] Not available on 80-pin package.

5.13 Pin Mode select register 4 (PINMODE4 - 0x4002 C050)

This register controls pull-up/pull-down resistor configuration for Port 2 pins 0 to 15. For details see Section 8–4 “Pin mode select register values”.

Table 90. Pin Mode select register 4 (PINMODE4 - address 0x4002 C050) bit description

PINMODE4	Symbol	Description	Reset value
1:0	P2.00MODE	Port 2 pin 0 control, see P0.00MODE.	00
3:2	P2.01MODE	Port 2 pin 1 control, see P0.00MODE.	00
5:4	P2.02MODE	Port 2 pin 2 control, see P0.00MODE.	00
7:6	P2.03MODE	Port 2 pin 3 control, see P0.00MODE.	00
9:8	P2.04MODE	Port 2 pin 4 control, see P0.00MODE.	00
11:10	P2.05MODE	Port 2 pin 5 control, see P0.00MODE.	00
13:12	P2.06MODE	Port 2 pin 6 control, see P0.00MODE.	00
15:14	P2.07MODE	Port 2 pin 7 control, see P0.00MODE.	00
17:16	P2.08MODE	Port 2 pin 8 control, see P0.00MODE.	00
19:18	P2.09MODE	Port 2 pin 9 control, see P0.00MODE.	00
21:20	P2.10MODE	Port 2 pin 10 control, see P0.00MODE.	00
23:22	P2.11MODE[1]	Port 2 pin 11 control, see P0.00MODE.	00
25:24	P2.12MODE[1]	Port 2 pin 12 control, see P0.00MODE.	00
27:26	P2.13MODE[1]	Port 2 pin 13 control, see P0.00MODE.	00
31:28	-	Reserved.	NA

[1] Not available on 80-pin package.

© NXP B.V. 2010. All rights reserved.

5.14 Pin Mode select register 7 (PINMODE7 - 0x4002 C05C)

This register controls pull-up/pull-down resistor configuration for Port 3 pins 16 to 31. For details see Section 8–4 “Pin mode select register values”.

Table 91. Pin Mode select register 7 (PINMODE7 - address 0x4002 C05C) bit description

PINMODE7	Symbol	Description	Reset value
17:0	-	Reserved	NA
19:18	P3.25MODE[1]	Port 3 pin 25 control, see P0.00MODE.	00
21:20	P3.26MODE[1]	Port 3 pin 26 control, see P0.00MODE.	00
31:22	-	Reserved.	NA

[1] Not available on 80-pin package.

5.15 Pin Mode select register 9 (PINMODE9 - 0x4002 C064)

This register controls pull-up/pull-down resistor configuration for Port 4 pins 16 to 31. For details see Section 8–4 “Pin mode select register values”.

Table 92. Pin Mode select register 9 (PINMODE9 - address 0x4002 C064) bit description

PINMODE9	Symbol	Description	Reset value
23:0	-	Reserved.	NA
25:24	P4.28MODE	Port 4 pin 28 control, see P0.00MODE.	00
27:26	P4.29MODE	Port 4 pin 29 control, see P0.00MODE.	00
31:28	-	Reserved.	NA

5.16 Open Drain Pin Mode select register 0 (PINMODE_OD0 - 0x4002 C068)

This register controls the open drain mode for Port 0 pins. For details see Section 8–4 “Pin mode select register values”.

Table 93. Open Drain Pin Mode select register 0 (PINMODE_OD0 - address 0x4002 C068) bit description

PINMODE_OD0	Symbol	Value	Description	Reset value
0	P0.00OD[3]		Port 0 pin 0 open drain mode control.	0
		0	P0.0 pin is in the normal (not open drain) mode.	
		1	P0.0 pin is in the open drain mode.	
1	P0.01OD[3]		Port 0 pin 1 open drain mode control, see P0.00OD	0
2	P0.02OD		Port 0 pin 2 open drain mode control, see P0.00OD	0
3	P0.03OD		Port 0 pin 3 open drain mode control, see P0.00OD	0
4	P0.04OD		Port 0 pin 4 open drain mode control, see P0.00OD	0
5	P0.05OD		Port 0 pin 5 open drain mode control, see P0.00OD	0
6	P0.06OD		Port 0 pin 6 open drain mode control, see P0.00OD	0
7	P0.07OD		Port 0 pin 7 open drain mode control, see P0.00OD	0
8	P0.08OD		Port 0 pin 8 open drain mode control, see P0.00OD	0
9	P0.09OD		Port 0 pin 9 open drain mode control, see P0.00OD	0

© NXP B.V. 2010. All rights reserved.

Table 93. Open Drain Pin Mode select register 0 (PINMODE_OD0 - address 0x4002 C068) bit description

PINMODE_OD0	Symbol	Value	Description	Reset value
10	P0.10OD[3]		Port 0 pin 10 open drain mode control, see P0.00OD	0
11	P0.11OD[3]		Port 0 pin 11 open drain mode control, see P0.00OD	0
14:12	-		Reserved.	NA
15	P0.15OD		Port 0 pin 15 open drain mode control, see P0.00OD	0
16	P0.16OD		Port 0 pin 16 open drain mode control, see P0.00OD	0
17	P0.17OD		Port 0 pin 17 open drain mode control, see P0.00OD	0
18	P0.18OD		Port 0 pin 18 open drain mode control, see P0.00OD	0
19	P0.19OD[3]		Port 0 pin 19 open drain mode control, see P0.00OD	0
20	P0.20OD[3]		Port 0 pin 20open drain mode control, see P0.00OD	0
21	P0.21OD		Port 0 pin 21 open drain mode control, see P0.00OD	0
22	P0.22OD		Port 0 pin 22 open drain mode control, see P0.00OD	0
23	P0.23OD		Port 0 pin 23 open drain mode control, see P0.00OD	0
24	P0.24OD		Port 0 pin 24open drain mode control, see P0.00OD	0
25	P0.25OD		Port 0 pin 25 open drain mode control, see P0.00OD	0
26	P0.26OD		Port 0 pin 26 open drain mode control, see P0.00OD	0
28:27	- [2]		Reserved.	NA
29	P0.29OD		Port 0 pin 29 open drain mode control, see P0.00OD	0
30	P0.30OD		Port 0 pin 30 open drain mode control, see P0.00OD	0
31	-		Reserved.	NA

[1] Not available on 80-pin package.

[2] Port 0 pins 27 and 28 should be set up using the I2CPADCFG register if they are used for an I^2C-bus. Bits 27 and 28 of PINMODE_OD0 do not have any affect on these pins, they are special open drain I^2C-bus compatible pins.

[3] Port 0 bits 1:0, 11:10, and 20:19 may potentially be used for I^2C-buses using standard port pins. If so, they should be configured for open drain mode via the related bits in PINMODE_OD0.

5.17 Open Drain Pin Mode select register 1 (PINMODE_OD1 - 0x4002 C06C)

This register controls the open drain mode for Port 1 pins. For details see Section 8–4 “Pin mode select register values”.

Table 94. Open Drain Pin Mode select register 1 (PINMODE_OD1 - address 0x4002 C06C) bit description

PINMODE_OD1	Symbol	Value	Description	Reset value
0	P1.00OD		Port 1 pin 0 open drain mode control.	0
		0	P1.0 pin is in the normal (not open drain) mode.	
		1	P1.0 pin is in the open drain mode.	
1	P1.01OD		Port 1 pin 1 open drain mode control, see P1.00OD	0
3:2	-		Reserved.	NA
4	P1.04OD		Port 1 pin 4 open drain mode control, see P1.00OD	0

© NXP B.V. 2010. All rights reserved.

Table 94. Open Drain Pin Mode select register 1 (PINMODE_OD1 - address 0x4002 C06C) bit description

PINMODE_OD1	Symbol	Value	Description	Reset value
7:5	-		Reserved.	NA
8	P1.08OD		Port 1 pin 8 open drain mode control, see P1.00OD	0
9	P1.09OD		Port 1 pin 9 open drain mode control, see P1.00OD	0
10	P1.10OD		Port 1 pin 10 open drain mode control, see P1.00OD	0
13:11	-		Reserved.	NA
14	P1.14OD		Port 1 pin 14 open drain mode control, see P1.00OD	0
15	P1.15OD		Port 1 pin 15 open drain mode control, see P1.00OD	0
16	P1.16OD[1]		Port 1 pin 16 open drain mode control, see P1.00OD	0
17	P1.17OD[1]		Port 1 pin 17 open drain mode control, see P1.00OD	0
18	P1.18OD		Port 1 pin 18 open drain mode control, see P1.00OD	0
19	P1.19OD		Port 1 pin 19 open drain mode control, see P1.00OD	0
20	P1.20OD		Port 1 pin 20open drain mode control, see P1.00OD	0
21	P1.21OD[1]		Port 1 pin 21 open drain mode control, see P1.00OD	0
22	P1.22OD		Port 1 pin 22 open drain mode control, see P1.00OD	0
23	P1.23OD		Port 1 pin 23 open drain mode control, see P1.00OD	0
24	P1.24OD		Port 1 pin 24open drain mode control, see P1.00OD	0
25	P1.25OD		Port 1 pin 25 open drain mode control, see P1.00OD	0
26	P1.26OD		Port 1 pin 26 open drain mode control, see P1.00OD	0
27	P1.27OD[1]		Port 1 pin 27 open drain mode control, see P1.00OD	0
28	P1.28OD		Port 1 pin 28 open drain mode control, see P1.00OD	0
29	P1.29OD		Port 1 pin 29 open drain mode control, see P1.00OD	0
30	P1.30OD		Port 1 pin 30 open drain mode control, see P1.00OD	0
31	P1.31OD		Port 1 pin 31 open drain mode control.	0

[1] Not available on 80-pin package.

5.18 Open Drain Pin Mode select register 2 (PINMODE_OD2 - 0x4002 C070)

This register controls the open drain mode for Port 2 pins. For details see Section 8–4 “Pin mode select register values”.

Table 95. Open Drain Pin Mode select register 2 (PINMODE_OD2 - address 0x4002 C070) bit description

PINMODE_OD2	Symbol	Value	Description	Reset value
0	P2.00OD		Port 2 pin 0 open drain mode control.	0
		0	P2.0 pin is in the normal (not open drain) mode.	
		1	P2.0 pin is in the open drain mode.	
1	P2.01OD		Port 2 pin 1 open drain mode control, see P2.00OD	0
2	P2.02OD		Port 2 pin 2 open drain mode control, see P2.00OD	0
3	P2.03OD		Port 2 pin 3 open drain mode control, see P2.00OD	0
4	P2.04OD		Port 2 pin 4 open drain mode control, see P2.00OD	0

© NXP B.V. 2010. All rights reserved.

Table 95. Open Drain Pin Mode select register 2 (PINMODE_OD2 - address 0x4002 C070) bit description

PINMODE_OD2	Symbol	Value	Description	Reset value
5	P2.05OD		Port 2 pin 5 open drain mode control, see P2.00OD	0
6	P2.06OD		Port 2 pin 6 open drain mode control, see P2.00OD	0
7	P2.07OD		Port 2 pin 7 open drain mode control, see P2.00OD	0
8	P2.08OD		Port 2 pin 8 open drain mode control, see P2.00OD	0
9	P2.09OD		Port 2 pin 9 open drain mode control, see P2.00OD	0
10	P2.10OD		Port 2 pin 10 open drain mode control, see P2.00OD	0
11	P2.11OD[1]		Port 2 pin 11 open drain mode control, see P2.00OD	0
12	P2.12OD[1]		Port 2 pin 12 open drain mode control, see P2.00OD	0
13	P2.13OD[1]		Port 2 pin 13 open drain mode control, see P2.00OD	0
31:14	-		Reserved.	NA

[1] Not available on 80-pin package.

5.19 Open Drain Pin Mode select register 3 (PINMODE_OD3 - 0x4002 C074)

This register controls the open drain mode for Port 3 pins. For details see Section 8–4 “Pin mode select register values”.

Table 96. Open Drain Pin Mode select register 3 (PINMODE_OD3 - address 0x4002 C074) bit description

PINMODE_OD3	Symbol	Value	Description	Reset value
24:0	-		Reserved.	NA
25	P3.25OD[1]		Port 3 pin 0 open drain mode control.	0
		0	P3.25 pin is in the normal (not open drain) mode.	
		1	P3.25 pin is in the open drain mode.	
26	P3.26OD[1]		Port 3 pin 26 open drain mode control, see P3.25OD	0
31:27	-		Reserved.	NA

[1] Not available on 80-pin package.

5.20 Open Drain Pin Mode select register 4 (PINMODE_OD4 - 0x4002 C078)

This register controls the open drain mode for Port 4 pins. For details see Section 8–4 “Pin mode select register values”.

Table 97. Open Drain Pin Mode select register 4 (PINMODE_OD4 - address 0x4002 C078) bit description

PINMODE_OD4	Symbol	Value	Description	Reset value
27:0	-		Reserved.	NA

© NXP B.V. 2010. All rights reserved.

Table 97. Open Drain Pin Mode select register 4 (PINMODE_OD4 - address 0x4002 C078) bit description

PINMODE_OD4	Symbol	Value	Description	Reset value
28	P4.28OD		Port 4 pin 28 open drain mode control.	0
		0	P4.28 pin is in the normal (not open drain) mode.	
		1	P4.28 pin is in the open drain mode.	
29	P4.28OD		Port 4 pin 29 open drain mode control, see P4.28OD	0
31:30	-		Reserved.	NA

5.21 I²C Pin Configuration register (I2CPADCFG - 0x4002 C07C)

The I2CPADCFG register allows configuration of the I²C pins for the I2C0 interface only, in order to support various I²C-bus operating modes. For use in standard or Fast Mode I²C, the 4 bits in I2CPADCFG should be 0, the default value for this register. For Fast Mode Plus, the SDADRV0 and SCLDRV0 bits should be 1. For non-I²C use of these pins, it may be desirable to turn off I²C filtering and slew rate control by setting SDAI2C0 and SCLI2C0 to 1. See Table 8–98 below.

Table 98. I²C Pin Configuration register (I2CPADCFG - address 0x4002 C07C) bit description

I2CPADCFG	Symbol	Value	Description	Reset value
0	SDADRV0		Drive mode control for the SDA0 pin, P0.27.	0
		0	The SDA0 pin is in the standard drive mode.	
		1	The SDA0 pin is in Fast Mode Plus drive mode.	
1	SDAI2C0		I²C mode control for the SDA0 pin, P0.27.	0
		0	The SDA0 pin has I²C glitch filtering and slew rate control enabled.	
		1	The SDA0 pin has I²C glitch filtering and slew rate control disabled.	
2	SCLDRV0		Drive mode control for the SCL0 pin, P0.28.	0
		0	The SCL0 pin is in the standard drive mode.	
		1	The SCL0 pin is in Fast Mode Plus drive mode.	
3	SCLI2C0		I²C mode control for the SCL0 pin, P0.28.	0
		0	The SCL0 pin has I²C glitch filtering and slew rate control enabled.	
		1	The SCL0 pin has I²C glitch filtering and slew rate control disabled.	
31:4	-		Reserved.	NA

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

GPIOs are configured using the following registers:

1. Power: always enabled.
2. Pins: See Section 8–3 for GPIO pins and their modes.
3. Wake-up: GPIO ports 0 and 2 can be used for wake-up if needed, see (Section 4–8.8).
4. Interrupts: Enable GPIO interrupts in IO0/2IntEnR (Table 9–113) or IO0/2IntEnF (Table 9–115). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

2. Features

2.1 Digital I/O ports

- Accelerated GPIO functions:
  - GPIO registers are located on a peripheral AHB bus for fast I/O timing.
  - Mask registers allow treating sets of port bits as a group, leaving other bits unchanged.
  - All GPIO registers are byte, half-word, and word addressable.
  - Entire port value can be written in one instruction.
  - GPIO registers are accessible by the GPDMA.
- Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.
- All GPIO registers support Cortex-M3 bit-banding.
- GPIO registers are accessible by the GPDMA controller to allow DMA of data to or from GPIOs, synchronized to any DMA request.
- Direction control of individual port bits.
- All I/Os default to input with pullup after reset.

2.2 Interrupt generating digital ports

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

© NXP B.V. 2010. All rights reserved.

- Registers provide a software view of pending rising edge interrupts, pending falling edge interrupts, and overall pending GPIO interrupts.
- GPIO0 and GPIO2 interrupts share the same position in the NVIC with External Interrupt 3.

3. Applications

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

4. Pin description

Table 99. GPIO pin description

Pin Name	Type	Description
P0[30:0][1]; P1[31:0][2]; P2[13:0]; P3[26:25]; P4[29:28]	Input/ Output	General purpose input/output. These are typically shared with other peripherals functions and will therefore not all be available in an application. Packaging options may affect the number of GPIOs available in a particular device. Some pins may be limited by requirements of the alternate functions of the pin. For example, the pins containing the I^2C0 functions are open-drain for any function selected on that pin. Details may be found in Section 7–1.1.

[1] P0[14:12] are not available.

[2] P1[2], P1[3], P1[7:5], P1[13:11] are not available.

© NXP B.V. 2010. All rights reserved.

5. Register description

Due to compatibility requirements with the LPC2300 series ARM7-based products, the LPC17xx implements portions of five 32-bit General Purpose I/O ports. Details on a specific GPIO port usage can be found in Section 8–3.

The registers in Table 9–100 represent the enhanced GPIO features available on all of the GPIO ports. These registers are located on an AHB bus for fast read and write timing. They can all be accessed in byte, half-word, and word sizes. A mask register allows access to a group of bits in a single GPIO port independently from other bits in the same port.

Table 100. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORTn Register Name & Address
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0x2009 C000 FIO1DIR - 0x2009 C020 FIO2DIR - 0x2009 C040 FIO3DIR - 0x2009 C060 FIO4DIR - 0x2009 C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK - 0x2009 C010 FIO1MASK - 0x2009 C030 FIO2MASK - 0x2009 C050 FIO3MASK - 0x2009 C070 FIO4MASK - 0x2009 C090
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC). The value read is masked by ANDing with inverted FIOMASK. Writing to this register places corresponding values in all bits enabled by zeros in FIOMASK. **Important:** if an FIOPIN register is read, its bit(s) masked with 1 in the FIOMASK register will be read as 0 regardless of the physical pin state.	R/W	0	FIO0PIN - 0x2009 C014 FIO1PIN - 0x2009 C034 FIO2PIN - 0x2009 C054 FIO3PIN - 0x2009 C074 FIO4PIN - 0x2009 C094
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by 0 in FIOMASK can be altered.	R/W	0	FIO0SET - 0x2009 C018 FIO1SET - 0x2009 C038 FIO2SET - 0x2009 C058 FIO3SET - 0x2009 C078 FIO4SET - 0x2009 C098
FIOCLR	Fast Port Output Clear register using FIOMASK. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by 0 in FIOMASK can be altered.	WO	0	FIO0CLR - 0x2009 C01C FIO1CLR - 0x2009 C03C FIO2CLR - 0x2009 C05C FIO3CLR - 0x2009 C07C FIO4CLR - 0x2009 C09C

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

© NXP B.V. 2010. All rights reserved.

Table 101. GPIO interrupt register map

Generic Name	Description	Access	Reset value[1]	PORTn Register Name & Address
IntEnR	GPIO Interrupt Enable for Rising edge.	R/W	0	IO0IntEnR - 0x4002 8090 IO2IntEnR - 0x4002 80B0
IntEnF	GPIO Interrupt Enable for Falling edge.	R/W	0	IO0IntEnR - 0x4002 8094 IO2IntEnR - 0x4002 80B4
IntStatR	GPIO Interrupt Status for Rising edge.	RO	0	IO0IntStatR - 0x4002 8084 IO2IntStatR - 0x4002 80A4
IntStatF	GPIO Interrupt Status for Falling edge.	RO	0	IO0IntStatF - 0x4002 8088 IO2IntStatF - 0x4002 80A8
IntClr	GPIO Interrupt Clear.	WO	0	IO0IntClr - 0x4002 808C IO2IntClr - 0x4002 80AC
IntStatus	GPIO overall Interrupt Status.	RO	0	IOIntStatus - 0x4002 8080

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

5.1 GPIO port Direction register FIOxDIR (FIO0DIR to FIO4DIR- 0x2009 C000 to 0x2009 C080)

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

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

Table 102. Fast GPIO port Direction register FIO0DIR to FIO4DIR - addresses 0x2009 C000 to 0x2009 C080) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0DIR FIO1DIR FIO2DIR FIO3DIR FIOI4DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.	0x0
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Aside from the 32-bit long and word only accessible FIODIR register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 9–103, too. Next to providing the same functions as the FIODIR register, these additional registers allow easier and faster access to the physical port pins.

© NXP B.V. 2010. All rights reserved.

Table 103. Fast GPIO port Direction control byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxDIR0	Fast GPIO Port x Direction control register 0. Bit 0 in FIOxDIR0 register corresponds to pin Px.0 … bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0DIR0 - 0x2009 C000 FIO1DIR0 - 0x2009 C020 FIO2DIR0 - 0x2009 C040 FIO3DIR0 - 0x2009 C060 FIO4DIR0 - 0x2009 C080
FIOxDIR1	Fast GPIO Port x Direction control register 1. Bit 0 in FIOxDIR1 register corresponds to pin Px.8 … bit 7 to pin Px.15.	8 (byte) R/W	0x00	FIO0DIR1 - 0x2009 C001 FIO1DIR1 - 0x2009 C021 FIO2DIR1 - 0x2009 C041 FIO3DIR1 - 0x2009 C061 FIO4DIR1 - 0x2009 C081
FIO0DIR2	Fast GPIO Port x Direction control register 2. Bit 0 in FIOxDIR2 register corresponds to pin Px.16 … bit 7 to pin Px.23.	8 (byte) R/W	0x00	FIO0DIR2 - 0x2009 C002 FIO1DIR2 - 0x2009 C022 FIO2DIR2 - 0x2009 C042 FIO3DIR2 - 0x2009 C062 FIO4DIR2 - 0x2009 C082
FIOxDIR3	Fast GPIO Port x Direction control register 3. Bit 0 in FIOxDIR3 register corresponds to pin Px.24 … bit 7 to pin Px.31.	8 (byte) R/W	0x00	FIO0DIR3 - 0x2009 C003 FIO1DIR3 - 0x2009 C023 FIO2DIR3 - 0x2009 C043 FIO3DIR3 - 0x2009 C063 FIO4DIR3 - 0x2009 C083
FIOxDIRL	Fast GPIO Port x Direction control Lower half-word register. Bit 0 in FIOxDIRL register corresponds to pin Px.0 … bit 15 to pin Px.15.	16 (half-word) R/W	0x0000	FIO0DIRL - 0x2009 C000 FIO1DIRL - 0x2009 C020 FIO2DIRL - 0x2009 C040 FIO3DIRL - 0x2009 C060 FIO4DIRL - 0x2009 C080
FIOxDIRU	Fast GPIO Port x Direction control Upper half-word register. Bit 0 in FIOxDIRU register corresponds to Px.16 … bit 15 to Px.31.	16 (half-word) R/W	0x0000	FIO0DIRU - 0x2009 C002 FIO1DIRU - 0x2009 C022 FIO2DIRU - 0x2009 C042 FIO3DIRU - 0x2009 C062 FIO4DIRU - 0x2009 C082

5.2 GPIO port output Set register FIOxSET (FIO0SET to FIO4SET - 0x2009 C018 to 0x2009 C098)

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 FIOxSET has no effect.

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

© NXP B.V. 2010. All rights reserved.

Access to a port pin via the FIOxSET register is conditioned by the corresponding bit of the FIOxMASK register (see Section 9–5.5).

Table 104. Fast GPIO port output Set register (FIO0SET to FIO4SET - addresses 0x2009 C018 to 0x2009 C098) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0SET FIO1SET FIO2SET FIO3SET FIO4SET		Fast GPIO output value Set bits. Bit 0 in FIOxSET controls pin Px.0, bit 31 in FIOxSET controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to HIGH.	

Aside from the 32-bit long and word only accessible FIOxSET register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 9–105, too. Next to providing the same functions as the FIOxSET register, these additional registers allow easier and faster access to the physical port pins.

Table 105. Fast GPIO port output Set byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxSET0	Fast GPIO Port x output Set register 0. Bit 0 in FIOxSET0 register corresponds to pin Px.0 … bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0SET0 - 0x2009 C018 FIO1SET0 - 0x2009 C038 FIO2SET0 - 0x2009 C058 FIO3SET0 - 0x2009 C078 FIO4SET0 - 0x2009 C098
FIOxSET1	Fast GPIO Port x output Set register 1. Bit 0 in FIOxSET1 register corresponds to pin Px.8 … bit 7 to pin Px.15.	8 (byte) R/W	0x00	FIO0SET1 - 0x2009 C019 FIO1SET1 - 0x2009 C039 FIO2SET1 - 0x2009 C059 FIO3SET1 - 0x2009 C079 FIO4SET1 - 0x2009 C099
FIOxSET2	Fast GPIO Port x output Set register 2. Bit 0 in FIOxSET2 register corresponds to pin Px.16 … bit 7 to pin Px.23.	8 (byte) R/W	0x00	FIO0SET2 - 0x2009 C01A FIO1SET2 - 0x2009 C03A FIO2SET2 - 0x2009 C05A FIO3SET2 - 0x2009 C07A FIO4SET2 - 0x2009 C09A
FIOxSET3	Fast GPIO Port x output Set register 3. Bit 0 in FIOxSET3 register corresponds to pin Px.24 … bit 7 to pin Px.31.	8 (byte) R/W	0x00	FIO0SET3 - 0x2009 C01B FIO1SET3 - 0x2009 C03B FIO2SET3 - 0x2009 C05B FIO3SET3 - 0x2009 C07B FIO4SET3 - 0x2009 C09B
FIOxSETL	Fast GPIO Port x output Set Lower half-word register. Bit 0 in FIOxSETL register corresponds to pin Px.0 … bit 15 to pin Px.15.	16 (half-word) R/W	0x0000	FIO0SETL - 0x2009 C018 FIO1SETL - 0x2009 C038 FIO2SETL - 0x2009 C058 FIO3SETL - 0x2009 C078 FIO4SETL - 0x2009 C098
FIOxSETU	Fast GPIO Port x output Set Upper half-word register. Bit 0 in FIOxSETU register corresponds to Px.16 … bit 15 to Px.31.	16 (half-word) R/W	0x0000	FIO0SETU - 0x2009 C01A FIO1SETU - 0x2009 C03A FIO2SETU - 0x2009 C05A FIO3SETU - 0x2009 C07A FIO4SETU - 0x2009 C09A

© NXP B.V. 2010. All rights reserved.

5.3 GPIO port output Clear register FIOxCLR (FIO0CLR to FIO4CLR- 0x2009 C01C to 0x2009 C09C)

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 FIOxSET register. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to FIOxCLR has no effect.

Access to a port pin via the FIOxCLR register is conditioned by the corresponding bit of the FIOxMASK register (see Section 9–5.5).

Table 106. Fast GPIO port output Clear register (FIO0CLR to FIO4CLR- addresses 0x2009 C01C to 0x2009 C09C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0CLR FIO1CLR FIO2CLR FIO3CLR FIO4CLR		Fast GPIO output value Clear bits. Bit 0 in FIOxCLR controls pin Px.0, bit 31 controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to LOW.	

Aside from the 32-bit long and word only accessible FIOxCLR register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 9–107, too. Next to providing the same functions as the FIOxCLR register, these additional registers allow easier and faster access to the physical port pins.

Table 107. Fast GPIO port output Clear byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxCLR0	Fast GPIO Port x output Clear register 0. Bit 0 in FIOxCLR0 register corresponds to pin Px.0 … bit 7 to pin Px.7.	8 (byte) WO	0x00	FIO0CLR0 - 0x2009 C01C FIO1CLR0 - 0x2009 C03C FIO2CLR0 - 0x2009 C05C FIO3CLR0 - 0x2009 C07C FIO4CLR0 - 0x2009 C09C
FIOxCLR1	Fast GPIO Port x output Clear register 1. Bit 0 in FIOxCLR1 register corresponds to pin Px.8 … bit 7 to pin Px.15.	8 (byte) WO	0x00	FIO0CLR1 - 0x2009 C01D FIO1CLR1 - 0x2009 C03D FIO2CLR1 - 0x2009 C05D FIO3CLR1 - 0x2009 C07D FIO4CLR1 - 0x2009 C09D
FIOxCLR2	Fast GPIO Port x output Clear register 2. Bit 0 in FIOxCLR2 register corresponds to pin Px.16 … bit 7 to pin Px.23.	8 (byte) WO	0x00	FIO0CLR2 - 0x2009 C01E FIO1CLR2 - 0x2009 C03E FIO2CLR2 - 0x2009 C05E FIO3CLR2 - 0x2009 C07E FIO4CLR2 - 0x2009 C09E

© NXP B.V. 2010. All rights reserved.

Table 107. Fast GPIO port output Clear byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxCLR3	Fast GPIO Port x output Clear register 3. Bit 0 in FIOxCLR3 register corresponds to pin Px.24 … bit 7 to pin Px.31.	8 (byte) WO	0x00	FIO0CLR3 - 0x2009 C01F FIO1CLR3 - 0x2009 C03F FIO2CLR3 - 0x2009 C05F FIO3CLR3 - 0x2009 C07F FIO4CLR3 - 0x2009 C09F
FIOxCLRL	Fast GPIO Port x output Clear Lower half-word register. Bit 0 in FIOxCLRL register corresponds to pin Px.0 … bit 15 to pin Px.15.	16 (half-word) WO	0x0000	FIO0CLRL - 0x2009 C01C FIO1CLRL - 0x2009 C03C FIO2CLRL - 0x2009 C05C FIO3CLRL - 0x2009 C07C FIO4CLRL - 0x2009 C09C
FIOxCLRU	Fast GPIO Port x output Clear Upper half-word register. Bit 0 in FIOxCLRU register corresponds to pin Px.16 … bit 15 to Px.31.	16 (half-word) WO	0x0000	FIO0CLRU - 0x2009 C01E FIO1CLRU - 0x2009 C03E FIO2CLRU - 0x2009 C05E FIO3CLRU - 0x2009 C07E FIO4CLRU - 0x2009 C09E

5.4 GPIO port Pin value register FIOxPIN (FIO0PIN to FIO4PIN- 0x2009 C014 to 0x2009 C094)

This register provides the value of port pins that are configured to perform only digital functions. The register will give the logic value of the pin regardless of whether the pin is configured for input or output, or as GPIO or an alternate digital function. As an example, a particular port pin may have GPIO input, GPIO output, UART receive, and PWM output as selectable functions. Any configuration of that pin will allow its current logic state to be read from the corresponding FIOxPIN 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 FIOxPIN register is not valid.

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

Access to a port pin via the FIOxPIN register is conditioned by the corresponding bit of the FIOxMASK register (see Section 9–5.5).

Only pins masked with zeros in the Mask register (see Section 9–5.5) will be correlated to the current content of the Fast GPIO port pin value register.

Table 108. Fast GPIO port Pin value register (FIO0PIN to FIO4PIN- addresses 0x2009 C014 to 0x2009 C094) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0VAL FIO1VAL FIO2VAL FIO3VAL FIO4VAL		Fast GPIO output value Set bits. Bit 0 in FIOxCLR corresponds to pin Px.0, bit 31 in FIOxCLR corresponds to pin Px.31.	0x0
		0	Controlled pin output is set to LOW.	
		1	Controlled pin output is set to HIGH.	

© NXP B.V. 2010. All rights reserved.

Aside from the 32-bit long and word only accessible FIOxPIN register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 9–109, too. Next to providing the same functions as the FIOxPIN register, these additional registers allow easier and faster access to the physical port pins.

Table 109. Fast GPIO port Pin value byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxPIN0	Fast GPIO Port x Pin value register 0. Bit 0 in FIOxPIN0 register corresponds to pin Px.0 … bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0PIN0 - 0x2009 C014 FIO1PIN0 - 0x2009 C034 FIO2PIN0 - 0x2009 C054 FIO3PIN0 - 0x2009 C074 FIO4PIN0 - 0x2009 C094
FIOxPIN1	Fast GPIO Port x Pin value register 1. Bit 0 in FIOxPIN1 register corresponds to pin Px.8 … bit 7 to pin Px.15.	8 (byte) R/W	0x00	FIO0PIN1 - 0x2009 C015 FIO1PIN1 - 0x2009 C035 FIO2PIN1 - 0x2009 C055 FIO3PIN1 - 0x2009 C075 FIO4PIN1 - 0x2009 C095
FIOxPIN2	Fast GPIO Port x Pin value register 2. Bit 0 in FIOxPIN2 register corresponds to pin Px.16 … bit 7 to pin Px.23.	8 (byte) R/W	0x00	FIO0PIN2 - 0x2009 C016 FIO1PIN2 - 0x2009 C036 FIO2PIN2 - 0x2009 C056 FIO3PIN2 - 0x2009 C076 FIO4PIN2 - 0x2009 C096
FIOxPIN3	Fast GPIO Port x Pin value register 3. Bit 0 in FIOxPIN3 register corresponds to pin Px.24 … bit 7 to pin Px.31.	8 (byte) R/W	0x00	FIO0PIN3 - 0x2009 C017 FIO1PIN3 - 0x2009 C037 FIO2PIN3 - 0x2009 C057 FIO3PIN3 - 0x2009 C077 FIO4PIN3 - 0x2009 C097
FIOxPINL	Fast GPIO Port x Pin value Lower half-word register. Bit 0 in FIOxPINL register corresponds to pin Px.0 … bit 15 to pin Px.15.	16 (half-word) R/W	0x0000	FIO0PINL - 0x2009 C014 FIO1PINL - 0x2009 C034 FIO2PINL - 0x2009 C054 FIO3PINL - 0x2009 C074 FIO4PINL - 0x2009 C094
FIOxPINU	Fast GPIO Port x Pin value Upper half-word register. Bit 0 in FIOxPINU register corresponds to pin Px.16 … bit 15 to Px.31.	16 (half-word) R/W	0x0000	FIO0PINU - 0x2009 C016 FIO1PINU - 0x2009 C036 FIO2PINU - 0x2009 C056 FIO3PINU - 0x2009 C076 FIO4PINU - 0x2009 C096

5.5 Fast GPIO port Mask register FIOxMASK (FIO0MASK to FIO4MASK - 0x2009 C010 to 0x2009 C090)

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

© NXP B.V. 2010. All rights reserved.

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 FIOxPIN register. For software examples, see Section 9–6.

Table 110. Fast GPIO port Mask register (FIO0MASK to FIO4MASK - addresses 0x2009 C010 to 0x2009 C090) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0MASK FIO1MASK FIO2MASK FIO3MASK FIO4MASK		Fast GPIO physical pin access control.	0x0
		0	Controlled pin is affected by writes to the port's FIOxSET, FIOxCLR, and FIOxPIN register(s). Current state of the pin can be read from the FIOxPIN register.	
		1	Controlled pin is not affected by writes into the port's FIOxSET, FIOxCLR and FIOxPIN register(s). When the FIOxPIN 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 FIOxMASK register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 9–111, too. Next to providing the same functions as the FIOxMASK register, these additional registers allow easier and faster access to the physical port pins.

Table 111. Fast GPIO port Mask byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxMASK0	Fast GPIO Port x Mask register 0. Bit 0 in FIOxMASK0 register corresponds to pin Px.0 … bit 7 to pin Px.7.	8 (byte) R/W	0x0	FIO0MASK0 - 0x2009 C010 FIO1MASK0 - 0x2009 C030 FIO2MASK0 - 0x2009 C050 FIO3MASK0 - 0x2009 C070 FIO4MASK0 - 0x2009 C090
FIOxMASK1	Fast GPIO Port x Mask register 1. Bit 0 in FIOxMASK1 register corresponds to pin Px.8 … bit 7 to pin Px.15.	8 (byte) R/W	0x0	FIO0MASK1 - 0x2009 C011 FIO1MASK1 - 0x2009 C031 FIO2MASK1 - 0x2009 C051 FIO3MASK1 - 0x2009 C071 FIO4MASK1 - 0x2009 C091
FIOxMASK2	Fast GPIO Port x Mask register 2. Bit 0 in FIOxMASK2 register corresponds to pin Px.16 … bit 7 to pin Px.23.	8 (byte) R/W	0x0	FIO0MASK2 - 0x2009 C012 FIO1MASK2 - 0x2009 C032 FIO2MASK2 - 0x2009 C052 FIO3MASK2 - 0x2009 C072 FIO4MASK2 - 0x2009 C092

© NXP B.V. 2010. All rights reserved.

Table 111. Fast GPIO port Mask byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxMASK3	Fast GPIO Port x Mask register 3. Bit 0 in FIOxMASK3 register corresponds to pin Px.24 … bit 7 to pin Px.31.	8 (byte) R/W	0x0	FIO0MASK3 - 0x2009 C013 FIO1MASK3 - 0x2009 C033 FIO2MASK3 - 0x2009 C053 FIO3MASK3 - 0x2009 C073 FIO4MASK3 - 0x2009 C093
FIOxMASKL	Fast GPIO Port x Mask Lower half-word register. Bit 0 in FIOxMASKL register corresponds to pin Px.0 … bit 15 to pin Px.15.	16 (half-word) R/W	0x0	FIO0MASKL - 0x2009 C010 FIO1MASKL - 0x2009 C030 FIO2MASKL - 0x2009 C050 FIO3MASKL - 0x2009 C070 FIO4MASKL - 0x2009 C090
FIOxMASKU	Fast GPIO Port x Mask Upper half-word register. Bit 0 in FIOxMASKU register corresponds to pin Px.16 … bit 15 to Px.31.	16 (half-word) R/W	0x0	FIO0MASKU - 0x2009 C012 FIO1MASKU - 0x2009 C032 FIO2MASKU - 0x2009 C052 FIO3MASKU - 0x2009 C072 FIO4MASKU - 0x2009 C092

© NXP B.V. 2010. All rights reserved.

5.6 GPIO interrupt registers

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

5.6.1 GPIO overall Interrupt Status register (IOIntStatus - 0x4002 8080)

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

Table 112. GPIO overall Interrupt Status register (IOIntStatus - address 0x4002 8080) bit description

Bit	Symbol	Value	Description	Reset value
0	P0Int		Port 0 GPIO interrupt pending.	0
		0	There are no pending interrupts on Port 0.	
		1	There is at least one pending interrupt on Port 0.	
1	-	-	Reserved. The value read from a reserved bit is not defined.	NA
2	P2Int		Port 2 GPIO interrupt pending.	0
		0	There are no pending interrupts on Port 2.	
		1	There is at least one pending interrupt on Port 2.	
31:2	-	-	Reserved. The value read from a reserved bit is not defined.	NA

5.6.2 GPIO Interrupt Enable for port 0 Rising Edge (IO0IntEnR - 0x4002 8090)

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

Table 113. GPIO Interrupt Enable for port 0 Rising Edge (IO0IntEnR - 0x4002 8090) bit description

Bit	Symbol	Value	Description	Reset value
0	P0.0ER		Enable rising edge interrupt for P0.0.	0
		0	Rising edge interrupt is disabled on P0.0.	
		1	Rising edge interrupt is enabled on P0.0.	
1	P0.1ER		Enable rising edge interrupt for P0.1.	0
2	P0.2ER		Enable rising edge interrupt for P0.2.	0
3	P0.3ER		Enable rising edge interrupt for P0.3.	0
4	P0.4ER[1]		Enable rising edge interrupt for P0.4.	0
5	P0.5ER[1]		Enable rising edge interrupt for P0.5.	0
6	P0.6ER		Enable rising edge interrupt for P0.6.	0
7	P0.7ER		Enable rising edge interrupt for P0.7.	0
8	P0.8ER		Enable rising edge interrupt for P0.8.	0
9	P0.9ER		Enable rising edge interrupt for P0.9.	0
10	P0.10ER		Enable rising edge interrupt for P0.10.	0
11	P0.11ER		Enable rising edge interrupt for P0.11.	0
14:12	-		Reserved	NA
15	P0.15ER		Enable rising edge interrupt for P0.15.	0
16	P0.16ER		Enable rising edge interrupt for P0.16.	0

© NXP B.V. 2010. All rights reserved.

Table 113. GPIO Interrupt Enable for port 0 Rising Edge (IO0IntEnR - 0x4002 8090) bit description

Bit	Symbol	Value	Description	Reset value
17	P0.17ER		Enable rising edge interrupt for P0.17.	0
18	P0.18ER		Enable rising edge interrupt for P0.18.	0
19	P0.19ER[1]		Enable rising edge interrupt for P0.19.	0
20	P0.20ER[1]		Enable rising edge interrupt for P0.20.	0
21	P0.21ER[1]		Enable rising edge interrupt for P0.21.	0
22	P0.22ER		Enable rising edge interrupt for P0.22.	0
23	P0.23ER[1]		Enable rising edge interrupt for P0.23.	0
24	P0.24ER[1]		Enable rising edge interrupt for P0.24.	0
25	P0.25ER		Enable rising edge interrupt for P0.25.	0
26	P0.26ER		Enable rising edge interrupt for P0.26.	0
27	P0.27ER[1]		Enable rising edge interrupt for P0.27.	0
28	P0.28ER[1]		Enable rising edge interrupt for P0.28.	0
29	P0.29ER		Enable rising edge interrupt for P0.29.	0
30	P0.30ER		Enable rising edge interrupt for P0.30.	0
31	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.3 GPIO Interrupt Enable for port 2 Rising Edge (IO2IntEnR - 0x4002 80B0)

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

Table 114. GPIO Interrupt Enable for port 2 Rising Edge (IO2IntEnR - 0x4002 80B0) bit description

Bit	Symbol	Value	Description	Reset value
0	P2.0ER		Enable rising edge interrupt for P2.0.	0
		0	Rising edge interrupt is disabled on P2.0.	
		1	Rising edge interrupt is enabled on P2.0.	
1	P2.1ER		Enable rising edge interrupt for P2.1.	0
2	P2.2ER		Enable rising edge interrupt for P2.2.	0
3	P2.3ER		Enable rising edge interrupt for P2.3.	0
4	P2.4ER		Enable rising edge interrupt for P2.4.	0
5	P2.5ER		Enable rising edge interrupt for P2.5.	0
6	P2.6ER		Enable rising edge interrupt for P2.6.	0
7	P2.7ER		Enable rising edge interrupt for P2.7.	0
8	P2.8ER		Enable rising edge interrupt for P2.8.	0
9	P2.9ER		Enable rising edge interrupt for P2.9.	0
10	P2.10ER		Enable rising edge interrupt for P2.10.	0
11	P2.11ER[1]		Enable rising edge interrupt for P2.11.	0

© NXP B.V. 2010. All rights reserved.

Table 114. GPIO Interrupt Enable for port 2 Rising Edge (IO2IntEnR - 0x4002 80B0) bit description

Bit	Symbol	Value	Description	Reset value
12	P2.12ER[1]		Enable rising edge interrupt for P2.12.	0
13	P2.13ER[1]		Enable rising edge interrupt for P2.13.	0
31:14	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.4 GPIO Interrupt Enable for port 0 Falling Edge (IO0IntEnF - 0x4002 8094)

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

Table 115. GPIO Interrupt Enable for port 0 Falling Edge (IO0IntEnF - address 0x4002 8094) bit description

Bit	Symbol	Value	Description	Reset value
0	P0.0EF		Enable falling edge interrupt for P0.0	0
		0	Falling edge interrupt is disabled on P0.0.	
		1	Falling edge interrupt is enabled on P0.0.	
1	P0.1EF		Enable falling edge interrupt for P0.1.	0
2	P0.2EF		Enable falling edge interrupt for P0.2.	0
3	P0.3EF		Enable falling edge interrupt for P0.3.	0
4	P0.4EF[1]		Enable falling edge interrupt for P0.4.	0
5	P0.5EF[1]		Enable falling edge interrupt for P0.5.	0
6	P0.6EF		Enable falling edge interrupt for P0.6.	0
7	P0.7EF		Enable falling edge interrupt for P0.7.	0
8	P0.8EF		Enable falling edge interrupt for P0.8.	0
9	P0.9EF		Enable falling edge interrupt for P0.9.	0
10	P0.10EF		Enable falling edge interrupt for P0.10.	0
11	P0.11EF		Enable falling edge interrupt for P0.11.	0
14:12	-		Reserved.	NA
15	P0.15EF		Enable falling edge interrupt for P0.15.	0
16	P0.16EF		Enable falling edge interrupt for P0.16.	0
17	P0.17EF		Enable falling edge interrupt for P0.17.	0
18	P0.18EF		Enable falling edge interrupt for P0.18.	0
19	P0.19EF[1]		Enable falling edge interrupt for P0.19.	0
20	P0.20EF[1]		Enable falling edge interrupt for P0.20.	0
21	P0.21EF[1]		Enable falling edge interrupt for P0.21.	0
22	P0.22EF		Enable falling edge interrupt for P0.22.	0
23	P0.23EF[1]		Enable falling edge interrupt for P0.23.	0
24	P0.24EF[1]		Enable falling edge interrupt for P0.24.	0
25	P0.25EF		Enable falling edge interrupt for P0.25.	0
26	P0.26EF		Enable falling edge interrupt for P0.26.	0
27	P0.27EF[1]		Enable falling edge interrupt for P0.27.	0

© NXP B.V. 2010. All rights reserved.

Table 115. GPIO Interrupt Enable for port 0 Falling Edge (IO0IntEnF - address 0x4002 8094) bit description

Bit	Symbol	Value	Description	Reset value
28	P0.28EF[1]		Enable falling edge interrupt for P0.28.	0
29	P0.29EF		Enable falling edge interrupt for P0.29.	0
30	P0.30EF		Enable falling edge interrupt for P0.30.	0
31	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.5 GPIO Interrupt Enable for port 2 Falling Edge (IO2IntEnF - 0x4002 80B4)

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

Table 116. GPIO Interrupt Enable for port 2 Falling Edge (IO2IntEnF - 0x4002 80B4) bit description

Bit	Symbol	Value	Description	Reset value
0	P2.0EF		Enable falling edge interrupt for P2.0	0
		0	Falling edge interrupt is disabled on P2.0.	
		1	Falling edge interrupt is enabled on P2.0.	
1	P2.1EF		Enable falling edge interrupt for P2.1.	0
2	P2.2EF		Enable falling edge interrupt for P2.2.	0
3	P2.3EF		Enable falling edge interrupt for P2.3.	0
4	P2.4EF		Enable falling edge interrupt for P2.4.	0
5	P2.5EF		Enable falling edge interrupt for P2.5.	0
6	P2.6EF		Enable falling edge interrupt for P2.6.	0
7	P2.7EF		Enable falling edge interrupt for P2.7.	0
8	P2.8EF		Enable falling edge interrupt for P2.8.	0
9	P2.9EF		Enable falling edge interrupt for P2.9.	0
10	P2.10EF		Enable falling edge interrupt for P2.10.	0
11	P2.11EF[1]		Enable falling edge interrupt for P2.11.	0
12	P2.12EF[1]		Enable falling edge interrupt for P2.12.	0
13	P2.13EF[1]		Enable falling edge interrupt for P2.13.	0
31:14	-		Reserved.	NA

[1] Not available on 80-pin package.

© NXP B.V. 2010. All rights reserved.

5.6.6 GPIO Interrupt Status for port 0 Rising Edge Interrupt (IO0IntStatR - 0x4002 8084)

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

Table 117. GPIO Interrupt Status for port 0 Rising Edge Interrupt (IO0IntStatR - 0x4002 8084) bit description

Bit	Symbol	Value	Description	Reset value
0	P0.0REI		Status of Rising Edge Interrupt for P0.0	0
		0	A rising edge has not been detected on P0.0.	
		1	Interrupt has been generated due to a rising edge on P0.0.	
1	P0.1REI		Status of Rising Edge Interrupt for P0.1.	0
2	P0.2REI		Status of Rising Edge Interrupt for P0.2.	0
3	P0.3REI		Status of Rising Edge Interrupt for P0.3.	0
4	P0.4REI[1]		Status of Rising Edge Interrupt for P0.4.	0
5	P0.5REI[1]		Status of Rising Edge Interrupt for P0.5.	0
6	P0.6REI		Status of Rising Edge Interrupt for P0.6.	0
7	P0.7REI		Status of Rising Edge Interrupt for P0.7.	0
8	P0.8REI		Status of Rising Edge Interrupt for P0.8.	0
9	P0.9REI		Status of Rising Edge Interrupt for P0.9.	0
10	P0.10REI		Status of Rising Edge Interrupt for P0.10.	0
11	P0.11REI		Status of Rising Edge Interrupt for P0.11.	0
14:12	-		Reserved.	NA
15	P0.15REI		Status of Rising Edge Interrupt for P0.15.	0
16	P0.16REI		Status of Rising Edge Interrupt for P0.16.	0
17	P0.17REI		Status of Rising Edge Interrupt for P0.17.	0
18	P0.18REI		Status of Rising Edge Interrupt for P0.18.	0
19	P0.19REI[1]		Status of Rising Edge Interrupt for P0.19.	0
20	P0.20REI[1]		Status of Rising Edge Interrupt for P0.20.	0
21	P0.21REI[1]		Status of Rising Edge Interrupt for P0.21.	0
22	P0.22REI		Status of Rising Edge Interrupt for P0.22.	0
23	P0.23REI[1]		Status of Rising Edge Interrupt for P0.23.	0
24	P0.24REI[1]		Status of Rising Edge Interrupt for P0.24.	0
25	P0.25REI		Status of Rising Edge Interrupt for P0.25.	0
26	P0.26REI		Status of Rising Edge Interrupt for P0.26.	0
27	P0.27REI[1]		Status of Rising Edge Interrupt for P0.27.	0
28	P0.28REI[1]		Status of Rising Edge Interrupt for P0.28.	0
29	P0.29REI		Status of Rising Edge Interrupt for P0.29.	0
30	P0.30REI		Status of Rising Edge Interrupt for P0.30.	0
31	-		Reserved.	NA

[1] Not available on 80-pin package.

© NXP B.V. 2010. All rights reserved.

5.6.7 GPIO Interrupt Status for port 2 Rising Edge Interrupt (IO2IntStatR - 0x4002 80A4)

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

Table 118. GPIO Interrupt Status for port 2 Rising Edge Interrupt (IO2IntStatR - 0x4002 80A4) bit description

Bit	Symbol	Value	Description	Reset value
0	P2.0REI		Status of Rising Edge Interrupt for P2.0	0
		0	A rising edge has not been detected on P2.0.	
		1	Interrupt has been generated due to a rising edge on P2.0.	
1	P2.1REI		Status of Rising Edge Interrupt for P2.1.	0
2	P2.2REI		Status of Rising Edge Interrupt for P2.2.	0
3	P2.3REI		Status of Rising Edge Interrupt for P2.3.	0
4	P2.4REI		Status of Rising Edge Interrupt for P2.4.	0
5	P2.5REI		Status of Rising Edge Interrupt for P2.5.	0
6	P2.6REI		Status of Rising Edge Interrupt for P2.6.	0
7	P2.7REI		Status of Rising Edge Interrupt for P2.7.	0
8	P2.8REI		Status of Rising Edge Interrupt for P2.8.	0
9	P2.9REI		Status of Rising Edge Interrupt for P2.9.	0
10	P2.10REI		Status of Rising Edge Interrupt for P2.10.	0
11	P2.11REI[1]		Status of Rising Edge Interrupt for P2.11.	0
12	P2.12REI[1]		Status of Rising Edge Interrupt for P2.12.	0
13	P2.13REI[1]		Status of Rising Edge Interrupt for P2.13.	0
31:14	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.8 GPIO Interrupt Status for port 0 Falling Edge Interrupt (IO0IntStatF - 0x4002 8088)

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

Table 119. GPIO Interrupt Status for port 0 Falling Edge Interrupt (IO0IntStatF - 0x4002 8088) bit description

Bit	Symbol	Value	Description	Reset value
0	P0.0FEI		Status of Falling Edge Interrupt for P0.0	0
		0	A falling edge has not been detected on P0.0.	
		1	Interrupt has been generated due to a falling edge on P0.0.	
1	P0.1FEI		Status of Falling Edge Interrupt for P0.1.	0
2	P0.2FEI		Status of Falling Edge Interrupt for P0.2.	0
3	P0.3FEI		Status of Falling Edge Interrupt for P0.3.	0
4	P0.4FEI[1]		Status of Falling Edge Interrupt for P0.4.	0
5	P0.5FEI[1]		Status of Falling Edge Interrupt for P0.5.	0
6	P0.6FEI		Status of Falling Edge Interrupt for P0.6.	0
7	P0.7FEI		Status of Falling Edge Interrupt for P0.7.	0

© NXP B.V. 2010. All rights reserved.

Table 119. GPIO Interrupt Status for port 0 Falling Edge Interrupt (IO0IntStatF - 0x4002 8088) bit description

Bit	Symbol	Value	Description	Reset value
8	P0.8FEI		Status of Falling Edge Interrupt for P0.8.	0
9	P0.9FEI		Status of Falling Edge Interrupt for P0.9.	0
10	P0.10FEI		Status of Falling Edge Interrupt for P0.10.	0
11	P0.11FEI		Status of Falling Edge Interrupt for P0.11.	0
14:12	-		Reserved.	NA
15	P0.15FEI		Status of Falling Edge Interrupt for P0.15.	0
16	P0.16FEI		Status of Falling Edge Interrupt for P0.16.	0
17	P0.17FEI		Status of Falling Edge Interrupt for P0.17.	0
18	P0.18FEI		Status of Falling Edge Interrupt for P0.18.	0
19	P0.19FEI[1]		Status of Falling Edge Interrupt for P0.19.	0
20	P0.20FEI[1]		Status of Falling Edge Interrupt for P0.20.	0
21	P0.21FEI[1]		Status of Falling Edge Interrupt for P0.21.	0
22	P0.22FEI		Status of Falling Edge Interrupt for P0.22.	0
23	P0.23FEI[1]		Status of Falling Edge Interrupt for P0.23.	0
24	P0.24FEI[1]		Status of Falling Edge Interrupt for P0.24.	0
25	P0.25FEI		Status of Falling Edge Interrupt for P0.25.	0
26	P0.26FEI		Status of Falling Edge Interrupt for P0.26.	0
27	P0.27FEI[1]		Status of Falling Edge Interrupt for P0.27.	0
28	P0.28FEI[1]		Status of Falling Edge Interrupt for P0.28.	0
29	P0.29FEI		Status of Falling Edge Interrupt for P0.29.	0
30	P0.30FEI		Status of Falling Edge Interrupt for P0.30.	0
31	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.9 GPIO Interrupt Status for port 2 Falling Edge Interrupt (IO2IntStatF - 0x4002 80A8)

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

Table 120. GPIO Interrupt Status for port 2 Falling Edge Interrupt (IO2IntStatF - 0x4002 80A8) bit description

Bit	Symbol	Value	Description	Reset value
0	P2.0FEI		Status of Falling Edge Interrupt for P2.0	0
		0	A falling edge has not been detected on P2.0.	
		1	Interrupt has been generated due to a falling edge on P2.0.	
1	P2.1FEI		Status of Falling Edge Interrupt for P2.1.	0
2	P2.2FEI		Status of Falling Edge Interrupt for P2.2.	0
3	P2.3FEI		Status of Falling Edge Interrupt for P2.3.	0
4	P2.4FEI		Status of Falling Edge Interrupt for P2.4.	0
5	P2.5FEI		Status of Falling Edge Interrupt for P2.5.	0
6	P2.6FEI		Status of Falling Edge Interrupt for P2.6.	0

© NXP B.V. 2010. All rights reserved.

Table 120. GPIO Interrupt Status for port 2 Falling Edge Interrupt (IO2IntStatF - 0x4002 80A8) bit description

Bit	Symbol	Value	Description	Reset value
7	P2.7FEI		Status of Falling Edge Interrupt for P2.7.	0
8	P2.8FEI		Status of Falling Edge Interrupt for P2.8.	0
9	P2.9FEI		Status of Falling Edge Interrupt for P2.9.	0
10	P2.10FEI		Status of Falling Edge Interrupt for P2.10.	0
11	P2.11FEI[1]		Status of Falling Edge Interrupt for P2.11.	0
12	P2.12FEI[1]		Status of Falling Edge Interrupt for P2.12.	0
13	P2.13FEI[1]		Status of Falling Edge Interrupt for P2.13.	0
31:14	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.10 GPIO Interrupt Clear register for port 0 (IO0IntClr - 0x4002 808C)

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

Table 121. GPIO Interrupt Clear register for port 0 (IO0IntClr - 0x4002 808C)) bit description

Bit	Symbol	Value	Description	Reset value
0	P0.0CI		Clear GPIO port Interrupts for P0.0	0
		0	Corresponding bits in IOxIntStatR and IOxIntStatF are unchanged.	
		1	Corresponding bits in IOxIntStatR and IOxStatF are cleared.	
1	P0.1CI		Clear GPIO port Interrupts for P0.1.	0
2	P0.2CI		Clear GPIO port Interrupts for P0.2.	0
3	P0.3CI		Clear GPIO port Interrupts for P0.3.	0
4	P0.4CI[1]		Clear GPIO port Interrupts for P0.4.	0
5	P0.5CI[1]		Clear GPIO port Interrupts for P0.5.	0
6	P0.6CI		Clear GPIO port Interrupts for P0.6.	0
7	P0.7CI		Clear GPIO port Interrupts for P0.7.	0
8	P0.8CI		Clear GPIO port Interrupts for P0.8.	0
9	P0.9CI		Clear GPIO port Interrupts for P0.9.	0
10	P0.10CI		Clear GPIO port Interrupts for P0.10.	0
11	P0.11CI		Clear GPIO port Interrupts for P0.11.	0
14:12	-		Reserved.	NA
15	P0.15CI		Clear GPIO port Interrupts for P0.15.	0
16	P0.16CI		Clear GPIO port Interrupts for P0.16.	0
17	P0.17CI		Clear GPIO port Interrupts for P0.17.	0
18	P0.18CI		Clear GPIO port Interrupts for P0.18.	0
19	P0.19CI[1]		Clear GPIO port Interrupts for P0.19.	0
20	P0.20CI[1]		Clear GPIO port Interrupts for P0.20.	0
21	P0.21CI[1]		Clear GPIO port Interrupts for P0.21.	0
22	P0.22CI		Clear GPIO port Interrupts for P0.22.	0

© NXP B.V. 2010. All rights reserved.

Table 121. GPIO Interrupt Clear register for port 0 (IO0IntClr - 0x4002 808C)) bit description

Bit	Symbol	Value	Description	Reset value
23	P0.23CI[1]		Clear GPIO port Interrupts for P0.23.	0
24	P0.24CI[1]		Clear GPIO port Interrupts for P0.24.	0
25	P0.25CI		Clear GPIO port Interrupts for P0.25.	0
26	P0.26CI		Clear GPIO port Interrupts for P0.26.	0
27	P0.27CI[1]		Clear GPIO port Interrupts for P0.27.	0
28	P0.28CI[1]		Clear GPIO port Interrupts for P0.28.	0
29	P0.29CI		Clear GPIO port Interrupts for P0.29.	0
30	P0.30CI		Clear GPIO port Interrupts for P0.30.	0
31	-		Reserved.	NA

[1] Not available on 80-pin package.

5.6.11 GPIO Interrupt Clear register for port 0 (IO2IntClr - 0x4002 80AC)

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

Table 122. GPIO Interrupt Clear register for port 0 (IO2IntClr - 0x4002 80AC) bit description

Bit	Symbol	Value	Description	Reset value
0	P2.0CI		Clear GPIO port Interrupts for P2.0	0
		0	Corresponding bits in IOxIntStatR and IOxIntStatF are unchanged.	
		1	Corresponding bits in IOxIntStatR and IOxStatF are cleared.	
1	P2.1CI		Clear GPIO port Interrupts for P2.1.	0
2	P2.2CI		Clear GPIO port Interrupts for P2.2.	0
3	P2.3CI		Clear GPIO port Interrupts for P2.3.	0
4	P2.4CI		Clear GPIO port Interrupts for P2.4.	0
5	P2.5CI		Clear GPIO port Interrupts for P2.5.	0
6	P2.6CI		Clear GPIO port Interrupts for P2.6.	0
7	P2.7CI		Clear GPIO port Interrupts for P2.7.	0
8	P2.8CI		Clear GPIO port Interrupts for P2.8.	0
9	P2.9CI		Clear GPIO port Interrupts for P2.9.	0
10	P2.10CI		Clear GPIO port Interrupts for P2.10.	0
11	P2.11CI[1]		Clear GPIO port Interrupts for P2.11.	0
12	P2.12CI[1]		Clear GPIO port Interrupts for P2.12.	0
13	P2.13CI[1]		Clear GPIO port Interrupts for P2.13.	0
31:14	-		Reserved.	NA

[1] Not available on 80-pin package.

© NXP B.V. 2010. All rights reserved.

6. GPIO usage notes

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

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

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

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

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

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

```
FIO0PIN1  = 0xA5;
```

6.2 Writing to FIOSET/FIOCLR vs. FIOPIN

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

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

A subset of a port's pins may be changed by using the FIOMASK register to define which pins are affected. FIOMASK is set up to contain zeroes in bits corresponding to pins that will be changed, and ones for all others. Solution 2 from Section 9–6.1 above illustrates output of 0xA5 on PORT0 pins 15 to 8 while preserving all other PORT0 output pins as they were before.

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The Ethernet controller is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCENET.

   Remark: On reset, the Ethernet block is disabled (PCENET = 0).
2. Clock: see Table 4–38.
3. Pins: Enable Ethernet pins through the PINSEL registers and select their modes through the PINMODE registers, see Section 8–5.
4. Wake-up: Activity on the Ethernet port can wake up the microcontroller from Power-down mode, see Section 4–8.8.
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. Initialization: see Section 10–17.2.

2. Introduction

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

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

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

Table 123. Ethernet acronyms, abbreviations, and definitions

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

© NXP B.V. 2010. All rights reserved.

Table 123. Ethernet acronyms, abbreviations, and definitions

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

3. Features

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

© NXP B.V. 2010. All rights reserved.

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

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

4. Architecture and operation

Figure 10–16 shows the internal architecture of the Ethernet block.

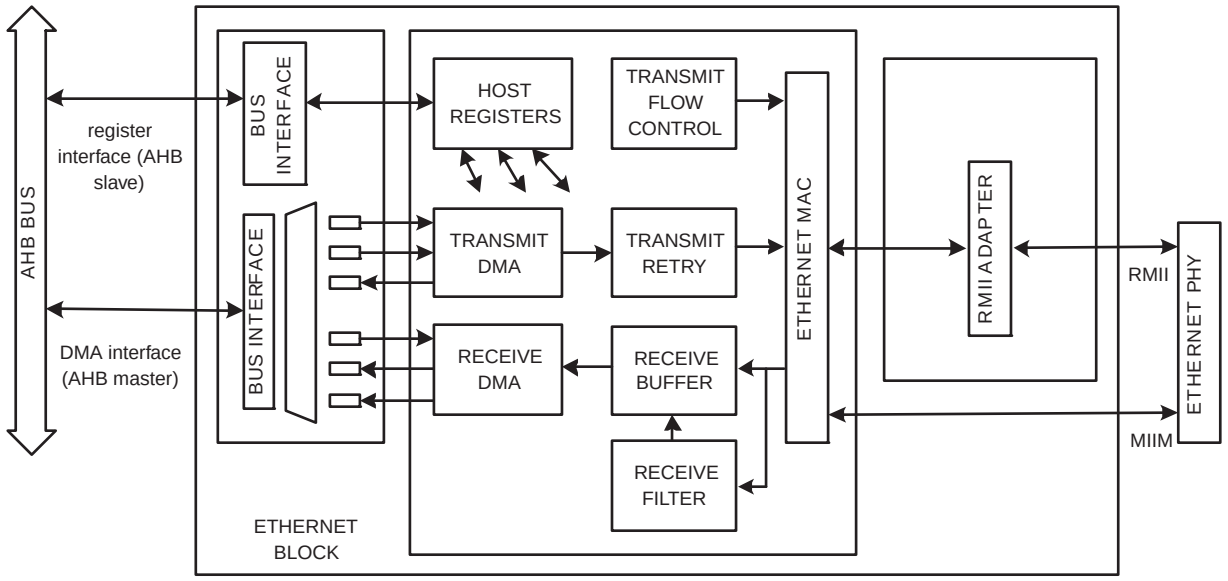


Fig 16. Ethernet block diagram

The block diagram for the Ethernet block consists of:

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

© NXP B.V. 2010. All rights reserved.

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

5. DMA engine functions

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

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

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

6. Overview of DMA operation

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

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

© NXP B.V. 2010. All rights reserved.

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

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

7. Ethernet Packet

Figure 10–17 illustrates the different fields in an Ethernet packet.

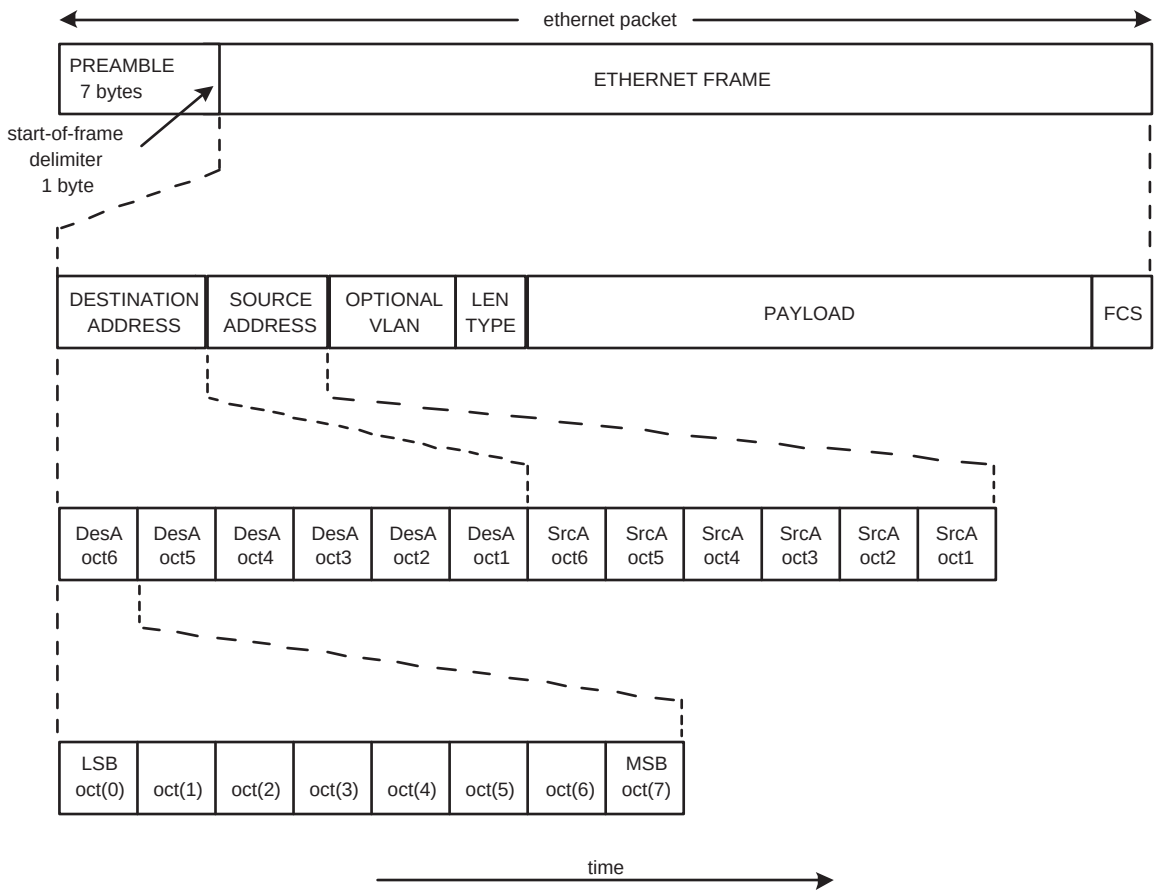


Fig 17. Ethernet packet fields

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

© NXP B.V. 2010. All rights reserved.

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

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

8. Overview

8.1 Partitioning

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

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

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

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

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

© NXP B.V. 2010. All rights reserved.

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

The Ethernet block has a standard Reduced Media Independent Interface (RMII) to connect to an external Ethernet PHY chip. Registers in the PHY chip are accessed via the AHB interface through the serial management connection of the MIIM bus, typically operating at 2.5 MHz.

8.2 Example PHY Devices

Some examples of compatible PHY devices are shown in Table 10–124.

Table 124. Example PHY Devices

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

9. Pin description

Table 10–125 shows the signals used for connecting the Reduced Media Independent Interface (RMII) to the external PHY.

Table 125. Ethernet RMII pin descriptions

Pin Name	Type	Pin Description
ENET_TX_EN	Output	Transmit data enable
ENET_TXD[1:0]	Output	Transmit data, 2 bits
ENET_RXD[1:0]	Input	Receive data, 2 bits.
ENET_RX_ER	Input	Receive error.
ENET_CRS	Input	Carrier sense/data valid.
ENET_REF_CLK/ ENET_RX_CLK	Input	Reference clock

Table 10–126 shows the signals used for Media Independent Interface Management (MIIM) of the external PHY.

Table 126. Ethernet MIIM pin descriptions

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

© NXP B.V. 2010. All rights reserved.

10. Registers and software interface

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

10.1 Register map

Table 10–127 lists the registers, register addresses and other basic information. The total AHB address space required is 4 kilobytes.

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

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

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

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

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

Table 127. Ethernet register definitions

Name	Description	Access	Reset Value	Address
MAC registers				
MAC1	MAC configuration register 1.	R/W	0x8000	0x5000 0000
MAC2	MAC configuration register 2.	R/W	0	0x5000 0004
IPGT	Back-to-Back Inter-Packet-Gap register.	R/W	0	0x5000 0008
IPGR	Non Back-to-Back Inter-Packet-Gap register.	R/W	0	0x5000 000C
CLRT	Collision window / Retry register.	R/W	0x370F	0x5000 0010
MAXF	Maximum Frame register.	R/W	0x0600	0x5000 0014
SUPP	PHY Support register.	R/W	0	0x5000 0018
TEST	Test register.	R/W	0	0x5000 001C
MCFG	MII Mgmt Configuration register.	R/W	0	0x5000 0020
MCMD	MII Mgmt Command register.	R/W	0	0x5000 0024
MADR	MII Mgmt Address register.	R/W	0	0x5000 0028
MWTD	MII Mgmt Write Data register.	WO	0	0x5000 002C
MRDD	MII Mgmt Read Data register.	RO	0	0x5000 0030
MIND	MII Mgmt Indicators register.	RO	0	0x5000 0034
SA0	Station Address 0 register.	R/W	0	0x5000 0040
SA1	Station Address 1 register.	R/W	0	0x5000 0044
SA2	Station Address 2 register.	R/W	0	0x5000 0048
Control registers				
Command	Command register.	R/W	0	0x5000 0100

© NXP B.V. 2010. All rights reserved.

Table 127. Ethernet register definitions

Name	Description	Access	Reset Value	Address
Status	Status register.	RO	0	0x5000 0104
RxDescriptor	Receive descriptor base address register.	R/W	0	0x5000 0108
RxStatus	Receive status base address register.	R/W	0	0x5000 010C
RxDescriptorNumber	Receive number of descriptors register.	R/W	0	0x5000 0110
RxProduceIndex	Receive produce index register.	RO	0	0x5000 0114
RxConsumeIndex	Receive consume index register.	R/W	0	0x5000 0118
TxDescriptor	Transmit descriptor base address register.	R/W	0	0x5000 011C
TxStatus	Transmit status base address register.	R/W	0	0x5000 0120
TxDescriptorNumber	Transmit number of descriptors register.	R/W	0	0x5000 0124
TxProduceIndex	Transmit produce index register.	R/W	0	0x5000 0128
TxConsumeIndex	Transmit consume index register.	RO	0	0x5000 012C
TSV0	Transmit status vector 0 register.	RO	0	0x5000 0158
TSV1	Transmit status vector 1 register.	RO	0	0x5000 015C
RSV	Receive status vector register.	RO	0	0x5000 0160
FlowControlCounter	Flow control counter register.	R/W	0	0x5000 0170
FlowControlStatus	Flow control status register.	RO	0	0x5000 0174
Rx filter registers				
RxFliterCtrl	Receive filter control register.		0	0x5000 0200
RxFilterWoLStatus	Receive filter WoL status register.		0	0x5000 0204
RxFilterWoLClear	Receive filter WoL clear register.		0	0x5000 0208
HashFilterL	Hash filter table LSBs register.		0	0x5000 0210
HashFilterH	Hash filter table MSBs register.		0	0x5000 0214
Module control registers				
IntStatus	Interrupt status register.	RO	0	0x5000 0FE0
IntEnable	Interrupt enable register.	R/W	0	0x5000 0FE4
IntClear	Interrupt clear register.	WO	0	0x5000 0FE8
IntSet	Interrupt set register.	WO	0	0x5000 0FEC
PowerDown	Power-down register.	R/W	0	0x5000 0FF4

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

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

© NXP B.V. 2010. All rights reserved.

11. Ethernet MAC register definitions

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

11.1 MAC Configuration Register 1 (MAC1 - 0x5000 0000)

The MAC configuration register 1 (MAC1) has an address of 0x5000 0000. Its bit definition is shown in Table 10–128.

Table 128. MAC Configuration register 1 (MAC1 - address 0x5000 0000) bit description

Bit	Symbol	Function	Reset value
0	RECEIVE ENABLE	Set this to allow receive frames to be received. Internally the MAC synchronizes this control bit to the incoming receive stream.	0
1	PASS ALL RECEIVE FRAMES	When enabled (set to '1'), the MAC will pass all frames regardless of type (normal vs. Control). When disabled, the MAC does not pass valid Control frames.	0
2	RX FLOW CONTROL	When enabled (set to '1'), the MAC acts upon received PAUSE Flow Control frames. When disabled, received PAUSE Flow Control frames are ignored.	0
3	TX FLOW CONTROL	When enabled (set to '1'), PAUSE Flow Control frames are allowed to be transmitted. When disabled, Flow Control frames are blocked.	0
4	LOOPBACK	Setting this bit will cause the MAC Transmit interface to be looped back to the MAC Receive interface. Clearing this bit results in normal operation.	0
7:5	-	Unused	0x0
8	RESET TX	Setting this bit will put the Transmit Function logic in reset.	0
9	RESET MCS / TX	Setting this bit resets the MAC Control Sublayer / Transmit logic. The MCS logic implements flow control.	0
10	RESET RX	Setting this bit will put the Ethernet receive logic in reset.	0
11	RESET MCS / RX	Setting this bit resets the MAC Control Sublayer / Receive logic. The MCS logic implements flow control.	0x0
13:12	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
14	SIMULATION RESET	Setting this bit will cause a reset to the random number generator within the Transmit Function.	0
15	SOFT RESET	Setting this bit will put all modules within the MAC in reset except the Host Interface.	1
31:16	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

11.2 MAC Configuration Register 2 (MAC2 - 0x5000 0004)

The MAC configuration register 2 (MAC2) has an address of 0x5000 0004. Its bit definition is shown in Table 10–129.

© NXP B.V. 2010. All rights reserved.

Table 129. MAC Configuration register 2 (MAC2 - address 0x5000 0004) bit description

Bit	Symbol	Function	Reset value
0	FULL-DUPLEX	When enabled (set to '1'), the MAC operates in Full-Duplex mode. When disabled, the MAC operates in Half-Duplex mode.	0
1	FRAME LENGTH CHECKING	When enabled (set to '1'), both transmit and receive frame lengths are compared to the Length/Type field. If the Length/Type field represents a length then the check is performed. Mismatches are reported in the StatusInfo word for each received frame.	0
2	HUGE FRAME ENABLE	When enabled (set to '1'), frames of any length are transmitted and received.	0
3	DELAYED CRC	This bit determines the number of bytes, if any, of proprietary header information that exist on the front of IEEE 802.3 frames. When 1, four bytes of header (ignored by the CRC function) are added. When 0, there is no proprietary header.	0
4	CRC ENABLE	Set this bit to append a CRC to every frame whether padding was required or not. Must be set if PAD/CRC ENABLE is set. Clear this bit if frames presented to the MAC contain a CRC.	0
5	PAD / CRC ENABLE	Set this bit to have the MAC pad all short frames. Clear this bit if frames presented to the MAC have a valid length. This bit is used in conjunction with AUTO PAD ENABLE and VLAN PAD ENABLE. See Table 10–131 - Pad Operation for details on the pad function.	0
6	VLAN PAD ENABLE	Set this bit to cause the MAC to pad all short frames to 64 bytes and append a valid CRC. Consult Table 10–131 - Pad Operation for more information on the various padding features. **Note:** This bit is ignored if PAD / CRC ENABLE is cleared.	0
7	AUTO DETECT PAD ENABLE	Set this bit to cause the MAC to automatically detect the type of frame, either tagged or un-tagged, by comparing the two octets following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly. Table 10–131 - Pad Operation provides a description of the pad function based on the configuration of this register. **Note:** This bit is ignored if PAD / CRC ENABLE is cleared.	0
8	PURE PREAMBLE ENFORCEMENT	When enabled (set to '1'), the MAC will verify the content of the preamble to ensure it contains 0x55 and is error-free. A packet with an incorrect preamble is discarded. When disabled, no preamble checking is performed.	0
9	LONG PREAMBLE ENFORCEMENT	When enabled (set to '1'), the MAC only allows receive packets which contain preamble fields less than 12 bytes in length. When disabled, the MAC allows any length preamble as per the Standard.	0
11:10	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
12	NO BACKOFF	When enabled (set to '1'), the MAC will immediately retransmit following a collision rather than using the Binary Exponential Backoff algorithm as specified in the Standard.	0
13	BACK PRESSURE / NO BACKOFF	When enabled (set to '1'), after the MAC incidentally causes a collision during back pressure, it will immediately retransmit without backoff, reducing the chance of further collisions and ensuring transmit packets get sent.	0
14	EXCESS DEFER	When enabled (set to '1') the MAC will defer to carrier indefinitely as per the Standard. When disabled, the MAC will abort when the excessive deferral limit is reached.	0
31:15	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

© NXP B.V. 2010. All rights reserved.

Table 130. Pad operation

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

11.3 Back-to-Back Inter-Packet-Gap Register (IPGT - 0x5000 0008)

The Back-to-Back Inter-Packet-Gap register (IPGT) has an address of 0x5000 0008. Its bit definition is shown in Table 10–131.

Table 131. Back-to-back Inter-packet-gap register (IPGT - address 0x5000 0008) bit description

Bit	Symbol	Function	Reset value
6:0	BACK-TO-BACK INTER-PACKET-GAP	This is a programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet to the beginning of the next. In Full-Duplex mode, the register value should be the desired period in nibble times minus 3. In Half-Duplex mode, the register value should be the desired period in nibble times minus 6. In Full-Duplex the recommended setting is 0x15 (21d), which represents the minimum IPG of 960 ns (in 100 Mbps mode) or 9.6 µs (in 10 Mbps mode). In Half-Duplex the recommended setting is 0x12 (18d), which also represents the minimum IPG of 960 ns (in 100 Mbps mode) or 9.6 µs (in 10 Mbps mode).	0x0
31:7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

11.4 Non Back-to-Back Inter-Packet-Gap Register (IPGR - 0x5000 000C)

The Non Back-to-Back Inter-Packet-Gap register (IPGR) has an address of 0x5000 000C. Its bit definition is shown in Table 10–132.

Table 132. Non Back-to-back Inter-packet-gap register (IPGR - address 0x5000 000C) bit description

Bit	Symbol	Function	Reset value
6:0	NON-BACK-TO-BACK INTER-PACKET-GAP PART2	This is a programmable field representing the Non-Back-to-Back Inter-Packet-Gap. The recommended value is 0x12 (18d), which represents the minimum IPG of 960 ns (in 100 Mbps mode) or 9.6 µs (in 10 Mbps mode).	0x0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
14:8	NON-BACK-TO-BACK INTER-PACKET-GAP PART1	This is a programmable field representing the optional carrierSense window referenced in IEEE 802.3/4.2.3.2.1 'Carrier Deference'. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x0 to IPGR2. The recommended value is 0xC (12d)	0x0
31:15	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0

© NXP B.V. 2010. All rights reserved.

11.5 Collision Window / Retry Register (CLRT - 0x5000 0010)

The Collision window / Retry register (CLRT) has an address of 0x5000 0010. Its bit definition is shown in Table 10–133.

Table 133. Collision Window / Retry register (CLRT - address 0x5000 0010) bit description

Bit	Symbol	Function	Reset value
3:0	RETRANSMISSION MAXIMUM	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The Standard specifies the attemptLimit to be 0xF (15d). See IEEE 802.3/4.2.3.2.5.	0xF
7:4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0x0
13:8	COLLISION WINDOW	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. The default value of 0x37 (55d) represents a 56 byte window following the preamble and SFD.	0x37
31:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11.6 Maximum Frame Register (MAXF - 0x5000 0014)

The Maximum Frame register (MAXF) has an address of 0x5000 0014. Its bit definition is shown in Table 10–134.

Table 134. Maximum Frame register (MAXF - address 0x5000 0014) bit description

Bit	Symbol	Function	Reset value
15:0	MAXIMUM FRAME LENGTH	This field resets to the value 0x0600, which represents a maximum receive frame of 1536 octets. An untagged maximum size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. If a shorter maximum length restriction is desired, program this 16-bit field.	0x0600
31:16	-	Unused	0x0

11.7 PHY Support Register (SUPP - 0x5000 0018)

The PHY Support register (SUPP) has an address of 0x5000 0018. The SUPP register provides additional control over the RMII interface. The bit definition of this register is shown in Table 10–135.

Table 135. PHY Support register (SUPP - address 0x5000 0018) bit description

Bit	Symbol	Function	Reset value
7:0	-	Unused	0x0
8	SPEED	This bit configures the Reduced MII logic for the current operating speed. When set, 100 Mbps mode is selected. When cleared, 10 Mbps mode is selected.	0
31:9	-	Unused	0x0

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

11.8 Test Register (TEST - 0x5000 001C)

The Test register (TEST) has an address of 0x5000 001C. The bit definition of this register is shown in Table 10–136. These bits are used for testing purposes only.

© NXP B.V. 2010. All rights reserved.

Table 136. Test register (TEST - address 0x5000) bit description

Bit	Symbol	Function	Reset value
0	SHORTCUT PAUSE QUANTA	This bit reduces the effective PAUSE quanta from 64 byte-times to 1 byte-time.	0
1	TEST PAUSE	This bit causes the MAC Control sublayer to inhibit transmissions, just as if a PAUSE Receive Control frame with a nonzero pause time parameter was received.	0
2	TEST BACKPRESSURE	Setting this bit will cause the MAC to assert backpressure on the link. Backpressure causes preamble to be transmitted, raising carrier sense. A transmit packet from the system will be sent during backpressure.	0
31:3	-	Unused	0x0

11.9 MII Mgmt Configuration Register (MCFG - 0x5000 0020)

The MII Mgmt Configuration register (MCFG) has an address of 0x5000 0020. The bit definition of this register is shown in Table 10–137.

Table 137. MII Mgmt Configuration register (MCFG - address 0x5000 0020) bit description

Bit	Symbol	Function	Reset value
0	SCAN INCREMENT	Set this bit to cause the MII Management hardware to perform read cycles across a range of PHYs. When set, the MII Management hardware will perform read cycles from address 1 through the value set in PHY ADDRESS[4:0]. Clear this bit to allow continuous reads of the same PHY.	0
1	SUPPRESS PREAMBLE	Set this bit to cause the MII Management hardware to perform read/write cycles without the 32-bit preamble field. Clear this bit to cause normal cycles to be performed. Some PHYs support suppressed preamble.	0
5:2	CLOCK SELECT	This field is used by the clock divide logic in creating the MII Management Clock (MDC) which IEEE 802.3u defines to be no faster than 2.5 MHz. Some PHYs support clock rates up to 12.5 MHz, however. The AHB bus clock (HCLK) is divided by the specified amount. Refer to Table 10–138 below for the definition of values for this field.	0
14:6	-	Unused	0x0
15	RESET MII MGMT	This bit resets the MII Management hardware.	0
31:16	-	Unused	0x0

Table 138. Clock select encoding

Clock Select	Bit 5	Bit 4	Bit 3	Bit 2	Maximum AHB clock supported
Host Clock divided by 4	0	0	0	x	10
Host Clock divided by 6	0	0	1	0	15
Host Clock divided by 8	0	0	1	1	20
Host Clock divided by 10	0	1	0	0	25
Host Clock divided by 14	0	1	0	1	35
Host Clock divided by 20	0	1	1	0	50
Host Clock divided by 28	0	1	1	1	70
Host Clock divided by 36	1	0	0	0	80[1]
Host Clock divided by 40	1	0	0	1	90[1]
Host Clock divided by 44	1	0	1	0	100[1]

© NXP B.V. 2010. All rights reserved.

Table 138. Clock select encoding

Clock Select	Bit 5	Bit 4	Bit 3	Bit 2	Maximum AHB clock supported
Host Clock divided by 48	1	0	1	1	120[1]
Host Clock divided by 52	1	1	0	0	130[1]
Host Clock divided by 56	1	1	0	1	140[1]
Host Clock divided by 60	1	1	1	0	150[1]
Host Clock divided by 64	1	1	1	1	160[1]

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

11.10 MII Mgmt Command Register (MCMD - 0x5000 0024)

The MII Mgmt Command register (MCMD) has an address of 0x5000 0024. The bit definition of this register is shown in Table 10–139.

Table 139. MII Mgmt Command register (MCMD - address 0x5000 0024) bit description

Bit	Symbol	Function	Reset value
0	READ	This bit causes the MII Management hardware to perform a single Read cycle. The Read data is returned in Register MRDD (MII Mgmt Read Data).	0
1	SCAN	This bit causes the MII Management hardware to perform Read cycles continuously. This is useful for monitoring Link Fail for example.	0
31:2	-	Unused	0x0

11.11 MII Mgmt Address Register (MADR - 0x5000 0028)

The MII Mgmt Address register (MADR) has an address of 0x5000 0028. The bit definition of this register is shown in Table 10–140.

Table 140. MII Mgmt Address register (MADR - address 0x5000 0028) bit description

Bit	Symbol	Function	Reset value
4:0	REGISTER ADDRESS	This field represents the 5-bit Register Address field of Mgmt cycles. Up to 32 registers can be accessed.	0x0
7:5	-	Unused	0x0
12:8	PHY ADDRESS	This field represents the 5-bit PHY Address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved).	0x0
31:13	-	Unused	0x0

11.12 MII Mgmt Write Data Register (MWTD - 0x5000 002C)

The MII Mgmt Write Data register (MWTD) is a write-only register with an address of 0x5000 002C. The bit definition of this register is shown in Table 10–141.

© NXP B.V. 2010. All rights reserved.

Table 141. MII Mgmt Write Data register (MWTD - address 0x5000 002C) bit description

Bit	Symbol	Function	Reset value
15:0	WRITE DATA	When written, an MII Mgmt write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from the MII Mgmt Address register (MADR).	0x0
31:16	-	Unused	0x0

11.13 MII Mgmt Read Data Register (MRDD - 0x5000 0030)

The MII Mgmt Read Data register (MRDD) is a read-only register with an address of 0x5000 0030. The bit definition of this register is shown in Table 10–142.

Table 142. MII Mgmt Read Data register (MRDD - address 0x5000 0030) bit description

Bit	Symbol	Function	Reset value
15:0	READ DATA	Following an MII Mgmt Read Cycle, the 16-bit data can be read from this location.	0x0
31:16	-	Unused	0x0

11.14 MII Mgmt Indicators Register (MIND - 0x5000 0034)

The MII Mgmt Indicators register (MIND) is a read-only register with an address of 0x5000 0034. The bit definition of this register is shown in Table 10–143.

Table 143. MII Mgmt Indicators register (MIND - address 0x5000 0034) bit description

Bit	Symbol	Function	Reset value
0	BUSY	When '1' is returned - indicates MII Mgmt is currently performing an MII Mgmt Read or Write cycle.	0
1	SCANNING	When '1' is returned - indicates a scan operation (continuous MII Mgmt Read cycles) is in progress.	0
2	NOT VALID	When '1' is returned - indicates MII Mgmt Read cycle has not completed and the Read Data is not yet valid.	0
3	MII Link Fail	When '1' is returned - indicates that an MII Mgmt link fail has occurred.	0
31:4	-	Unused	0x0

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

For PHY Write if scan is not used:

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

For PHY Read if scan is not used:

1. Write 1 to MCMD
2. Write PHY address and register address to MADR

© NXP B.V. 2010. All rights reserved.

3. Wait for busy bit to be cleared in MIND
4. Write 0 to MCMD
5. Read data from MRDD

11.15 Station Address 0 Register (SA0 - 0x5000 0040)

The Station Address 0 register (SA0) has an address of 0x5000 0040. The bit definition of this register is shown in Table 10–144.

Table 144. Station Address register (SA0 - address 0x5000 0040) bit description

Bit	Symbol	Function	Reset value
7:0	STATION ADDRESS, 2nd octet	This field holds the second octet of the station address.	0x0
15:8	STATION ADDRESS, 1st octet	This field holds the first octet of the station address.	0x0
31:16	-	Unused	0x0

The station address is used for perfect address filtering and for sending pause control frames. For the ordering of the octets in the packet please refer to Figure 10–17.

11.16 Station Address 1 Register (SA1 - 0x5000 0044)

The Station Address 1 register (SA1) has an address of 0x5000 0044. The bit definition of this register is shown in Table 10–145.

Table 145. Station Address register (SA1 - address 0x5000 0044) bit description

Bit	Symbol	Function	Reset value
7:0	STATION ADDRESS, 4th octet	This field holds the fourth octet of the station address.	0x0
15:8	STATION ADDRESS, 3rd octet	This field holds the third octet of the station address.	0x0
31:16	-	Unused	0x0

The station address is used for perfect address filtering and for sending pause control frames. For the ordering of the octets in the packet please refer to Figure 10–17.

11.17 Station Address 2 Register (SA2 - 0x5000 0048)

The Station Address 2 register (SA2) has an address of 0x5000 0048. The bit definition of this register is shown in Table 10–146.

Table 146. Station Address register (SA2 - address 0x5000 0048) bit description

Bit	Symbol	Function	Reset value
7:0	STATION ADDRESS, 6th octet	This field holds the sixth octet of the station address.	0x0
15:8	STATION ADDRESS, 5th octet	This field holds the fifth octet of the station address.	0x0
31:16	-	Unused	0x0

© NXP B.V. 2010. All rights reserved.

The station address is used for perfect address filtering and for sending pause control frames. For the ordering of the octets in the packet please refer to Figure 10–17.

12. Control register definitions

12.1 Command Register (Command - 0x5000 0100)

The Command register (Command) register has an address of 0x5000 0100. Its bit definition is shown in Table 10–147.

Table 147. Command register (Command - address 0x5000 0100) bit description

Bit	Symbol	Function	Reset value
0	RxEnable	Enable receive.	0
1	TxEnable	Enable transmit.	0
2	-	Unused	0x0
3	RegReset	When a '1' is written, all datapaths and the host registers are reset. The MAC needs to be reset separately.	0
4	TxReset	When a '1' is written, the transmit datapath is reset.	0
5	RxReset	When a '1' is written, the receive datapath is reset.	0
6	PassRuntFrame	When set to '1', passes runt frames smaller than 64 bytes to memory unless they have a CRC error. If '0' runt frames are filtered out.	0
7	PassRxFilter	When set to '1', disables receive filtering i.e. all frames received are written to memory.	0
8	TxFlowControl	Enable IEEE 802.3 / clause 31 flow control sending pause frames in full duplex and continuous preamble in half duplex.	0
9	RMII	When set to "1", RMII mode is selected. This bit must be set to one during Ethernet initialization. See Section 10–17.2.	0
10	FullDuplex	When set to "1", indicates full duplex operation.	0
31:11	-	Unused	0x0

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

12.2 Status Register (Status - 0x5000 0104)

The Status register (Status) is a read-only register with an address of 0x5000 0104. Its bit definition is shown in Table 10–148.

Table 148. Status register (Status - address 0x5000 0104) bit description

Bit	Symbol	Function	Reset value
0	RxStatus	If 1, the receive channel is active. If 0, the receive channel is inactive.	0
1	TxStatus	If 1, the transmit channel is active. If 0, the transmit channel is inactive.	0
31:2	-	Unused	0x0

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

© NXP B.V. 2010. All rights reserved.

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

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

12.3 Receive Descriptor Base Address Register (RxDescriptor - 0x5000 0108)

The Receive Descriptor base address register (RxDescriptor) has an address of 0x5000 0108. Its bit definition is shown in Table 10–149.

Table 149. Receive Descriptor Base Address register (RxDescriptor - address 0x5000 0108) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to ’00’	-
31:2	RxDescriptor	MSBs of receive descriptor base address.	0x0

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

12.4 Receive Status Base Address Register (RxStatus - 0x5000 010C)

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

Table 150. receive Status Base Address register (RxStatus - address 0x5000 010C) bit description

Bit	Symbol	Function	Reset value
2:0	-	Fixed to ’000’	-
31:3	RxStatus	MSBs of receive status base address.	0x0

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

12.5 Receive Number of Descriptors Register (RxDescriptor - 0x5000 0110)

The Receive Number of Descriptors register (RxDescriptorNumber) has an address of 0x5000 0110. Its bit definition is shown in Table 10–151.

© NXP B.V. 2010. All rights reserved.

Table 151. Receive Number of Descriptors register (RxDescriptor - address 0x5000 0110) bit description

Bit	Symbol	Function	Reset value
15:0	RxDescriptorNumber	Number of descriptors in the descriptor array for which RxDescriptor is the base address. The number of descriptors is minus one encoded.	0x0
31:16	-	Unused	0x0

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

12.6 Receive Produce Index Register (RxProduceIndex - 0x5000 0114)

The Receive Produce Index register (RxProduceIndex) is a read-only register with an address of 0x5000 0114. Its bit definition is shown in Table 10–152.

Table 152. Receive Produce Index register (RxProduceIndex - address 0x5000 0114) bit description

Bit	Symbol	Function	Reset value
15:0	RxProduceIndex	Index of the descriptor that is going to be filled next by the receive datapath.	0x0
31:16	-	Unused	0x0

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

12.7 Receive Consume Index Register (RxConsumeIndex - 0x5000 0118)

The Receive consume index register (RxConsumeIndex) has an address of 0x5000 0118. Its bit definition is shown in Table 10–153.

Table 153. Receive Consume Index register (RxConsumeIndex - address 0x5000 0118) bit description

Bit	Symbol	Function	Reset value
15:0	RxConsumeIndex	Index of the descriptor that is going to be processed next by the receive	
31:16	-	Unused	0x0

The receive consume register defines the descriptor that is going to be processed next by the software receive driver. The receive array is empty as long as RxProduceIndex equals RxConsumeIndex. As soon as the array is not empty, software can process the frame pointed to by RxConsumeIndex. After a frame has been processed by software, software should increment the RxConsumeIndex. The value must be wrapped to 0 once the value

© NXP B.V. 2010. All rights reserved.

of RxDescriptorNumber has been reached. If the RxProduceIndex equals RxConsumeIndex - 1, the array is full and any further frames being received will cause a buffer overrun error.

12.8 Transmit Descriptor Base Address Register (TxDescriptor - 0x5000 011C)

The Transmit Descriptor base address register (TxDescriptor) has an address of 0x5000 011C. Its bit definition is shown in Table 10–154.

Table 154. Transmit Descriptor Base Address register (TxDescriptor - address 0x5000 011C) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to ’00’	-
31:2	TxDescriptor	MSBs of transmit descriptor base address.	0x0

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

12.9 Transmit Status Base Address Register (TxStatus - 0x5000 0120)

The Transmit Status base address register (TxStatus) has an address of 0x5000 0120. Its bit definition is shown in Table 10–155.

Table 155. Transmit Status Base Address register (TxStatus - address 0x5000 0120) bit description

Bit	Symbol	Function	Reset value
1:0	-	Fixed to ’00’	-
31:2	TxStatus	MSBs of transmit status base address.	0x0

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

12.10 Transmit Number of Descriptors Register (TxDescriptorNumber - 0x5000 0124)

The Transmit Number of Descriptors register (TxDescriptorNumber) has an address of 0x5000 0124. Its bit definition is shown in Table 10–156.

Table 156. Transmit Number of Descriptors register (TxDescriptorNumber - address 0x5000 0124) bit description

Bit	Symbol	Function	Reset value
15:0	TxDescriptorNumber	Number of descriptors in the descriptor array for which TxDescriptor is the base address. The register is minus one encoded.	
31:16	-	Unused	0x0

© NXP B.V. 2010. All rights reserved.

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

12.11 Transmit Produce Index Register (TxProduceIndex - 0x5000 0128)

The Transmit Produce Index register (TxProduceIndex) has an address of 0x5000 0128. Its bit definition is shown in Table 10–157.

Table 157. Transmit Produce Index register (TxProduceIndex - address 0x5000 0128) bit description

Bit	Symbol	Function	Reset value
15:0	TxProduceIndex	Index of the descriptor that is going to be filled next by the transmit software driver.	0x0
31:16	-	Unused	0x0

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

12.12 Transmit Consume Index Register (TxConsumeIndex - 0x5000 012C)

The Transmit Consume Index register (TxConsumeIndex) is a read-only register with an address of 0x5000 012C. Its bit definition is shown in Table 10–158.

Table 158. Transmit Consume Index register (TxConsumeIndex - address 0x5000 012C) bit description

Bit	Symbol	Function	Reset value
15:0	TxConsumeIndex	Index of the descriptor that is going to be transmitted next by the transmit datapath.	0x0
31:16	-	Unused	0x0

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

12.13 Transmit Status Vector 0 Register (TSV0 - 0x5000 0158)

The Transmit Status Vector 0 register (TSV0) is a read-only register with an address of 0x5000 0158. The transmit status vector registers store the most recent transmit status returned by the MAC. Since the status vector consists of more than 4 bytes, status is

© NXP B.V. 2010. All rights reserved.

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

Table 10–159 lists the bit definitions of the TSV0 register.

Table 159. Transmit Status Vector 0 register (TSV0 - address 0x5000 0158) bit description

Bit	Symbol	Function	Reset value
0	CRC error	The attached CRC in the packet did not match the internally generated CRC.	0
1	Length check error	Indicates the frame length field does not match the actual number of data items and is not a type field.	0
2	Length out of range[1]	Indicates that frame type/length field was larger than 1500 bytes.	0
3	Done	Transmission of packet was completed.	0
4	Multicast	Packet's destination was a multicast address.	0
5	Broadcast	Packet's destination was a broadcast address.	0
6	Packet Defer	Packet was deferred for at least one attempt, but less than an excessive defer.	0
7	Excessive Defer	Packet was deferred in excess of 6071 nibble times in 100 Mbps or 24287 bit times in 10 Mbps mode.	0
8	Excessive Collision	Packet was aborted due to exceeding of maximum allowed number of collisions.	0
9	Late Collision	Collision occurred beyond collision window, 512 bit times.	0
10	Giant	Byte count in frame was greater than can be represented in the transmit byte count field in TSV1.	0
11	Underrun	Host side caused buffer underrun.	0
27:12	Total bytes	The total number of bytes transferred including collided attempts.	0x0
28	Control frame	The frame was a control frame.	0
29	Pause	The frame was a control frame with a valid PAUSE opcode.	0
30	Backpressure	Carrier-sense method backpressure was previously applied.	0
31	VLAN	Frame's length/type field contained 0x8100 which is the VLAN protocol identifier.	0

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

12.14 Transmit Status Vector 1 Register (TSV1 - 0x5000 015C)

The Transmit Status Vector 1 register (TSV1) is a read-only register with an address of 0x5000 015C. The transmit status vector registers store the most recent transmit status returned by the MAC. Since the status vector consists of more than 4 bytes, status is distributed over two registers TSV0 and TSV1. These registers are provided for debug

© NXP B.V. 2010. All rights reserved.

purposes, because the communication between driver software and the Ethernet block takes place primarily through the frame descriptors. The status register contents are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.Table 10–160 lists the bit definitions of the TSV1 register.

Table 160. Transmit Status Vector 1 register (TSV1 - address 0x5000 015C) bit description

Bit	Symbol	Function	Reset value
15:0	Transmit byte count	The total number of bytes in the frame, not counting the collided bytes.	0x0
19:16	Transmit collision count	Number of collisions the current packet incurred during transmission attempts. The maximum number of collisions (16) cannot be represented.	0x0
31:20	-	Unused	0x0

12.15 Receive Status Vector Register (RSV - 0x5000 0160)

The Receive status vector register (RSV) is a read-only register with an address of 0x5000 0160. The receive status vector register stores the most recent receive status returned by the MAC. This register is provided for debug purposes, because the communication between driver software and the Ethernet block takes place primarily through the frame descriptors. The status register contents are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

Table 10–161 lists the bit definitions of the RSV register.

Table 161. Receive Status Vector register (RSV - address 0x5000 0160) bit description

Bit	Symbol	Function	Reset value
15:0	Received byte count	Indicates length of received frame.	0x0
16	Packet previously ignored	Indicates that a packet was dropped.	0
17	RXDV event previously seen	Indicates that the last receive event seen was not long enough to be a valid packet.	0
18	Carrier event previously seen	Indicates that at some time since the last receive statistics, a carrier event was detected.	0
19	Receive code violation	Indicates that received PHY data does not represent a valid receive code.	0
20	CRC error	The attached CRC in the packet did not match the internally generated CRC.	0
21	Length check error	Indicates the frame length field does not match the actual number of data items and is not a type field.	0
22	Length out of range[1]	Indicates that frame type/length field was larger than 1518 bytes.	0
23	Receive OK	The packet had valid CRC and no symbol errors.	0
24	Multicast	The packet destination was a multicast address.	0
25	Broadcast	The packet destination was a broadcast address.	0

© NXP B.V. 2010. All rights reserved.

Table 161. Receive Status Vector register (RSV - address 0x5000 0160) bit description

Bit	Symbol	Function	Reset value
26	Dribble Nibble	Indicates that after the end of packet another 1-7 bits were received. A single nibble, called dribble nibble, is formed but not sent out.	0
27	Control frame	The frame was a control frame.	0
28	PAUSE	The frame was a control frame with a valid PAUSE opcode.	0
29	Unsupported Opcode	The current frame was recognized as a Control Frame but contains an unknown opcode.	0
30	VLAN	Frame's length/type field contained 0x8100 which is the VLAN protocol identifier.	0
31	-	Unused	0x0

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

12.16 Flow Control Counter Register (FlowControlCounter - 0x5000 0170)

The Flow Control Counter register (FlowControlCounter) has an address of 0x5000 0170. Table 10–162 lists the bit definitions of the register.

Table 162. Flow Control Counter register (FlowControlCounter - address 0x5000 0170) bit description

Bit	Symbol	Function	Reset value
15:0	MirrorCounter	In full duplex mode the MirrorCounter specifies the number of cycles before re-issuing the Pause control frame.	0x0
31:16	PauseTimer	In full-duplex mode the PauseTimer specifies the value that is inserted into the pause timer field of a pause flow control frame. In half duplex mode the PauseTimer specifies the number of backpressure cycles.	0x0

12.17 Flow Control Status Register (FlowControlStatus - 0x5000 0174)

The Flow Control Status register (FlowControlStatus) is a read-only register with an address of 0x5000 8174. Table 10–163 lists the bit definitions of the register.

Table 163. Flow Control Status register (FlowControlStatus - address 0x5000 8174) bit description

Bit	Symbol	Function	Reset value
15:0	MirrorCounterCurrent	In full duplex mode this register represents the current value of the datapath's mirror counter which counts up to the value specified by the MirrorCounter field in the FlowControlCounter register. In half duplex mode the register counts until it reaches the value of the PauseTimer bits in the FlowControlCounter register.	0x0
31:16	-	Unused	0x0

© NXP B.V. 2010. All rights reserved.

13. Receive filter register definitions

13.1 Receive Filter Control Register (RxFilterCtrl - 0x5000 0200)

The Receive Filter Control register (RxFilterCtrl) has an address of 0x5000 0200. Table 10–164 lists the definition of the individual bits in the register.

Table 164. Receive Filter Control register (RxFilterCtrl - address 0x5000 0200) bit description

Bit	Symbol	Function	Reset value
0	AcceptUnicastEn	When set to ’1’, all unicast frames are accepted.	0
1	AcceptBroadcastEn	When set to ’1’, all broadcast frames are accepted.	0
2	AcceptMulticastEn	When set to ’1’, all multicast frames are accepted.	0
3	AcceptUnicastHashEn	When set to ’1’, unicast frames that pass the imperfect hash filter are accepted.	0
4	AcceptMulticastHashEn	When set to ’1’, multicast frames that pass the imperfect hash filter are accepted.	0
5	AcceptPerfectEn	When set to ’1’, the frames with a destination address identical to the station address are accepted.	0
11:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
12	MagicPacketEnWoL	When set to ’1’, the result of the magic packet filter will generate a WoL interrupt when there is a match.	0
13	RxFilterEnWoL	When set to ’1’, the result of the perfect address matching filter and the imperfect hash filter will generate a WoL interrupt when there is a match.	0
31:14	-	Unused	0x0

13.2 Receive Filter WoL Status Register (RxFilterWoLStatus - 0x5000 0204)

The Receive Filter Wake-up on LAN Status register (RxFilterWoLStatus) is a read-only register with an address of 0x5000 0204.

Table 10–165 lists the definition of the individual bits in the register.

Table 165. Receive Filter WoL Status register (RxFilterWoLStatus - address 0x5000 0204) bit description

Bit	Symbol	Function	Reset value
0	AcceptUnicastWoL	When the value is ’1’, a unicast frames caused WoL.	0
1	AcceptBroadcastWoL	When the value is ’1’, a broadcast frame caused WoL.	0
2	AcceptMulticastWoL	When the value is ’1’, a multicast frame caused WoL.	0
3	AcceptUnicastHashWoL	When the value is ’1’, a unicast frame that passes the imperfect hash filter caused WoL.	0
4	AcceptMulticastHashWoL	When the value is ’1’, a multicast frame that passes the imperfect hash filter caused WoL.	0
5	AcceptPerfectWoL	When the value is ’1’, the perfect address matching filter caused WoL.	0

© NXP B.V. 2010. All rights reserved.

Table 165. Receive Filter WoL Status register (RxFilterWoLStatus - address 0x5000 0204) bit description

Bit	Symbol	Function	Reset value
6	-	Unused	0x0
7	RxFilterWoL	When the value is '1', the receive filter caused WoL.	0
8	MagicPacketWoL	When the value is '1', the magic packet filter caused WoL.	0
31:9	-	Unused	0x0

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

13.3 Receive Filter WoL Clear Register (RxFilterWoLClear - 0x5000 0208)

The Receive Filter Wake-up on LAN Clear register (RxFilterWoLClear) is a write-only register with an address of 0x5000 0208.

Table 10–166 lists the definition of the individual bits in the register.

Table 166. Receive Filter WoL Clear register (RxFilterWoLClear - address 0x5000 0208) bit description

Bit	Symbol	Function	Reset value
0	AcceptUnicastWoLClr	When a '1' is written to one of these bits (0 to 5), the corresponding status bit in the RxFilterWoLStatus register is cleared.	0
1	AcceptBroadcastWoLClr		0
2	AcceptMulticastWoLClr		0
3	AcceptUnicastHashWoLClr		0
4	AcceptMulticastHashWoLClr		0
5	AcceptPerfectWoLClr		0
6	-	Unused	0x0
7	RxFilterWoLClr	When a '1' is written to one of these bits (7 and/or 8), the corresponding status bit in the RxFilterWoLStatus register is cleared.	0
8	MagicPacketWoLClr		0
31:9	-	Unused	0x0

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

13.4 Hash Filter Table LSBs Register (HashFilterL - 0x5000 0210)

The Hash Filter table LSBs register (HashFilterL) has an address of 0x5000 0210. Table 10–167 lists the bit definitions of the register. Details of Hash filter table use can be found in Section 10–17.10 "Receive filtering" on page 196.

Table 167. Hash Filter Table LSBs register (HashFilterL - address 0x5000 0210) bit description

Bit	Symbol	Function	Reset value
31:0	HashFilterL	Bits 31:0 of the imperfect filter hash table for receive filtering.	0x0

© NXP B.V. 2010. All rights reserved.

13.5 Hash Filter Table MSBs Register (HashFilterH - 0x5000 0214)

The Hash Filter table MSBs register (HashFilterH) has an address of 0x5000 0214. Table 10–168 lists the bit definitions of the register. Details of Hash filter table use can be found in Section 10–17.10 “Receive filtering” on page 196.

Table 168. Hash Filter MSBs register (HashFilterH - address 0x5000 0214) bit description

Bit	Symbol	Function	Reset value
31:0	HashFilterH	Bits 63:32 of the imperfect filter hash table for receive filtering.	0x0

14. Module control register definitions

14.1 Interrupt Status Register (IntStatus - 0x5000 0FE0)

The Interrupt Status register (IntStatus) is a read-only register with an address of 0x5000 0FE0. The interrupt status register bit definition is shown in Table 10–169. Note that all bits are flip-flops with an asynchronous set in order to be able to generate interrupts if there are wake-up events while clocks are disabled.

Table 169. Interrupt Status register (IntStatus - address 0x5000 0FE0) bit description

Bit	Symbol	Function	Reset value
0	RxOverrunInt	Interrupt set on a fatal overrun error in the receive queue. The fatal interrupt should be resolved by a Rx soft-reset. The bit is not set when there is a nonfatal overrun error.	0
1	RxErrorInt	Interrupt trigger on receive errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError or NoDescriptor or Overrun.	0
2	RxFinishedInt	Interrupt triggered when all receive descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
3	RxDoneInt	Interrupt triggered when a receive descriptor has been processed while the Interrupt bit in the Control field of the descriptor was set.	0
4	TxUnderrunInt	Interrupt set on a fatal underrun error in the transmit queue. The fatal interrupt should be resolved by a Tx soft-reset. The bit is not set when there is a nonfatal underrun error.	0
5	TxErrorInt	Interrupt trigger on transmit errors: LateCollision, ExcessiveCollision and ExcessiveDefer, NoDescriptor or Underrun.	0
6	TxFinishedInt	Interrupt triggered when all transmit descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
7	TxDoneInt	Interrupt triggered when a descriptor has been transmitted while the Interrupt bit in the Control field of the descriptor was set.	0
11:8	-	Unused	0x0
12	SoftInt	Interrupt triggered by software writing a 1 to the SoftintSet bit in the IntSet register.	0
13	WakeupInt	Interrupt triggered by a Wake-up event detected by the receive filter.	0
31:14	-	Unused	0x0

© NXP B.V. 2010. All rights reserved.

The interrupt status register is read-only. Setting can be done via the IntSet register. Reset can be accomplished via the IntClear register.

14.2 Interrupt Enable Register (IntEnable - 0x5000 0FE4)

The Interrupt Enable register (IntEnable) has an address of 0x5000 0FE4. The interrupt enable register bit definition is shown in Table 10–170.

Table 170. Interrupt Enable register (intEnable - address 0x5000 0FE4) bit description

Bit	Symbol	Function	Reset value
0	RxOverrunIntEn	Enable for interrupt trigger on receive buffer overrun or descriptor underrun situations.	0
1	RxErrorIntEn	Enable for interrupt trigger on receive errors.	0
2	RxFinishedIntEn	Enable for interrupt triggered when all receive descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
3	RxDoneIntEn	Enable for interrupt triggered when a receive descriptor has been processed while the Interrupt bit in the Control field of the descriptor was set.	0
4	TxUnderrunIntEn	Enable for interrupt trigger on transmit buffer or descriptor underrun situations.	0
5	TxErrorIntEn	Enable for interrupt trigger on transmit errors.	0
6	TxFinishedIntEn	Enable for interrupt triggered when all transmit descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex.	0
7	TxDoneIntEn	Enable for interrupt triggered when a descriptor has been transmitted while the Interrupt bit in the Control field of the descriptor was set.	0
11:8	-	Unused	0x0
12	SoftIntEn	Enable for interrupt triggered by the SoftInt bit in the IntStatus register, caused by software writing a 1 to the SoftIntSet bit in the IntSet register.	0
13	WakeupIntEn	Enable for interrupt triggered by a Wake-up event detected by the receive filter.	0
31:14	-	Unused	0x0

14.3 Interrupt Clear Register (IntClear - 0x5000 0FE8)

The Interrupt Clear register (IntClear) is a write-only register with an address of 0x5000 0FE8. The interrupt clear register bit definition is shown in Table 10–171.

© NXP B.V. 2010. All rights reserved.

Table 171. Interrupt Clear register (IntClear - address 0x5000 0FE8) bit description

Bit	Symbol	Function	Reset value
0	RxOverrunIntClr	Writing a '1' to one of these bits clears (0 to 7) the corresponding status bit in interrupt status register IntStatus.	0
1	RxErrorIntClr		0
2	RxFinishedIntClr		0
3	RxDoneIntClr		0
4	TxUnderrunIntClr		0
5	TxErrorIntClr		0
6	TxFinishedIntClr		0
7	TxDoneIntClr		0
11:8	-	Unused	0x0
12	SoftIntClr	Writing a '1' to one of these bits (12 and/or 13) clears the corresponding status bit in interrupt status register IntStatus.	0
13	WakeupIntClr		0
31:14	-	Unused	0x0

The interrupt clear register is write-only. Writing a 1 to a bit of the IntClear register clears the corresponding bit in the status register. Writing a 0 will not affect the interrupt status.

14.4 Interrupt Set Register (IntSet - 0x5000 0FEC)

The Interrupt Set register (IntSet) is a write-only register with an address of 0x5000 0FEC. The interrupt set register bit definition is shown in Table 10–172.

Table 172. Interrupt Set register (IntSet - address 0x5000 0FEC) bit description

Bit	Symbol	Function	Reset value
0	RxOverrunIntSet	Writing a '1' to one of these bits (0 to 7) sets the corresponding status bit in interrupt status register IntStatus.	0
1	RxErrorIntSet		0
2	RxFinishedIntSet		0
3	RxDoneIntSet		0
4	TxUnderrunIntSet		0
5	TxErrorIntSet		0
6	TxFinishedIntSet		0
7	TxDoneIntSet		0
11:8	-	Unused	0x0
12	SoftIntSet	Writing a '1' to one of these bits (12 and/or 13) sets the corresponding status bit in interrupt status register IntStatus.	0
13	WakeupIntSet		0
31:14	-	Unused	0x0

The interrupt set register is write-only. Writing a 1 to a bit of the IntSet register sets the corresponding bit in the status register. Writing a 0 will not affect the interrupt status.

© NXP B.V. 2010. All rights reserved.

14.5 Power-Down Register (PowerDown - 0x5000 0FF4)

The Power-Down register (PowerDown) is used to block all AHB accesses except accesses to the Power-Down register. The register has an address of 0x5000 0FF4. The bit definition of the register is listed in Table 10–173.

Table 173. Power-Down register (PowerDown - address 0x5000 0FF4) bit description

Bit	Symbol	Function	Reset value
30:0	-	Unused	0x0
31	PowerDownMACAHB	If true, all AHB accesses will return a read/write error, except accesses to the Power-Down register.	0

Setting the bit will return an error on all read and write accesses on the MACAHB interface except for accesses to the Power-Down register.

© NXP B.V. 2010. All rights reserved.

15. Descriptor and status formats

This section defines the descriptor format for the transmit and receive scatter/gather DMA engines. Each Ethernet frame can consist of one or more fragments. Each fragment corresponds to a single descriptor. The DMA managers in the Ethernet block scatter (for receive) and gather (for transmit) multiple fragments for a single Ethernet frame.

15.1 Receive descriptors and statuses

Figure 10–18 depicts the layout of the receive descriptors in memory.

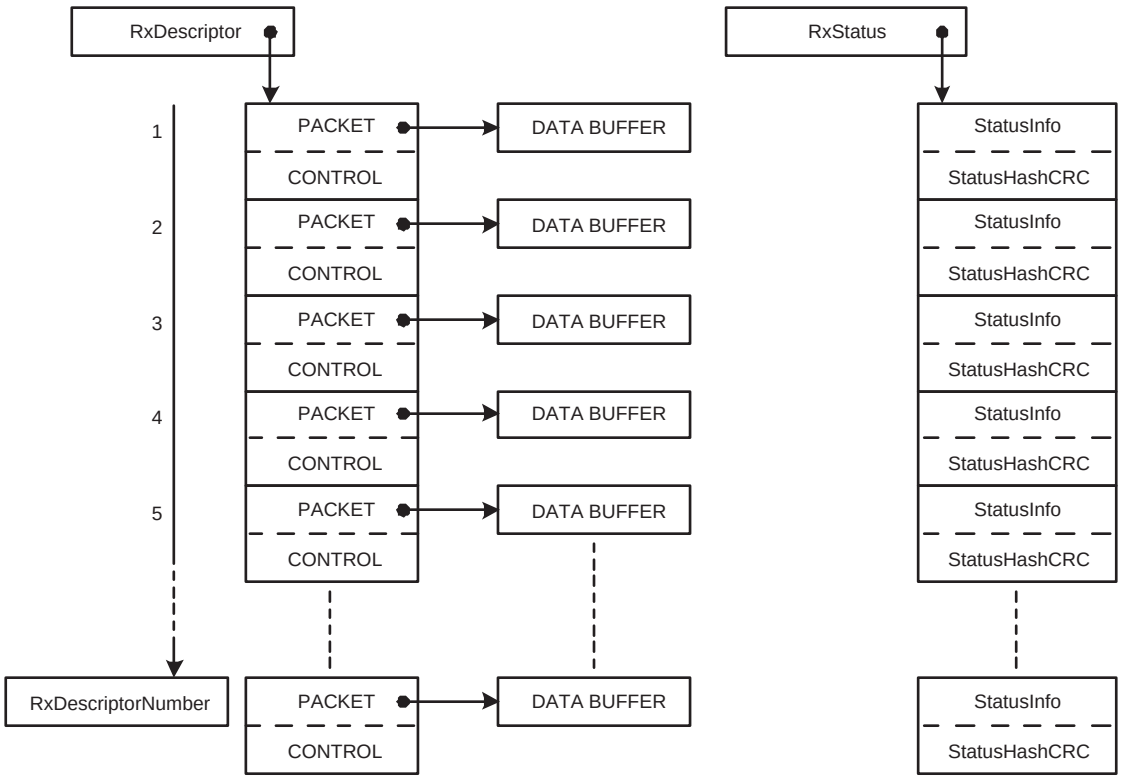


Fig 18. Receive descriptor memory layout

Receive descriptors are stored in an array in memory. The base address of the array is stored in the RxDescriptor register, and should be aligned on a 4 byte address boundary. The number of descriptors in the array is stored in the RxDescriptorNumber register using a minus one encoding style e.g. if the array has 8 elements the register value should be 7. Parallel to the descriptors there is an array of statuses. For each element of the descriptor array there is an associated status field in the status array. The base address of the status array is stored in the RxStatus register, and must be aligned on an 8 byte address boundary. During operation (when the receive data path is enabled) the RxDescriptor, RxStatus and RxDescriptorNumber registers should not be modified.

Two registers, RxConsumeIndex and RxProduceIndex, define the descriptor locations that will be used next by hardware and software. Both registers act as counters starting at 0 and wrapping when they reach the value of RxDescriptorNumber. The RxProduceIndex contains the index of the descriptor that is going to be filled with the next frame being

© NXP B.V. 2010. All rights reserved.

received. The RxConsumeIndex is programmed by software and is the index of the next descriptor that the software receive driver is going to process. When RxProduceIndex == RxConsumeIndex, the receive buffer is empty. When RxProduceIndex == RxConsumeIndex -1 (taking wraparound into account), the receive buffer is full and newly received data would generate an overflow unless the software driver frees up one or more descriptors.

Each receive descriptor takes two word locations (8 bytes) in memory. Likewise each status field takes two words (8 bytes) in memory. Each receive descriptor consists of a pointer to the data buffer for storing receive data (Packet) and a control word (Control). The Packet field has a zero address offset, the control field has a 4 byte address offset with respect to the descriptor address as defined in Table 10–174.

Table 174. Receive Descriptor Fields

Symbol	Address offset	Bytes	Description
Packet	0x0	4	Base address of the data buffer for storing receive data.
Control	0x4	4	Control information, see Table 10–175.

The data buffer pointer (Packet) is a 32-bit, byte aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in Table 10–175.

Table 175. Receive Descriptor Control Word

Bit	Symbol	Description
10:0	Size	Size in bytes of the data buffer. This is the size of the buffer reserved by the device driver for a frame or frame fragment i.e. the byte size of the buffer pointed to by the Packet field. The size is -1 encoded e.g. if the buffer is 8 bytes the size field should be equal to 7.
30:11	-	Unused
31	Interrupt	If true generate an RxDone interrupt when the data in this frame or frame fragment and the associated status information has been committed to memory.

Table 10–176 lists the fields in the receive status elements from the status array.

Table 176. Receive Status Fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Receive status return flags, see Table 10–178.
StatusHashCRC	0x4	4	The concatenation of the destination address hash CRC and the source address hash CRC.

Each receive status consists of two words. The StatusHashCRC word contains a concatenation of the two 9-bit hash CRCs calculated from the destination and source addresses contained in the received frame. After detecting the destination and source addresses, StatusHashCRC is calculated once, then held for every fragment of the same frame.

The concatenation of the two CRCs is shown in Table 10–177:

© NXP B.V. 2010. All rights reserved.

Table 177. Receive Status HashCRC Word

Bit	Symbol	Description
8:0	SAHashCRC	Hash CRC calculated from the source address.
15:9	-	Unused
24:16	DAHashCRC	Hash CRC calculated from the destination address.
31:25	-	Unused

The StatusInfo word contains flags returned by the MAC and flags generated by the receive data path reflecting the status of the reception. Table 10–178 lists the bit definitions in the StatusInfo word.

Table 178. Receive status information word

Bit	Symbol	Description
10:0	RxSize	The size in bytes of the actual data transferred into one fragment buffer. In other words, this is the size of the frame or fragment as actually written by the DMA manager for one descriptor. This may be different from the Size bits of the Control field in the descriptor that indicate the size of the buffer allocated by the device driver. Size is -1 encoded e.g. if the buffer has 8 bytes the RxSize value will be 7.
17:11	-	Unused
18	ControlFrame	Indicates this is a control frame for flow control, either a pause frame or a frame with an unsupported opcode.
19	VLAN	Indicates a VLAN frame.
20	FailFilter	Indicates this frame has failed the Rx filter. These frames will not normally pass to memory. But due to the limitation of the size of the buffer, part of this frame may already be passed to memory. Once the frame is found to have failed the Rx filter, the remainder of the frame will be discarded without being passed to the memory. However, if the PassRxFilter bit in the Command register is set, the whole frame will be passed to memory.
21	Multicast	Set when a multicast frame is received.
22	Broadcast	Set when a broadcast frame is received.
23	CRCError	The received frame had a CRC error.
24	SymbolError	The PHY reports a bit error over the PHY interface during reception.
25	LengthError	The frame length field value in the frame specifies a valid length, but does not match the actual data length.
26	RangeError[1]	The received packet exceeds the maximum packet size.
27	AlignmentError	An alignment error is flagged when dribble bits are detected and also a CRC error is detected. This is in accordance with IEEE std. 802.3/clause 4.3.2.
28	Overrun	Receive overrun. The adapter can not accept the data stream.
29	NoDescriptor	No new Rx descriptor is available and the frame is too long for the buffer size in the current receive descriptor.
30	LastFlag	When set to 1, indicates this descriptor is for the last fragment of a frame. If the frame consists of a single fragment, this bit is also set to 1.
31	Error	An error occurred during reception of this frame. This is a logical OR of AlignmentError, RangeError, LengthError, SymbolError, CRCError, and Overrun.

© NXP B.V. 2010. All rights reserved.

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

For multi-fragment frames, the value of the AlignmentError, RangeError, LengthError, SymbolError and CRCError bits in all but the last fragment in the frame will be 0; likewise the value of the FailFilter, Multicast, Broadcast, VLAN and ControlFrame bits is undefined. The status of the last fragment in the frame will copy the value for these bits from the MAC. All fragment statuses will have valid LastFrag, RxSize, Error, Overrun and NoDescriptor bits.

15.2 Transmit descriptors and statuses

Figure 10–19 depicts the layout of the transmit descriptors in memory.

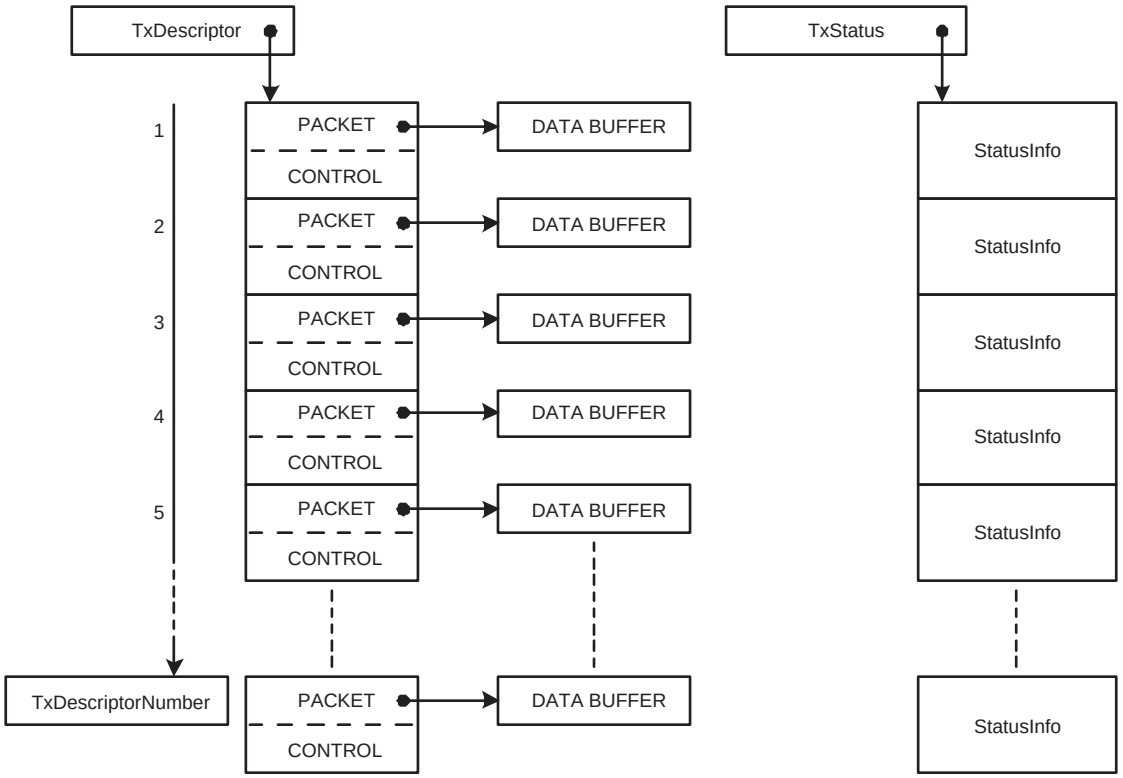


Fig 19. Transmit descriptor memory layout

Transmit descriptors are stored in an array in memory. The lowest address of the transmit descriptor array is stored in the TxDescriptor register, and must be aligned on a 4 byte address boundary. The number of descriptors in the array is stored in the TxDescriptorNumber register using a minus one encoding style i.e. if the array has 8 elements the register value should be 7. Parallel to the descriptors there is an array of statuses. For each element of the descriptor array there is an associated status field in the status array. The base address of the status array is stored in the TxStatus register, and must be aligned on a 4 byte address boundary. During operation (when the transmit data path is enabled) the TxDescriptor, TxStatus, and TxDescriptorNumber registers should not be modified.

© NXP B.V. 2010. All rights reserved.

Two registers, TxConsumeIndex and TxProduceIndex, define the descriptor locations that will be used next by hardware and software. Both register act as counters starting at 0 and wrapping when they reach the value of TxDescriptorNumber. The TxProduceIndex contains the index of the next descriptor that is going to be filled by the software driver. The TxConsumeIndex contains the index of the next descriptor going to be transmitted by the hardware. When TxProduceIndex == TxConsumeIndex, the transmit buffer is empty. When TxProduceIndex == TxConsumeIndex -1 (taking wraparound into account), the transmit buffer is full and the software driver cannot add new descriptors until the hardware has transmitted one or more frames to free up descriptors.

Each transmit descriptor takes two word locations (8 bytes) in memory. Likewise each status field takes one word (4 bytes) in memory. Each transmit descriptor consists of a pointer to the data buffer containing transmit data (Packet) and a control word (Control). The Packet field has a zero address offset, whereas the control field has a 4 byte address offset, see Table 10–179.

Table 179. Transmit descriptor fields

Symbol	Address offset	Bytes	Description
Packet	0x0	4	Base address of the data buffer containing transmit data.
Control	0x4	4	Control information, see Table 10–180.

The data buffer pointer (Packet) is a 32-bit, byte aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in Table 10–180.

Table 180. Transmit descriptor control word

Bit	Symbol	Description
10:0	Size	Size in bytes of the data buffer. This is the size of the frame or fragment as it needs to be fetched by the DMA manager. In most cases it will be equal to the byte size of the data buffer pointed to by the Packet field of the descriptor. Size is -1 encoded e.g. a buffer of 8 bytes is encoded as the Size value 7.
25:11	-	Unused
26	Override	Per frame override. If true, bits 30:27 will override the defaults from the MAC internal registers. If false, bits 30:27 will be ignored and the default values from the MAC will be used.
27	Huge	If true, enables huge frame, allowing unlimited frame sizes. When false, prevents transmission of more than the maximum frame length (MAXF[15:0]).
28	Pad	If true, pad short frames to 64 bytes.
29	CRC	If true, append a hardware CRC to the frame.
30	Last	If true, indicates that this is the descriptor for the last fragment in the transmit frame. If false, the fragment from the next descriptor should be appended.
31	Interrupt	If true, a TxDone interrupt will be generated when the data in this frame or frame fragment has been sent and the associated status information has been committed to memory.

Table 10–181 shows the one field transmit status.

Table 181. Transmit status fields

Symbol	Address offset	Bytes	Description
StatusInfo	0x0	4	Transmit status return flags, see Table 10–182.

© NXP B.V. 2010. All rights reserved.

The transmit status consists of one word which is the StatusInfo word. It contains flags returned by the MAC and flags generated by the transmit data path reflecting the status of the transmission. Table 10–182 lists the bit definitions in the StatusInfo word.

Table 182. Transmit status information word

Bit	Symbol	Description
20:0	-	Unused
24:21	CollisionCount	The number of collisions this packet incurred, up to the Retransmission Maximum.
25	Defer	This packet incurred deferral, because the medium was occupied. This is not an error unless excessive deferral occurs.
26	ExcessiveDefer	This packet incurred deferral beyond the maximum deferral limit and was aborted.
27	ExcessiveCollision	Indicates this packet exceeded the maximum collision limit and was aborted.
28	LateCollision	An Out of window Collision was seen, causing packet abort.
29	Underrun	A Tx underrun occurred due to the adapter not producing transmit data.
30	NoDescriptor	The transmit stream was interrupted because a descriptor was not available.
31	Error	An error occurred during transmission. This is a logical OR of Underrun, LateCollision, ExcessiveCollision, and ExcessiveDefer.

For multi-fragment frames, the value of the LateCollision, ExcessiveCollision, ExcessiveDefer, Defer and CollissionCount bits in all but the last fragment in the frame will be 0. The status of the last fragment in the frame will copy the value for these bits from the MAC. All fragment statuses will have valid Error, NoDescriptor and Underrun bits.

16. Ethernet block functional description

This section defines the functions of the DMA capable 10/100 Ethernet MAC. After introducing the DMA concepts of the Ethernet block, and a description of the basic transmit and receive functions, this section elaborates on advanced features such as flow control, receive filtering, etc.

16.1 Overview

The Ethernet block can transmit and receive Ethernet packets from an off-chip Ethernet PHY connected through the RMII interface.

Typically during system start-up, the Ethernet block will be initialized. Software initialization of the Ethernet block should include initialization of the descriptor and status arrays as well as the receiver fragment buffers.

Remark: when initializing the Ethernet block, it is important to first configure the PHY and insure that reference clocks (ENET_REF_CLK signal in RMII mode, or both ENET_RX_CLK and ENET_TX_CLK signals in MII mode) are present at the external pins and connected to the EMAC module (selecting the appropriate pins using the PINSEL registers) prior to continuing with Ethernet configuration. Otherwise the CPU can become locked and no further functionality will be possible. This will cause JTAG lose communication with the target, if debug mode is being used.

© NXP B.V. 2010. All rights reserved.

To transmit a packet the software driver has to set up the appropriate Control registers and a descriptor to point to the packet data buffer before transferring the packet to hardware by incrementing the TxProduceIndex register. After transmission, hardware will increment TxConsumeIndex and optionally generate an interrupt.

The hardware will receive packets from the PHY and apply filtering as configured by the software driver. While receiving a packet the hardware will read a descriptor from memory to find the location of the associated receiver data buffer. Receive data is written in the data buffer and receive status is returned in the receive descriptor status word. Optionally an interrupt can be generated to notify software that a packet has been received. Note that the DMA manager will prefetch and buffer up to three descriptors.

16.2 AHB interface

The registers of the Ethernet block connect to an AHB slave interface to allow access to the registers from the CPU.

The AHB interface has a 32-bit data path, which supports only word accesses and has an address aperture of 4 kB. Table 10–127 lists the registers of the Ethernet block.

All AHB write accesses to registers are posted except for accesses to the IntSet, IntClear and IntEnable registers. AHB write operations are executed in order.

If the PowerDown bit of the PowerDown register is set, all AHB read and write accesses will return a read or write error except for accesses to the PowerDown register.

Bus Errors

The Ethernet block generates errors for several conditions:

- The AHB interface will return a read error when there is an AHB read access to a write-only register; likewise a write error is returned when there is an AHB write access to the read-only register. An AHB read or write error will be returned on AHB read or write accesses to reserved registers. These errors are propagated back to the CPU. Registers defined as read-only and write-only are identified in Table 10–127.
- If the PowerDown bit is set all accesses to AHB registers will result in an error response except for accesses to the PowerDown register.

17. Interrupts

The Ethernet block has a single interrupt request output to the CPU (via the NVIC).

The interrupt service routine must read the IntStatus register to determine the origin of the interrupt. All interrupt statuses can be set by software writing to the IntSet register; statuses can be cleared by software writing to the IntClear register.

The transmit and receive data paths can only set interrupt statuses, they cannot clear statuses. The SoftInt interrupt cannot be set by hardware and can be used by software for test purposes.

17.1 Direct Memory Access (DMA)

Descriptor arrays

© NXP B.V. 2010. All rights reserved.

The Ethernet block includes two DMA managers. The DMA managers make it possible to transfer frames directly to and from memory with little support from the processor and without the need to trigger an interrupt for each frame.

The DMA managers work with arrays of frame descriptors and statuses that are stored in memory. The descriptors and statuses act as an interface between the Ethernet hardware and the device driver software. There is one descriptor array for receive frames and one descriptor array for transmit frames. Using buffering for frame descriptors, the memory traffic and memory bandwidth utilization of descriptors can be kept small.

Each frame descriptor contains two 32-bit fields: the first field is a pointer to a data buffer containing a frame or a fragment, whereas the second field is a control word related to that frame or fragment.

The software driver must write the base addresses of the descriptor and status arrays in the TxDescriptor/RxDescriptor and TxStatus/RxStatus registers. The number of descriptors/statuses in each array must be written in the TxDescriptorNumber/RxDescriptorNumber registers. The number of descriptors in an array corresponds to the number of statuses in the associated status array.

Transmit descriptor arrays, receive descriptor arrays and transmit status arrays must be aligned on a 4 byte (32bit)address boundary, while the receive status array must be aligned on a 8 byte (64bit) address boundary.

Ownership of descriptors

Both device driver software and Ethernet hardware can read and write the descriptor arrays at the same time in order to produce and consume descriptors. A descriptor is "owned" either by the device driver or by the Ethernet hardware. Only the owner of a descriptor reads or writes its value. Typically, the sequence of use and ownership of descriptors and statuses is as follows: a descriptor is owned and set up by the device driver; ownership of the descriptor/status is passed by the device driver to the Ethernet block, which reads the descriptor and writes information to the status field; the Ethernet block passes ownership of the descriptor back to the device driver, which uses the status information and then recycles the descriptor to be used for another frame. Software must pre-allocate the memory used to hold the descriptor arrays.

Software can hand over ownership of descriptors and statuses to the hardware by incrementing (and wrapping if on the array boundary) the TxProduceIndex/RxConsumeIndex registers. Hardware hands over descriptors and status to software by updating the TxConsumeIndex/ RxProduceIndex registers.

After handing over a descriptor to the receive and transmit DMA hardware, device driver software should not modify the descriptor or reclaim the descriptor by decrementing the TxProduceIndex/ RxConsumeIndex registers because descriptors may have been prefetched by the hardware. In this case the device driver software will have to wait until the frame has been transmitted or the device driver has to soft-reset the transmit and/or receive data paths which will also reset the descriptor arrays.

Sequential order with wrap-around

When descriptors are read from and statuses are written to the arrays, this is done in sequential order with wrap-around. Sequential order means that when the Ethernet block has finished reading/writing a descriptor/status, the next descriptor/status it reads/writes is

© NXP B.V. 2010. All rights reserved.

the one at the next higher, adjacent memory address. Wrap around means that when the Ethernet block has finished reading/writing the last descriptor/status of the array (with the highest memory address), the next descriptor/status it reads/writes is the first descriptor/status of the array at the base address of the array.

Full and Empty state of descriptor arrays

The descriptor arrays can be empty, partially full or full. A descriptor array is empty when all descriptors are owned by the producer. A descriptor array is partially full if both producer and consumer own part of the descriptors and both are busy processing those descriptors. A descriptor array is full when all descriptors (except one) are owned by the consumer, so that the producer has no more room to process frames. Ownership of descriptors is indicated with the use of a consume index and a produce index. The produce index is the first element of the array owned by the producer. It is also the index of the array element that is next going to be used by the producer of frames (it may already be busy using it and subsequent elements). The consume index is the first element of the array that is owned by the consumer. It is also the number of the array element next to be consumed by the consumer of frames (it and subsequent elements may already be in the process of being consumed). If the consume index and the produce index are equal, the descriptor array is empty and all array elements are owned by the producer. If the consume index equals the produce index plus one, then the array is full and all array elements (except the one at the produce index) are owned by the consumer. With a full descriptor array, still one array element is kept empty, to be able to easily distinguish the full or empty state by looking at the value of the produce index and consume index. An array must have at least 2 elements to be able to indicate a full descriptor array with a produce index of value 0 and a consume index of value 1. The wrap around of the arrays is taken into account when determining if a descriptor array is full, so a produce index that indicates the last element in the array and a consume index that indicates the first element in the array, also means the descriptor array is full. When the produce index and the consume index are unequal and the consume index is not the produce index plus one (with wrap around taken into account), then the descriptor array is partially full and both the consumer and producer own enough descriptors to be able to operate actively on the descriptor array.

Interrupt bit

The descriptors have an Interrupt bit, which is programmed by software. When the Ethernet block is processing a descriptor and finds this bit set, it will allow triggering an interrupt (after committing status to memory) by passing the RxDoneInt or TxDoneInt bits in the IntStatus register to the interrupt output pin. If the Interrupt bit is not set in the descriptor, then the RxDoneInt or TxDoneInt are not set and no interrupt is triggered (note that the corresponding bits in IntEnable must also be set to trigger interrupts). This offers flexible ways of managing the descriptor arrays. For instance, the device driver could add 10 frames to the Tx descriptor array, and set the Interrupt bit in descriptor number 5 in the descriptor array. This would invoke the interrupt service routine before the transmit descriptor array is completely exhausted. The device driver could add another batch of frames to the descriptor array, without interrupting continuous transmission of frames.

Frame fragments

For maximum flexibility in frame storage, frames can be split up into multiple frame fragments with fragments located in different places in memory. In this case one descriptor is used for each frame fragment. So, a descriptor can point to a single frame or

© NXP B.V. 2010. All rights reserved.

to a fragment of a frame. By using fragments, scatter/gather DMA can be done: transmit frames are gathered from multiple fragments in memory and receive frames can be scattered to multiple fragments in memory.

By stringing together fragments it is possible to create large frames from small memory areas. Another use of fragments is to be able to locate a frame header and frame payload in different places and to concatenate them without copy operations in the device driver.

For transmissions, the Last bit in the descriptor Control field indicates if the fragment is the last in a frame; for receive frames, the LastFrag bit in the StatusInfo field of the status words indicates if the fragment is the last in the frame. If the Last(Frag) bit is 0 the next descriptor belongs to the same Ethernet frame, If the Last(Frag) bit is 1 the next descriptor is a new Ethernet frame.

17.2 Initialization

After reset, the Ethernet software driver needs to initialize the Ethernet block. During initialization the software needs to:

- Remove the soft reset condition from the MAC
- Configure the PHY via the MIIM interface of the MAC.

  Remark: it is important to configure the PHY and insure that reference clocks (ENET_REF_CLK signal in RMII mode, or both ENET_RX_CLK and ENET_TX_CLK signals in MII mode) are present at the external pins and connected to the EMAC module (selecting the appropriate pins using the PINSEL registers) prior to continuing with Ethernet configuration. Otherwise the CPU can become locked and no further functionality will be possible. This will cause JTAG lose communication with the target, if debug mode is being used.
- Select RMII mode
- Configure the transmit and receive DMA engines, including the descriptor arrays
- Configure the host registers (MAC1,MAC2 etc.) in the MAC
- Enable the receive and transmit data paths

Depending on the PHY, the software needs to initialize registers in the PHY via the MII Management interface. The software can read and write PHY registers by programming the MCFG, MCMD, MADR registers of the MAC. Write data should be written to the MWTD register; read data and status information can be read from the MRDD and MIND registers.

The Ethernet block supports RMII PHYs. During initialization software must select RMII mode by programming the Command register.

Before switching to RMII mode the default soft reset (MAC1 register bit 15) has to be de-asserted. The phy_ref_clk must be running and internally connected during this operation.

Transmit and receive DMA engines should be initialized by the device driver by allocating the descriptor and status arrays in memory. Transmit and receive functions have their own dedicated descriptor and status arrays. The base addresses of these arrays need to be programmed in the TxDescriptor/TxStatus and RxDescriptor/RxStatus registers. The number of descriptors in an array matches the number of statuses in an array.

© NXP B.V. 2010. All rights reserved.

Please note that the transmit descriptors, receive descriptors and receive statuses are 8 bytes each while the transmit statuses are 4 bytes each. All descriptor arrays and transmit statuses need to be aligned on 4 byte boundaries; receive status arrays need to be aligned on 8 byte boundaries. The number of descriptors in the descriptor arrays needs to be written to the TxDescriptorNumber/RxDescriptorNumber registers using a -1 encoding i.e. the value in the registers is the number of descriptors minus one e.g. if the descriptor array has 4 descriptors the value of the number of descriptors register should be 3.

After setting up the descriptor arrays, frame buffers need to be allocated for the receive descriptors before enabling the receive data path. The Packet field of the receive descriptors needs to be filled with the base address of the frame buffer of that descriptor. Amongst others the Control field in the receive descriptor needs to contain the size of the data buffer using -1 encoding.

The receive data path has a configurable filtering function for discarding/ignoring specific Ethernet frames. The filtering function should also be configured during initialization.

After an assertion of the hardware reset, the soft reset bit in the MAC will be asserted. The soft reset condition must be removed before the Ethernet block can be enabled.

Enabling of the receive function is located in two places. The receive DMA manager needs to be enabled and the receive data path of the MAC needs to be enabled. To prevent overflow in the receive DMA engine the receive DMA engine should be enabled by setting the RxEnable bit in the Command register before enabling the receive data path in the MAC by setting the RECEIVE ENABLE bit in the MAC1 register.

The transmit DMA engine can be enabled at any time by setting the TxEnable bit in the Command register.

Before enabling the data paths, several options can be programmed in the MAC, such as automatic flow control, transmit to receive loop-back for verification, full/half duplex modes, etc.

Base addresses of descriptor arrays and descriptor array sizes cannot be modified without a (soft) reset of the receive and transmit data paths.

17.3 Transmit process

Overview

This section outlines the transmission process.

Device driver sets up descriptors and data

If the descriptor array is full the device driver should wait for the descriptor arrays to become not full before writing to a descriptor in the descriptor array. If the descriptor array is not full, the device driver should use the descriptor numbered TxProduceIndex of the array pointed to by TxDescriptor.

The Packet pointer in the descriptor is set to point to a data frame or frame fragment to be transmitted. The Size field in the Command field of the descriptor should be set to the number of bytes in the fragment buffer, -1 encoded. Additional control information can be indicated in the Control field in the descriptor (bits Interrupt, Last, CRC, Pad).

© NXP B.V. 2010. All rights reserved.

After writing the descriptor the descriptor needs to be handed over to the hardware by incrementing (and possibly wrapping) the TxProduceIndex register.

If the transmit data path is disabled, the device driver should not forget to enable the transmit data path by setting the TxEnable bit in the Command register.

When there is a multi-fragment transmission for fragments other than the last, the Last bit in the descriptor must be set to 0; for the last fragment the Last bit must be set to 1. To trigger an interrupt when the frame has been transmitted and transmission status has been committed to memory, set the Interrupt bit in the descriptor Control field to 1. To have the hardware add a CRC in the frame sequence control field of this Ethernet frame, set the CRC bit in the descriptor. This should be done if the CRC has not already been added by software. To enable automatic padding of small frames to the minimum required frame size, set the Pad bit in the Control field of the descriptor to 1. In typical applications bits CRC and Pad are both set to 1.

The device driver can set up interrupts using the IntEnable register to wait for a signal of completion from the hardware or can periodically inspect (poll) the progress of transmission. It can also add new frames at the end of the descriptor array, while hardware consumes descriptors at the start of the array.

The device driver can stop the transmit process by resetting the TxEnable bit in the Command register to 0. The transmission will not stop immediately; frames already being transmitted will be transmitted completely and the status will be committed to memory before deactivating the data path. The status of the transmit data path can be monitored by the device driver reading the TxStatus bit in the Status register.

As soon as the transmit data path is enabled and the corresponding TxConsumeIndex and TxProduceIndex are not equal i.e. the hardware still needs to process frames from the descriptor array, the TxStatus bit in the Status register will return to 1 (active).

Tx DMA manager reads the Tx descriptor array

When the TxEnable bit is set, the Tx DMA manager reads the descriptors from memory at the address determined by TxDescriptor and TxConsumeIndex. The number of descriptors requested is determined by the total number of descriptors owned by the hardware: TxProduceIndex - TxConsumeIndex. Block transferring descriptors minimizes memory loading. Read data returned from memory is buffered and consumed as needed.

Tx DMA manager transmits data

After reading the descriptor the transmit DMA engine reads the associated frame data from memory and transmits the frame. After transfer completion, the Tx DMA manager writes status information back to the StatusInfo and StatusHashCRC words of the status field. The value of the TxConsumeIndex is only updated after status information has been committed to memory, which is checked by an internal tag protocol in the memory interface. The Tx DMA manager continues to transmit frames until the descriptor array is empty. If the transmit descriptor array is empty the TxStatus bit in the Status register will return to 0 (inactive). If the descriptor array is empty the Ethernet hardware will set the TxFinishedInt bit of the IntStatus register. The transmit data path will still be enabled.

The Tx DMA manager inspects the Last bit of the descriptor Control field when loading the descriptor. If the Last bit is 0, this indicates that the frame consists of multiple fragments. The Tx DMA manager gathers all the fragments from the host memory, visiting a string of

© NXP B.V. 2010. All rights reserved.

frame descriptors, and sends them out as one Ethernet frame on the Ethernet connection. When the Tx DMA manager finds a descriptor with the Last bit in the Control field set to 1, this indicates the last fragment of the frame and thus the end of the frame is found.

Update ConsumeIndex

Each time the Tx DMA manager commits a status word to memory it completes the transmission of a descriptor and it increments the TxConsumeIndex (taking wrap around into account) to hand the descriptor back to the device driver software. Software can re-use the descriptor for new transmissions after hardware has handed it back.

The device driver software can keep track of the progress of the DMA manager by reading the TxConsumeIndex register to see how far along the transmit process is. When the Tx descriptor array is emptied completely, the TxConsumeIndex register retains its last value.

Write transmission status

After the frame has been transmitted over the RMII bus, the StatusInfo word of the frame descriptor is updated by the DMA manager.

If the descriptor is for the last fragment of a frame (or for the whole frame if there are no fragments), then depending on the success or failure of the frame transmission, error flags (Error, LateCollision, ExcessiveCollision, Underrun, ExcessiveDefer, Defer) are set in the status. The CollisionCount field is set to the number of collisions the frame incurred, up to the Retransmission Maximum programmed in the Collision window/retry register of the MAC.

Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx DMA manager. Even if the descriptor is for a frame fragment other than the last fragment, the error flags are returned via the AHB interface. If the Ethernet block detects a transmission error during transmission of a (multi-fragment) frame, all remaining fragments of the frame are still read via the AHB interface. After an error, the remaining transmit data is discarded by the Ethernet block. If there are errors during transmission of a multi-fragment frame the error statuses will be repeated until the last fragment of the frame. Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx DMA manager. These may include error information if the error is detected early enough. The status for the last fragment in the frame will only be written after the transmission has completed on the Ethernet connection. Thus, the status for the last fragment will always reflect any error that occurred anywhere in the frame.

The status of the last frame transmission can also be inspected by reading the TSV0 and TSV1 registers. These registers do not report statuses on a fragment basis and do not store information of previously sent frames. They are provided primarily for debug purposes, because the communication between driver software and the Ethernet block takes place through the frame descriptors. The status registers are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

Transmission error handling

If an error occurs during the transmit process, the Tx DMA manager will report the error via the transmission StatusInfo word written in the Status array and the IntStatus interrupt status register.

© NXP B.V. 2010. All rights reserved.

The transmission can generate several types of errors: LateCollision, ExcessiveCollision, ExcessiveDefer, Underrun, and NoDescriptor. All have corresponding bits in the transmission StatusInfo word. In addition to the separate bits in the StatusInfo word, LateCollision, ExcessiveCollision, and ExcessiveDefer are ORed together into the Error bit of the Status. Errors are also propagated to the IntStatus register; the TxError bit in the IntStatus register is set in the case of a LateCollision, ExcessiveCollision, ExcessiveDefer, or NoDescriptor error; Underrun errors are reported in the TxUnderrun bit of the IntStatus register.

Underrun errors can have three causes:

- The next fragment in a multi-fragment transmission is not available. This is a nonfatal error. A NoDescriptor status will be returned on the previous fragment and the TxError bit in IntStatus will be set.
- The transmission fragment data is not available when the Ethernet block has already started sending the frame. This is a nonfatal error. An Underrun status will be returned on transfer and the TxError bit in IntStatus will be set.
- The flow of transmission statuses stalls and a new status has to be written while a previous status still waits to be transferred across the memory interface. This is a fatal error which can only be resolved by a soft reset of the hardware.

The first and second situations are nonfatal and the device driver has to re-send the frame or have upper software layers re-send the frame. In the third case the hardware is in an undefined state and needs to be soft reset by setting the TxReset bit in the Command register.

After reporting a LateCollision, ExcessiveCollision, ExcessiveDefer or Underrun error, the transmission of the erroneous frame will be aborted, remaining transmission data and frame fragments will be discarded and transmission will continue with the next frame in the descriptor array.

Device drivers should catch the transmission errors and take action.

Transmit triggers interrupts

The transmit data path can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set, the Tx DMA will set the TxDoneInt bit in the IntStatus register after sending the fragment and committing the associated transmission status to memory. Even if a descriptor (fragment) is not the last in a multi-fragment frame the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor array is empty while the Ethernet hardware is enabled the hardware will set the TxFinishedInt bit of the IntStatus register.
- If the AHB interface does not consume the transmission statuses at a sufficiently high bandwidth the transmission may underrun in which case the TxUnderrun bit will be set in the IntStatus register. This is a fatal error which requires a soft reset of the transmission queue.

© NXP B.V. 2010. All rights reserved.

- In the case of a transmission error (LateCollision, ExcessiveCollision, or ExcessiveDefer) or a multi-fragment frame where the device driver did provide the initial fragments but did not provide the rest of the fragments (NoDescriptor) or in the case of a nonfatal overrun, the hardware will set the TxErrorInt bit of the IntStatus register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the IntEnable register. Enabling or disabling does not affect the IntStatus register contents, only the propagation of the interrupt status to the CPU (via the NVIC).

The interrupts, either of individual frames or of the whole list, are a good means of communication between the DMA manager and the device driver, triggering the device driver to inspect the status words of descriptors that have been processed.

Transmit example

Figure 10–20 illustrates the transmit process in an example transmitting uses a frame header of 8 bytes and a frame payload of 12 bytes.

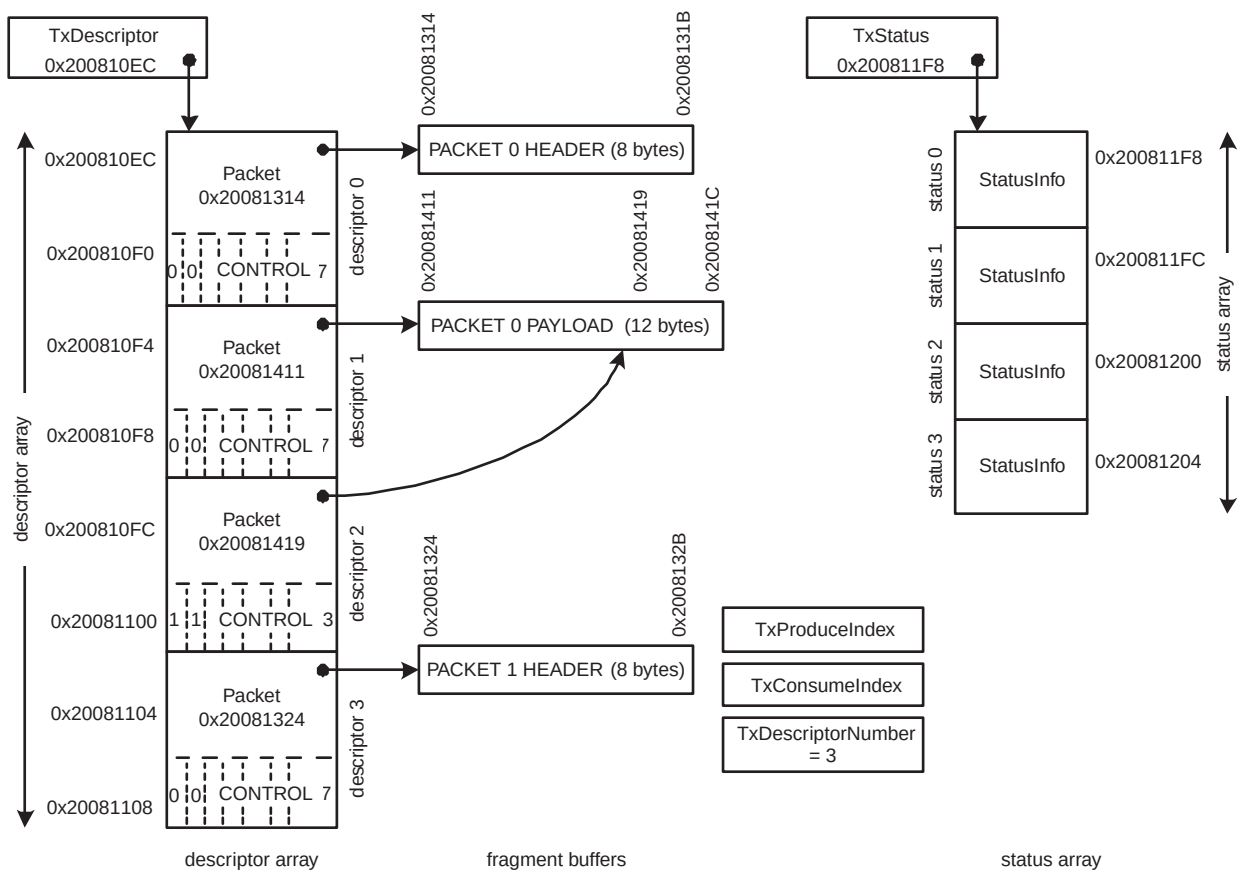


Fig 20. Transmit example memory and registers

After reset the values of the DMA registers will be zero. During initialization the device driver will allocate the descriptor and status array in memory. In this example, an array of four descriptors is allocated; the array is 4x2x4 bytes and aligned on a 4 byte address

© NXP B.V. 2010. All rights reserved.

boundary. Since the number of descriptors matches the number of statuses the status array consists of four elements; the array is 4x1x4 bytes and aligned on a 4 byte address boundary. The device driver writes the base address of the descriptor array (0x2008 10EC) to the TxDescriptor register and the base address of the status array (0x2008 11F8) to the TxStatus register. The device driver writes the number of descriptors and statuses minus 1(3) to the TxDescriptorNumber register. The descriptors and statuses in the arrays need not be initialized, yet.

At this point, the transmit data path may be enabled by setting the TxEnable bit in the Command register. If the transmit data path is enabled while there are no further frames to send the TxFinishedInt interrupt flag will be set. To reduce the processor interrupt load only the desired interrupts can be enabled by setting the relevant bits in the IntEnable register.

Now suppose application software wants to transmit a frame of 12 bytes using a TCP/IP protocol (in real applications frames will be larger than 12 bytes). The TCP/IP stack will add a header to the frame. The frame header need not be immediately in front of the payload data in memory. The device driver can program the Tx DMA to collect header and payload data. To do so, the device driver will program the first descriptor to point at the frame header; the Last flag in the descriptor will be set to false/0 to indicate a multi-fragment transmission. The device driver will program the next descriptor to point at the actual payload data. The maximum size of a payload buffer is 2 kB so a single descriptor suffices to describe the payload buffer. For the sake of the example though the payload is distributed across two descriptors. After the first descriptor in the array describing the header, the second descriptor in the array describes the initial 8 bytes of the payload; the third descriptor in the array describes the remaining 4 bytes of the frame. In the third descriptor the Last bit in the Control word is set to true/1 to indicate it is the last descriptor in the frame. In this example the Interrupt bit in the descriptor Control field is set in the last fragment of the frame in order to trigger an interrupt after the transmission completed. The Size field in the descriptor’s Control word is set to the number of bytes in the fragment buffer, -1 encoded.

Note that in real device drivers, the payload will typically only be split across multiple descriptors if it is more than 2 kB. Also note that transmission payload data is forwarded to the hardware without the device driver copying it (zero copy device driver).

After setting up the descriptors for the transaction the device driver increments the TxProduceIndex register by 3 since three descriptors have been programmed. If the transmit data path was not enabled during initialization the device driver needs to enable the data path now.

If the transmit data path is enabled the Ethernet block will start transmitting the frame as soon as it detects the TxProduceIndex is not equal to TxConsumeIndex - both were zero after reset. The Tx DMA will start reading the descriptors from memory. The memory system will return the descriptors and the Ethernet block will accept them one by one while reading the transmit data fragments.

As soon as transmission read data is returned from memory, the Ethernet block will try to start transmission on the Ethernet connection via the RMII interface.

© NXP B.V. 2010. All rights reserved.

After transmitting each fragment of the frame the Tx DMA will write the status of the fragment's transmission. Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx DMA manager. The status for the last fragment in the frame will only be written after the transmission has completed on the Ethernet connection.

Since the Interrupt bit in the descriptor of the last fragment is set, after committing the status of the last fragment to memory the Ethernet block will trigger a TxDoneInt interrupt, which triggers the device driver to inspect the status information.

In this example the device driver cannot add new descriptors as long as the Ethernet block has not incremented the TxConsumeIndex because the descriptor array is full (even though one descriptor is not programmed yet). Only after the hardware commits the status for the first fragment to memory and the TxConsumeIndex is set to 1 by the DMA manager can the device driver program the next (the fourth) descriptor. The fourth descriptor can already be programmed before completely transmitting the first frame.

In this example the hardware adds the CRC to the frame. If the device driver software adds the CRC, the CRC trailer can be considered another frame fragment which can be added by doing another gather DMA.

Each data byte is transmitted across the RMII interface as four 2-bit values. The Ethernet block adds the preamble, frame delimiter leader, and the CRC trailer if hardware CRC is enabled. Once transmission on the RMII interface commences the transmission cannot be interrupted without generating an underrun error, which is why descriptors and data read commands are issued as soon as possible and pipelined.

Using an RMII PHY, the data communication between the Ethernet block and the PHY is communicated at 50 MHz. In 10 Mbps mode data will only be transmitted once every 10 clock cycles.

17.4 Receive process

This section outlines the receive process including the activities in the device driver software.

Device driver sets up descriptors

After initializing the receive descriptor and status arrays to receive frames from the Ethernet connection, the receive data path should be enabled in the MAC1 register and the Control register.

During initialization, each Packet pointer in the descriptors is set to point to a data fragment buffer. The size of the buffer is stored in the Size bits of the Control field of the descriptor. Additionally, the Control field in the descriptor has an Interrupt bit. The Interrupt bit allows generation of an interrupt after a fragment buffer has been filled and its status has been committed to memory.

After the initialization and enabling of the receive data path, all descriptors are owned by the receive hardware and should not be modified by the software unless hardware hands over the descriptor by incrementing the RxProduceIndex, indicating that a frame has been received. The device driver is allowed to modify the descriptors after a (soft) reset of the receive data path.

© NXP B.V. 2010. All rights reserved.

Rx DMA manager reads Rx descriptor arrays

When the RxEnable bit in the Command register is set, the Rx DMA manager reads the descriptors from memory at the address determined by RxDescriptor and RxProduceIndex. The Ethernet block will start reading descriptors even before actual receive data arrives on the RMII interface (descriptor prefetching). The block size of the descriptors to be read is determined by the total number of descriptors owned by the hardware: RxConsumeIndex - RxProduceIndex - 1. Block transferring of descriptors minimizes memory load. Read data returned from memory is buffered and consumed as needed.

RX DMA manager receives data

After reading the descriptor, the receive DMA engine waits for the MAC to return receive data from the RMII interface that passes the receive filter. Receive frames that do not match the filtering criteria are not passed to memory. Once a frame passes the receive filter, the data is written in the fragment buffer associated with the descriptor. The Rx DMA does not write beyond the size of the buffer. When a frame is received that is larger than a descriptor's fragment buffer, the frame will be written to multiple fragment buffers of consecutive descriptors. In the case of a multi-fragment reception, all but the last fragment in the frame will return a status where the LastFrag bit is set to 0. Only on the last fragment of a frame the LastFrag bit in the status will be set to 1. If a fragment buffer is the last of a frame, the buffer may not be filled completely. The first receive data of the next frame will be written to the fragment buffer of the next descriptor.

After receiving a fragment, the Rx DMA manager writes status information back to the StatusInfo and StatusHashCRC words of the status. The Ethernet block writes the size in bytes of a descriptor's fragment buffer in the RxSize field of the Status word. The value of the RxProduceIndex is only updated after the fragment data and the fragment status information has been committed to memory, which is checked by an internal tag protocol in the memory interface. The Rx DMA manager continues to receive frames until the descriptor array is full. If the descriptor array is full, the Ethernet hardware will set the RxFinishedInt bit of the IntStatus register. The receive data path will still be enabled. If the receive descriptor array is full any new receive data will generate an overflow error and interrupt.

Update ProduceIndex

Each time the Rx DMA manager commits a data fragment and the associated status word to memory, it completes the reception of a descriptor and increments the RxProduceIndex (taking wrap around into account) in order to hand the descriptor back to the device driver software. Software can re-use the descriptor for new receptions by handing it back to hardware when the receive data has been processed.

The device driver software can keep track of the progress of the DMA manager by reading the RxProduceIndex register to see how far along the receive process is. When the Rx descriptor array is emptied completely, the RxProduceIndex retains its last value.

Write reception status

After the frame has been received from the RMII bus, the StatusInfo and StatusHashCRC words of the frame descriptor are updated by the DMA manager.

© NXP B.V. 2010. All rights reserved.

If the descriptor is for the last fragment of a frame (or for the whole frame if there are no fragments), then depending on the success or failure of the frame reception, error flags (Error, NoDescriptor, Overrun, AlignmentError, RangeError, LengthError, SymbolError, or CRCError) are set in StatusInfo. The RxSize field is set to the number of bytes actually written to the fragment buffer, -1 encoded. For fragments not being the last in the frame the RxSize will match the size of the buffer. The hash CRCs of the destination and source addresses of a packet are calculated once for all the fragments belonging to the same packet and then stored in every StatusHashCRC word of the statuses associated with the corresponding fragments. If the reception reports an error, any remaining data in the receive frame is discarded and the LastFrag bit will be set in the receive status field, so the error flags in all but the last fragment of a frame will always be 0.

The status of the last received frame can also be inspected by reading the RSV register. The register does not report statuses on a fragment basis and does not store information of previously received frames. RSV is provided primarily for debug purposes, because the communication between driver software and the Ethernet block takes place through the frame descriptors.

Reception error handling

When an error occurs during the receive process, the Rx DMA manager will report the error via the receive StatusInfo written in the Status array and the IntStatus interrupt status register.

The receive process can generate several types of errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError, Overrun, and NoDescriptor. All have corresponding bits in the receive StatusInfo. In addition to the separate bits in the StatusInfo, AlignmentError, RangeError, LengthError, SymbolError, and CRCError are ORed together into the Error bit of the StatusInfo. Errors are also propagated to the IntStatus register; the RxError bit in the IntStatus register is set if there is an AlignmentError, RangeError, LengthError, SymbolError, CRCError, or NoDescriptor error; nonfatal overrun errors are reported in the RxError bit of the IntStatus register; fatal Overrun errors are report in the RxOverrun bit of the IntStatus register. On fatal overrun errors, the Rx data path needs to be soft reset by setting the RxReset bit in the Command register.

Overrun errors can have three causes:

- In the case of a multi-fragment reception, the next descriptor may be missing. In this case the NoDescriptor field is set in the status word of the previous descriptor and the RxError in the IntStatus register is set. This error is nonfatal.
- The data flow on the receiver data interface stalls, corrupting the packet. In this case the overrun bit in the status word is set and the RxError bit in the IntStatus register is set. This error is nonfatal.
- The flow of reception statuses stalls and a new status has to be written while a previous status still waits to be transferred across the memory interface. This error will corrupt the hardware state and requires the hardware to be soft reset. The error is detected and sets the Overrun bit in the IntStatus register.

The first overrun situation will result in an incomplete frame with a NoDescriptor status and the RxError bit in IntStatus set. Software should discard the partially received frame. In the second overrun situation the frame data will be corrupt which results in the Overrun status bit being set in the Status word while the IntError interrupt bit is set. In the third case

© NXP B.V. 2010. All rights reserved.

receive errors cannot be reported in the receiver Status arrays which corrupts the hardware state; the errors will still be reported in the IntStatus register's Overrun bit. The RxReset bit in the Command register should be used to soft reset the hardware.

Device drivers should catch the above receive errors and take action.

Receive triggers interrupts

The receive data path can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set, the Rx DMA will set the RxDoneInt bit in the IntStatus register after receiving a fragment and committing the associated data and status to memory. Even if a descriptor (fragment) is not the last in a multi-fragment frame, the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor array is full while the Ethernet hardware is enabled, the hardware will set the RxFinishedInt bit of the IntStatus register.
- If the AHB interface does not consume receive statuses at a sufficiently high bandwidth, the receive status process may overrun, in which case the RxOverrun bit will be set in the IntStatus register.
- If there is a receive error (AlignmentError, RangeError, LengthError, SymbolError, or CRCError), or a multi-fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments, or if a nonfatal data Overrun occurred, the hardware will set the RxErrorInt bit of the IntStatus register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the IntEnable register. Enabling or disabling does not affect the IntStatus register contents, only the propagation of the interrupt status to the CPU (via the NVIC).

The interrupts, either of individual frames or of the whole list, are a good means of communication between the DMA manager and the device driver, triggering the device driver to inspect the status words of descriptors that have been processed.

Device driver processes receive data

As a response to status (e.g. RxDoneInt) interrupts or polling of the RxProduceIndex, the device driver can read the descriptors that have been handed over to it by the hardware (RxProduceIndex - RxConsumeIndex). The device driver should inspect the status words in the status array to check for multi-fragment receptions and receive errors.

The device driver can forward receive data and status to upper software layers. After processing of data and status, the descriptors, statuses and data buffers may be recycled and handed back to hardware by incrementing the RxConsumeIndex.

Receive example

Figure 10–21 illustrates the receive process in an example receiving a frame of 19 bytes.

© NXP B.V. 2010. All rights reserved.

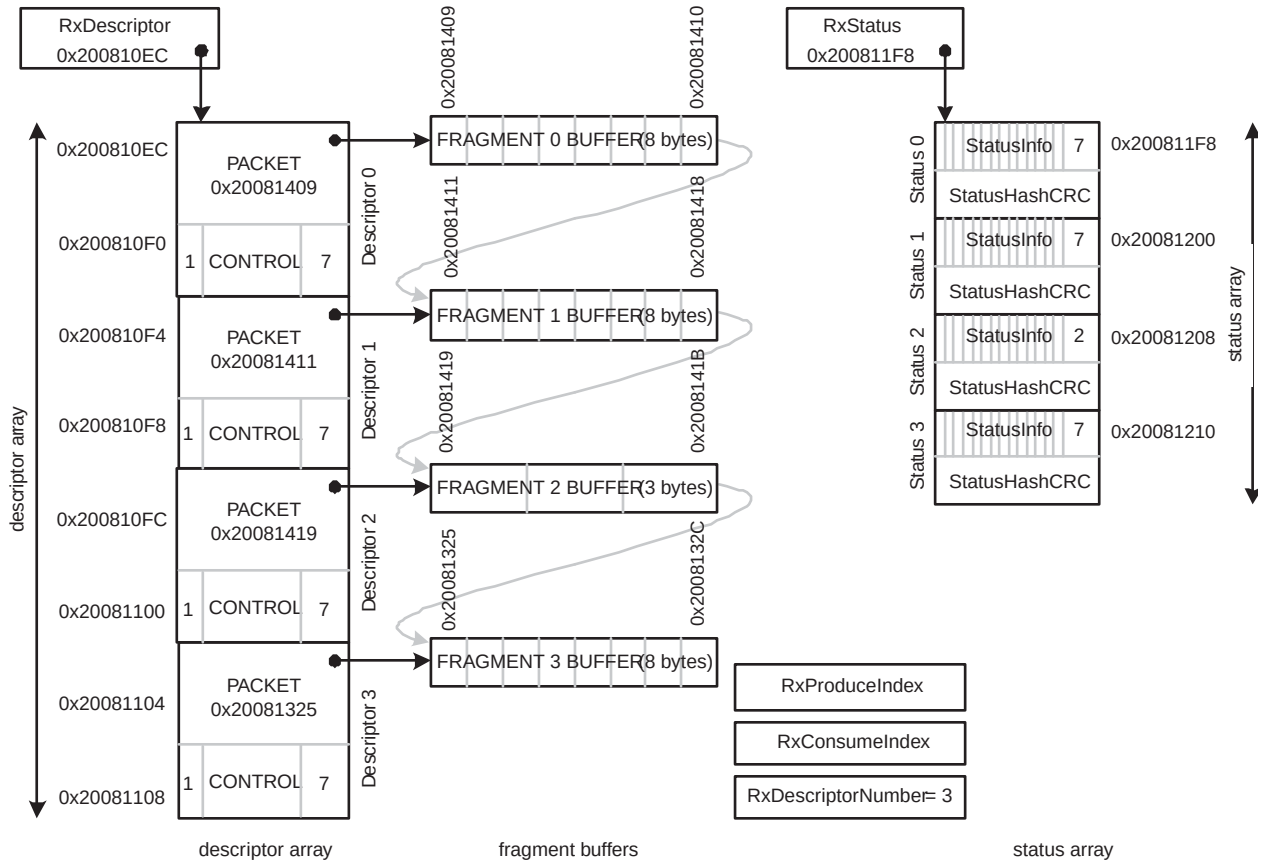


Fig 21. Receive Example Memory and Registers

After reset, the values of the DMA registers will be zero. During initialization, the device driver will allocate the descriptor and status array in memory. In this example, an array of four descriptors is allocated; the array is 4x2x4 bytes and aligned on a 4 byte address boundary. Since the number of descriptors matches the number of statuses, the status array consists of four elements; the array is 4x2x4 bytes and aligned on a 8 byte address boundary. The device driver writes the base address of the descriptor array (0x2008 10EC) in the RxDescriptor register, and the base address of the status array (0x2008 11F8) in the RxStatus register. The device driver writes the number of descriptors and statuses minus 1 (3) in the RxDescriptorNumber register. The descriptors and statuses in the arrays need not be initialized yet.

After allocating the descriptors, a fragment buffer needs to be allocated for each of the descriptors. Each fragment buffer can be between 1 byte and 2 k bytes. The base address of the fragment buffer is stored in the Packet field of the descriptors. The number of bytes in the fragment buffer is stored in the Size field of the descriptor Control word. The Interrupt field in the Control word of the descriptor can be set to generate an interrupt as soon as the descriptor has been filled by the receive process. In this example the fragment buffers are 8 bytes, so the value of the Size field in the Control word of the descriptor is set to 7. Note that in this example, the fragment buffers are actually a

© NXP B.V. 2010. All rights reserved.

continuous memory space; even when a frame is distributed over multiple fragments it will typically be in a linear, continuous memory space; when the descriptors wrap at the end of the descriptor array the frame will not be in a continuous memory space.

The device driver should enable the receive process by writing a 1 to the RxEnable bit of the Command register, after which the MAC needs to be enabled by writing a 1 to the 'RECEIVE ENABLE' bit of the MAC1 configuration register. The Ethernet block will now start receiving Ethernet frames. To reduce the processor interrupt load, some interrupts can be disabled by setting the relevant bits in the IntEnable register.

After the Rx DMA manager is enabled, it will start issuing descriptor read commands. In this example the number of descriptors is 4. Initially the RxProduceIndex and RxConsumeIndex are 0. Since the descriptor array is considered full if RxProduceIndex == RxConsumeIndex - 1, the Rx DMA manager can only read (RxConsumeIndex - RxProduceIndex - 1 =) 3 descriptors; note the wrapping.

After enabling the receive function in the MAC, data reception will begin starting at the next frame i.e. if the receive function is enabled while the RMII interface is halfway through receiving a frame, the frame will be discarded and reception will start at the next frame. The Ethernet block will strip the preamble and start of frame delimiter from the frame. If the frame passes the receive filtering, the Rx DMA manager will start writing the frame to the first fragment buffer.

Suppose the frame is 19 bytes long. Due to the buffer sizes specified in this example, the frame will be distributed over three fragment buffers. After writing the initial 8 bytes in the first fragment buffer, the status for the first fragment buffer will be written and the Rx DMA will continue filling the second fragment buffer. Since this is a multi-fragment receive, the status of the first fragment will have a 0 for the LastFrag bit in the StatusInfo word; the RxSize field will be set to 7 (8, -1 encoded). After writing the 8 bytes in the second fragment the Rx DMA will continue writing the third fragment. The status of the second fragment will be like the status of the first fragment: LastFrag = 0, RxSize = 7. After writing the three bytes in the third fragment buffer, the end of the frame has been reached and the status of the third fragment is written. The third fragment's status will have the LastFrag bit set to 1 and the RxSize equal to 2 (3, -1 encoded).

The next frame received from the RMII interface will be written to the fourth fragment buffer i.e. five bytes of the third buffer will be unused.

The Rx DMA manager uses an internal tag protocol in the memory interface to check that the receive data and status have been committed to memory. After the status of the fragments are committed to memory, an RxDoneInt interrupt will be triggered, which activates the device driver to inspect the status information. In this example, all descriptors have the Interrupt bit set in the Control word i.e. all descriptors will generate an interrupt after committing data and status to memory.

In this example the receive function cannot read new descriptors as long as the device driver does not increment the RxConsumeIndex, because the descriptor array is full (even though one descriptor is not programmed yet). Only after the device driver has forwarded the receive data to application software, and after the device driver has updated the RxConsumeIndex by incrementing it, will the Ethernet block can continue reading descriptors and receive data. The device driver will probably increment the RxConsumeIndex by 3, since the driver will forward the complete frame consisting of three fragments to the application, and hence free up three descriptors at the same time.

© NXP B.V. 2010. All rights reserved.

Each four pairs of bits transferred on the RMII interface are transferred as a byte on the data write interface after being delayed by 128 or 136 cycles for filtering by the receive filter and buffer modules. The Ethernet block removes preamble, frame start delimiter, and CRC from the data and checks the CRC. To limit the buffer NoDescriptor error probability, three descriptors are buffered. The value of the RxProduceIndex is only updated after status information has been committed to memory, which is checked by an internal tag protocol in the memory interface. The software device driver will process the receive data, after which the device driver will update the RxConsumeIndex.

17.5 Transmission retry

If a collision on the Ethernet occurs, it usually takes place during the collision window spanning the first 64 bytes of a frame. If collision is detected, the Ethernet block will retry the transmission. For this purpose, the first 64 bytes of a frame are buffered, so that this data can be used during the retry. A transmission retry within the first 64 bytes in a frame is fully transparent to the application and device driver software.

When a collision occurs outside of the 64 byte collision window, a LateCollision error is triggered, and the transmission is aborted. After a LateCollision error, the remaining data in the transmit frame will be discarded. The Ethernet block will set the Error and LateCollision bits in the frame's status fields. The TxError bit in the IntStatus register will be set. If the corresponding bit in the IntEnable register is set, the TxError bit in the IntStatus register will be propagated to the CPU (via the NVIC). The device driver software should catch the interrupt and take appropriate actions.

The 'RETRANSMISSION MAXIMUM' field of the CLRT register can be used to configure the maximum number of retries before aborting the transmission.

17.6 Status hash CRC calculations

For each received frame, the Ethernet block is able to detect the destination address and source address and from them calculate the corresponding hash CRCs. To perform the computation, the Ethernet block features two internal blocks: one is a controller synchronized with the beginning and the end of each frame, the second block is the CRC calculator.

When a new frame is detected, internal signaling notifies the controller.The controller starts counting the incoming bytes of the frame, which correspond to the destination address bytes. When the sixth (and last) byte is counted, the controller notifies the calculator to store the corresponding 32-bit CRC into a first inner register. Then the controller repeats counting the next incoming bytes, in order to get synchronized with the source address. When the last byte of the source address is encountered, the controller again notifies the CRC calculator, which freezes until the next new frame. When the calculator receives this second notification, it stores the present 32-bit CRC into a second inner register. Then the CRCs remain frozen in their own registers until new notifications arise.

The destination address and source address hash CRCs being written in the StatusHashCRC word are the nine most significant bits of the 32-bit CRCs as calculated by the CRC calculator.

© NXP B.V. 2010. All rights reserved.

17.7 Duplex modes

The Ethernet block can operate in full duplex and half duplex mode. Half or full duplex mode needs to be configured by the device driver software during initialization.

For a full duplex connection the FullDuplex bit of the Command register needs to be set to 1 and the FULL-DUPLEX bit of the MAC2 configuration register needs to be set to 1; for half duplex the same bits need to be set to 0.

17.8 IEE 802.3/Clause 31 flow control

Overview

For full duplex connections, the Ethernet block supports IEEE 802.3/clause 31 flow control using pause frames. This type of flow control may be used in full-duplex point-to-point connections. Flow control allows a receiver to stall a transmitter e.g. when the receive buffers are (almost) full. For this purpose, the receiving side sends a pause frame to the transmitting side.

Pause frames use units of 512 bit times corresponding to 128 rx_clk/tx_clk cycles.

Receive flow control

In full-duplex mode, the Ethernet block will suspend its transmissions when the it receives a pause frame. Rx flow control is initiated by the receiving side of the transmission. It is enabled by setting the ‘RX FLOW CONTROL’ bit in the MAC1 configuration register. If the RX FLOW CONTROL’ bit is zero, then the Ethernet block ignores received pause control frames. When a pause frame is received on the Rx side of the Ethernet block, transmission on the Tx side will be interrupted after the currently transmitting frame has completed, for an amount of time as indicated in the received pause frame. The transmit data path will stop transmitting data for the number of 512 bit slot times encoded in the pause-timer field of the received pause control frame.

By default the received pause control frames are not forwarded to the device driver. To forward the receive flow control frames to the device driver, set the ‘PASS ALL RECEIVE FRAMES’ bit in the MAC1 configuration register.

Transmit flow control

If case device drivers need to stall the receive data e.g. because software buffers are full, the Ethernet block can transmit pause control frames. Transmit flow control needs to be initiated by the device driver software; there is no IEEE 802.3/31 flow control initiated by hardware, such as the DMA managers.

With software flow control, the device driver can detect a situation in which the process of receiving frames needs to be interrupted by sending out Tx pause frames. Note that due to Ethernet delays, a few frames can still be received before the flow control takes effect and the receive stream stops.

Transmit flow control is activated by writing 1 to the TxFlowControl bit of the Command register. When the Ethernet block operates in full duplex mode, this will result in transmission of IEEE 802.3/31 pause frames. The flow control continues until a 0 is written to TxFlowControl bit of the Command register.

© NXP B.V. 2010. All rights reserved.

If the MAC is operating in full-duplex mode, then setting the TxFlowControl bit of the Command register will start a pause frame transmission. The value inserted into the pause-timer value field of transmitted pause frames is programmed via the PauseTimer[15:0] bits in the FlowControlCounter register. When the TxFlowControl bit is de-asserted, another pause frame having a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission.

When flow control be in force for an extended time, a sequence of pause frames must be transmitted. This is supported with a mirror counter mechanism. To enable mirror counting, a nonzero value is written to the MirrorCounter[15:0] bits in the FlowControlCounter register. When the TxFlowControl bit is asserted, a pause frame is transmitted. After sending the pause frame, an internal mirror counter is initialized to zero. The internal mirror counter starts incrementing one every 512 bit-slot times. When the internal mirror counter reaches the MirrorCounter value, another pause frame is transmitted with pause-timer value equal to the PauseTimer field from the FlowControlCounter register, the internal mirror counter is reset to zero and restarts counting. The register MirrorCounter[15:0] is usually set to a smaller value than register PauseTimer[15:0] to ensure an early expiration of the mirror counter, allowing time to send a new pause frame before the transmission on the other side can resume. By continuing to send pause frames before the transmitting side finishes counting the pause timer, the pause can be extended as long as TxFlowControl is asserted. This continues until TxFlowControl is de-asserted when a final pause frame having a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission. To disable the mirror counter mechanism, write the value 0 to MirrorCounter field in the FlowControlCounter register. When using the mirror counter mechanism, account for time-of-flight delays, frame transmission time, queuing delays, crystal frequency tolerances, and response time delays by programming the MirrorCounter conservatively, typically about 80% of the PauseTimer value.

If the software device driver sets the MirrorCounter field of the FlowControlCounter register to zero, the Ethernet block will only send one pause control frame. After sending the pause frame an internal pause counter is initialized at zero; the internal pause counter is incremented by one every 512 bit-slot times. Once the internal pause counter reaches the value of the PauseTimer register, the TxFlowControl bit in the Command register will be reset. The software device driver can poll the TxFlowControl bit to detect when the pause completes.

The value of the internal counter in the flow control module can be read out via the FlowControlStatus register. If the MirrorCounter is nonzero, the FlowControlStatus register will return the value of the internal mirror counter; if the MirrorCounter is zero the FlowControlStatus register will return the value of the internal pause counter value.

The device driver is allowed to dynamically modify the MirrorCounter register value and switch between zero MirrorCounter and nonzero MirrorCounter modes.

Transmit flow control is enabled via the ‘TX FLOW CONTROL’ bit in the MAC1 configuration register. If the ‘TX FLOW CONTROL’ bit is zero, then the MAC will not transmit pause control frames, software must not initiate pause frame transmissions, and the TxFlowControl bit in the Command register should be zero.

Transmit flow control example

Figure 10–22 illustrates the transmit flow control.

© NXP B.V. 2010. All rights reserved.

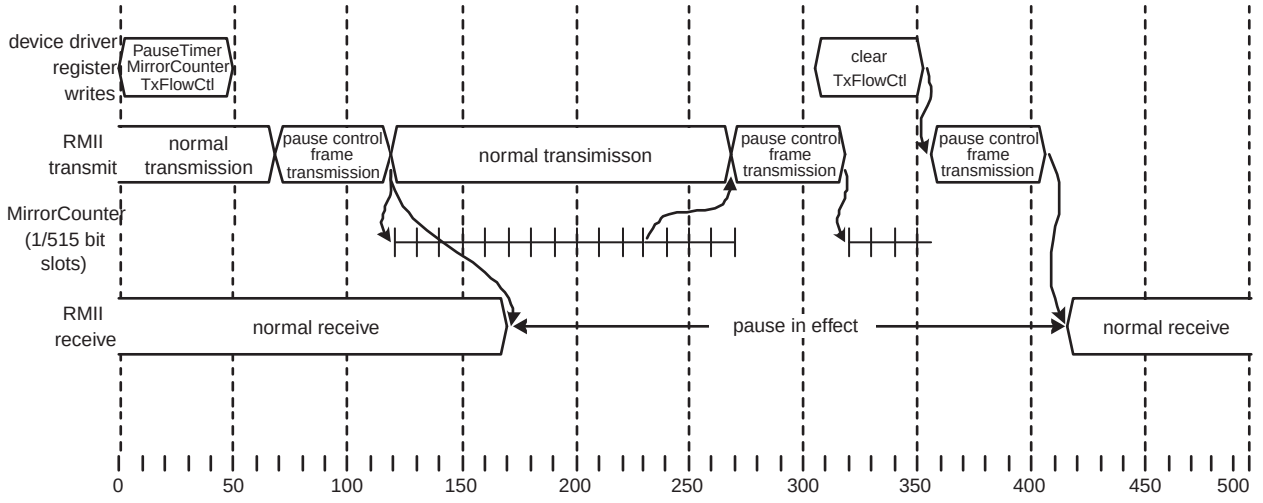


Fig 22. Transmit Flow Control

In this example, a frame is received while transmitting another frame (full duplex.) The device driver detects that some buffer might overrun and enables the transmit flow control by programming the PauseTimer and MirrorCounter fields of the FlowControlCounter register, after which it enables the transmit flow control by setting the TxFlowControl bit in the Command register.

As a response to the enabling of the flow control a pause control frame will be sent after the currently transmitting frame has been transmitted. When the pause frame transmission completes the internal mirror counter will start counting bit slots; as soon as the counter reaches the value in the MirrorCounter field another pause frame is transmitted. While counting the transmit data path will continue normal transmissions.

As soon as software disables transmit flow control a zero pause control frame is transmitted to resume the receive process.

17.9 Half-Duplex mode backpressure

When in half-duplex mode, backpressure can be generated to stall receive packets by sending continuous preamble that basically jams any other transmissions on the Ethernet medium. When the Ethernet block operates in half duplex mode, asserting the TxFlowControl bit in the Command register will result in applying continuous preamble on the Ethernet wire, effectively blocking traffic from any other Ethernet station on the same segment.

In half duplex mode, when the TxFlowControl bit goes high, continuous preamble is sent until TxFlowControl is de-asserted. If the medium is idle, the Ethernet block begins transmitting preamble, which raises carrier sense causing all other stations to defer. In the event the transmitting of preamble causes a collision, the backpressure ‘rides through’ the collision. The colliding station backs off and then defers to the backpressure. If during backpressure, the user wishes to send a frame, the backpressure is interrupted, the frame sent and then the backpressure resumed. If TxFlowControl is asserted for longer than 3.3 ms in 10 Mbps mode or 0.33 ms in 100 Mbps mode, backpressure will cease sending preamble for several byte times to avoid the jabber limit.

© NXP B.V. 2010. All rights reserved.

17.10 Receive filtering

Features of receive filtering

The Ethernet MAC has several receive packet filtering functions that can be configured from the software driver:

- Perfect address filter: allows packets with a perfectly matching station address to be identified and passed to the software driver.
- Hash table filter: allows imperfect filtering of packets based on the station address.
- Unicast/multicast/broadcast filtering: allows passing of all unicast, multicast, and/or broadcast packets.
- Magic packet filter: detection of magic packets to generate a Wake-on-LAN interrupt.

The filtering functions can be logically combined to create complex filtering functions. Furthermore, the Ethernet block can pass or reject runt packets smaller than 64 bytes; a promiscuous mode allows all packets to be passed to software.

Overview

The Ethernet block has the capability to filter out receive frames by analyzing the Ethernet destination address in the frame. This capability greatly reduces the load on the host system, because Ethernet frames that are addressed to other stations would otherwise need to be inspected and rejected by the device driver software, using up bandwidth, memory space, and host CPU time. Address filtering can be implemented using the perfect address filter or the (imperfect) hash filter. The latter produces a 6-bit hash code which can be used as an index into a 64 entry programmable hash table. Figure 10–23 depicts a functional view of the receive filter.

At the top of the diagram the Ethernet receive frame enters the filters. Each filter is controlled by signals from control registers; each filter produces a ‘Ready’ output and a ‘Match’ output. If ‘Ready’ is 0 then the Match value is ‘don’t care’; if a filter finishes filtering then it will assert its Ready output; if the filter finds a matching frame it will assert the Match output along with the Ready output. The results of the filters are combined by logic functions into a single RxAbort output. If the RxAbort output is asserted, the frame does not need to be received.

In order to reduce memory traffic, the receive data path has a buffer of 68 bytes. The Ethernet MAC will only start writing a frame to memory after 68 byte delays. If the RxAbort signal is asserted during the initial 68 bytes of the frame, the frame can be discarded and removed from the buffer and not stored to memory at all, not using up receive descriptors, etc. If the RxAbort signal is asserted after the initial 68 bytes in a frame (probably due to reception of a Magic Packet), part of the frame is already written to memory and the Ethernet MAC will stop writing further data in the frame to memory; the FailFilter bit in the status word of the frame will be set to indicate that the software device driver can discard the frame immediately.

© NXP B.V. 2010. All rights reserved.

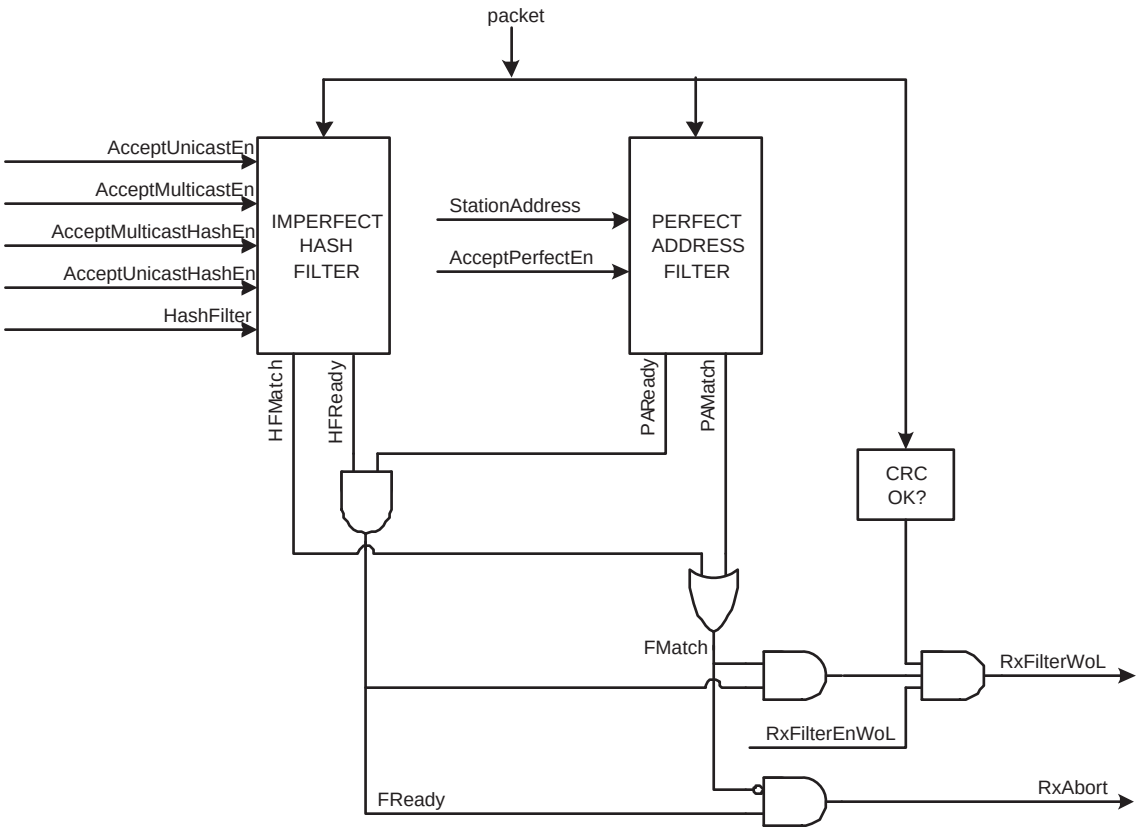


Fig 23. Receive filter block diagram

Unicast, broadcast and multicast

Generic filtering based on the type of frame (unicast, multicast or broadcast) can be programmed using the AcceptUnicastEn, AcceptMulticastEn, or AcceptBroadcastEn bits of the RxFilterCtrl register. Setting the AcceptUnicast, AcceptMulticast, and AcceptBroadcast bits causes all frames of types unicast, multicast and broadcast, respectively, to be accepted, ignoring the Ethernet destination address in the frame. To program promiscuous mode, i.e. to accept all frames, set all 3 bits to 1.

Perfect address match

When a frame with a unicast destination address is received, a perfect filter compares the destination address with the 6 byte station address programmed in the station address registers SA0, SA1, SA2. If the AcceptPerfectEn bit in the RxFilterCtrl register is set to 1, and the address matches, the frame is accepted.

Imperfect hash filtering

An imperfect filter is available, based on a hash mechanism. This filter applies a hash function to the destination address and uses the hash to access a table that indicates if the frame should be accepted. The advantage of this type of filter is that a small table can cover any possible address. The disadvantage is that the filtering is imperfect, i.e. sometimes frames are accepted that should have been discarded.

© NXP B.V. 2010. All rights reserved.

- Hash function:
  - The standard Ethernet cyclic redundancy check (CRC) function is calculated from the 6 byte destination address in the Ethernet frame (this CRC is calculated anyway as part of calculating the CRC of the whole frame), then bits [28:23] out of the 32-bit CRC result are taken to form the hash. The 6-bit hash is used to access the hash table: it is used as an index in the 64-bit HashFilter register that has been programmed with accept values. If the selected accept value is 1, the frame is accepted.
  - The device driver can initialize the hash filter table by writing to the registers HashFilterL and HashfilterH. HashFilterL contains bits 0 through 31 of the table and HashFilterH contains bit 32 through 63 of the table. So, hash value 0 corresponds to bit 0 of the HashfilterL register and hash value 63 corresponds to bit 31 of the HashFilterH register.
- Multicast and unicast
  - The imperfect hash filter can be applied to multicast addresses, by setting the AcceptMulticastHashEn bit in the RxFilter register to 1.
  - The same imperfect hash filter that is available for multicast addresses can also be used for unicast addresses. This is useful to be able to respond to a multitude of unicast addresses without enabling all unicast addresses. The hash filter can be applied to unicast addresses by setting the AcceptUnicastHashEn bit in the RxFilter register to 1.

Enabling and disabling filtering

The filters as defined in the sections above can be bypassed by setting the PassRxFilter bit in the Command register. When the PassRxFilter bit is set, all receive frames will be passed to memory. In this case the device driver software has to implement all filtering functionality in software. Setting the PassRxFilter bit does not affect the runt frame filtering as defined in the next section.

Runt frames

A frame with less than 64 bytes (or 68 bytes for VLAN frames) is shorter than the minimum Ethernet frame size and therefore considered erroneous; they might be collision fragments. The receive data path automatically filters and discards these runt frames without writing them to memory and using a receive descriptor.

When a runt frame has a correct CRC there is a possibility that it is intended to be useful. The device driver can receive the runt frames with correct CRC by setting the PassRuntFrame bit of the Command register to 1.

17.11 Power management

The Ethernet block supports power management by means of clock switching. All clocks in the Ethernet core can be switched off. If Wake-up on LAN is needed, the rx_clk should not be switched off.

17.12 Wake-up on LAN

Overview

© NXP B.V. 2010. All rights reserved.

The Ethernet block supports power management with remote wake-up over LAN. The host system can be powered down, even including part of the Ethernet block itself, while the Ethernet block continues to listen to packets on the LAN. Appropriately formed packets can be received and recognized by the Ethernet block and used to trigger the host system to wake up from its power-down state.

Wake-up of the system takes effect through an interrupt. When a wake-up event is detected, the WakeupInt bit in the IntStatus register is set. The interrupt status will trigger an interrupt if the corresponding WakeupIntEn bit in the IntEnable register is set. This interrupt should be used by system power management logic to wake up the system.

While in a power-down state the packet that generates a Wake-up on LAN event is lost.

There are two ways in which Ethernet packets can trigger wake-up events: generic Wake-up on LAN and Magic Packet. Magic Packet filtering uses an additional filter for Magic Packet detection. In both cases a Wake-up on LAN event is only triggered if the triggering packet has a valid CRC. Figure 10–23 shows the generation of the wake-up signal.

The RxFilterWoLStatus register can be read by the software to inspect the reason for a Wake-up event. Before going to power-down the power management software should clear the register by writing the RxFilterWolClear register.

NOTE: when entering in power-down mode, a receive frame might be not entirely stored into the Rx buffer. In this situation, after turning exiting power-down mode, the next receive frame is corrupted due to the data of the previous frame being added in front of the last received frame. Software drivers have to reset the receive data path just after exiting power-down mode.

The following subsections describe the two Wake-up on LAN mechanisms.

Filtering for WoL

The receive filter functionality can be used to generate Wake-up on LAN events. If the RxFilterEnWoL bit of the RxFilterCtrl register is set, the receive filter will set the WakeupInt bit of the IntStatus register if a frame is received that passes the filter. The interrupt will only be generated if the CRC of the frame is correct.

Magic Packet WoL

The Ethernet block supports wake-up using Magic Packet technology (see ‘Magic Packet technology’, Advanced Micro Devices). A Magic Packet is a specially formed packet solely intended for wake-up purposes. This packet can be received, analyzed and recognized by the Ethernet block and used to trigger a wake-up event.

A Magic Packet is a packet that contains in its data portion the station address repeated 16 times with no breaks or interruptions, preceded by 6 Magic Packet synchronization bytes with the value 0xFF. Other data may be surrounding the Magic Packet pattern in the data portion of the packet. The whole packet must be a well-formed Ethernet frame.

The magic packet detection unit analyzes the Ethernet packets, extracts the packet address and checks the payload for the Magic Packet pattern. The address from the packet is used for matching the pattern (not the address in the SA0/1/2 registers.) A magic

© NXP B.V. 2010. All rights reserved.

packet only sets the wake-up interrupt status bit if the packet passes the receive filter as illustrated in Figure 10–23: the result of the receive filter is ANDed with the magic packet filter result to produce the result.

Magic Packet filtering is enabled by setting the MagicPacketEnWoL bit of the RxFilterCtrl register. Note that when doing Magic Packet WoL, the RxFilterEnWoL bit in the RxFilterCtrl register should be 0. Setting the RxFilterEnWoL bit to 1 would accept all packets for a matching address, not just the Magic Packets i.e. WoL using Magic Packets is more strict.

When a magic packet is detected, apart from the WakeupInt bit in the IntStatus register, the MagicPacketWoL bit is set in the RxFilterWoLStatus register. Software can reset the bit writing a 1 to the corresponding bit of the RxFilterWoLClear register.

Example: An example of a Magic Packet with station address 0x11 0x22 0x33 0x44 0x55 0x66 is the following (MISC indicates miscellaneous additional data bytes in the packet):

```
<DESTINATION> <SOURCE> <MISC>
FF FF FF FF FF FF
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
<MISC> <CRC>
```

17.13 Enabling and disabling receive and transmit

Enabling and disabling reception

After reset, the receive function of the Ethernet block is disabled. The receive function can be enabled by the device driver setting the RxEnable bit in the Command register and the “RECEIVE ENABLE’ bit in the MAC1 configuration register (in that order).

The status of the receive data path can be monitored by the device driver by reading the RxStatus bit of the Status register. Figure 10–24 illustrates the state machine for the generation of the RxStatus bit.

© NXP B.V. 2010. All rights reserved.

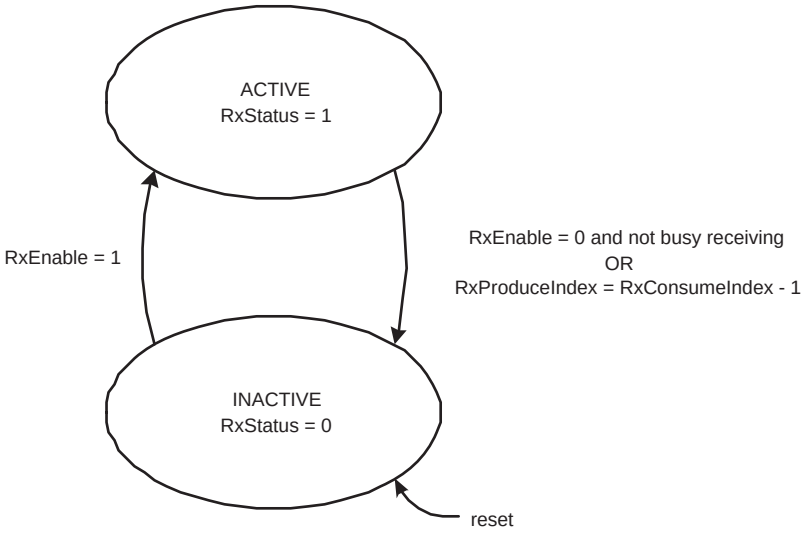


Fig 24. Receive Active/Inactive state machine

After a reset, the state machine is in the INACTIVE state. As soon as the RxEnable bit is set in the Command register, the state machine transitions to the ACTIVE state. As soon as the RxEnable bit is cleared, the state machine returns to the INACTIVE state. If the receive data path is busy receiving a packet while the receive data path gets disabled, the packet will be received completely, stored to memory along with its status before returning to the INACTIVE state. Also if the Receive descriptor array is full, the state machine will return to the INACTIVE state.

For the state machine in Figure 10–24, a soft reset is like a hardware reset assertion, i.e. after a soft reset the receive data path is inactive until the data path is re-enabled.

Enabling and disabling transmission

After reset, the transmit function of the Ethernet block is disabled. The Tx transmit data path can be enabled by the device driver setting the TxEnable bit in the Command register to 1.

The status of the transmit data paths can be monitored by the device driver reading the TxStatus bit of the Status register. Figure 10–25 illustrates the state machine for the generation of the TxStatus bit.

© NXP B.V. 2010. All rights reserved.

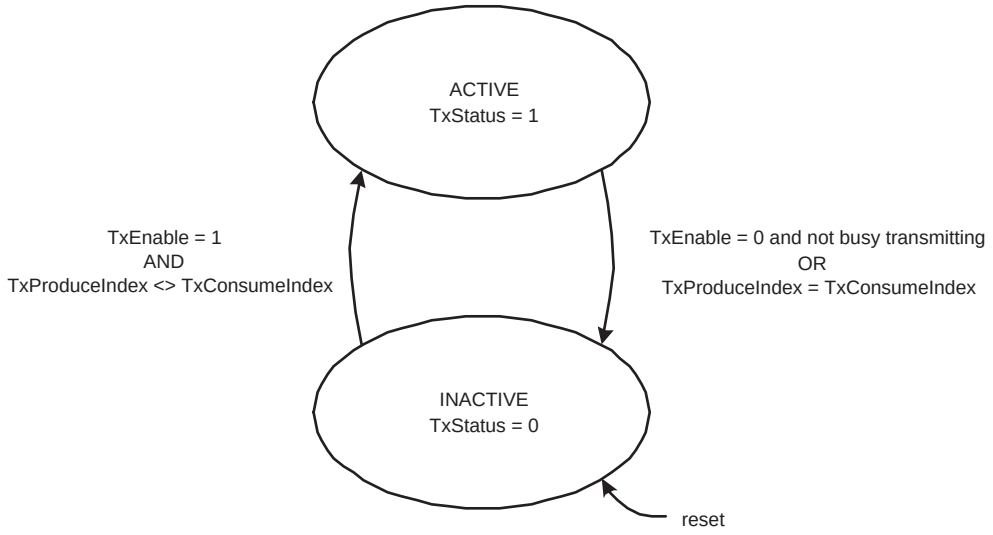


Fig 25. Transmit Active/Inactive state machine

After reset, the state machine is in the INACTIVE state. As soon as the TxEnable bit is set in the Command register and the Produce and Consume indices are not equal, the state machine transitions to the ACTIVE state. As soon as the TxEnable bit is cleared and the transmit data path has completed all pending transmissions, including committing the transmission status to memory, the state machine returns to the INACTIVE state. The state machine will also return to the INACTIVE state if the Produce and Consume indices are equal again i.e. all frames have been transmitted.

For the state machine in Figure 10–25, a soft reset is like a hardware reset assertion, i.e. after a soft reset the transmit data path is inactive until the data path is re-enabled.

17.14 Transmission padding and CRC

In the case of a frame of less than 60 bytes (or 64 bytes for VLAN frames), the Ethernet block can pad the frame to 64 or 68 bytes including a 4 bytes CRC Frame Check Sequence (FCS). Padding is affected by the value of the ‘AUTO DETECT PAD ENABLE’ (ADPEN), ‘VLAN PAD ENABLE’ (VLPEN) and ‘PAD/CRC ENABLE’ (PADEN) bits of the MAC2 configuration register, as well as the Override and Pad bits from the transmit descriptor Control word. CRC generation is affected by the ‘CRC ENABLE’ (CRCE) and ‘DELAYED CRC’ (DCRC) bits of the MAC2 configuration register, and the Override and CRC bits from the transmit descriptor Control word.

The effective pad enable (EPADEN) is equal to the ‘PAD/CRC ENABLE’ bit from the MAC2 register if the Override bit in the descriptor is 0. If the Override bit is 1, then EPADEN will be taken from the descriptor Pad bit. Likewise the effective CRC enable (ECRCE) equals CRCE if the Override bit is 0, otherwise it equal the CRC bit from the descriptor.

If padding is required and enabled, a CRC will always be appended to the padded frames. A CRC will only be appended to the non-padded frames if ECRCE is set.

If EPADEN is 0, the frame will not be padded and no CRC will be added unless ECRCE is set.

© NXP B.V. 2010. All rights reserved.

If EPADEN is 1, then small frames will be padded and a CRC will always be added to the padded frames. In this case if ADPEN and VLPEN are both 0, then the frames will be padded to 60 bytes and a CRC will be added creating 64 bytes frames; if VLPEN is 1, the frames will be padded to 64 bytes and a CRC will be added creating 68 bytes frames; if ADPEN is 1, while VLPEN is 0 VLAN frames will be padded to 64 bytes, non VLAN frames will be padded to 60 bytes, and a CRC will be added to padded frames, creating 64 or 68 bytes padded frames.

If CRC generation is enabled, CRC generation can be delayed by four bytes by setting the DELAYED CRC bit in the MAC2 register, in order to skip proprietary header information.

17.15 Huge frames and frame length checking

The ‘HUGE FRAME ENABLE’ bit in the MAC2 configuration register can be set to 1 to enable transmission and reception of frames of any length. Huge frame transmission can be enabled on a per frame basis by setting the Override and Huge bits in the transmit descriptor Control word.

When enabling huge frames, the Ethernet block will not check frame lengths and report frame length errors (RangeError and LengthError). If huge frames are enabled, the received byte count in the RSV register may be invalid because the frame may exceed the maximum size; the RxSize fields from the receive status arrays will be valid.

Frame lengths are checked by comparing the length/type field of the frame to the actual number of bytes in the frame. A LengthError is reported by setting the corresponding bit in the receive StatusInfo word.

The MAXF register allows the device driver to specify the maximum number of bytes in a frame. The Ethernet block will compare the actual receive frame to the MAXF value and report a RangeError in the receive StatusInfo word if the frame is larger.

17.16 Statistics counters

Generally, Ethernet applications maintain many counters that track Ethernet traffic statistics. There are a number of standards specifying such counters, such as IEEE std 802.3 / clause 30. Other standards are RFC 2665 and RFC 2233.

The approach taken here is that by default all counters are implemented in software. With the help of the StatusInfo field in frame statuses, many of the important statistics events listed in the standards can be counted by software.

17.17 MAC status vectors

Transmit and receive status information as detected by the MAC are available in registers TSV0, TSV1 and RSV so that software can poll them. These registers are normally of limited use because the communication between driver software and the Ethernet block takes place primarily through frame descriptors. Statistical events can be counted by software in the device driver. However, for debug purposes the transmit and receive status vectors are made visible. They are valid as long as the internal status of the MAC is valid and should typically only be read when the transmit and receive processes are halted.

© NXP B.V. 2010. All rights reserved.

17.18 Reset

The Ethernet block has a hard reset input which is connected to the chip reset, as well as several soft resets which can be activated by setting the appropriate bit(s) in registers. All registers in the Ethernet block have a value of 0 after a hard reset, unless otherwise specified.

Hard reset

After a hard reset, all registers will be set to their default value.

Soft reset

Parts of the Ethernet block can be soft reset by setting bits in the Command register and the MAC1 configuration register.The MAC1 register has six different reset bits:

- SOFT RESET: Setting this bit will put all modules in the MAC in reset, except for the MAC registers (at addresses 0x000 to 0x0FC). The value of the soft reset after a hardware reset assertion is 1, i.e. the soft reset needs to be cleared after a hardware reset.
- SIMULATION RESET: Resets the random number generator in the Transmit Function. The value after a hardware reset assertion is 0.
- RESET MCS/Rx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the receive function in the MAC. The value after a hardware reset assertion is 0.
- RESET Rx: Setting this bit will reset the receive function in the MAC. The value after a hardware reset assertion is 0.
- RESET MCS/Tx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the transmit function in the MAC. The value after a hardware reset assertion is 0.
- RESET Tx: Setting this bit will reset the transmit function of the MAC. The value after a hardware reset assertion is 0.

The above reset bits must be cleared by software.

The Command register has three different reset bits:

- TxReset: Writing a ‘1’ to the TxReset bit will reset the transmit data path, excluding the MAC portions, including all (read-only) registers in the transmit data path, as well as the TxProduceIndex register in the host registers module. A soft reset of the transmit data path will abort all AHB transactions of the transmit data path. The reset bit will be cleared autonomously by the Ethernet block. A soft reset of the Tx data path will clear the TxStatus bit in the Status register.
- RxReset: Writing a ‘1’ to the RxReset bit will reset the receive data path, excluding the MAC portions, including all (read-only) registers in the receive data path, as well as the RxConsumeIndex register in the host registers module. A soft reset of the receive data path will abort all AHB transactions of the receive data path. The reset bit will be cleared autonomously by the Ethernet block. A soft reset of the Rx data path will clear the RxStatus bit in the Status register.

© NXP B.V. 2010. All rights reserved.

- RegReset: Resets all of the data paths and registers in the host registers module, excluding the registers in the MAC. A soft reset of the registers will also abort all AHB transactions of the transmit and receive data path. The reset bit will be cleared autonomously by the Ethernet block.

To do a full soft reset of the Ethernet block, device driver software must:

- Set the ‘SOFT RESET’ bit in the MAC1 register to 1.
- Set the RegReset bit in the Command register, this bit clears automatically.
- Re-initialize the MAC registers (0x000 to 0x0FC).
- Reset the ‘SOFT RESET’ bit in the MAC1 register to 0.

To reset just the transmit data path, the device driver software has to:

- Set the ‘RESET MCS/Tx’ bit in the MAC1 register to 1.
- Disable the Tx DMA managers by setting the TxEnable bits in the Command register to 0.
- Set the TxReset bit in the Command register, this bit clears automatically.
- Reset the ‘RESET MCS/Tx’ bit in the MAC1 register to 0.

To reset just the receive data path, the device driver software has to:

- Disable the receive function by resetting the ‘RECEIVE ENABLE’ bit in the MAC1 configuration register and resetting of the RxEnable bit of the Command register.
- Set the ‘RESET MCS/Rx’ bit in the MAC1 register to 1.
- Set the RxReset bit in the Command register, this bit clears automatically.
- Reset the ‘RESET MCS/Rx’ bit in the MAC1 register to 0.

17.19 Ethernet errors

The Ethernet block generates errors for the following conditions:

- A reception can cause an error: AlignmentError, RangeError, LengthError, SymbolError, CRCError, NoDescriptor, or Overrun. These are reported back in the receive StatusInfo and in the interrupt status register (IntStatus).
- A transmission can cause an error: LateCollision, ExcessiveCollision, ExcessiveDefer, NoDescriptor, or Underrun. These are reported back in the transmission StatusInfo and in the interrupt status register (IntStatus).

© NXP B.V. 2010. All rights reserved.

18. AHB bandwidth

The Ethernet block is connected to an AHB bus which must carry all of the data and control information associated with all Ethernet traffic in addition to the CPU accesses required to operate the Ethernet block and deal with message contents.

18.1 DMA access

Assumptions

By making some assumptions, the bandwidth needed for each type of AHB transfer can be calculated and added in order to find the overall bandwidth requirement.

The flexibility of the descriptors used in the Ethernet block allows the possibility of defining memory buffers in a range of sizes. In order to analyze bus bandwidth requirements, some assumptions must be made about these buffers. The "worst case" is not addressed since that would involve all descriptors pointing to single byte buffers, with most of the memory occupied in holding descriptors and very little data. It can easily be shown that the AHB cannot handle the huge amount of bus traffic that would be caused by such a degenerate (and illogical) case.

For this analysis, an Ethernet packet is assumed to consist of a 64 byte frame. Continuous traffic is assumed on both the transmit and receive channels.

This analysis does not reflect the flow of Ethernet traffic over time, which would include inter-packet gaps in both the transmit and receive channels that reduce the bandwidth requirements over a larger time frame.

Types of DMA access and their bandwidth requirements

The interface to an external Ethernet PHY is via RMII. RMII operates at 50 MHz, transferring a byte in 4 clock cycles. The data transfer rate is 12.5 Mbps.

The Ethernet block initiates DMA accesses for the following cases:

- Tx descriptor read:
  - Transmit descriptors occupy 2 words (8 bytes) of memory and are read once for each use of a descriptor.
  - Two word read happens once every 64 bytes (16 words) of transmitted data.
  - This gives 1/8th of the data rate, which = 1.5625 Mbps.
- Rx descriptor read:
  - Receive descriptors occupy 2 words (8 bytes) of memory and are read once for each use of a descriptor.
  - Two word read happens once every 64 bytes (16 words) of received data.
  - This gives 1/8th of the data rate, which = 1.5625 Mbps.
- Tx status write:
  - Transmit status occupies 1 word (4 bytes) of memory and is written once for each use of a descriptor.
  - One word write happens once every 64 bytes (16 words) of transmitted data.
  - This gives 1/16th of the data rate, which = 0.7813 Mbps.

© NXP B.V. 2010. All rights reserved.

- Rx status write:
  - Receive status occupies 2 words (8 bytes) of memory and is written once for each use of a descriptor.
  - Two word write happens once every 64 bytes (16 words) of received data.
  - This gives 1/8 of the data rate, which = 1.5625 Mbps.
- Tx data read:
  - Data transmitted in an Ethernet frame, the size is variable.
  - Basic Ethernet rate = 12.5 Mbps.
- Rx data write:
  - Data to be received in an Ethernet frame, the size is variable.
  - Basic Ethernet rate = 12.5 Mbps.

This gives a total rate of 30.5 Mbps for the traffic generated by the Ethernet DMA function.

18.2 Types of CPU access

- Accesses that mirror each of the DMA access types:
  - All or part of status values must be read, and all or part of descriptors need to be written after each use, transmitted data must be stored in the memory by the CPU, and eventually received data must be retrieved from the memory by the CPU.
  - This gives roughly the same or slightly lower rate as the combined DMA functions, which = 30.5 Mbps.
- Access to registers in the Ethernet block:
  - The CPU must read the RxProduceIndex, TxConsumeIndex, and IntStatus registers, and both read and write the RxConsumeIndex and TxProduceIndex registers.
  - 7 word read/writes once every 64 bytes (16 words) of transmitted and received data.
  - This gives 7/16 of the data rate, which = 5.4688 Mbps.

This gives a total rate of 36 Mbps for the traffic generated by the Ethernet DMA function.

18.3 Overall bandwidth

Overall traffic on the AHB is the sum of DMA access rates and CPU access rates, which comes to approximately 66.5 MB/s.

The peak bandwidth requirement can be somewhat higher due to the use of small memory buffers, in order to hold often used addresses (e.g. the station address) for example. Driver software can determine how to build frames in an efficient manner that does not overutilize the AHB.

The bandwidth available on the AHB bus depends on the system clock frequency. As an example, assume that the system clock is set at 60 MHz. All or nearly all of bus accesses related to the Ethernet will be word transfers. The raw AHB bandwidth can be approximated as 4 bytes per two system clocks, which equals 2 times the system clock rate. With a 60 MHz system clock, the bandwidth is 120 MB/s, giving about 55% utilization

© NXP B.V. 2010. All rights reserved.

for Ethernet traffic during simultaneous transmit and receive operations. This shows that it is not necessary to use the maximum CPU frequency for the Ethernet to work with plenty of bandwidth headroom.

© NXP B.V. 2010. All rights reserved.

19. CRC calculation

The calculation is used for several purposes:

- Generation the FCS at the end of the Ethernet frame.
- Generation of the hash table index for the hash table filtering.
- Generation of the destination and source address hash CRCs.

The C pseudocode function below calculates the CRC on a frame taking the frame (without FCS) and the number of bytes in the frame as arguments. The function returns the CRC as a 32-bit integer.

```
int crc_calc(char frame_no_fcs[], int frame_len) {
     int  i;   // iterator
     int  j;   // another iterator
     char byte;// current byte
     int  crc; // CRC result
     int q0, q1, q2, q3;  // temporary variables
     crc = 0xFFFFFFFF;
     for (i = 0; i < frame_len; i++) {
          byte = *frame_no_fcs++;
          for (j = 0; j < 2; j++) {
               if (((crc >> 28) ^ (byte >> 3)) & 0x00000001) {
                    q3 = 0x04C11DB7;
               } else {
                    q3 = 0x00000000;
               }
               if (((crc >> 29) ^ (byte >> 2)) & 0x00000001) {
                    q2 = 0x09823B6E;
               } else {
                    q2 = 0x00000000;
               }
               if (((crc >> 30) ^ (byte >> 1)) & 0x00000001) {
                    q1 = 0x130476DC;
               } else {
                    q1 = 0x00000000;
               }
               if (((crc >> 31) ^ (byte >> 0)) & 0x00000001) {
                    q0 = 0x2608EDB8;
               } else {
                    q0 = 0x00000000;
               }
               crc = (crc << 4) ^ q3 ^ q2 ^ q1 ^ q0;
               byte >>= 4;
          }
     }
     return crc;
}
```

For FCS calculation, this function is passed a pointer to the first byte of the frame and the length of the frame without the FCS.

© NXP B.V. 2010. All rights reserved.

For hash filtering, this function is passed a pointer to the destination address part of the frame and the CRC is only calculated on the 6 address bytes. The hash filter uses bits [28:23] for indexing the 64-bits { HashFilterH, HashFilterL } vector. If the corresponding bit is set the packet is passed, otherwise it is rejected by the hash filter.

For obtaining the destination and source address hash CRCs, this function calculates first both the 32-bit CRCs, then the nine most significant bits from each 32-bit CRC are extracted, concatenated, and written in every StatusHashCRC word of every fragment status.

© NXP B.V. 2010. All rights reserved.

1. How to read this chapter

This chapter describes the USB controller which is present on all LPC17xx devices except the LPC1767. On some LPC17xx family devices, the USB controller can also be configured for Host or OTG operation.

2. Basic configuration

The USB controller is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCUSB.

   Remark: On reset, the USB block is disabled (PCUSB = 0).
2. Clock: The USB block can be used with a dedicated USB PLL (PLL1) to obtain the USB clock or with the Main PLL (PLL0). See Section 4–6.1.
3. Pins: Select USB pins and their modes in PINSEL0 to PINSEL5 and PINMODE0 to PINMODE5 (Section 8–5).
4. Wake-up: Activity on the USB bus port can wake up the microcontroller from Power-down mode, see Section 4–8.8.
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. Initialization: see Section 11–13.

3. Introduction

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 1 ms frames. Each frame contains a Start-Of-Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 16 logical or 32 physical endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device. Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the rate of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

For more information on the Universal Serial Bus, see the USB Implementers Forum website.

The USB device controller on the LPC17xx enables full-speed (12 Mb/s) data exchange with a USB host controller.

© NXP B.V. 2010. All rights reserved.

Table 183. USB related acronyms, abbreviations, and definitions used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-performance bus
ATLE	Auto Transfer Length Extraction
ATX	Analog Transceiver
DD	DMA Descriptor
DDP	DMA Description Pointer
DMA	Direct Memory Access
EOP	End-Of-Packet
EP	Endpoint
EP_RAM	Endpoint RAM
FS	Full Speed
LED	Light Emitting Diode
LS	Low Speed
MPS	Maximum Packet Size
NAK	Negative Acknowledge
PLL	Phase Locked Loop
RAM	Random Access Memory
SOF	Start-Of-Frame
SIE	Serial Interface Engine
SRAM	Synchronous RAM
UDCA	USB Device Communication Area
USB	Universal Serial Bus

4. Features

- Fully compliant with the USB 2.0 specification (full speed).
- Supports 32 physical (16 logical) endpoints.
- Supports Control, Bulk, Interrupt and Isochronous endpoints.
- Scalable realization of endpoints at run time.
- Endpoint maximum packet size selection (up to USB maximum specification) by software at run time.
- Supports SoftConnect and GoodLink features.
- Supports DMA transfers on all non-control endpoints.
- Allows dynamic switching between CPU controlled and DMA modes.
- Double buffer implementation for Bulk and Isochronous endpoints.

5. Fixed endpoint configuration

Table 11–184 shows the supported endpoint configurations. Endpoints are realized and configured at run time using the Endpoint realization registers, documented in Section 11–10.4 “Endpoint realization registers”.

© NXP B.V. 2010. All rights reserved.

Table 184. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Packet size (bytes)	Double buffer
0	0	Control	Out	8, 16, 32, 64	No
0	1	Control	In	8, 16, 32, 64	No
1	2	Interrupt	Out	1 to 64	No
1	3	Interrupt	In	1 to 64	No
2	4	Bulk	Out	8, 16, 32, 64	Yes
2	5	Bulk	In	8, 16, 32, 64	Yes
3	6	Isochronous	Out	1 to 1023	Yes
3	7	Isochronous	In	1 to 1023	Yes
4	8	Interrupt	Out	1 to 64	No
4	9	Interrupt	In	1 to 64	No
5	10	Bulk	Out	8, 16, 32, 64	Yes
5	11	Bulk	In	8, 16, 32, 64	Yes
6	12	Isochronous	Out	1 to 1023	Yes
6	13	Isochronous	In	1 to 1023	Yes
7	14	Interrupt	Out	1 to 64	No
7	15	Interrupt	In	1 to 64	No
8	16	Bulk	Out	8, 16, 32, 64	Yes
8	17	Bulk	In	8, 16, 32, 64	Yes
9	18	Isochronous	Out	1 to 1023	Yes
9	19	Isochronous	In	1 to 1023	Yes
10	20	Interrupt	Out	1 to 64	No
10	21	Interrupt	In	1 to 64	No
11	22	Bulk	Out	8, 16, 32, 64	Yes
11	23	Bulk	In	8, 16, 32, 64	Yes
12	24	Isochronous	Out	1 to 1023	Yes
12	25	Isochronous	In	1 to 1023	Yes
13	26	Interrupt	Out	1 to 64	No
13	27	Interrupt	In	1 to 64	No
14	28	Bulk	Out	8, 16, 32, 64	Yes
14	29	Bulk	In	8, 16, 32, 64	Yes
15	30	Bulk	Out	8, 16, 32, 64	Yes
15	31	Bulk	In	8, 16, 32, 64	Yes

6. Functional description

The architecture of the USB device controller is shown below in Figure 11–26.

© NXP B.V. 2010. All rights reserved.

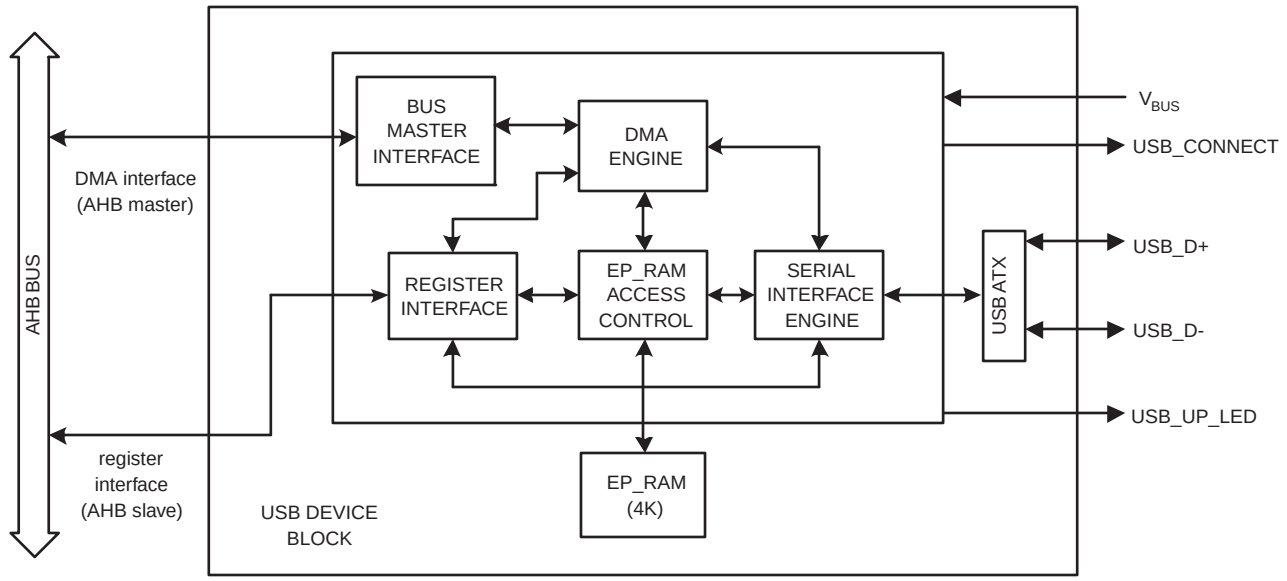


Fig 26. USB device controller block diagram

6.1 Analog transceiver

The USB Device Controller has a built-in analog transceiver (ATX). The USB ATX sends/receives the bi-directional D+ and D- signals of the USB bus.

6.2 Serial Interface Engine (SIE)

The SIE implements the full USB protocol layer. It is completely hardwired for speed and needs no firmware intervention. It handles transfer of data between the endpoint buffers in EP_RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

6.3 Endpoint RAM (EP_RAM)

Each endpoint buffer is implemented as an SRAM based FIFO. The SRAM dedicated for this purpose is called the EP_RAM. Each realized endpoint has a reserved space in the EP_RAM. The total EP_RAM space required depends on the number of realized endpoints, the maximum packet size of the endpoint, and whether the endpoint supports double buffering.

6.4 EP_RAM access control

The EP_RAM Access Control logic handles transfer of data from/to the EP_RAM and the three sources that can access it: the CPU (via the Register Interface), the SIE, and the DMA Engine.

© NXP B.V. 2010. All rights reserved.

6.5 DMA engine and bus master interface

When enabled for an endpoint, the DMA Engine transfers data between RAM on the AHB bus and the endpoint's buffer in EP_RAM. A single DMA channel is shared between all endpoints. When transferring data, the DMA Engine functions as a master on the AHB bus through the bus master interface.

6.6 Register interface

The Register Interface allows the CPU to control the operation of the USB Device Controller. It also provides a way to write transmit data to the controller and read receive data from the controller.

6.7 SoftConnect

The connection to the USB is accomplished by bringing D+ (for a full-speed device) HIGH through a 1.5 kOhm pull-up resistor. The SoftConnect feature can be used to allow software to finish its initialization sequence before deciding to establish connection to the USB. Re-initialization of the USB bus connection can also be performed without having to unplug the cable.

To use the SoftConnect feature, the CONNECT signal should control an external switch that connects the 1.5 kOhm resistor between D+ and +3.3V. Software can then control the CONNECT signal by writing to the CON bit using the SIE Set Device Status command.

6.8 GoodLink

Good USB connection indication is provided through GoodLink technology. When the device is successfully enumerated and configured, the LED indicator will be permanently ON. During suspend, the LED will be OFF.

This feature provides a user-friendly indicator on the status of the USB device. It is a useful field diagnostics tool to isolate faulty equipment.

To use the GoodLink feature the UP_LED signal should control an LED. The UP_LED signal is controlled using the SIE Configure Device command.

7. Operational overview

Transactions on the USB bus transfer data between device endpoints and the host. The direction of a transaction is defined with respect to the host. OUT transactions transfer data from the host to the device. IN transactions transfer data from the device to the host. All transactions are initiated by the host controller.

For an OUT transaction, the USB ATX receives the bi-directional D+ and D- signals of the USB bus. The Serial Interface Engine (SIE) receives the serial data from the ATX and converts it into a parallel data stream. The parallel data is written to the corresponding endpoint buffer in the EP_RAM.

For IN transactions, the SIE reads the parallel data from the endpoint buffer in EP_RAM, converts it into serial data, and transmits it onto the USB bus using the USB ATX.

© NXP B.V. 2010. All rights reserved.

Once data has been received or sent, the endpoint buffer can be read or written. How this is accomplished depends on the endpoint's type and operating mode. The two operating modes for each endpoint are Slave (CPU-controlled) mode, and DMA mode.

In Slave mode, the CPU transfers data between RAM and the endpoint buffer using the Register Interface. See Section 11–14 "Slave mode operation" for a detailed description of this mode.

In DMA mode, the DMA transfers data between RAM and the endpoint buffer. See Section 11–15 "DMA operation" for a detailed description of this mode.

8. Pin description

Table 185. USB external interface

Name	Direction	Description
V_{BUS}	I	V_{BUS} status input. When this function is not enabled via its corresponding PINSEL register, it is driven HIGH internally.
USB_CONNECT	O	SoftConnect control signal.
USB_UP_LED	O	GoodLink LED control signal.
USB_D+	I/O	Positive differential data.
USB_D-	I/O	Negative differential data.

9. Clocking and power management

This section describes the clocking and power management features of the USB Device Controller.

9.1 Power requirements

The USB protocol insists on power management by the device. This becomes very critical if the device draws power from the bus (bus-powered device). The following constraints should be met by a bus-powered device:

1. A device in the non-configured state should draw a maximum of 100 mA from the bus.
2. A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
3. A suspended device can draw a maximum of 2.5 mA.

9.2 Clocks

The USB device controller clocks are shown in Table 11–186

© NXP B.V. 2010. All rights reserved.

Table 186. USB device controller clock sources

Clock source	Description
AHB master clock	Clock for the AHB master bus interface and DMA
AHB slave clock	Clock for the AHB slave interface
usbclk	48 MHz clock from the dedicated USB PLL (PLL1) or the Main PLL (PLL0), used to recover the 12 MHz clock from the USB bus

9.3 Power management support

To help conserve power, the USB device controller automatically disables the AHB master clock and usbclk when not in use.

When the USB Device Controller goes into the suspend state (bus is idle for 3 ms), the usbclk input to the device controller is automatically disabled, helping to conserve power. However, if software wishes to access the device controller registers, usbclk must be active. To allow access to the device controller registers while in the suspend state, the USBClkCtrl and USBClkSt registers are provided.

When software wishes to access the device controller registers, it should first ensure usbclk is enabled by setting DEV_CLK_EN in the USBClkCtrl register, and then poll the corresponding DEV_CLK_ON bit in USBClkSt until set. Once set, usbclk will remain enabled until DEV_CLK_EN is cleared by software.

When a DMA transfer occurs, the device controller automatically turns on the AHB master clock. Once asserted, it remains active for a minimum of 2 ms (2 frames), to help ensure that DMA throughput is not affected by turning off the AHB master clock. 2 ms after the last DMA access, the AHB master clock is automatically disabled to help conserve power. If desired, software also has the capability of forcing this clock to remain enabled using the USBClkCtrl register.

Note that the AHB slave clock is always enabled as long as the PCUSB bit of PCONP is set. When the device controller is not in use, all of the device controller clocks may be disabled by clearing PCUSB.

The USB_NEED_CLK signal is used to facilitate going into and waking up from chip Power-down mode. USB_NEED_CLK is asserted if any of the bits of the USBClkSt register are asserted.

After entering the suspend state with DEV_CLK_EN and AHB_CLK_EN cleared, the DEV_CLK_ON and AHB_CLK_ON will be cleared when the corresponding clock turns off. When both bits are zero, USB_NEED_CLK will be low, indicating that the chip can be put into Power-down mode by writing to the PCON register. The status of USB_NEED_CLK can be read from the USBIntSt register.

Any bus activity in the suspend state will cause the USB_NEED_CLK signal to be asserted. When the chip is in Power-down mode and the USB interrupt is enabled, the assertion of USB_NEED_CLK causes the chip to wake up from Power-down mode.

© NXP B.V. 2010. All rights reserved.

9.4 Remote wake-up

The USB device controller supports software initiated remote wake-up. Remote wake-up involves resume signaling on the USB bus initiated from the device. This is done by clearing the SUS bit in the SIE Set Device Status register. Before writing into the register, all the clocks to the device controller have to be enabled using the USBClkCtrl register.

10. Register description

Table 11–187 shows the USB Device Controller registers directly accessible by the CPU. The Serial Interface Engine (SIE) has other registers that are indirectly accessible via the SIE command registers. See Section 11–12 “Serial interface engine command description” for more info.

Table 187. USB device register map

Name	Description	Access	Reset value[1]	Address
Clock control registers				
USBClkCtrl	USB Clock Control	R/W	0	0x5000 CFF4
USBClkSt	USB Clock Status	RO	0	0x5000 CFF8
Device interrupt registers				
USBIntSt	USB Interrupt Status	R/W	0x8000 0000	0x400F C1C0
USBDevIntSt	USB Device Interrupt Status	RO	0x10	0x5000 C200
USBDevIntEn	USB Device Interrupt Enable	R/W	0	0x5000 C204
USBDevIntClr	USB Device Interrupt Clear	WO	0	0x5000 C208
USBDevIntSet	USB Device Interrupt Set	WO	0	0x5000 C20C
USBDevIntPri	USB Device Interrupt Priority	WO	0	0x5000 C22C
Endpoint interrupt registers				
USBEpIntSt	USB Endpoint Interrupt Status	RO	0	0x5000 C230
USBEpIntEn	USB Endpoint Interrupt Enable	R/W	0	0x5000 C234
USBEpIntClr	USB Endpoint Interrupt Clear	WO	0	0x5000 C238
USBEpIntSet	USB Endpoint Interrupt Set	WO	0	0x5000 C23C
USBEpIntPri	USB Endpoint Priority	WO[2]	0	0x5000 C240
Endpoint realization registers				
USBReEp	USB Realize Endpoint	R/W	0x3	0x5000 C244
USBEpInd	USB Endpoint Index	WO[2]	0	0x5000 C248
USBMaxPSize	USB MaxPacketSize	R/W	0x8	0x5000 C24C
USB transfer registers				
USBRxData	USB Receive Data	RO	0	0x5000 C218
USBRxPLen	USB Receive Packet Length	RO	0	0x5000 C220
USBTxData	USB Transmit Data	WO[2]	0	0x5000 C21C
USBTxPLen	USB Transmit Packet Length	WO[2]	0	0x5000 C224
USBCtrl	USB Control	R/W	0	0x5000 C228
SIE Command registers				
USBCmdCode	USB Command Code	WO[2]	0	0x5000 C210
USBCmdData	USB Command Data	RO	0	0x5000 C214

© NXP B.V. 2010. All rights reserved.

Table 187. USB device register map

Name	Description	Access	Reset value[1]	Address
DMA registers				
USBDMARSt	USB DMA Request Status	RO	0	0x5000 C250
USBDMARClr	USB DMA Request Clear	WO[2]	0	0x5000 C254
USBDMARSet	USB DMA Request Set	WO[2]	0	0x5000 C258
USBUDCAH	USB UDCA Head	R/W	0	0x5000 C280
USBEpDMASt	USB Endpoint DMA Status	RO	0	0x5000 C284
USBEpDMAEn	USB Endpoint DMA Enable	WO[2]	0	0x5000 C288
USBEpDMADis	USB Endpoint DMA Disable	WO[2]	0	0x5000 C28C
USBDMAIntSt	USB DMA Interrupt Status	RO	0	0x5000 C290
USBDMAIntEn	USB DMA Interrupt Enable	R/W	0	0x5000 C294
USBEoTIntSt	USB End of Transfer Interrupt Status	RO	0	0x5000 C2A0
USBEoTIntClr	USB End of Transfer Interrupt Clear	WO[2]	0	0x5000 C2A4
USBEoTIntSet	USB End of Transfer Interrupt Set	WO[2]	0	0x5000 C2A8
USBNDDRIntSt	USB New DD Request Interrupt Status	RO	0	0x5000 C2AC
USBNDDRIntClr	USB New DD Request Interrupt Clear	WO[2]	0	0x5000 C2B0
USBNDDRIntSet	USB New DD Request Interrupt Set	WO[2]	0	0x5000 C2B4
USBSysErrIntSt	USB System Error Interrupt Status	RO	0	0x5000 C2B8
USBSysErrIntClr	USB System Error Interrupt Clear	WO[2]	0	0x5000 C2BC
USBSysErrIntSet	USB System Error Interrupt Set	WO[2]	0	0x5000 C2C0

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

[2] Reading WO register will return an invalid value.

10.1 Clock control registers

10.1.1 USB Clock Control register (USBClkCtrl - 0x5000 CFF4)

This register controls the clocking of the USB Device Controller. Whenever software wants to access the device controller registers, both DEV_CLK_EN and AHB_CLK_EN must be set. The PORTSEL_CLK_EN bit need only be set when accessing the OTGStCtrl register (see Section 13–8.6) when the USB is used in OTG configuration.

The software does not have to repeat this exercise for every register access, provided that the corresponding USBClkCtrl bits are already set. Note that this register is functional only when the PCUSB bit of PCONP is set; when PCUSB is cleared, all clocks to the device controller are disabled irrespective of the contents of this register. USBClkCtrl is a read/write register.

Table 188. USBClkCtrl register (USBClkCtrl - address 0x5000 CFF4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	DEV_CLK_EN	Device clock enable. Enables the usbclk input to the device controller	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

Table 188. USBClkCtrl register (USBClkCtrl - address 0x5000 CFF4) bit description

Bit	Symbol	Description	Reset value
3	PORTSEL_CLK_EN	Port select register clock enable.	NA
4	AHB_CLK_EN	AHB clock enable	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.1.2 USB Clock Status register (USBClkSt - 0x5000 CFF8)

This register holds the clock availability status. The bits of this register are ORed together to form the USB_NEED_CLK signal. When enabling a clock via USBClkCtrl, software should poll the corresponding bit in USBClkSt. If it is set, then software can go ahead with the register access. Software does not have to repeat this exercise for every access, provided that the USBClkCtrl bits are not disturbed. USBClkSt is a read-only register.

Table 189. USB Clock Status register (USBClkSt - address 0x5000 CFF8) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	DEV_CLK_ON	Device clock on. The usbclk input to the device controller is active.	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PORTSEL_CLK_ON	Port select register clock on.	NA
4	AHB_CLK_ON	AHB clock on.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.2 Device interrupt registers

10.2.1 USB Interrupt Status register (USBIntSt - 0x5000 C1C0)

The USB Device Controller has three interrupt lines. This register allows software to determine their status with a single read operation. All three interrupt lines are ORed together to a single channel of the vectored interrupt controller. This register also contains the USB_NEED_CLK status and EN_USB_INTS control bits. USBIntSt is a read/write register.

Table 190. USB Interrupt Status register (USBIntSt - address 0x5000 C1C0) bit description

Bit	Symbol	Description	Reset value
0	USB_INT_REQ_LP	Low priority interrupt line status. This bit is read-only.	0
1	USB_INT_REQ_HP	High priority interrupt line status. This bit is read-only.	0
2	USB_INT_REQ_DMA	DMA interrupt line status. This bit is read-only.	0
7:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

Table 190. USB Interrupt Status register (USBIntSt - address 0x5000 C1C0) bit description

Bit	Symbol	Description	Reset value
8	USB_NEED_CLK	USB need clock indicator. This bit is set to 1 when USB activity or a change of state on the USB data pins is detected, and it indicates that a PLL supplied clock of 48 MHz is needed. Once USB_NEED_CLK becomes one, it resets to zero 5 ms after the last packet has been received/sent, or 2 ms after the Suspend Change (SUS_CH) interrupt has occurred. A change of this bit from 0 to 1 can wake up the microcontroller if activity on the USB bus is selected to wake up the part from the Power-down mode (see Section 4–8.8 "Wake-up from Reduced Power Modes" for details). Also see Section 4–5.9 "PLL0 and Power-down mode" and Section 4–8.9 "Power Control for Peripherals register (PCONP - 0x400F C0C4)" for considerations about the PLL and invoking the Power-down mode. This bit is read-only.	1
30:9	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31	EN_USB_INTS	Enable all USB interrupts. When this bit is cleared, the Vectored Interrupt Controller does not see the ORed output of the USB interrupt lines.	1

10.2.2 USB Device Interrupt Status register (USBDevIntSt - 0x5000 C200)

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

Table 191. USB Device Interrupt Status register (USBDevIntSt - address 0x5000 C200) bit allocation

Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 192. USB Device Interrupt Status register (USBDevIntSt - address 0x5000 C200) bit description

Bit	Symbol	Description	Reset value
0	FRAME	The frame interrupt occurs every 1 ms. This is used in isochronous packet transfers.	0
1	EP_FAST	Fast endpoint interrupt. If an Endpoint Interrupt Priority register (USBEpIntPri) bit is set, the corresponding endpoint interrupt will be routed to this bit.	0
2	EP_SLOW	Slow endpoints interrupt. If an Endpoint Interrupt Priority Register (USBEpIntPri) bit is not set, the corresponding endpoint interrupt will be routed to this bit.	0
3	DEV_STAT	Set when USB Bus reset, USB suspend change or Connect change event occurs. Refer to Section 11–12.6 "Set Device Status (Command: 0xFE, Data: write 1 byte)" on page 245.	0
4	CCEMPTY	The command code register (USBCmdCode) is empty (New command can be written).	1
5	CDFULL	Command data register (USBCmdData) is full (Data can be read now).	0
6	RxENDPKT	The current packet in the endpoint buffer is transferred to the CPU.	0
7	TxENDPKT	The number of data bytes transferred to the endpoint buffer equals the number of bytes programmed in the TxPacket length register (USBTxPLen).	0

© NXP B.V. 2010. All rights reserved.

Table 192. USB Device Interrupt Status register (USBDevIntSt - address 0x5000 C200) bit description

Bit	Symbol	Description	Reset value
8	EP_RLZED	Endpoints realized. Set when Realize Endpoint register (USBReEp) or MaxPacketSize register (USBMaxPSize) is updated and the corresponding operation is completed.	0
9	ERR_INT	Error Interrupt. Any bus error interrupt from the USB device. Refer to Section 11–12.9 "Read Error Status (Command: 0xFB, Data: read 1 byte)" on page 247	0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.2.3 USB Device Interrupt Enable register (USBDevIntEn - 0x5000 C204)

Writing a one to a bit in this register enables the corresponding bit in USBDevIntSt to generate an interrupt on one of the interrupt lines when set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri. USBDevIntEn is a read/write register.

Table 193. USB Device Interrupt Enable register (USBDevIntEn - address 0x5000 C204) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 194. USB Device Interrupt Enable register (USBDevIntEn - address 0x5000 C204) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBDevIntEn bit allocation table above	0	No interrupt is generated.	0
		1	An interrupt will be generated when the corresponding bit in the Device Interrupt Status (USBDevIntSt) register (Table 11–191) is set. By default, the interrupt is routed to the USB_INT_REQ_LP interrupt line. Optionally, either the EP_FAST or FRAME interrupt may be routed to the USB_INT_REQ_HP interrupt line by changing the value of USBDevIntPri.	

10.2.4 USB Device Interrupt Clear register (USBDevIntClr - 0x5000 C208)

Writing one to a bit in this register clears the corresponding bit in USBDevIntSt. Writing a zero has no effect.

Remark: Before clearing the EP_SLOW or EP_FAST interrupt bits, the corresponding endpoint interrupts in USBEpIntSt should be cleared.

USBDevIntClr is a write-only register.

Table 195. USB Device Interrupt Clear register (USBDevIntClr - address 0x5000 C208) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-

© NXP B.V. 2010. All rights reserved.

Bit	23	22	21	20	19	18	17	16
Symbol	-	-	-	-	-	-	-	-
Bit	15	14	13	12	11	10	9	8
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	7	6	5	4	3	2	1	0
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 196. USB Device Interrupt Clear register (USBDevIntClr - address 0x5000 C208) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBDevIntClr bit allocation table above	0	No effect.	0
		1	The corresponding bit in USBDevIntSt (Section 11–10.2.2) is cleared.	

10.2.5 USB Device Interrupt Set register (USBDevIntSet - 0x5000 C20C)

Writing one to a bit in this register sets the corresponding bit in the USBDevIntSt. Writing a zero has no effect

USBDevIntSet is a write-only register.

Table 197. USB Device Interrupt Set register (USBDevIntSet - address 0x5000 C20C) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Bit	23	22	21	20	19	18	17	16
Symbol	-	-	-	-	-	-	-	-
Bit	15	14	13	12	11	10	9	8
Symbol	-	-	-	-	-	-	ERR_INT	EP_RLZED
Bit	7	6	5	4	3	2	1	0
Symbol	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

Table 198. USB Device Interrupt Set register (USBDevIntSet - address 0x5000 C20C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBDevIntSet bit allocation table above	0	No effect.	0
		1	The corresponding bit in USBDevIntSt (Section 11–10.2.2) is set.	

10.2.6 USB Device Interrupt Priority register (USBDevIntPri - 0x5000 C22C)

Writing one to a bit in this register causes the corresponding interrupt to be routed to the USB_INT_REQ_HP interrupt line. Writing zero causes the interrupt to be routed to the USB_INT_REQ_LP interrupt line. Either the EP_FAST or FRAME interrupt can be routed to USB_INT_REQ_HP, but not both. If the software attempts to set both bits to one, no interrupt will be routed to USB_INT_REQ_HP. USBDevIntPri is a write-only register.

© NXP B.V. 2010. All rights reserved.

Table 199. USB Device Interrupt Priority register (USBDevIntPri - address 0x5000 C22C) bit description

Bit	Symbol	Value	Description	Reset value
0	FRAME	0	FRAME interrupt is routed to USB_INT_REQ_LP.	0
		1	FRAME interrupt is routed to USB_INT_REQ_HP.	
1	EP_FAST	0	EP_FAST interrupt is routed to USB_INT_REQ_LP.	0
		1	EP_FAST interrupt is routed to USB_INT_REQ_HP.	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.3 Endpoint interrupt registers

The registers in this group facilitate handling of endpoint interrupts. Endpoint interrupts are used in Slave mode operation.

10.3.1 USB Endpoint Interrupt Status register (USBEpIntSt - 0x5000 C230)

Each physical non-isochronous endpoint is represented by a bit in this register to indicate that it has generated an interrupt. All non-isochronous OUT endpoints generate an interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet is successfully transmitted, or when a NAK handshake is sent on the bus and the interrupt on NAK feature is enabled (see Section 11–12.3 “Set Mode (Command: 0xF3, Data: write 1 byte)” on page 244). A bit set to one in this register causes either the EP_FAST or EP_SLOW bit of USBDevIntSt to be set depending on the value of the corresponding bit of USBEpDevIntPri. USBEpIntSt is a read-only register.

Note that for Isochronous endpoints, handling of packet data is done when the FRAME interrupt occurs.

Table 200. USB Endpoint Interrupt Status register (USBEpIntSt - address 0x5000 C230) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 201. USB Endpoint Interrupt Status register (USBEpIntSt - address 0x5000 C230) bit description

Bit	Symbol	Description	Reset value
0	EP0RX	Endpoint 0, Data Received Interrupt bit.	0
1	EP0TX	Endpoint 0, Data Transmitted Interrupt bit or sent a NAK.	0
2	EP1RX	Endpoint 1, Data Received Interrupt bit.	0
3	EP1TX	Endpoint 1, Data Transmitted Interrupt bit or sent a NAK.	0
4	EP2RX	Endpoint 2, Data Received Interrupt bit.	0
5	EP2TX	Endpoint 2, Data Transmitted Interrupt bit or sent a NAK.	0

© NXP B.V. 2010. All rights reserved.

Table 201. USB Endpoint Interrupt Status register (USBEpIntSt - address 0x5000 C230) bit description

Bit	Symbol	Description	Reset value
6	EP3RX	Endpoint 3, Isochronous endpoint.	NA
7	EP3TX	Endpoint 3, Isochronous endpoint.	NA
8	EP4RX	Endpoint 4, Data Received Interrupt bit.	0
9	EP4TX	Endpoint 4, Data Transmitted Interrupt bit or sent a NAK.	0
10	EP5RX	Endpoint 5, Data Received Interrupt bit.	0
11	EP5TX	Endpoint 5, Data Transmitted Interrupt bit or sent a NAK.	0
12	EP6RX	Endpoint 6, Isochronous endpoint.	NA
13	EP6TX	Endpoint 6, Isochronous endpoint.	NA
14	EP7RX	Endpoint 7, Data Received Interrupt bit.	0
15	EP7TX	Endpoint 7, Data Transmitted Interrupt bit or sent a NAK.	0
16	EP8RX	Endpoint 8, Data Received Interrupt bit.	0
17	EP8TX	Endpoint 8, Data Transmitted Interrupt bit or sent a NAK.	0
18	EP9RX	Endpoint 9, Isochronous endpoint.	NA
19	EP9TX	Endpoint 9, Isochronous endpoint.	NA
20	EP10RX	Endpoint 10, Data Received Interrupt bit.	0
21	EP10TX	Endpoint 10, Data Transmitted Interrupt bit or sent a NAK.	0
22	EP11RX	Endpoint 11, Data Received Interrupt bit.	0
23	EP11TX	Endpoint 11, Data Transmitted Interrupt bit or sent a NAK.	0
24	EP12RX	Endpoint 12, Isochronous endpoint.	NA
25	EP12TX	Endpoint 12, Isochronous endpoint.	NA
26	EP13RX	Endpoint 13, Data Received Interrupt bit.	0
27	EP13TX	Endpoint 13, Data Transmitted Interrupt bit or sent a NAK.	0
28	EP14RX	Endpoint 14, Data Received Interrupt bit.	0
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

10.3.2 USB Endpoint Interrupt Enable register (USBEpIntEn - 0x5000 C234)

Setting a bit to 1 in this register causes the corresponding bit in USBEpIntSt to be set when an interrupt occurs for the associated endpoint. Setting a bit to 0 causes the corresponding bit in USBDMARSt to be set when an interrupt occurs for the associated endpoint. USBEpIntEn is a read/write register.

Table 202. USB Endpoint Interrupt Enable register (USBEpIntEn - address 0x5000 C234) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX

© NXP B.V. 2010. All rights reserved.

Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 203. USB Endpoint Interrupt Enable register (USBEpIntEn - address 0x5000 C234) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntEn bit allocation table above	0	The corresponding bit in USBDMARSt is set when an interrupt occurs for this endpoint.	0
		1	The corresponding bit in USBEpIntSt is set when an interrupt occurs for this endpoint. Implies Slave mode for this endpoint.	

10.3.3 USB Endpoint Interrupt Clear register (USBEpIntClr - 0x5000 C238)

Writing a one to this a bit in this register causes the SIE Select Endpoint/Clear Interrupt command to be executed (Table 11–247) for the corresponding physical endpoint. Writing zero has no effect. Before executing the Select Endpoint/Clear Interrupt command, the CDFULL bit in USBDevIntSt is cleared by hardware. On completion of the command, the CDFULL bit is set, USBCmdData contains the status of the endpoint, and the corresponding bit in USBEpIntSt is cleared.

Notes:

- When clearing interrupts using USBEpIntClr, software should wait for CDFULL to be set to ensure the corresponding interrupt has been cleared before proceeding.
- While setting multiple bits in USBEpIntClr simultaneously is possible, it is not recommended; only the status of the endpoint corresponding to the least significant interrupt bit cleared will be available at the end of the operation.
- Alternatively, the SIE Select Endpoint/Clear Interrupt command can be directly invoked using the SIE command registers, but using USBEpIntClr is recommended because of its ease of use.

Each physical endpoint has its own reserved bit in this register. The bit field definition is the same as that of USBEpIntSt shown in Table 11–200. USBEpIntClr is a write-only register.

Table 204. USB Endpoint Interrupt Clear register (USBEpIntClr - address 0x5000 C238) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

© NXP B.V. 2010. All rights reserved.

Table 205. USB Endpoint Interrupt Clear register (USBEpIntClr - address 0x5000 C238) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntClr bit allocation table above	0	No effect.	0
		1	Clears the corresponding bit in USBEpIntSt, by executing the SIE Select Endpoint/Clear Interrupt command for this endpoint.	

10.3.4 USB Endpoint Interrupt Set register (USBEpIntSet - 0x5000 C23C)

Writing a one to a bit in this register sets the corresponding bit in USBEpIntSt. Writing zero has no effect. Each endpoint has its own bit in this register. USBEpIntSet is a write-only register.

Table 206. USB Endpoint Interrupt Set register (USBEpIntSet - address 0x5000 C23C) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

Table 207. USB Endpoint Interrupt Set register (USBEpIntSet - address 0x5000 C23C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntSet bit allocation table above	0	No effect.	0
		1	Sets the corresponding bit in USBEpIntSt.	

10.3.5 USB Endpoint Interrupt Priority register (USBEpIntPri - 0x5000 C240)

This register determines whether an endpoint interrupt is routed to the EP_FAST or EP_SLOW bits of USBDevIntSt. If a bit in this register is set to one, the interrupt is routed to EP_FAST, if zero it is routed to EP_SLOW. Routing of multiple endpoints to EP_FAST or EP_SLOW is possible.

Note that the USBDevIntPri register determines whether the EP_FAST interrupt is routed to the USB_INT_REQ_HP or USB_INT_REQ_LP interrupt line.

USBEpIntPri is a write-only register.

Table 208. USB Endpoint Interrupt Priority register (USBEpIntPri - address 0x5000 C240) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP15TX	EP15RX	EP14TX	E14RX	EP13TX	EP13RX	EP12TX	EP12RX
Bit	23	22	21	20	19	18	17	16
Symbol	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
Bit	15	14	13	12	11	10	9	8
Symbol	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
Bit	7	6	5	4	3	2	1	0
Symbol	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

© NXP B.V. 2010. All rights reserved.

Table 209. USB Endpoint Interrupt Priority register (USBEpIntPri - address 0x5000 C240) bit description

Bit	Symbol	Value	Description	Reset value
31:0	See USBEpIntPri bit allocation table above	0	The corresponding interrupt is routed to the EP_SLOW bit of USBDevIntSt	0
		1	The corresponding interrupt is routed to the EP_FAST bit of USBDevIntSt	

10.4 Endpoint realization registers

The registers in this group allow realization and configuration of endpoints at run time.

10.4.1 EP RAM requirements

The USB device controller uses a RAM based FIFO for each endpoint buffer. The RAM dedicated for this purpose is called the Endpoint RAM (EP_RAM). Each endpoint has space reserved in the EP_RAM. The EP_RAM space required for an endpoint depends on its MaxPacketSize and whether it is double buffered. 32 words of EP_RAM are used by the device for storing the endpoint buffer pointers. The EP_RAM is word aligned but the MaxPacketSize is defined in bytes hence the RAM depth has to be adjusted to the next word boundary. Also, each buffer has one word header showing the size of the packet length received.

The EP_ RAM space (in words) required for the physical endpoint can be expressed as

$$EPRAMspace = \left(\frac{MaxPacketSize + 3}{4} + 1\right) \times dbstatus$$

where dbstatus = 1 for a single buffered endpoint and 2 for double a buffered endpoint.

Since all the realized endpoints occupy EP_RAM space, the total EP_RAM requirement is

$$TotalEPRAMspace = 32 + \sum_{n=0}^{N} EPRAMspace(n)$$

where N is the number of realized endpoints. Total EP_RAM space should not exceed 4096 bytes (4 kB, 1 kwords).

10.4.2 USB Realize Endpoint register (USBReEp - 0x5000 C244)

Writing one to a bit in this register causes the corresponding endpoint to be realized. Writing zeros causes it to be unrealized. This register returns to its reset state when a bus reset occurs. USBReEp is a read/write register.

© NXP B.V. 2010. All rights reserved.

Table 210. USB Realize Endpoint register (USBReEp - address 0x5000 C244) bit allocation

Reset value: 0x0000 0003

Bit	31	30	29	28	27	26	25	24
Symbol	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
Bit	23	22	21	20	19	18	17	16
Symbol	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16
Bit	15	14	13	12	11	10	9	8
Symbol	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
Bit	7	6	5	4	3	2	1	0
Symbol	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

Table 211. USB Realize Endpoint register (USBReEp - address 0x5000 C244) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint EP0 is not realized.	1
		1	Control endpoint EP0 is realized.	
1	EP1	0	Control endpoint EP1 is not realized.	1
		1	Control endpoint EP1 is realized.	
31:2	EPxx	0	Endpoint EPxx is not realized.	0
		1	Endpoint EPxx is realized.	

On reset, only the control endpoints are realized. Other endpoints, if required, are realized by programming the corresponding bits in USBReEp. To calculate the required EP_RAM space for the realized endpoints, see Section 11–10.4.1.

Realization of endpoints is a multi-cycle operation. Pseudo code for endpoint realization is shown below.

```
Clear EP_RLZED bit in USBDevIntSt;

for every endpoint to be realized,
{
      /* OR with the existing value of the Realize Endpoint register */
      USBReEp |= (UInt32) ((0x1 << endpt));
      /* Load Endpoint index Reg with physical endpoint no.*/
      USBEpIn = (UInt32) endpointnumber;

      /* load the max packet size Register */
      USBEpMaxPSize = MPS;

      /* check whether the EP_RLZED bit in the Device Interrupt Status register is set
      */
      while (!(USBDevIntSt & EP_RLZED))
      {
            /* wait until endpoint realization is complete */
      }
      /* Clear the EP_RLZED bit */
      Clear EP_RLZED bit in USBDevIntSt;
}
```


© NXP B.V. 2010. All rights reserved.

The device will not respond to any transactions to unrealized endpoints. The SIE Configure Device command will only cause realized and enabled endpoints to respond to transactions. For details see Table 11–242.

10.4.3 USB Endpoint Index register (USBEpIn - 0x5000 C248)

Each endpoint has a register carrying the MaxPacketSize value for that endpoint. This is in fact a register array. Hence before writing, this register is addressed through the USBEpIn register.

The USBEpIn register will hold the physical endpoint number. Writing to USBMaxPSize will set the array element pointed to by USBEpIn. USBEpIn is a write-only register.

Table 212. USB Endpoint Index register (USBEpIn - address 0x5000 C248) bit description

Bit	Symbol	Description	Reset value
4:0	PHY_EP	Physical endpoint number (0-31)	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

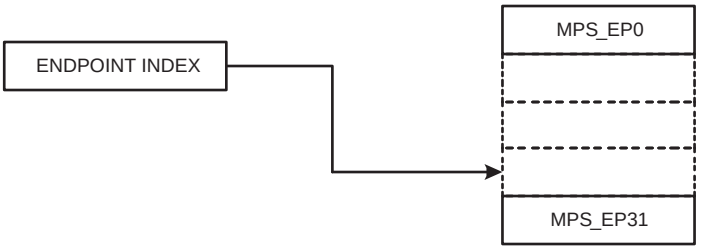
10.4.4 USB MaxPacketSize register (USBMaxPSize - 0x5000 C24C)

On reset, the control endpoint is assigned the maximum packet size of 8 bytes. Other endpoints are assigned 0. Modifying USBMaxPSize will cause the endpoint buffer addresses within the EP_RAM to be recalculated. This is a multi-cycle process. At the end, the EP_RLZED bit will be set in USBDevIntSt (Table 11–191). USBMaxPSize array indexing is shown in Figure 11–27. USBMaxPSize is a read/write register.

Table 213. USB MaxPacketSize register (USBMaxPSize - address 0x5000 C24C) bit description

Bit	Symbol	Description	Reset value
9:0	MPS	The maximum packet size value.	0x008[1]
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Reset value for EP0 and EP1. All other endpoints have a reset value of 0x0.



The Endpoint Index is set via the USBEpIn register. MPS_EP0 to MPS_EP31 are accessed via the USBMaxPSize register.

Fig 27. USB MaxPacketSize register array indexing

10.5 USB transfer registers

The registers in this group are used for transferring data between endpoint buffers and RAM in Slave mode operation. See Section 11–14 “Slave mode operation”.

© NXP B.V. 2010. All rights reserved.

10.5.1 USB Receive Data register (USBRxData - 0x5000 C218)

For an OUT transaction, the CPU reads the endpoint buffer data from this register. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. On reading this register, data from the selected endpoint buffer is fetched. The data is in little endian format: the first byte received from the USB bus will be available in the least significant byte of USBRxData. USBRxData is a read-only register.

Table 214. USB Receive Data register (USBRxData - address 0x5000 C218) bit description

Bit	Symbol	Description	Reset value
31:0	RX_DATA	Data received.	0x0000 0000

10.5.2 USB Receive Packet Length register (USBRxPLen - 0x5000 C220)

This register contains the number of bytes remaining in the endpoint buffer for the current packet being read via the USBRxData register, and a bit indicating whether the packet is valid or not. Before reading this register, the RD_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately. This register is updated on each read of the USBRxData register. USBRxPLen is a read-only register.

Table 215. USB Receive Packet Length register (USBRxPlen - address 0x5000 C220) bit description

Bit	Symbol	Value	Description	Reset value
9:0	PKT_LNGTH	-	The remaining number of bytes to be read from the currently selected endpoint's buffer. When this field decrements to 0, the RxENDPKT bit will be set in USBDevIntSt.	0
10	DV		Data valid. This bit is useful for isochronous endpoints. Non-isochronous endpoints do not raise an interrupt when an erroneous data packet is received. But invalid data packet can be produced with a bus reset. For isochronous endpoints, data transfer will happen even if an erroneous packet is received. In this case DV bit will not be set for the packet.	0
		0	Data is invalid.	
		1	Data is valid.	
11	PKT_RDY	-	The PKT_LNGTH field is valid and the packet is ready for reading.	0
31:12	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.5.3 USB Transmit Data register (USBTxData - 0x5000 C21C)

For an IN transaction, the CPU writes the endpoint data into this register. Before writing to this register, the WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set appropriately, and the packet length should be written to the USBTxPlen register. On writing this register, the data is written to the selected endpoint buffer. The data is in little endian format: the first byte sent on the USB bus will be the least significant byte of USBTxData. USBTxData is a write-only register.

Table 216. USB Transmit Data register (USBTxData - address 0x5000 C21C) bit description

Bit	Symbol	Description	Reset value
31:0	TX_DATA	Transmit Data.	0x0000 0000

© NXP B.V. 2010. All rights reserved.

10.5.4 USB Transmit Packet Length register (USBTxPLen - 0x5000 C224)

This register contains the number of bytes transferred from the CPU to the selected endpoint buffer. Before writing data to USBTxData, software should first write the packet length (≤ MaxPacketSize) to this register. After each write to USBTxData, hardware decrements USBTxPLen by 4. The WR_EN bit and LOG_ENDPOINT field of the USBCtrl register should be set to select the desired endpoint buffer before starting this process.

For data buffers larger than the endpoint's MaxPacketSize, software should submit data in packets of MaxPacketSize, and send the remaining extra bytes in the last packet. For example, if the MaxPacketSize is 64 bytes and the data buffer to be transferred is of length 130 bytes, then the software sends two 64-byte packets and the remaining 2 bytes in the last packet. So, a total of 3 packets are sent on USB. USBTxPLen is a write-only register.

Table 217. USB Transmit Packet Length register (USBTxPLen - address 0x5000 C224) bit description

Bit	Symbol	Value	Description	Reset value
9:0	PKT_LNGTH	-	The remaining number of bytes to be written to the selected endpoint buffer. This field is decremented by 4 by hardware after each write to USBTxData. When this field decrements to 0, the TxENDPKT bit will be set in USBDevIntSt.	0x000
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.5.5 USB Control register (USBCtrl - 0x5000 C228)

This register controls the data transfer operation of the USB device. It selects the endpoint buffer that is accessed by the USBRxData and USBTxData registers, and enables reading and writing them. USBCtrl is a read/write register.

Table 218. USB Control register (USBCtrl - address 0x5000 C228) bit description

Bit	Symbol	Value	Description	Reset value
0	RD_EN		Read mode control. Enables reading data from the OUT endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBRxData register. This bit is cleared by hardware when the last word of the current packet is read from USBRxData.	0
		0	Read mode is disabled.	
		1	Read mode is enabled.	
1	WR_EN		Write mode control. Enables writing data to the IN endpoint buffer for the endpoint specified in the LOG_ENDPOINT field using the USBTxData register. This bit is cleared by hardware when the number of bytes in USBTxLen have been sent.	0
		0	Write mode is disabled.	
		1	Write mode is enabled.	
5:2	LOG_ENDPOINT	-	Logical Endpoint number.	0x0
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.6 SIE command code registers

The SIE command code registers are used for communicating with the Serial Interface Engine. See Section 11–12 "Serial interface engine command description" for more information.

© NXP B.V. 2010. All rights reserved.

10.6.1 USB Command Code register (USBCmdCode - 0x5000 C210)

This register is used for sending the command and write data to the SIE. The commands written here are propagated to the SIE and executed there. After executing the command, the register is empty, and the CCEMPTY bit of USBDevIntSt register is set. See Section 11–12 for details. USBCmdCode is a write-only register.

Table 219. USB Command Code register (USBCmdCode - address 0x5000 C210) bit description

Bit	Symbol	Value	Description	Reset value
7:0	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:8	CMD_PHASE		The command phase:	0x00
		0x01	Read	
		0x02	Write	
		0x05	Command	
23:16	CMD_CODE/ CMD_WDATA		This is a multi-purpose field. When CMD_PHASE is Command or Read, this field contains the code for the command (CMD_CODE). When CMD_PHASE is Write, this field contains the command write data (CMD_WDATA).	0x00
31:24	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.6.2 USB Command Data register (USBCmdData - 0x5000 C214)

This register contains the data retrieved after executing a SIE command. When the data is ready to be read, the CD_FULL bit of the USBDevIntSt register is set. See Table 11–191 for details. USBCmdData is a read-only register.

Table 220. USB Command Data register (USBCmdData - address 0x5000 C214) bit description

Bit	Symbol	Description	Reset value
7:0	CMD_RDATA	Command Read Data.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.7 DMA registers

The registers in this group are used for the DMA mode of operation (see Section 11–15 “DMA operation”)

10.7.1 USB DMA Request Status register (USBDMARSt - 0x5000 C250)

A bit in this register associated with a non-isochronous endpoint is set by hardware when an endpoint interrupt occurs (see the description of USBEpIntSt) and the corresponding bit in USBEpIntEn is 0. A bit associated with an isochronous endpoint is set when the corresponding bit in USBEpIntEn is 0 and a FRAME interrupt occurs. A set bit serves as a flag for the DMA engine to start the data transfer if the DMA is enabled for the corresponding endpoint in the USBEpDMASt register. The DMA cannot be enabled for control endpoints (EP0 and EP1). USBDMARSt is a read-only register.

Table 221. USB DMA Request Status register (USBDMARSt - address 0x5000 C250) bit allocation
Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24

© NXP B.V. 2010. All rights reserved.

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 222. USB DMA Request Status register (USBDMARSt - address 0x5000 C250) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and EP0 bit must be 0).	0
1	EP1	0	Control endpoint IN (DMA cannot be enabled for this endpoint and EP1 bit must be 0).	0
31:2	EPxx		Endpoint xx ($2 \leq xx \leq 31$) DMA request.	0
		0	DMA not requested by endpoint xx.	
		1	DMA requested by endpoint xx.	

[1] DMA can not be enabled for this endpoint and the corresponding bit in the USBDMARSt must be 0.

10.7.2 USB DMA Request Clear register (USBDMARClr - 0x5000 C254)

Writing one to a bit in this register will clear the corresponding bit in the USBDMARSt register. Writing zero has no effect.

This register is intended for initialization prior to enabling the DMA for an endpoint. When the DMA is enabled for an endpoint, hardware clears the corresponding bit in USBDMARSt on completion of a packet transfer. Therefore, software should not clear the bit using this register while the endpoint is enabled for DMA operation.

USBDMARClr is a write-only register.

The USBDMARClr bit allocation is identical to the USBDMARSt register (Table 11–221).

Table 223. USB DMA Request Clear register (USBDMARClr - address 0x5000 C254) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).	0
1	EP1	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0
31:2	EPxx		Clear the endpoint xx ($2 \leq xx \leq 31$) DMA request.	0
		0	No effect.	
		1	Clear the corresponding bit in USBDMARSt.	

10.7.3 USB DMA Request Set register (USBDMARSet - 0x5000 C258)

Writing one to a bit in this register sets the corresponding bit in the USBDMARSt register. Writing zero has no effect.

© NXP B.V. 2010. All rights reserved.

This register allows software to raise a DMA request. This can be useful when switching from Slave to DMA mode of operation for an endpoint: if a packet to be processed in DMA mode arrives before the corresponding bit of USBEpIntEn is cleared, the DMA request is not raised by hardware. Software can then use this register to manually start the DMA transfer.

Software can also use this register to initiate a DMA transfer to proactively fill an IN endpoint buffer before an IN token packet is received from the host.

USBDMARSet is a write-only register.

The USBDMARSet bit allocation is identical to the USBDMARSt register (Table 11–221).

Table 224. USB DMA Request Set register (USBDMARSet - address 0x5000 C258) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0 bit must be 0).	0
1	EP1	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1 bit must be 0).	0
31:2	EPxx		Set the endpoint xx (2 ≤ xx ≤ 31) DMA request.	0
		0	No effect.	
		1	Set the corresponding bit in USBDMARSt.	

10.7.4 USB UDCA Head register (USBUDCAH - 0x5000 C280)

The UDCA (USB Device Communication Area) Head register maintains the address where the UDCA is located in the RAM. Refer to Section 11–15.2 “USB device communication area” and Section 11–15.4 “The DMA descriptor” for more details on the UDCA and DMA descriptors. USBUDCAH is a read/write register.

Table 225. USB UDCA Head register (USBUDCAH - address 0x5000 C280) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved. Software should not write ones to reserved bits. The UDCA is aligned to 128-byte boundaries.	0x00
31:7	UDCA_ADDR	Start address of the UDCA.	0

10.7.5 USB EP DMA Status register (USBEpDMASt - 0x5000 C284)

Bits in this register indicate whether DMA operation is enabled for the corresponding endpoint. A DMA transfer for an endpoint can start only if the corresponding bit is set in this register. USBEpDMASt is a read-only register.

Table 226. USB EP DMA Status register (USBEpDMASt - address 0x5000 C284) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0_DMA_ENABLE	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_ENABLE bit must be 0).	0
1	EP1_DMA_ENABLE	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_ENABLE bit must be 0).	0
31:2	EPxx_DMA_ENABLE		endpoint xx (2 ≤ xx ≤ 31) DMA enabled bit.	0
		0	The DMA for endpoint EPxx is disabled.	
		1	The DMA for endpoint EPxx is enabled.	

© NXP B.V. 2010. All rights reserved.

10.7.6 USB EP DMA Enable register (USBEpDMAEn - 0x5000 C288)

Writing one to a bit to this register will enable the DMA operation for the corresponding endpoint. Writing zero has no effect.The DMA cannot be enabled for control endpoints EP0 and EP1. USBEpDMAEn is a write-only register.

Table 227. USB EP DMA Enable register (USBEpDMAEn - address 0x5000 C288) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0_DMA_ENABLE	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_ENABLE bit value must be 0).	0
1	EP1_DMA_ENABLE	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_ENABLE bit must be 0).	0
31:2	EPxx_DMA_ENABLE		Endpoint xx($2 \leq xx \leq 31$) DMA enable control bit.	0
		0	No effect.	
		1	Enable the DMA operation for endpoint EPxx.	

10.7.7 USB EP DMA Disable register (USBEpDMADis - 0x5000 C28C)

Writing a one to a bit in this register clears the corresponding bit in USBEpDMASt. Writing zero has no effect on the corresponding bit of USBEpDMASt. Any write to this register clears the internal DMA_PROCEED flag. Refer to Section 11–15.5.4 “Optimizing descriptor fetch” for more information on the DMA_PROCEED flag. If a DMA transfer is in progress for an endpoint when its corresponding bit is cleared, the transfer is completed before the DMA is disabled. When an error condition is detected during a DMA transfer, the corresponding bit is cleared by hardware. USBEpDMADis is a write-only register.

Table 228. USB EP DMA Disable register (USBEpDMADis - address 0x5000 C28C) bit description

Bit	Symbol	Value	Description	Reset value
0	EP0_DMA_DISABLE	0	Control endpoint OUT (DMA cannot be enabled for this endpoint and the EP0_DMA_DISABLE bit value must be 0).	0
1	EP1_DMA_DISABLE	0	Control endpoint IN (DMA cannot be enabled for this endpoint and the EP1_DMA_DISABLE bit value must be 0).	0
31:2	EPxx_DMA_DISABLE		Endpoint xx ($2 \leq xx \leq 31$) DMA disable control bit.	0
		0	No effect.	
		1	Disable the DMA operation for endpoint EPxx.	

10.7.8 USB DMA Interrupt Status register (USBDMAIntSt - 0x5000 C290)

Each bit of this register reflects whether any of the 32 bits in the corresponding interrupt status register are set. USBDMAIntSt is a read-only register.

Table 229. USB DMA Interrupt Status register (USBDMAIntSt - address 0x5000 C290) bit description

Bit	Symbol	Value	Description	Reset value
0	EOT		End of Transfer Interrupt bit.	0
		0	All bits in the USBEoTIntSt register are 0.	
		1	At least one bit in the USBEoTIntSt is set.	
1	NDDR		New DD Request Interrupt bit.	0
		0	All bits in the USBNDDRIntSt register are 0.	
		1	At least one bit in the USBNDDRIntSt is set.	

© NXP B.V. 2010. All rights reserved.

Table 229. USB DMA Interrupt Status register (USBDMAIntSt - address 0x5000 C290) bit description

Bit	Symbol	Value	Description	Reset value
2	ERR		System Error Interrupt bit.	0
		0	All bits in the USBSysErrIntSt register are 0.	
		1	At least one bit in the USBSysErrIntSt is set.	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.7.9 USB DMA Interrupt Enable register (USBDMAIntEn - 0x5000 C294)

Writing a one to a bit in this register enables the corresponding bit in USBDMAIntSt to generate an interrupt on the USB_INT_REQ_DMA interrupt line when set. USBDMAIntEn is a read/write register.

Table 230. USB DMA Interrupt Enable register (USBDMAIntEn - address 0x5000 C294) bit description

Bit	Symbol	Value	Description	Reset value
0	EOT		End of Transfer Interrupt enable bit.	0
		0	The End of Transfer Interrupt is disabled.	
		1	The End of Transfer Interrupt is enabled.	
1	NDDR		New DD Request Interrupt enable bit.	0
		0	The New DD Request Interrupt is disabled.	
		1	The New DD Request Interrupt is enabled.	
2	ERR		System Error Interrupt enable bit.	0
		0	The System Error Interrupt is disabled.	
		1	The System Error Interrupt is enabled.	
31:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.7.10 USB End of Transfer Interrupt Status register (USBEoTIntSt - 0x5000 C2A0)

When the DMA transfer completes for the current DMA descriptor, either normally (descriptor is retired) or because of an error, the bit corresponding to the endpoint is set in this register. The cause of the interrupt is recorded in the DD_status field of the descriptor. USBEoTIntSt is a read-only register.

Table 231. USB End of Transfer Interrupt Status register (USBEoTIntSt - address 0x5000 C2A0s) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Endpoint xx ($2 \leq xx \leq 31$) End of Transfer Interrupt request.	0
		0	There is no End of Transfer interrupt request for endpoint xx.	
		1	There is an End of Transfer Interrupt request for endpoint xx.	

10.7.11 USB End of Transfer Interrupt Clear register (USBEoTIntClr - 0x5000 C2A4)

Writing one to a bit in this register clears the corresponding bit in the USBEoTIntSt register. Writing zero has no effect. USBEoTIntClr is a write-only register.

© NXP B.V. 2010. All rights reserved.

Table 232. USB End of Transfer Interrupt Clear register (USBEoTIntClr - address 0x5000 C2A4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Clear endpoint xx (2 ≤ xx ≤ 31) End of Transfer Interrupt request.	0
		0	No effect.	
		1	Clear the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	

10.7.12 USB End of Transfer Interrupt Set register (USBEoTIntSet - 0x5000 C2A8)

Writing one to a bit in this register sets the corresponding bit in the USBEoTIntSt register. Writing zero has no effect. USBEoTIntSet is a write-only register.

Table 233. USB End of Transfer Interrupt Set register (USBEoTIntSet - address 0x5000 C2A8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Set endpoint xx (2 ≤ xx ≤ 31) End of Transfer Interrupt request.	0
		0	No effect.	
		1	Set the EPxx End of Transfer Interrupt request in the USBEoTIntSt register.	

10.7.13 USB New DD Request Interrupt Status register (USBNDDRIntSt - 0x5000 C2AC)

A bit in this register is set when a transfer is requested from the USB device and no valid DD is detected for the corresponding endpoint. USBNDDRIntSt is a read-only register.

Table 234. USB New DD Request Interrupt Status register (USBNDDRIntSt - address 0x5000 C2AC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request.	0
		0	There is no new DD interrupt request for endpoint xx.	
		1	There is a new DD interrupt request for endpoint xx.	

10.7.14 USB New DD Request Interrupt Clear register (USBNDDRIntClr - 0x5000 C2B0)

Writing one to a bit in this register clears the corresponding bit in the USBNDDRIntSt register. Writing zero has no effect. USBNDDRIntClr is a write-only register.

Table 235. USB New DD Request Interrupt Clear register (USBNDDRIntClr - address 0x5000 C2B0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Clear endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request.	0
		0	No effect.	
		1	Clear the EPxx new DD interrupt request in the USBNDDRIntSt register.	

10.7.15 USB New DD Request Interrupt Set register (USBNDDRIntSet - 0x5000 C2B4)

Writing one to a bit in this register sets the corresponding bit in the USBNDDRIntSt register. Writing zero has no effect. USBNDDRIntSet is a write-only register

© NXP B.V. 2010. All rights reserved.

Table 236. USB New DD Request Interrupt Set register (USBNDDRIntSet - address 0x5000 C2B4) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Set endpoint xx (2 ≤ xx ≤ 31) new DD interrupt request.	0
		0	No effect.	
		1	Set the EPxx new DD interrupt request in the USBNDDRIntSt register.	

10.7.16 USB System Error Interrupt Status register (USBSysErrIntSt - 0x5000 C2B8)

If a system error (AHB bus error) occurs when transferring the data or when fetching or updating the DD the corresponding bit is set in this register. USBSysErrIntSt is a read-only register.

Table 237. USB System Error Interrupt Status register (USBSysErrIntSt - address 0x5000 C2B8) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Endpoint xx (2 ≤ xx ≤ 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.	

10.7.17 USB System Error Interrupt Clear register (USBSysErrIntClr - 0x5000 C2BC)

Writing one to a bit in this register clears the corresponding bit in the USBSysErrIntSt register. Writing zero has no effect. USBSysErrIntClr is a write-only register.

Table 238. USB System Error Interrupt Clear register (USBSysErrIntClr - address 0x5000 C2BC) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Clear endpoint xx (2 ≤ xx ≤ 31) System Error Interrupt request.	0
		0	No effect.	
		1	Clear the EPxx System Error Interrupt request in the USBSysErrIntSt register.	

10.7.18 USB System Error Interrupt Set register (USBSysErrIntSet - 0x5000 C2C0)

Writing one to a bit in this register sets the corresponding bit in the USBSysErrIntSt register. Writing zero has no effect. USBSysErrIntSet is a write-only register.

Table 239. USB System Error Interrupt Set register (USBSysErrIntSet - address 0x5000 C2C0) bit description

Bit	Symbol	Value	Description	Reset value
31:0	EPxx		Set endpoint xx (2 ≤ xx ≤ 31) System Error Interrupt request.	0
		0	No effect.	
		1	Set the EPxx System Error Interrupt request in the USBSysErrIntSt register.	

11. Interrupt handling

This section describes how an interrupt event on any of the endpoints is routed to the Nested Vectored Interrupt Controller (NVIC). For a diagram showing interrupt event handling, see Figure 11–28.

All non-isochronous OUT endpoints (control, bulk, and interrupt endpoints) generate an interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet has been successfully transmitted or when a NAK signal is sent and interrupts on NAK are enabled by the SIE Set Mode command, see Section 11–12.3. For isochronous endpoints, a frame interrupt is generated every 1 ms.

© NXP B.V. 2010. All rights reserved.

The interrupt handling is different for Slave and DMA mode.

Slave mode

If an interrupt event occurs on an endpoint and the endpoint interrupt is enabled in the USBEpIntEn register, the corresponding status bit in the USBEpIntSt is set. For non-isochronous endpoints, all endpoint interrupt events are divided into two types by the corresponding USBEpIntPri[n] registers: fast endpoint interrupt events and slow endpoint interrupt events. All fast endpoint interrupt events are ORed and routed to bit EP_FAST in the USBDevIntSt register. All slow endpoint interrupt events are ORed and routed to the EP_SLOW bit in USBDevIntSt.

For isochronous endpoints, the FRAME bit in USBDevIntSt is set every 1 ms.

The USBDevIntSt register holds the status of all endpoint interrupt events as well as the status of various other interrupts (see Section 11–10.2.2). By default, all interrupts (if enabled in USBDevIntEn) are routed to the USB_INT_REQ_LP bit in the USBIntSt register to request low priority interrupt handling. However, the USBDevIntPri register can route either the FRAME or the EP_FAST bit to the USB_INT_REQ_HP bit in the USBIntSt register.

Only one of the EP_FAST and FRAME interrupt events can be routed to the USB_INT_REQ_HP bit. If routing both bits to USB_INT_REQ_HP is attempted, both interrupt events are routed to USB_INT_REQ_LP.

Slow endpoint interrupt events are always routed directly to the USB_INT_REQ_LP bit for low priority interrupt handling by software.

The final interrupt signal to the NVIC is gated by the EN_USB_INTS bit in the USBIntSt register. The USB interrupts are routed to the NVIC only if EN_USB_INTS is set.

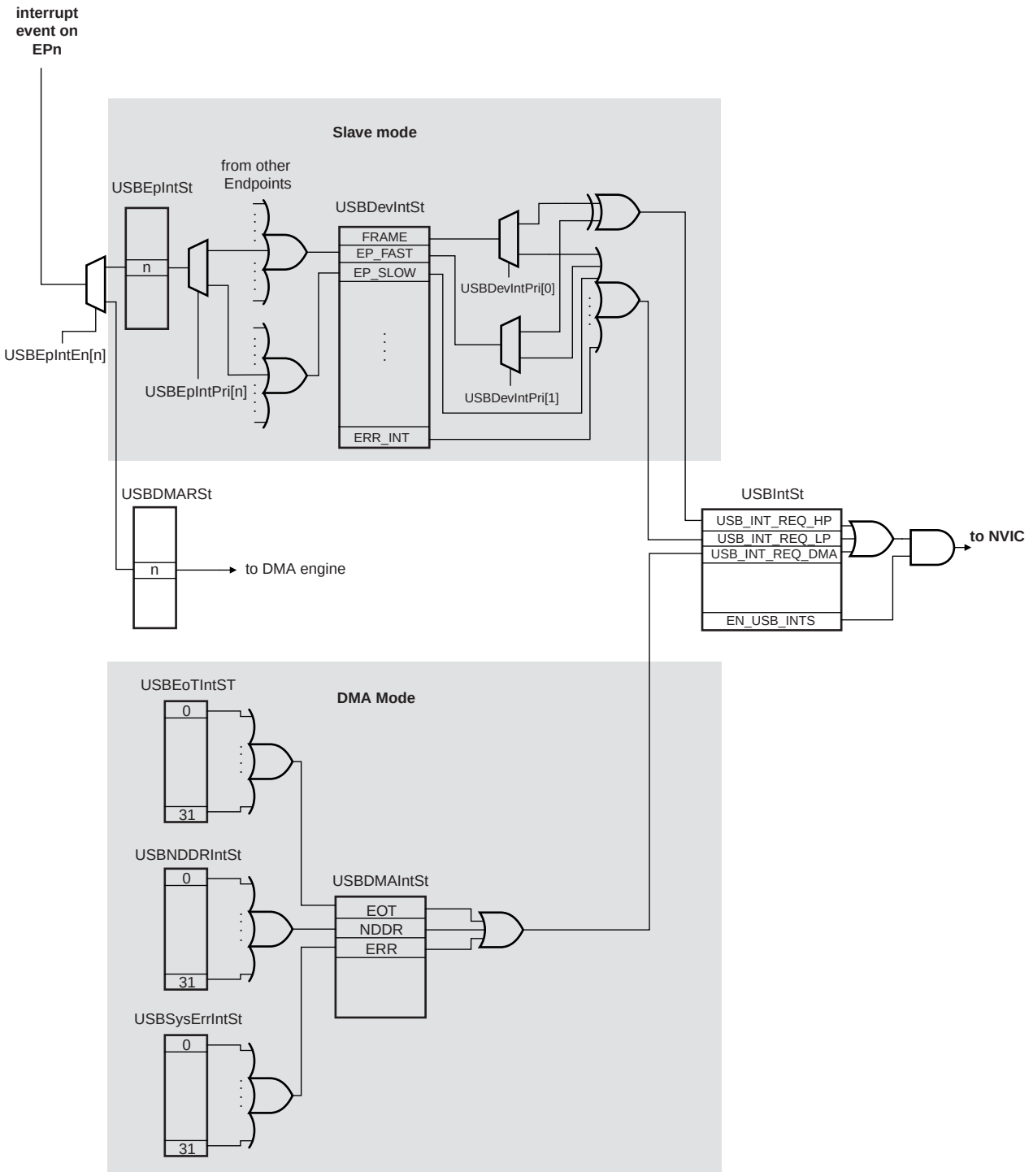
DMA mode

If an interrupt event occurs on a non-control endpoint and the endpoint interrupt is not enabled in the USBEpIntEn register, the corresponding status bit in the USBDMARSt is set by hardware. This serves as a flag for the DMA engine to transfer data if DMA transfer is enabled for the corresponding endpoint in the USBEpDMASt register.

Three types of interrupts can occur for each endpoint for data transfers in DMA mode: End of transfer interrupt, new DD request interrupt, and system error interrupt. These interrupt events set a bit for each endpoint in the respective registers USBEoTIntSt, USBNDDRIntSt, and USBSysErrIntSt. The End of transfer interrupts from all endpoints are then Ored and routed to the EOT bit in USBDMAIntSt. Likewise, all New DD request interrupts and system error interrupt events are routed to the NDDR and ERR bits respectively in the USBDMAStInt register.

The EOT, NDDR, and ERR bits (if enabled in USBDMAIntEn) are ORed to set the USB_INT_REQ_DMA bit in the USBIntSt register. If the EN_USB_INTS bit is set in USBIntSt, the interrupt is routed to the NVIC.

© NXP B.V. 2010. All rights reserved.



For simplicity, USBDevIntEn and USBDMAIntEn are not shown.

Fig 28. Interrupt event handling

© NXP B.V. 2010. All rights reserved.

12. Serial interface engine command description

The functions and registers of the Serial Interface Engine (SIE) are accessed using commands, which consist of a command code followed by optional data bytes (read or write action). The USBCmdCode (Table 11–219) and USBCmdData (Table 11–220) registers are used for these accesses.

A complete access consists of two phases:

1. **Command phase:** the USBCmdCode register is written with the CMD_PHASE field set to the value 0x05 (Command), and the CMD_CODE field set to the desired command code. On completion of the command, the CCEMPTY bit of USBDevIntSt is set.
2. **Data phase (optional):** for writes, the USBCmdCode register is written with the CMD_PHASE field set to the value 0x01 (Write), and the CMD_WDATA field set with the desired write data. On completion of the write, the CCEMPTY bit of USBDevIntSt is set. For reads, USBCmdCode register is written with the CMD_PHASE field set to the value 0x02 (Read), and the CMD_CODE field set with command code the read corresponds to. On completion of the read, the CDFULL bit of USBDevInSt will be set, indicating the data is available for reading in the USBCmdData register. In the case of multi-byte registers, the least significant byte is accessed first.

An overview of the available commands is given in Table 11–240.

Here is an example of the Read Current Frame Number command (reading 2 bytes):

```
USBDevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL
USBCmdCode = 0x00F50500;       // CMD_CODE=0xF5, CMD_PHASE=0x05(Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY interrupt bit.
USBCmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
USBDevIntClr = 0x20;           // Clear CDFULL.
CurFrameNum = USBCmdData;      // Read Frame number LSB byte.
USBCmdCode = 0x00F50200;       // CMD_CODE=0xF5, CMD_PHASE=0x02(Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
Temp = USBCmdData;             // Read Frame number MSB byte
USBDevIntClr = 0x20;           // Clear CDFULL interrupt bit.
CurFrameNum = CurFrameNum | (Temp << 8);
```

Here is an example of the Set Address command (writing 1 byte):

```
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x00D00500;       // CMD_CODE=0xD0, CMD_PHASE=0x05(Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x008A0100;       // CMD_WDATA=0x8A(DEV_EN=1, DEV_ADDR=0xA),
                               // CMD_PHASE=0x01(Write)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
```


© NXP B.V. 2010. All rights reserved.

Table 240. SIE command code table

Command name	Recipient	Code (Hex)	Data phase
Device commands			
Set Address	Device	D0	Write 1 byte
Configure Device	Device	D8	Write 1 byte
Set Mode	Device	F3	Write 1 byte
Read Current Frame Number	Device	F5	Read 1 or 2 bytes
Read Test Register	Device	FD	Read 2 bytes
Set Device Status	Device	FE	Write 1 byte
Get Device Status	Device	FE	Read 1 byte
Get Error Code	Device	FF	Read 1 byte
Read Error Status	Device	FB	Read 1 byte
Endpoint Commands			
Select Endpoint	Endpoint 0	00	Read 1 byte (optional)
	Endpoint 1	01	Read 1 byte (optional)
	Endpoint xx	xx	Read 1 byte (optional)
Select Endpoint/Clear Interrupt	Endpoint 0	40	Read 1 byte
	Endpoint 1	41	Read 1 byte
	Endpoint xx	xx + 40	Read 1 byte
Set Endpoint Status	Endpoint 0	40	Write 1 byte
	Endpoint 1	41	Write 1 byte
	Endpoint xx	xx + 40	Write 1 byte
Clear Buffer	Selected Endpoint	F2	Read 1 byte (optional)
Validate Buffer	Selected Endpoint	FA	None

12.1 Set Address (Command: 0xD0, Data: write 1 byte)

The Set Address command is used to set the USB assigned address and enable the (embedded) function. The address set in the device will take effect after the status stage of the control transaction. After a bus reset, DEV_ADDR is set to 0x00, and DEV_EN is set to 1. The device will respond to packets for function address 0x00, endpoint 0 (default endpoint).

Table 241. Set Address command bit description

Bit	Symbol	Description	Reset value
6:0	DEV_ADDR	Device address set by the software. After a bus reset this field is set to 0x00.	0x00
7	DEV_EN	Device Enable. After a bus reset this bit is set to 1. 0: Device will not respond to any packets. 1: Device will respond to packets for function address DEV_ADDR.	0

12.2 Configure Device (Command: 0xD8, Data: write 1 byte)

A value of 1 written to the register indicates that the device is configured and all the enabled non-control endpoints will respond. Control endpoints are always enabled and respond even if the device is not configured, in the default state.

© NXP B.V. 2010. All rights reserved.

Table 242. Configure Device command bit description

Bit	Symbol	Description	Reset value
0	CONF_DEVICE	Device is configured. All enabled non-control endpoints will respond. This bit is cleared by hardware when a bus reset occurs. When set, the UP_LED signal is driven LOW if the device is not in the suspended state (SUS=0).	
7:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.3 Set Mode (Command: 0xF3, Data: write 1 byte)

Table 243. Set Mode command bit description

Bit	Symbol	Value	Description	Reset value
0	AP_CLK		Always PLL Clock.	0
		0	USB_NEED_CLK is functional; the 48 MHz clock can be stopped when the device enters suspend state.	
		1	USB_NEED_CLK is fixed to 1; the 48 MHz clock cannot be stopped when the device enters suspend state.	
1	INAK_CI		Interrupt on NAK for Control IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
2	INAK_CO		Interrupt on NAK for Control OUT endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
3	INAK_II		Interrupt on NAK for Interrupt IN endpoint.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
4	INAK_IO[1]		Interrupt on NAK for Interrupt OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
5	INAK_BI		Interrupt on NAK for Bulk IN endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed IN transactions generate interrupts.	
6	INAK_BO[2]		Interrupt on NAK for Bulk OUT endpoints.	0
		0	Only successful transactions generate an interrupt.	
		1	Both successful and NAKed OUT transactions generate interrupts.	
7	-	-	Reserved, 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.

12.4 Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes)

Returns the frame number of the last successfully received SOF. The frame number is eleven bits wide. The frame number returns least significant byte first. In case the user is only interested in the lower 8 bits of the frame number, only the first byte needs to be read.

© NXP B.V. 2010. All rights reserved.

- In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF.
- In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.

12.5 Read Test Register (Command: 0xFD, Data: read 2 bytes)

The test register is 16 bits wide. It returns the value of 0xA50F if the USB clocks (usbclk and AHB slave clock) are running.

12.6 Set Device Status (Command: 0xFE, Data: write 1 byte)

The Set Device Status command sets bits in the Device Status Register.

Table 244. Set Device Status command bit description

Bit	Symbol	Value	Description	Reset value
0	CON		The Connect bit indicates the current connect status of the device. It controls the CONNECT output pin, used for SoftConnect. Reading the connect bit returns the current connect status. This bit is cleared by hardware when the V_{BUS} status input is LOW for more than 3 ms. The 3 ms delay filters out temporary dips in the V_{BUS} voltage.	0
		0	Writing a 0 will make the CONNECT pin go HIGH.	
		1	Writing a 1 will make the CONNECT pin go LOW.	
1	CON_CH		Connect Change.	0
		0	This bit is cleared when read.	
		1	This bit is set when the device's pull-up resistor is disconnected because V_{BUS} disappeared. The DEV_STAT interrupt is generated when this bit is 1.	
2	SUS		Suspend: The Suspend bit represents the current suspend state. When the device is suspended (SUS = 1) and the CPU writes a 0 into it, the device will generate a remote wake-up. This will only happen when the device is connected (CON = 1). When the device is not connected or not suspended, writing a 0 has no effect. Writing a 1 to this bit has no effect.	0
		0	This bit is reset to 0 on any activity.	
		1	This bit is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 ms.	
3	SUS_CH		Suspend (SUS) bit change indicator. The SUS bit can toggle because: • The device goes into the suspended state. • The device is disconnected. • The device receives resume signalling on its upstream port. This bit is cleared when read.	0
		0	SUS bit not changed.	
		1	SUS bit changed. At the same time a DEV_STAT interrupt is generated.	

© NXP B.V. 2010. All rights reserved.

Table 244. Set Device Status command bit description

Bit	Symbol	Value	Description	Reset value
4	RST		Bus Reset bit. On a bus reset, the device will automatically go to the default state. In the default state: • Device is unconfigured. • Will respond to address 0. • Control endpoint will be in the Stalled state. • All endpoints are unrealized except control endpoints EP0 and EP1. • Data toggling is reset for all endpoints. • All buffers are cleared. • There is no change to the endpoint interrupt status. • DEV_STAT interrupt is generated. Note: Bus resets are ignored when the device is not connected (CON=0).	0
		0	This bit is cleared when read.	
		1	This bit is set when the device receives a bus reset. A DEV_STAT interrupt is generated.	
7:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.7 Get Device Status (Command: 0xFE, Data: read 1 byte)

The Get Device Status command returns the Device Status Register. Reading the device status returns 1 byte of data. The bit field definition is same as the Set Device Status Register as shown in Table 11–244.

Remark: To ensure correct operation, the DEV_STAT bit of USBDevIntSt must be cleared before executing the Get Device Status command.

12.8 Get Error Code (Command: 0xFF, Data: read 1 byte)

Different error conditions can arise inside the SIE. The Get Error Code command returns the last error code that occurred. The 4 least significant bits form the error code.

© NXP B.V. 2010. All rights reserved.

Table 245. Get Error Code command 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

12.9 Read Error Status (Command: 0xFB, Data: read 1 byte)

This command reads the 8-bit Error register from the USB device. This register records which error events have recently occurred in the SIE. If any of these bits are set, the ERR_INT bit of USBDevIntSt is set. The error bits are cleared after reading this register.

Table 246. Read Error Status command bit description

Bit	Symbol	Description	Reset value
0	PID_ERR	PID encoding error or Unknown PID or Token CRC.	0
1	UEPKT	Unexpected Packet - any packet sequence violation from the specification.	0
2	DCRC	Data CRC error.	0
3	TIMEOUT	Time out error.	0
4	EOP	End of packet error.	0
5	B_OVRN	Buffer Overrun.	0
6	BTSTF	Bit stuff error.	0
7	TGL_ERR	Wrong toggle bit in data PID, ignored data.	0

12.10 Select Endpoint (Command: 0x00 - 0x1F, Data: read 1 byte (optional))

The Select Endpoint command initializes an internal pointer to the start of the selected buffer in EP_RAM. Optionally, this command can be followed by a data read, which returns some additional information on the packet(s) in the endpoint buffer(s). The command code of the Select Endpoint command is equal to the physical endpoint number. In the case of a single buffered endpoint the B_2_FULL bit is not valid.

© NXP B.V. 2010. All rights reserved.

Table 247. Select Endpoint command bit description

Bit	Symbol	Value	Description	Reset value
0	FE		Full/Empty. This bit indicates the full or empty status of the endpoint buffer(s). For IN endpoints, the FE bit gives the ANDed result of the B_1_FULL and B_2_FULL bits. For OUT endpoints, the FE bit gives ORed result of the B_1_FULL and B_2_FULL bits. For single buffered endpoints, this bit simply reflects the status of B_1_FULL.	0
		0	For an IN endpoint, at least one write endpoint buffer is empty.	
		1	For an OUT endpoint, at least one endpoint read buffer is full.	
1	ST		Stalled endpoint indicator.	0
		0	The selected endpoint is not stalled.	
		1	The selected endpoint is stalled.	
2	STP		SETUP bit: the value of this bit is updated after each successfully received packet (i.e. an ACKed package on that particular physical endpoint).	0
		0	The STP bit is cleared by doing a Select Endpoint/Clear Interrupt on this endpoint.	
		1	The last received packet for the selected endpoint was a SETUP packet.	
3	PO		Packet over-written bit.	0
		0	The PO bit is cleared by the ‘Select Endpoint/Clear Interrupt’ command.	
		1	The previously received packet was over-written by a SETUP packet.	
4	EPN		EP NAKed bit indicates sending of a NAK. If the host sends an OUT packet to a filled OUT buffer, the device returns NAK. If the host sends an IN token packet to an empty IN buffer, the device returns NAK.	0
		0	The EPN bit is reset after the device has sent an ACK after an OUT packet or when the device has seen an ACK after sending an IN packet.	
		1	The EPN bit is set when a NAK is sent and the interrupt on NAK feature is enabled.	
5	B_1_FULL		The buffer 1 status.	0
		0	Buffer 1 is empty.	
		1	Buffer 1 is full.	
6	B_2_FULL		The buffer 2 status.	0
		0	Buffer 2 is empty.	
		1	Buffer 2 is full.	
7	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.11 Select Endpoint/Clear Interrupt (Command: 0x40 - 0x5F, Data: read 1 byte)

Commands 0x40 to 0x5F are identical to their Select Endpoint equivalents, with the following differences:

- They clear the bit corresponding to the endpoint in the USBEpIntSt register.
- In case of a control OUT endpoint, they clear the STP and PO bits in the corresponding Select Endpoint Register.
- Reading one byte is obligatory.

© NXP B.V. 2010. All rights reserved.

Remark: This command may be invoked by using the USBCmdCode and USBCmdData registers, or by setting the corresponding bit in USBEpIntClr. For ease of use, using the USBEpIntClr register is recommended.

12.12 Set Endpoint Status (Command: 0x40 - 0x55, Data: write 1 byte (optional))

The Set Endpoint Status command sets status bits 7:5 and 0 of the endpoint. The Command Code of Set Endpoint Status is equal to the sum of 0x40 and the physical endpoint number in hex. Not all bits can be set for all types of endpoints.

Table 248. Set Endpoint Status command bit description

Bit	Symbol	Value	Description	Reset value
0	ST		Stalled endpoint bit. A Stalled control endpoint is automatically unstalled when it receives a SETUP token, regardless of the content of the packet. If the endpoint should stay in its stalled state, the CPU can stall it again by setting this bit. When a stalled endpoint is unstalled - either by the Set Endpoint Status command or by receiving a SETUP token - it is also re-initialized. This flushes the buffer: in case of an OUT buffer it waits for a DATA 0 PID; in case of an IN buffer it writes a DATA 0 PID. There is no change of the interrupt status of the endpoint. When already unstalled, writing a zero to this bit initializes the endpoint. When an endpoint is stalled by the Set Endpoint Status command, it is also re-initialized.	0
		0	The endpoint is unstalled.	
		1	The endpoint is stalled.	
4:1	-	-	Reserved, 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 STP bit is set in the Select Endpoint register. It is defined only for control OUT endpoints.	

12.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional))

When an OUT packet sent by the host has been received successfully, an internal hardware FIFO status Buffer_Full flag is set. All subsequent packets will be refused by returning a NAK. When the device software has read the data, it should free the buffer by issuing the Clear Buffer command. This clears the internal Buffer_Full flag. When the buffer is cleared, new packets will be accepted.

When bit 0 of the optional data byte is 1, the previously received packet was over-written by a SETUP packet. The Packet over-written bit is used only in control transfers. According to the USB specification, a SETUP packet should be accepted irrespective of the buffer status. The software should always check the status of the PO bit after reading

© NXP B.V. 2010. All rights reserved.

the SETUP data. If it is set then it should discard the previously read data, clear the PO bit by issuing a Select Endpoint/Clear Interrupt command, read the new SETUP data and again check the status of the PO bit.

See Section 11–14 “Slave mode operation” for a description of when this command is used.

Table 249. Clear Buffer command bit description

Bit	Symbol	Value	Description	Reset value
0	PO		Packet over-written bit. This bit is only applicable to the control endpoint EP0.	0
		0	The previously received packet is intact.	
		1	The previously received packet was over-written by a later SETUP packet.	
7:1	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.14 Validate Buffer (Command: 0xFA, Data: none)

When the CPU has written data into an IN buffer, software should issue a Validate Buffer command. This tells hardware that the buffer is ready for sending on the USB bus. Hardware will send the contents of the buffer when the next IN token packet is received.

Internally, there is a hardware FIFO status flag called Buffer_Full. This flag is set by the Validate Buffer command and cleared when the data has been sent on the USB bus and the buffer is empty.

A control IN buffer cannot be validated when its corresponding OUT buffer has the Packet Over-written (PO) bit (see the Clear Buffer Register) set or contains a pending SETUP packet. For the control endpoint the validated buffer will be invalidated when a SETUP packet is received.

See Section 11–14 “Slave mode operation” for a description of when this command is used.

13. USB device controller initialization

The LPC17xx USB device controller initialization includes the following steps:

1. Enable the device controller by setting the PCUSB bit of PCONP.
2. Configure and enable the PLL and Clock Dividers to provide 48 MHz for usbclk and the desired frequency for cclk. For correct operation of synchronization logic in the device controller, the minimum cclk frequency is 18 MHz. For the procedure for determining the PLL setting and configuration, see Section 4–5.11 “Procedure for determining PLL0 settings”.
3. Enable the device controller clocks by setting DEV_CLK_EN and AHB_CLK_EN bits in the USBClkCtrl register. Poll the respective clock bits in the USBClkSt register until they are set.
4. Enable the USB pin functions by writing to the corresponding PINSEL register.
5. Disable the pull-ups and pull-downs on the V_{BUS} pin using the corresponding PINMODE register by putting the pin in the “pin has neither pull-up nor pull-down resistor enabled” mode. See Section 8–4 “Pin mode select register values”.

© NXP B.V. 2010. All rights reserved.

6. Set USBEpIn and USBMaxPSize registers for EP0 and EP1, and wait until the EP_RLZED bit in USBDevIntSt is set so that EP0 and EP1 are realized.
7. Enable endpoint interrupts (Slave mode):
   - Clear all endpoint interrupts using USBEpIntClr.
   - Clear any device interrupts using USBDevIntClr.
   - Enable Slave mode for the desired endpoints by setting the corresponding bits in USBEpIntEn.
   - Set the priority of each enabled interrupt using USBEpIntPri.
   - Configure the desired interrupt mode using the SIE Set Mode command.
   - Enable device interrupts using USBDevIntEn (normally DEV_STAT, EP_SLOW, and possibly EP_FAST).
8. Configure the DMA (DMA mode):
   - Disable DMA operation for all endpoints using USBEpDMADis.
   - Clear any pending DMA requests using USBDMARClr.
   - Clear all DMA interrupts using USBEoTIntClr, USBNDDRIntClr, and USBSysErrIntClr.
   - Prepare the UDCA in system memory.
   - Write the desired address for the UDCA to USBUDCAH (for example 0x7FD0 0000).
   - Enable the desired endpoints for DMA operation using USBEpDMAEn.
   - Set EOT, DDR, and ERR bits in USBDMAIntEn.
9. Install USB interrupt handler in the NVIC by writing its address to the appropriate vector table location and enabling the USB interrupt in the NVIC.
10. Set default USB address to 0x0 and DEV_EN to 1 using the SIE Set Address command. A bus reset will also cause this to happen.
11. Set CON bit to 1 to make CONNECT active using the SIE Set Device Status command.

The configuration of the endpoints varies depending on the software application. By default, all the endpoints are disabled except control endpoints EP0 and EP1. Additional endpoints are enabled and configured by software after a SET_CONFIGURATION or SET_INTERFACE device request is received from the host.

14. Slave mode operation

In Slave mode, the CPU transfers data between RAM and the endpoint buffer using the Register Interface.

14.1 Interrupt generation

In slave mode, data packet transfer between RAM and an endpoint buffer can be initiated in response to an endpoint interrupt. Endpoint interrupts are enabled using the USBEpIntEn register, and are observable in the USBEpIntSt register.

© NXP B.V. 2010. All rights reserved.

All non-isochronous OUT endpoints generate an endpoint interrupt when they receive a packet without an error. All non-isochronous IN endpoints generate an interrupt when a packet is successfully transmitted, or when a NAK handshake is sent on the bus and the interrupt on NAK feature is enabled.

For Isochronous endpoints, transfer of data is done when the FRAME interrupt (in USBDevIntSt) occurs.

14.2 Data transfer for OUT endpoints

When the software wants to read the data from an endpoint buffer it should set the RD_EN bit and program LOG_ENDPOINT with the desired endpoint number in the USBCtrl register. The control logic will fetch the packet length to the USBRxPLen register, and set the PKT_RDY bit (Table 11–215).

Software can now start reading the data from the USBRxData register (Table 11–214). When the end of packet is reached, the RD_EN bit is cleared, and the RxENDPKT bit is set in the USBDevSt register. Software now issues a Clear Buffer (refer to Table 11–249) command. The endpoint is now ready to accept the next packet. For OUT isochronous endpoints, the next packet will be received irrespective of whether the buffer has been cleared. Any data not read from the buffer before the end of the frame is lost. See Section 11–16 “Double buffered endpoint operation” for more details.

If the software clears RD_EN before the entire packet is read, reading is terminated, and the data remains in the endpoint’s buffer. When RD_EN is set again for this endpoint, the data will be read from the beginning.

14.3 Data transfer for IN endpoints

When writing data to an endpoint buffer, WR_EN (Section 11–10.5.5 “USB Control register (USBCtrl - 0x5000 C228)”) is set and software writes to the number of bytes it is going to send in the packet to the USBTxPLen register (Section 11–10.5.4). It can then write data continuously in the USBTxData register.

When the number of bytes programmed in USBTxPLen have been written to USBTxData, the WR_EN bit is cleared, and the TxENDPKT bit is set in the USBDevIntSt register. Software issues a Validate Buffer (Section 11–12.14 “Validate Buffer (Command: 0xFA, Data: none)”) command. The endpoint is now ready to send the packet. For IN isochronous endpoints, the data in the buffer will be sent only if the buffer is validated before the next FRAME interrupt occurs; otherwise, an empty packet will be sent in the next frame. If the software clears WR_EN before the entire packet is written, writing will start again from the beginning the next time WR_EN is set for this endpoint.

Both RD_EN and WR_EN can be high at the same time for the same logical endpoint. Interleaved read and write operation is possible.

15. DMA operation

In DMA mode, the DMA transfers data between RAM and the endpoint buffer.

The following sections discuss DMA mode operation. Background information is given in sections Section 11–15.2 “USB device communication area” and Section 11–15.3 “Triggering the DMA engine”. The fields of the DMA Descriptor are described in Section

© NXP B.V. 2010. All rights reserved.

11–15.4 “The DMA descriptor”. The last three sections describe DMA operation: Section 11–15.5 “Non-isochronous endpoint operation”, Section 11–15.6 “Isochronous endpoint operation”, and Section 11–15.7 “Auto Length Transfer Extraction (ATLE) mode operation”.

15.1 Transfer terminology

Within this section three types of transfers are mentioned:

1. USB transfers – transfer of data over the USB bus. The USB 2.0 specification refers to these simply as transfers. Within this section they are referred to as USB transfers to distinguish them from DMA transfers. A USB transfer is composed of transactions. Each transaction is composed of packets.
2. DMA transfers – the transfer of data between an endpoint buffer and system memory (RAM).
3. Packet transfers – in this section, a packet transfer refers to the transfer of a packet of data between an endpoint buffer and system memory (RAM). A DMA transfer is composed of one or more packet transfers.

15.2 USB device communication area

The CPU and DMA controller communicate through a common area of memory, called the USB Device Communication Area, or UDCA. The UDCA is a 32-word array of DMA Descriptor Pointers (DDPs), each of which corresponds to a physical endpoint. Each DDP points to the start address of a DMA Descriptor, if one is defined for the endpoint. DDPs for unrealized endpoints and endpoints disabled for DMA operation are ignored and can be set to a NULL (0x0) value.

The start address of the UDCA is stored in the USBUDCAH register. The UDCA can reside at any 128-byte boundary of RAM that is accessible to both the CPU and DMA controller.

Figure 11–29 illustrates the UDCA and its relationship to the UDCA Head (USBUDCAH) register and DMA Descriptors.

© NXP B.V. 2010. All rights reserved.

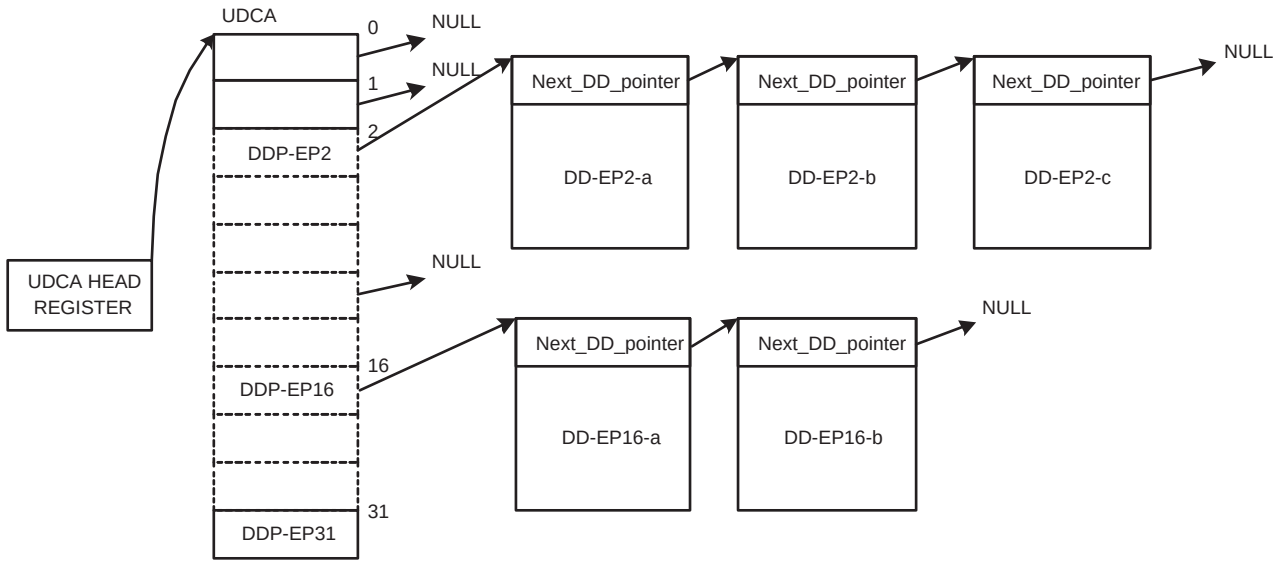


Fig 29. UDCA Head register and DMA Descriptors

15.3 Triggering the DMA engine

An endpoint raises a DMA request when Slave mode is disabled by setting the corresponding bit in the USBEpIntEn register to 0 (Section 11–10.3.2) and an endpoint interrupt occurs (see Section 11–10.7.1 “USB DMA Request Status register (USBDMARSt - 0x5000_C250)”).

A DMA transfer for an endpoint starts when the endpoint is enabled for DMA operation in USBEpDMASt, the corresponding bit in USBDMARSt is set, and a valid DD is found for the endpoint.

All endpoints share a single DMA channel to minimize hardware overhead. If more than one DMA request is active in USBDMARSt, the endpoint with the lowest physical endpoint number is processed first.

In DMA mode, the bits corresponding to Interrupt on NAK for Bulk OUT and Interrupt OUT endpoints (INAK_BO and INAK_IO) should be set to 0 using the SIE Set Mode command (Section 11–12.3).

15.4 The DMA descriptor

DMA transfers are described by a data structure called the DMA Descriptor (DD).

DDs are placed in RAM. These descriptors can be located anywhere in on-chip RAM at word-aligned addresses.

DDs for non-isochronous endpoints are four words long. DDs for isochronous endpoints are five words long.

The parameters associated with a DMA transfer are:

- The start address of the DMA buffer

© NXP B.V. 2010. All rights reserved.

- The length of the DMA buffer
- The start address of the next DMA descriptor
- Control information
- Count information (number of bytes transferred)
- Status information

Table 11–250 lists the DMA descriptor fields.

Table 250. DMA descriptor

Word position	Access (H/W)	Access (S/W)	Bit position	Description
0	R	R/W	31:0	Next_DD_pointer
1	R	R/W	1:0	DMA_mode (00 -Normal; 01 - ATLE)
	R	R/W	2	Next_DD_valid (1 - valid; 0 - invalid)
	-	-	3	Reserved
	R	R/W	4	Isochronous_endpoint (1 - isochronous; 0 - non-isochronous)
	R	R/W	15:5	Max_packet_size
	R/W[1]	R/W	31:16	DMA_buffer_length This value is specified in bytes for non-isochronous endpoints and in number of packets for isochronous endpoints.
2	R/W	R/W	31:0	DMA_buffer_start_addr
3	R/W	R/I	0	DD_retired (To be initialized to 0)
	W	R/I	4:1	DD_status (To be initialized to 0000): 0000 - NotServiced 0001 - BeingServiced 0010 - NormalCompletion 0011 - DataUnderrun (short packet) 1000 - DataOverrun 1001 - SystemError
	W	R/I	5	Packet_valid (To be initialized to 0)
	W	R/I	6	LS_byte_extracted (ATLE mode) (To be initialized to 0)
	W	R/I	7	MS_byte_extracted (ATLE mode) (To be initialized to 0)
	R	W	13:8	Message_length_position (ATLE mode)
	-	-	15:14	Reserved
	R/W	R/I	31:16	Present_DMA_count (To be initialized to 0)
4	R/W	R/W	31:0	Isochronous_packetsize_memory_address

[1] Write-only in ATLE mode

Legend: R - Read; W - Write; I - Initialize

15.4.1 Next_DD_pointer

Pointer to the memory location from where the next DMA descriptor will be fetched.

© NXP B.V. 2010. All rights reserved.

15.4.2 DMA_mode

Specifies the DMA mode of operation. Two modes have been defined: Normal and Automatic Transfer Length Extraction (ATLE) mode. In normal mode, software initializes the DMA_buffer_length for OUT endpoints. In ATLE mode, the DMA_buffer_length is extracted from the incoming data. See Section 11–15.7 “Auto Length Transfer Extraction (ATLE) mode operation” on page 261 for more details.

15.4.3 Next_DD_valid

This bit indicates whether the software has prepared the next DMA descriptor. If set, the DMA engine fetches the new descriptor when it is finished with the current one.

15.4.4 Isochronous_endpoint

When set, this bit indicates that the descriptor belongs to an isochronous endpoint. Hence 5 words have to be read when fetching it.

15.4.5 Max_packet_size

The maximum packet size of the endpoint. This parameter is used while transferring the data for IN endpoints from the memory. It is used for OUT endpoints to detect the short packet. This is applicable to non-isochronous endpoints only. This field should be set to the same MPS value that is assigned for the endpoint using the USBMaxPSize register.

15.4.6 DMA_buffer_length

This indicates the depth of the DMA buffer allocated for transferring the data. The DMA engine will stop using this descriptor when this limit is reached and will look for the next descriptor.

In Normal mode operation, software sets this value for both IN and OUT endpoints. In ATLE mode operation, software sets this value for IN endpoints only. For OUT endpoints, hardware sets this value using the extracted length of the data stream.

For isochronous endpoints, DMA_buffer_length is specified in number of packets, for non-isochronous endpoints in bytes.

15.4.7 DMA_buffer_start_addr

The address where the data is read from or written to. This field is updated each time the DMA engine finishes transferring a packet.

15.4.8 DD_retired

This bit is set by hardware when the DMA engine finishes the current descriptor. This happens when the end of the buffer is reached, a short packet is transferred (non-isochronous endpoints), or an error condition is detected.

15.4.9 DD_status

The status of the DMA transfer is encoded in this field. The following codes are defined:

- **NotServiced** - No packet has been transferred yet.
- **BeingServiced** - At least one packet is transferred.
- **NormalCompletion** - The DD is retired because the end of the buffer is reached and there were no errors. The DD_retired bit is also set.

© NXP B.V. 2010. All rights reserved.

- **DataUnderrun** - Before reaching the end of the DMA buffer, the USB transfer is terminated because a short packet is received. The DD_retired bit is also set.
- **DataOverrun** - The end of the DMA buffer is reached in the middle of a packet transfer. This is an error situation. The DD_retired bit is set. The present DMA count field is equal to the value of DMA_buffer_length. The packet must be re-transmitted from the endpoint buffer in another DMA transfer. The corresponding EPxx_DMA_ENABLE bit in USBEpDMASt is cleared.
- **SystemError** - The DMA transfer being serviced is terminated because of an error on the AHB bus. The DD_retired bit is not set in this case. The corresponding EPxx_DMA_ENABLE in USBEpDMASt is cleared. Since a system error can happen while updating the DD, the DD fields in RAM may be unreliable.

15.4.10 Packet_valid

This bit is used for isochronous endpoints. It indicates whether the last packet transferred to the memory is received with errors or not. This bit is set if the packet is valid, i.e., it was received without errors. See Section 11–15.6 “Isochronous endpoint operation” on page 259 for isochronous endpoint operation.

This bit is unnecessary for non-isochronous endpoints because a DMA request is generated only for packets without errors, and thus Packet_valid will always be set when the request is generated.

15.4.11 LS_byte_extracted

Used in ATLE mode. When set, this bit indicates that the Least Significant Byte (LSB) of the transfer length has been extracted. The extracted size is reflected in the DMA_buffer_length field, bits 23:16.

15.4.12 MS_byte_extracted

Used in ATLE mode. When set, this bit indicates that the Most Significant Byte (MSB) of the transfer size has been extracted. The size extracted is reflected in the DMA_buffer_length field, bits 31:24. Extraction stops when LS_Byte_extracted and MS_byte_extracted bits are set.

15.4.13 Present_DMA_count

The number of bytes transferred by the DMA engine. The DMA engine updates this field after completing each packet transfer.

For isochronous endpoints, Present_DMA_count is the number of packets transferred; for non-isochronous endpoints, Present_DMA_count is the number of bytes.

15.4.14 Message_length_position

Used in ATLE mode. This field gives the offset of the message length position embedded in the incoming data packets. This is applicable only for OUT endpoints. Offset 0 indicates that the message length starts from the first byte of the first packet.

15.4.15 Isochronous_packetsize_memory_address

The memory buffer address where the packet size information along with the frame number has to be transferred or fetched. See Figure 11–30. This is applicable to isochronous endpoints only.

© NXP B.V. 2010. All rights reserved.

15.5 Non-isochronous endpoint operation

15.5.1 Setting up DMA transfers

Software prepares the DMA Descriptors (DDs) for those physical endpoints to be enabled for DMA transfer. These DDs are present in on-chip RAM. The start address of the first DD is programmed into the DMA Description pointer (DDP) location for the corresponding endpoint in the UDCA. Software then sets the EPxx_DMA_ENABLE bit for this endpoint in the USBEpDMAEn register (Section 11–10.7.6).The DMA_mode bit field in the descriptor is set to '00' for normal mode operation. All other DD fields are initialized as specified in Table 11–250.

DMA operation is not supported for physical endpoints 0 and 1 (default control endpoints).

15.5.2 Finding DMA Descriptor

When there is a trigger for a DMA transfer for an endpoint, the DMA engine will first determine whether a new descriptor has to the fetched or not. A new descriptor does not have to be fetched if the last packet transferred was for the same endpoint and the DD is not yet in the retired state. An internal flag called DMA_PROCEED is used to identify this condition (see Section 11–15.5.4 "Optimizing descriptor fetch" on page 258).

If a new descriptor has to be read, the DMA engine will calculate the location of the DDP for this endpoint and will fetch the start address of the DD from this location. A DD start address at location zero is considered invalid. In this case the NDDR interrupt is raised. All other word-aligned addresses are considered valid.

When the DD is fetched, the DD status word (word 3) is read first and the status of the DD_retired bit is checked. If not set, DDP points to a valid DD. If DD_retired is set, the DMA engine will read the control word (word 1) of the DD.

If Next_DD_valid bit is set, the DMA engine will fetch the Next_DD_pointer field (word 0) of the DD and load it to the DDP. The new DDP is written to the UDCA area.

The full DD (4 words) will then be fetched from the address in the DDP. The DD will give the details of the DMA transfer to be done. The DMA engine will load its hardware resources with the information fetched from the DD (start address, DMA count etc.).

If Next_DD_valid is not set and DD_retired bit is set, the DMA engine raises the NDDR interrupt for this endpoint and clears the corresponding EPxx_DMA_ENABLE bit.

15.5.3 Transferring the data

For OUT endpoints, the current packet is read from the EP_RAM by the DMA Engine and transferred to on-chip RAM memory locations starting from DMA_buffer_start_addr. For IN endpoints, the data is fetched from on-chip RAM at DMA_buffer_start_addr and written to the EP_RAM. The DMA_buffer_start_addr and Present_DMA_count fields are updated after each packet is transferred.

15.5.4 Optimizing descriptor fetch

A DMA transfer normally involves multiple packet transfers. Hardware will not re-fetch a new DD from memory unless the endpoint changes. To indicate an ongoing multi-packet transfer, hardware sets an internal flag called DMA_PROCEED.

© NXP B.V. 2010. All rights reserved.

The DMA_PROCEED flag is cleared after the required number of bytes specified in the DMA_buffer_length field is transferred. It is also cleared when the software writes into the USBEpDMADis register. The ability to clear the DMA_PROCEED flag allows software to to force the DD to be re-fetched for the next packet transfer. Writing all zeros into the USBEpDMADis register clears the DMA_PROCEED flag without disabling DMA operation for any endpoint.

15.5.5 Ending the packet transfer

On completing a packet transfer, the DMA engine writes back the DD with updated status information to the same memory location from where it was read. The DMA_buffer_start_addr, Present_DMA_count, and the DD_status fields in the DD are updated.

A DD can have the following types of completion:

Normal completion - If the current packet is fully transferred and the Present_DMA_count field equals the DMA_buffer_length, the DD has completed normally. The DD will be written back to memory with DD_retired set and DD_status set to NormalCompletion. The EOT interrupt is raised for this endpoint.

USB transfer end completion - If the current packet is fully transferred and its size is less than the Max_packet_size field, and the end of the DMA buffer is still not reached, the USB transfer end completion occurs. The DD will be written back to the memory with DD_retired set and DD_Status set to the DataUnderrun completion code. The EOT interrupt is raised for this endpoint.

Error completion - If the current packet is partially transferred i.e. the end of the DMA buffer is reached in the middle of the packet transfer, an error situation occurs. The DD is written back with DD_retired set and DD_status set to the DataOverrun status code. The EOT interrupt is raised for this endpoint and the corresponding bit in USBEpDMASt register is cleared. The packet will be re-sent from the endpoint buffer to memory when the corresponding EPxx_DMA_ENABLE bit is set again using the USBEpDMAEn register.

15.5.6 No_Packet DD

For an IN transfer, if the system does not have any data to send for a while, it can respond to an NDDR interrupt by programming a No_Packet DD. This is done by setting both the Max_packet_size and DMA_buffer_length fields in the DD to 0. On processing a No_Packet DD, the DMA engine clears the DMA request bit in USBDMARSt corresponding to the endpoint without transferring a packet. The DD is retired with a status code of NormalCompletion. This can be repeated as often as necessary. The device will respond to IN token packets on the USB bus with a NAK until a DD with a data packet is programmed and the DMA transfers the packet into the endpoint buffer.

15.6 Isochronous endpoint operation

For isochronous endpoints, the packet size can vary for each packet. There is one packet per isochronous endpoint for each frame.

15.6.1 Setting up DMA transfers

Software sets the isochronous endpoint bit to 1 in the DD, and programs the initial value of the Isochronous_packetsize_memory_address field. All other fields are initialized the same as for non-isochronous endpoints.

© NXP B.V. 2010. All rights reserved.

For isochronous endpoints, the DMA_buffer_length and Present_DMA_count fields are in frames rather than bytes.

15.6.2 Finding the DMA Descriptor

Finding the descriptors is done in the same way as that for a non-isochronous endpoint.

A DMA request will be placed for DMA-enabled isochronous endpoints on every FRAME interrupt. On processing the request, the DMA engine will fetch the descriptor and if Isochronous_endpoint is set, will fetch the Isochronous_packetsize_memory_address from the fifth word of the DD.

15.6.3 Transferring the Data

The data is transferred to or from the memory location DMA_buffer_start_addr. After the end of the packet transfer the Present_DMA_count value is incremented by 1.

The isochronous packet size is stored in memory as shown in Figure 11–30. Each word in the packet size memory shown is divided into fields: Frame_number (bits 31 to 17), Packet_valid (bit 16), and Packet_length (bits 15 to 0). The space allocated for the packet size memory for a given DD should be DMA_buffer_length words in size – one word for each packet to transfer.

OUT endpoints

At the completion of each frame, the packet size is written to the address location in Isochronous_packet_size_memory_address, and Isochronous_packet_size_memory_address is incremented by 4.

IN endpoints

Only the Packet_length field of the isochronous packet size word is used. For each frame, an isochronous data packet of size specified by this field is transferred from the USB device to the host, and Isochronous_packet_size_memory_address is incremented by 4 at the end of the packet transfer. If Packet_length is zero, an empty packet will be sent by the USB device.

15.6.4 DMA descriptor completion

DDs for isochronous endpoints can only end with a status code of NormalCompletion since there is no short packet on Isochronous endpoints, and the USB transfer continues indefinitely until a SystemError occurs. There is no DataOverrun detection for isochronous endpoints.

15.6.5 Isochronous OUT Endpoint Operation Example

Assume that an isochronous endpoint is programmed for the transfer of 10 frames and that the transfer begins when the frame number is 21. After transferring four frames with packet sizes of 10,15, 8 and 20 bytes without errors, the descriptor and memory map appear as shown in Figure 11–30.

The_total_number_of_bytes_transferred = 0x0A + 0x0F + 0x08 + 0x14 = 0x35.

The Packet_valid bit (bit 16) of all the words in the packet length memory is set to 1.

© NXP B.V. 2010. All rights reserved.

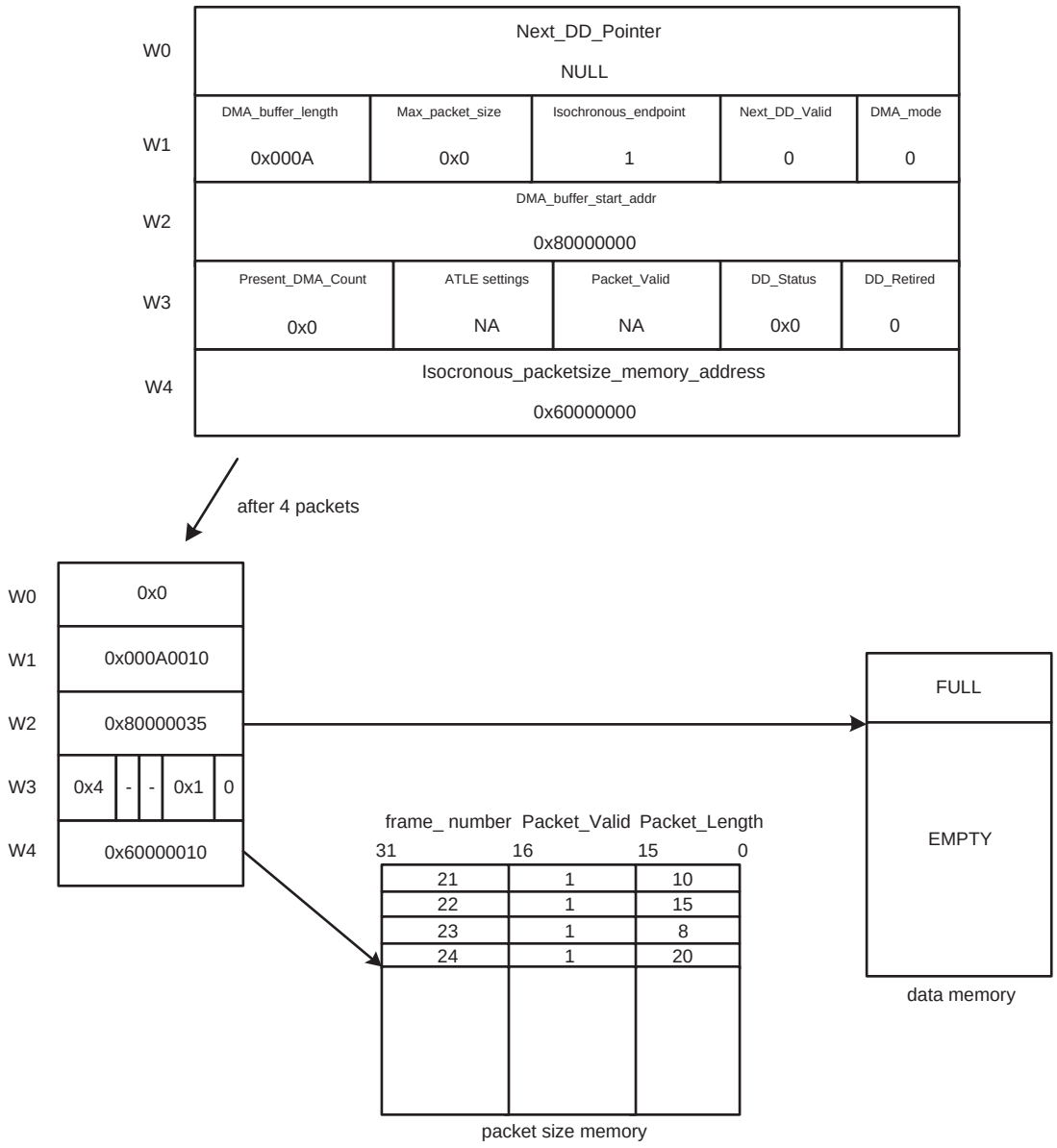


Fig 30. Isochronous OUT endpoint operation example

15.7 Auto Length Transfer Extraction (ATLE) mode operation

Some host drivers such as NDIS (Network Driver Interface Specification) host drivers are capable of concatenating small USB transfers (delta transfers) to form a single large USB transfer. For OUT USB transfers, the device hardware has to break up this concatenated transfer back into the original delta transfers and transfer them to separate DMA buffers. This is achieved by setting the DMA mode to Auto Transfer Length Extraction (ATLE) mode in the DMA descriptor. ATLE mode is supported for Bulk endpoints only.

OUT transfers in ATLE mode

© NXP B.V. 2010. All rights reserved.

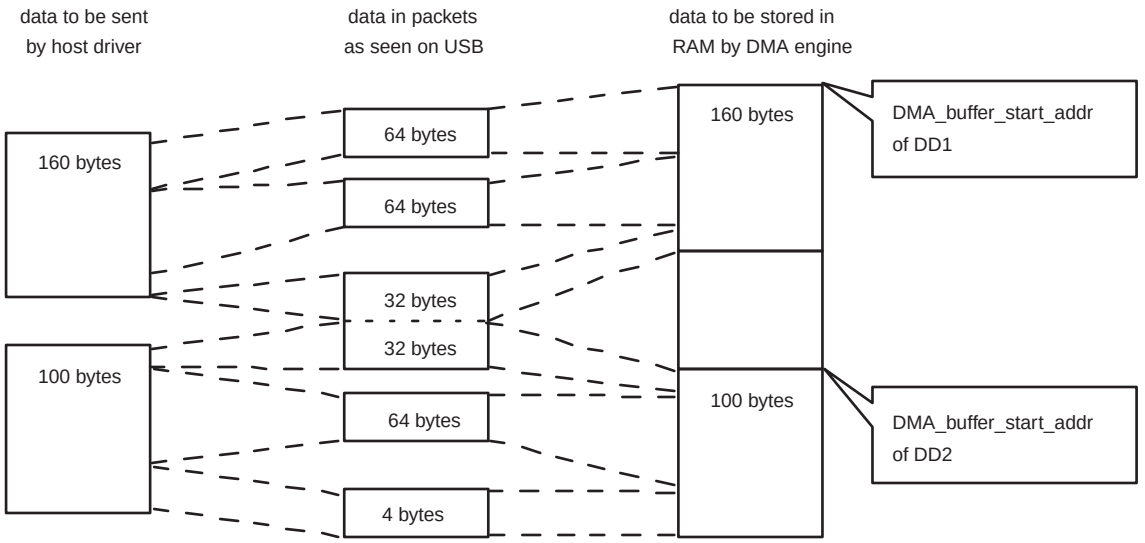


Fig 31. Data transfer in ATLE mode

Figure 11–31 shows a typical OUT USB transfer in ATLE mode, where the host concatenates two USB transfers of 160 bytes and 100 bytes, respectively. Given a MaxPacketSize of 64, the device hardware interprets this USB transfer as four packets of 64 bytes and a short packet of 4 bytes. The third and fourth packets are concatenated. Note that in Normal mode, the USB transfer would be interpreted as packets of 64, 64, 32, and 64 and 36 bytes.

It is now the responsibility of the DMA engine to separate these two USB transfers and put them in the memory locations in the DMA_buffer_start_addr field of DMA Descriptor 1 (DD1) and DMA Descriptor 2 (DD2).

Hardware reads the two-byte-wide DMA_buffer_length at the offset (from the start of the USB transfer) specified by Message_length_position from the incoming data packets and writes it in the DMA_buffer_length field of the DD. To ensure that both bytes of the DMA_buffer_length are extracted in the event they are split between two packets, the flags LS_byte_extracted and MS_byte_extracted are set by hardware after the respective byte is extracted. After the extraction of the MS byte, the DMA transfer continues as in the normal mode.

The flags LS_byte_extracted and MS_byte_extracted are set to 0 by software when preparing a new DD. Therefore, once a DD is retired, the transfer length is extracted again for the next DD.

If DD1 is retired during the transfer of a concatenated packet (such as the third packet in Figure 11–31), and DD2 is not programmed (Next_DD_valid field of DD1 is 0), then DD1 is retired with DD_status set to the DataOverrun status code. This is treated as an error condition and the corresponding EPxx_DMA_ENABLE bit of USBEpDMASt is cleared by hardware.

© NXP B.V. 2010. All rights reserved.

In ATLE mode, the last buffer length to be transferred always ends with a short or empty packet indicating the end of the USB transfer. If the concatenated transfer lengths are such that the USB transfer ends on a MaxPacketSize packet boundary, the (NDIS) host will send an empty packet to mark the end of the USB transfer.

IN transfers in ATLE mode

For IN USB transfers from the device to the host, DMA_buffer_length is set by the device software as in normal mode.

In ATLE mode, the device concatenates data from multiple DDs to form a single USB transfer. If a DD is retired in the middle of a packet (packet size is less than MaxPacketSize), the next DD referenced by Next_DD_pointer is fetched, and the remaining bytes to form a packet of MaxPacketSize are transferred from the next DD's buffer.

If the next DD is not programmed (i.e. Next_DD_valid field in DD is 0), and the DMA buffer length for the current DD has completed before the MaxPacketSize packet boundary, then the available bytes from current DD are sent as a short packet on USB, which marks the end of the USB transfer for the host.

If the last buffer length completes on a MaxPacketSize packet boundary, the device software must program the next DD with DMA_buffer_length field 0, so that an empty packet is sent by the device to mark the end of the USB transfer for the host.

15.7.1 Setting up the DMA transfer

For OUT endpoints, the host hardware needs to set the field Message_length_position in the DD. This indicates the start location of the message length in the incoming data packets. Also the device software has to set the DMA_buffer_length field to 0 for OUT endpoints because this field is updated by the device hardware after the extraction of the buffer length.

For IN endpoints, descriptors are set in the same way as in normal mode operation.

Since a single packet can be split between two DDs, software should always keep two DDs ready, except for the last DMA transfer which ends with a short or empty packet.

15.7.2 Finding the DMA Descriptor

DMA descriptors are found in the same way as the normal mode operation.

15.7.3 Transferring the Data

OUT endpoints

If the LS_byte_extracted or MS_byte_extracted bit in the status field is not set, the hardware will extract the transfer length from the data stream and program DMA_buffer_length. Once the extraction is complete both the LS_byte_extracted and MS_byte_extracted bits will be set.

IN endpoints

The DMA transfer proceeds as in normal mode and continues until the number of bytes transferred equals the DMA_buffer_length.

© NXP B.V. 2010. All rights reserved.

15.7.4 Ending the packet transfer

The DMA engine proceeds with the transfer until the number of bytes specified in the field DMA_buffer_length is transferred to or from on-chip RAM. Then the EOT interrupt will be generated. If this happens in the middle of the packet, the linked DD will get loaded and the remaining part of the packet gets transferred to or from the address pointed by the new DD.

OUT endpoints

If the linked DD is not valid and the packet is partially transferred to memory, the DD ends with DataOverrun status code set, and the DMA will be disabled for this endpoint. Otherwise DD_status will be updated with the NormalCompletion status code.

IN endpoints

If the linked DD is not valid and the packet is partially transferred to USB, the DD ends with a status code of NormalCompletion in the DD_status field. This situation corresponds to the end of the USB transfer, and the packet will be sent as a short packet. Also, when the linked DD is valid and buffer length is 0, an empty packet will be sent to indicate the end of the USB transfer.

16. Double buffered endpoint operation

The Bulk and Isochronous endpoints of the USB Device Controller are double buffered to increase data throughput.

When a double-buffered endpoint is realized, enough space for both endpoint buffers is automatically allocated in the EP_RAM. See Section 11–10.4.1.

For the following discussion, the endpoint buffer currently accessible to the CPU or DMA engine for reading or writing is said to be the active buffer.

16.1 Bulk endpoints

For Bulk endpoints, the active endpoint buffer is switched by the SIE Clear Buffer or Validate Buffer commands.

The following example illustrates how double buffering works for a Bulk OUT endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty, and that the active buffer is B_1.

1. The host sends a data packet to the endpoint. The device hardware puts the packet into B_1, and generates an endpoint interrupt.
2. Software clears the endpoint interrupt and begins reading the packet data from B_1. While B_1 is still being read, the host sends a second packet, which device hardware places in B_2, and generates an endpoint interrupt.
3. Software is still reading from B_1 when the host attempts to send a third packet. Since both B_1 and B_2 are full, the device hardware responds with a NAK.
4. Software finishes reading the first packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.

© NXP B.V. 2010. All rights reserved.

5. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. Software finds that the active buffer (B_2) has data (FE=1). Software clears the endpoint interrupt and begins reading the contents of B_2.
6. The host re-sends the third packet which device hardware places in B_1. An endpoint interrupt is generated.
7. Software finishes reading the second packet from B_2 and sends a SIE Clear Buffer command to free B_2 to receive another packet. B_1 becomes the active buffer. Software waits for the next endpoint interrupt to occur (it already has been generated back in step 6).
8. Software responds to the endpoint interrupt by clearing it and begins reading the third packet from B_1.
9. Software finishes reading the third packet from B_1 and sends a SIE Clear Buffer command to free B_1 to receive another packet. B_2 becomes the active buffer.
10. Software tests the FE bit and finds that the active buffer (B_2) is empty (FE=0).
11. Both B_1 and B_2 are empty. Software waits for the next endpoint interrupt to occur. The active buffer is now B_2. The next data packet sent by the host will be placed in B_2.

The following example illustrates how double buffering works for a Bulk IN endpoint in Slave mode:

Assume that both buffer 1 (B_1) and buffer 2 (B_2) are empty and that the active buffer is B_1. The interrupt on NAK feature is enabled.

1. The host requests a data packet by sending an IN token packet. The device responds with a NAK and generates an endpoint interrupt.
2. Software clears the endpoint interrupt. The device has three packets to send. Software fills B_1 with the first packet and sends a SIE Validate Buffer command. The active buffer is switched to B_2.
3. Software sends the SIE Select Endpoint command to read the Select Endpoint Register and test the FE bit. It finds that B_2 is empty (FE=0) and fills B_2 with the second packet. Software sends a SIE Validate Buffer command, and the active buffer is switched to B_1.
4. Software waits for the endpoint interrupt to occur.
5. The device successfully sends the packet in B_1 and clears the buffer. An endpoint interrupt occurs.
6. Software clears the endpoint interrupt. Software fills B_1 with the third packet and validates it using the SIE Validate Buffer command. The active buffer is switched to B_2.
7. The device successfully sends the second packet from B_2 and generates an endpoint interrupt.
8. Software has no more packets to send, so it simply clears the interrupt.
9. The device successfully sends the third packet from B_1 and generates an endpoint interrupt.
10. Software has no more packets to send, so it simply clears the interrupt.

© NXP B.V. 2010. All rights reserved.

11. Both B_1 and B_2 are empty, and the active buffer is B_2. The next packet written by software will go into B_2.

In DMA mode, switching of the active buffer is handled automatically in hardware. For Bulk IN endpoints, proactively filling an endpoint buffer to take advantage of the double buffering can be accomplished by manually starting a packet transfer using the USBDMARSet register.

16.2 Isochronous endpoints

For isochronous endpoints, the active data buffer is switched by hardware when the FRAME interrupt occurs. The SIE Clear Buffer and Validate Buffer commands do not cause the active buffer to be switched.

Double buffering allows the software to make full use of the frame interval writing or reading a packet to or from the active buffer, while the packet in the other buffer is being sent or received on the bus.

For an OUT isochronous endpoint, any data not read from the active buffer before the end of the frame is lost when it switches.

For an IN isochronous endpoint, if the active buffer is not validated before the end of the frame, an empty packet is sent on the bus when the active buffer is switched, and its contents will be overwritten when it becomes active again.

© NXP B.V. 2010. All rights reserved.

1. How to read this chapter

The USB host controller is available on the LPC1768, LPC1766, LPC1765, LPC1758, LPC1756, and LPC1754. On these devices, the USB controller can be configured for device, Host, or OTG operation.

2. Basic configuration

The USB controller is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCUSB.

   Remark: On reset, the USB block is disabled (PCUSB = 0).
2. Clock: The USB block can be used with a dedicated USB PLL (PLL1) to obtain the USB clock or with the Main PLL (PLL0). See Section 4–6.1.
3. Pins: Select USB pins and their modes in PINSEL0 to PINSEL5 and PINMODE0 to PINMODE5 (Section 8–5).
4. Wake-up: Activity on the USB bus port can wake up the microcontroller from Power-down mode, see Section 4–8.8.
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. Initialization: see Section 13–11.

3. Introduction

This section describes the host portion of the USB 2.0 OTG dual role core which integrates the host controller (OHCI compliant), device controller, and I^2C interface. The I^2C interface controls the external OTG ATX.

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

The host controller enables data exchange with various USB devices attached to the bus. It consists of register interface, serial interface engine and DMA controller. The register interface complies to the OHCI specification.

Table 251. USB (OHCI) related acronyms and abbreviations used in this chapter

Acronym/abbreviation	Description
AHB	Advanced High-Performance Bus
ATX	Analog Transceiver
DMA	Direct Memory Access
FS	Full Speed

© NXP B.V. 2010. All rights reserved.

Table 251. USB (OHCI) related acronyms and abbreviations used in this chapter

Acronym/abbreviation	Description
LS	Low Speed
OHCI	Open Host Controller Interface
USB	Universal Serial Bus

3.1 Features

- OHCI compliant.
- OpenHCI specifies the operation and interface of the USB Host Controller and SW Driver
  - USBOperational: Process Lists and generate SOF Tokens.
  - USBReset: Forces reset signaling on the bus, SOF disabled.
  - USBSuspend: Monitor USB for wake-up activity.
  - USBResume: Forces resume signaling on the bus.
- The Host Controller has four USB states visible to the SW Driver.
- HCCA register points to Interrupt and Isochronous Descriptors List.
- ControlHeadED and BulkHeadED registers point to Control and Bulk Descriptors List.

3.2 Architecture

The architecture of the USB host controller is shown below in Figure 12–32.

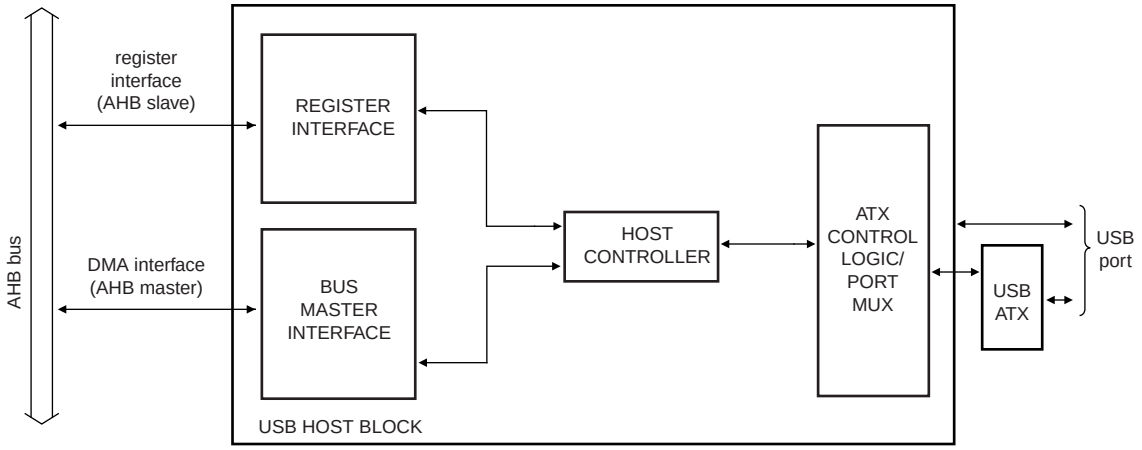


Fig 32. USB Host controller block diagram

4. Interfaces

The USB interface is controlled by the OTG controller. It has one USB port.

© NXP B.V. 2010. All rights reserved.

4.1 Pin description

Table 252. USB Host port pins

Pin name	Direction	Description	Type
USB_D+	I/O	Positive differential data	USB Connector
USB_D−	I/O	Negative differential data	USB Connector
USB_UP_LED	O	GoodLink LED control signal	Control
$\overline{\text{USB\_PPWR}}$	O	Port power enable	Host power switch
USB_PWRD	I	Port power status	Host power switch
$\overline{\text{USB\_OVRCR}}$	I	Over-current status	Host power switch

4.1.1 USB host usage note

The USB block can be configured as USB host. For details on how to connect the USB port, see the USB OTG chapter, Section 13–7.

The USB device/host/OTG controller is disabled after RESET and must be enabled by writing a 1 to the PCUSB bit in the PCONP register, see Table 4–46.

4.2 Software interface

The software interface of the USB host block consists of a register view and the format definitions for the endpoint descriptors. For details on these two aspects see the OHCI specification. The register map is shown in the next subsection.

4.2.1 Register map

The following registers are located in the AHB clock ‘cclk’ domain. They can be accessed directly by the processor. All registers are 32-bit wide and aligned in the word address boundaries.

Table 253. USB Host register address definitions

Name	Address	R/W[1]	Function	Reset value
HcRevision	0x5000 C000	R	BCD representation of the version of the HCI specification that is implemented by the Host Controller.	0x10
HcControl	0x5000 C004	R/W	Defines the operating modes of the HC.	0x0
HcCommandStatus	0x5000 C008	R/W	This register is used to receive the commands from the Host Controller Driver (HCD). It also indicates the status of the HC.	0x0
HcInterruptStatus	0x5000 C00C	R/W	Indicates the status on various events that cause hardware interrupts by setting the appropriate bits.	0x0
HcInterruptEnable	0x5000 C010	R/W	Controls the bits in the HcInterruptStatus register and indicates which events will generate a hardware interrupt.	0x0
HcInterruptDisable	0x5000 C014	R/W	The bits in this register are used to disable corresponding bits in the HCInterruptStatus register and in turn disable that event leading to hardware interrupt.	0x0
HcHCCA	0x5000 C018	R/W	Contains the physical address of the host controller communication area.	0x0
HcPeriodCurrentED	0x5000 C01C	R	Contains the physical address of the current isochronous or interrupt endpoint descriptor.	0x0

© NXP B.V. 2010. All rights reserved.

Table 253. USB Host register address definitions *…continued*

Name	Address	R/W[1]	Function	Reset value
HcControlHeadED	0x5000 C020	R/W	Contains the physical address of the first endpoint descriptor of the control list.	0x0
HcControlCurrentED	0x5000 C024	R/W	Contains the physical address of the current endpoint descriptor of the control list	0x0
HcBulkHeadED	0x5000 C028	R/W	Contains the physical address of the first endpoint descriptor of the bulk list.	0x0
HcBulkCurrentED	0x5000 C02C	R/W	Contains the physical address of the current endpoint descriptor of the bulk list.	0x0
HcDoneHead	0x5000 C030	R	Contains the physical address of the last transfer descriptor added to the ‘Done’ queue.	0x0
HcFmInterval	0x5000 C034	R/W	Defines the bit time interval in a frame and the full speed maximum packet size which would not cause an overrun.	0x2EDF
HcFmRemaining	0x5000 C038	R	A 14-bit counter showing the bit time remaining in the current frame.	0x0
HcFmNumber	0x5000 C03C	R	Contains a 16-bit counter and provides the timing reference among events happening in the HC and the HCD.	0x0
HcPeriodicStart	0x5000 C040	R/W	Contains a programmable 14-bit value which determines the earliest time HC should start processing a periodic list.	0x0
HcLSThreshold	0x5000 C044	R/W	Contains 11-bit value which is used by the HC to determine whether to commit to transfer a maximum of 8-byte LS packet before EOF.	0x628h
HcRhDescriptorA	0x5000 C048	R/W	First of the two registers which describes the characteristics of the root hub.	0xFF000902
HcRhDescriptorB	0x5000 C04C	R/W	Second of the two registers which describes the characteristics of the Root Hub.	0x60000h
HcRhStatus	0x5000 C050	R/W	This register is divided into two parts. The lower D-word represents the hub status field and the upper word represents the hub status change field.	0x0
HcRhPortStatus[1]	0x5000 C054	R/W	Controls and reports the port events on a per-port basis.	0x0
HcRhPortStatus[2]	0x5000 C058	R/W	Controls and reports the port events on a per port basis.	0x0
Module_ID/Ver_Rev_ID	0x5000 C0FC	R	IP number, where yy (0x00) is unique version number and zz (0x00) is a unique revision number.	0x3505yyzz

[1] The R/W column in Table 12–253 lists the accessibility of the register:

a) Registers marked ‘R’ for access will return their current value when read.

b) Registers marked ‘R/W’ allow both read and write.

4.2.2 USB Host Register Definitions

Refer to the OHCI specification document on the Compaq website for register definitions.

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 13: LPC17xx USB OTG controller

Rev. 01 — 4 January 2010 User manual

1. How to read this chapter

The USB OTG controller is available in the LPC1768, LPC1766, LPC1765, LPC1758, LPC1756, and LPC1754. On these devices, the USB controller can be configured for device, Host, or OTG operation.

2. Basic configuration

The USB controller is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCUSB.

   Remark: On reset, the USB block is disabled (PCUSB = 0).
2. Clock: The USB clock can generated using the dedicated USB PLL (PLL1) or with the Main PLL (PLL0). See Section 4–6.1.
3. Pins: Select USB pins and their modes in PINSEL0 to PINSEL5 and PINMODE0 to PINMODE5 (Section 8–5).
4. Wake-up: Activity on the USB bus port can wake up the microcontroller from Power-down mode (see Section 13–10.2 and Section 4–8.8).
5. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
6. Initialization: see Section 13–11.

3. Introduction

This chapter describes the OTG and I^2C portions of the USB 2.0 OTG dual role device controller which integrates the (OHCI) host controller, device controller, and I^2C. The I^2C interface that is part of the USB block is intended to control an external OTG transceiver, and is not the same as the I^2C peripherals described in Section 19–1.

USB OTG (On-The-Go) is a supplement to the USB 2.0 specification that augments the capability of existing mobile devices and USB peripherals by adding host functionality for connection to USB peripherals. The specification and more information on USB OTG can be found on the USB Implementers Forum web site.

4. Features

- Fully compliant with On-The-Go supplement to the *USB 2.0 Specification, Revision 1.0a*.
- Hardware support for Host Negotiation Protocol (HNP).
- Includes a programmable timer required for HNP and SRP.
- Supports any OTG transceiver compliant with the *OTG Transceiver Specification (CEA-2011), Rev. 1.0*.

© NXP B.V. 2010. All rights reserved.

5. Architecture

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

The host, device, OTG, and I^2C controllers can be programmed through the register interface. The OTG controller enables dynamic switching between host and device roles through the HNP protocol. One port may be connected to an external OTG transceiver to support an OTG connection. The communication between the register interface and an external OTG transceiver is handled through an I^2C interface and through the external OTG transceiver interrupt signal.

For USB connections that use the device or host controller only (not OTG), the ports use an embedded USB Analog Transceiver (ATX).

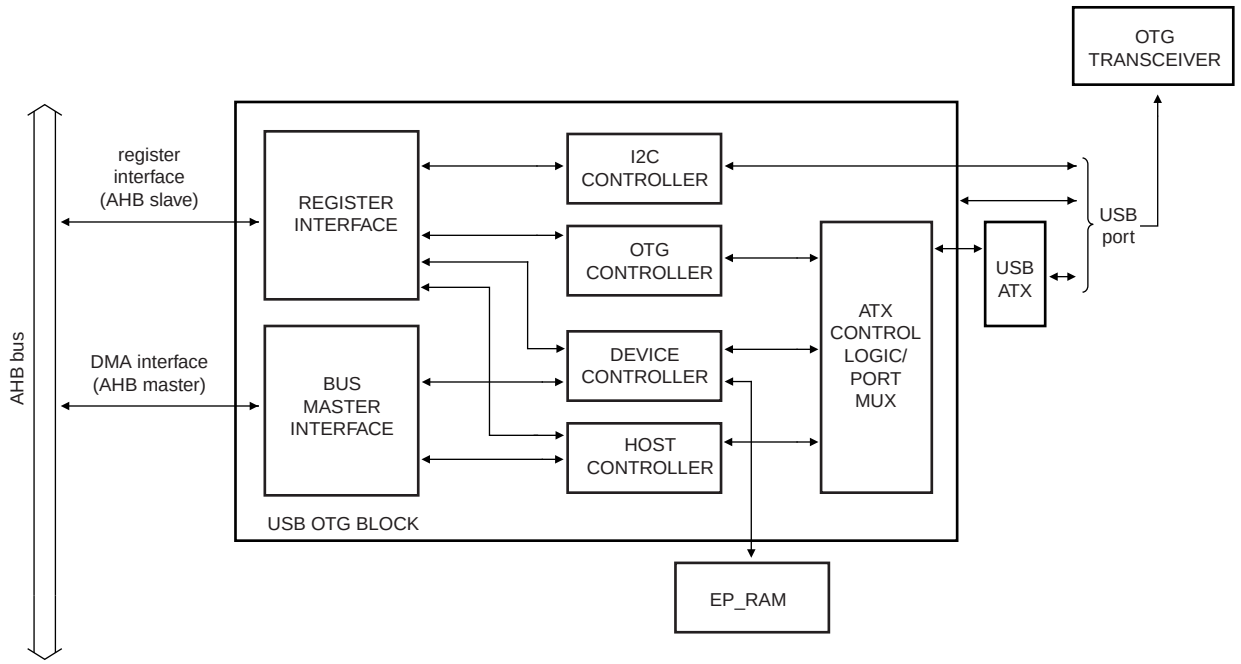


Fig 33. USB OTG controller block diagram

6. Modes of operation

The OTG controller is capable of operating in the following modes:

- Host mode (see Figure 13–34)
- Device mode (see Figure 13–35)
- OTG mode (see Figure 13–36)

© NXP B.V. 2010. All rights reserved.

7. Pin configuration

The OTG controller has one USB port.

Table 254. USB OTG port pins

Pin name	Direction	Description	Pin category
USB_D+	I/O	Positive differential data	USB Connector
USB_D−	I/O	Negative differential data	USB Connector
USB_UP_LED	O	GoodLink LED control signal	Control
USB_SCL	I/O	I^2C serial clock	External OTG transceiver
USB_SDA	I/O	I^2C serial data	External OTG transceiver

The following figures show different ways to realize connections to an USB device. The example described here uses an ISP1302 (ST-Ericsson) for the external OTG transceiver and the USB Host power switch LM3526-L (National Semiconductors).

7.1 Connecting the USB port to an external OTG transceiver

For OTG functionality an external OTG transceiver must be connected to the LPC17xx: Use the internal USB transceiver for USB signalling and use the external OTG transceiver for OTG functionality only (see Figure 13–34). This option uses the internal transceiver in VP/VM mode.

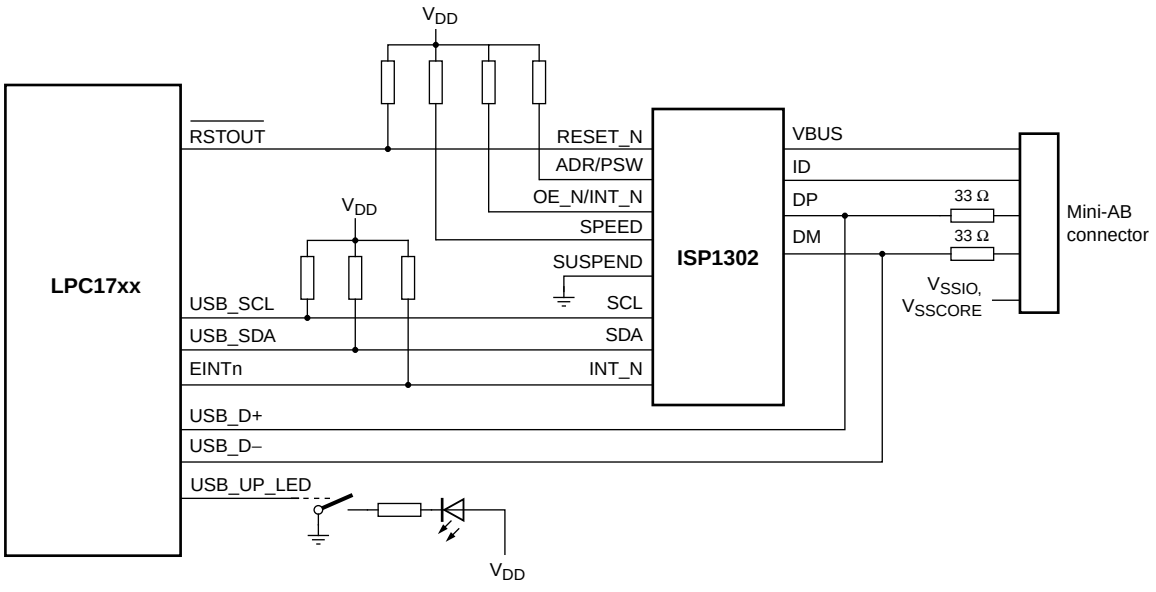


Fig 34. USB OTG port configuration

© NXP B.V. 2010. All rights reserved.

7.2 Connecting USB as a host

The USB port is connected as host using an embedded USB transceiver. There is no OTG functionality on the port.

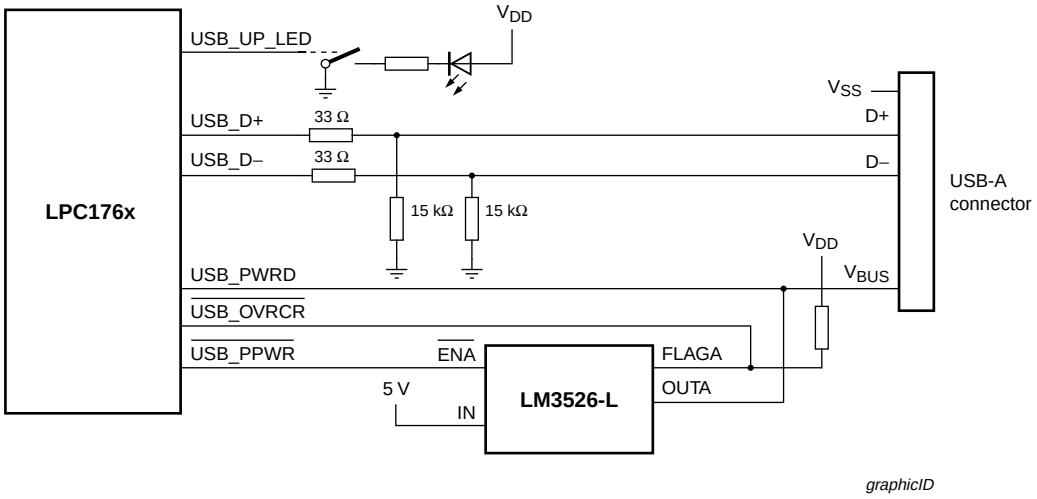


Fig 35. USB host port configuration

7.3 Connecting USB as device

The USB port is connected as device. There is no OTG functionality on the USB port.

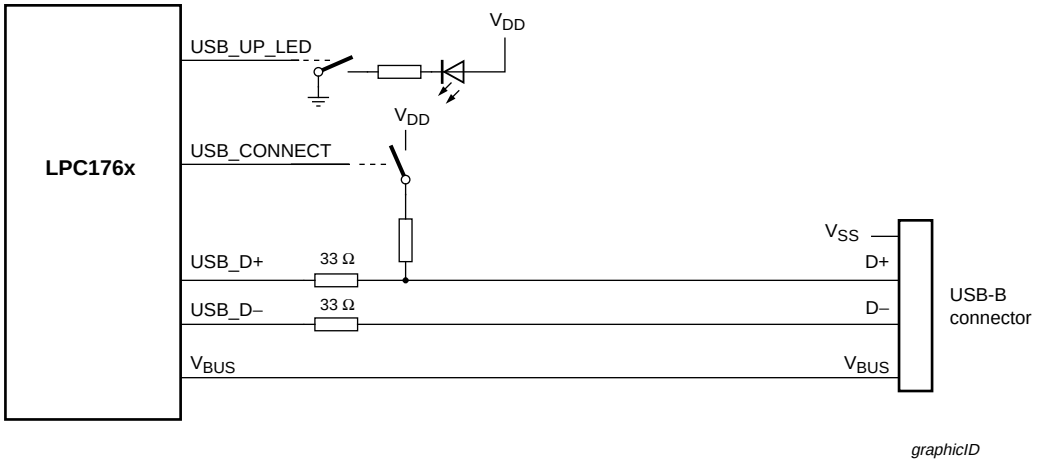


Fig 36. USB device port configuration

© NXP B.V. 2010. All rights reserved.

8. Register description

The OTG and I²C registers are summarized in the following table.

The Device and Host registers are explained in Table 12–253 and Table 11–187 in the USB Device Controller and USB Host (OHCI) Controller chapters. All registers are 32 bits wide and aligned to word address boundaries.

Table 255. USB OTG and I²C register address definitions

Name	Description	Access	Reset value	Address
Interrupt register				
USBIntSt	USB Interrupt Status	R/W	0x8000 0100	0x400F C1C0
OTG registers				
OTGIntSt	OTG Interrupt Status	RO	0	0x5000 C100
OTGIntEn	OTG Interrupt Enable	R/W	0	0x5000 C104
OTGIntSet	OTG Interrupt Set	WO	NA	0x5000 C108
OTGIntClr	OTG Interrupt Clear	WO	NA	0x5000 C10C
OTGStCtrl	OTG Status and Control	R/W	0	0x5000 C110
OTGTmr	OTG Timer	R/W	0xFFFF	0x5000 C114
I²C registers				
I2C_RX	I²C Receive	RO	NA	0x5000 C300
I2C_TX	I²C Transmit	WO	NA	0x5000 C300
I2C_STS	I²C Status	RO	0x0A00	0x5000 C304
I2C_CTL	I²C Control	R/W	0	0x5000 C308
I2C_CLKHI	I²C Clock High	R/W	0xB9	0x5000 C30C
I2C_CLKLO	I²C Clock Low	WO	0xB9	0x5000 C310
Clock control registers				
OTGClkCtrl	OTG clock controller	R/W	0	0x5000 CFF4
OTGClkSt	OTG clock status	RO	0	0x5000 CFF8

8.1 USB Interrupt Status Register (USBIntSt - 0x5000 C1C0)

The USB OTG controller has seven interrupt lines. This register allows software to determine their status with a single read operation.

The interrupt lines are ORed together to a single channel of the vectored interrupt controller.

Table 256. USB Interrupt Status register - (USBIntSt - address 0x5000 C1C0) bit description

Bit	Symbol	Description	Reset Value
0	USB_INT_REQ_LP	Low priority interrupt line status. This bit is read-only.	0
1	USB_INT_REQ_HP	High priority interrupt line status. This bit is read-only.	0
2	USB_INT_REQ_DMA	DMA interrupt line status. This bit is read-only.	0
3	USB_HOST_INT	USB host interrupt line status. This bit is read-only.	0
4	USB_ATX_INT	External ATX interrupt line status. This bit is read-only.	0
5	USB_OTG_INT	OTG interrupt line status. This bit is read-only.	0

© NXP B.V. 2010. All rights reserved.

Table 256. USB Interrupt Status register - (USBIntSt - address 0x5000 C1C0) bit description

Bit	Symbol	Description	Reset Value
6	USB_I2C_INT	I^2C module interrupt line status. This bit is read-only.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	USB_NEED_CLK	USB need clock indicator. This bit is read-only.	1
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 NVIC does not see the ORed output of the USB interrupt lines.	1

8.2 OTG Interrupt Status Register (OTGIntSt - 0x5000 C100)

Bits in this register are set by hardware when the interrupt event occurs during the HNP handoff sequence. See Section 13–9 for more information on when these bits are set.

Table 257. OTG Interrupt Status register (OTGIntSt - address 0x5000 C100) bit description

Bit	Symbol	Description	Reset Value
0	TMR	Timer time-out.	0
1	REMOVE_PU	Remove pull-up. This bit is set by hardware to indicate that software needs to disable the D+ pull-up resistor.	0
2	HNP_FAILURE	HNP failed. This bit is set by hardware to indicate that the HNP switching has failed.	0
3	HNP_SUCCESS	HNP succeeded. This bit is set by hardware to indicate that the HNP switching has succeeded.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.3 OTG Interrupt Enable Register (OTGIntEn - 0x5000 C104)

Writing a one to a bit in this register enables the corresponding bit in OTGIntSt to generate an interrupt on one of the interrupt lines. The interrupt is routed to the USB_OTG_INT interrupt line in the USBIntSt register.

The bit allocation and reset value of OTGIntEn is the same as OTGIntSt.

8.4 OTG Interrupt Set Register (OTGIntSet - 0x5000 C20C)

Writing a one to a bit in this register will set the corresponding bit in the OTGIntSt register. Writing a zero has no effect. The bit allocation of OTGIntSet is the same as in OTGIntSt.

© NXP B.V. 2010. All rights reserved.

8.5 OTG Interrupt Clear Register (OTGIntClr - 0x5000 C10C)

Writing a one to a bit in this register will clear the corresponding bit in the OTGIntSt register. Writing a zero has no effect. The bit allocation of OTGIntClr is the same as in OTGIntSt.

8.6 OTG Status and Control Register (OTGStCtrl - 0x5000 C110)

The OTGStCtrl register allows enabling hardware tracking during the HNP hand over sequence, controlling the OTG timer, monitoring the timer count, and controlling the functions mapped to port U1 and U2.

Time critical events during the switching sequence are controlled by the OTG timer. The timer can operate in two modes:

1. Monoshot mode: an interrupt is generated at the end of TIMEOUT_CNT (see Section 13–8.7 “OTG Timer Register (OTGTmr - 0x5000 C114)”), the TMR bit is set in OTGIntSt, and the timer will be disabled.
2. Free running mode: an interrupt is generated at the end of TIMEOUT_CNT (see Section 13–8.7 “OTG Timer Register (OTGTmr - 0x5000 C114)”), the TMR bit is set, and the timer value is reloaded into the counter. The timer is not disabled in this mode.

Table 258. OTG Status Control register (OTGStCtrl - address 0x5000 C110) bit description

Bit	Symbol	Description	Reset Value
1:0	PORT_FUNC	Controls port function. Bit 0 is set or cleared by hardware when B_HNP_TRACK or A_HNP_TRACK is set and HNP succeeds. See Section 13–9. Bit 1 is reserved.	-
3:2	TMR_SCALE	Timer scale selection. This field determines the duration of each timer count. 00: 10 μs (100 KHz) 01: 100 μs (10 KHz) 10: 1000 μs (1 KHz) 11: Reserved	0x0
4	TMR_MODE	Timer mode selection. 0: monoshot 1: free running	0
5	TMR_EN	Timer enable. When set, TMR_CNT increments. When cleared, TMR_CNT is reset to 0.	0
6	TMR_RST	Timer reset. Writing one to this bit resets TMR_CNT to 0. This provides a single bit control for the software to restart the timer when the timer is enabled.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	B_HNP_TRACK	Enable HNP tracking for B-device (peripheral), see Section 13–9. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
9	A_HNP_TRACK	Enable HNP tracking for A-device (host), see Section 13–9. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0

© NXP B.V. 2010. All rights reserved.

Table 258. OTG Status Control register (OTGStCtrl - address 0x5000 C110) bit description

Bit	Symbol	Description	Reset Value
10	PU_REMOVED	When the B-device changes its role from peripheral to host, software sets this bit when it removes the D+ pull-up, see Section 13–9. Hardware clears this bit when HNP_SUCCESS or HNP_FAILURE is set.	0
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31:16	TMR_CNT	Current timer count value.	0x0

8.7 OTG Timer Register (OTGTmr - 0x5000 C114)

Table 259. OTG Timer register (OTGTmr - address 0x5000 C114) bit description

Bit	Symbol	Description	Reset Value
15:0	TIMEOUT_CNT	The TMR interrupt is set when TMR_CNT reaches this value.	0xFFFF
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.8 OTG Clock Control Register (OTGClkCtrl - 0x5000 CFF4)

This register controls the clocking of the OTG controller. Whenever software wants to access the registers, the corresponding clock control bit needs to be set. The software does not have to repeat this exercise for every register access, provided that the corresponding OTGClkCtrl bits are already set.

Table 260. OTG clock control register (OTG_clock_control - address 0x5000 CFF4) bit description

Bit	Symbol	Value	Description	Reset Value
0	HOST_CLK_EN		Host clock enable	0
		0	Disable the Host clock.	
		1	Enable the Host clock.	
1	DEV_CLK_EN		Device clock enable	0
		0	Disable the Device clock.	
		1	Enable the Device clock.	
2	I2C_CLK_EN		I^2C clock enable	0
		0	Disable the I^2C clock.	
		1	Enable the I^2C clock.	
3	OTG_CLK_EN		OTG clock enable	0
		0	Disable the OTG clock.	
		1	Enable the OTG clock.	

© NXP B.V. 2010. All rights reserved.

Table 260. OTG clock control register (OTG_clock_control - address 0x5000 CFF4) bit description

Bit	Symbol	Value	Description	Reset Value
4	AHB_CLK_EN		AHB master clock enable	0
		0	Disable the AHB clock.	
		1	Enable the AHB clock.	
31:5	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.9 OTG Clock Status Register (OTGClkSt - 0x5000 CFF8)

This register holds the clock availability status. When enabling a clock via OTGClkCtrl, software should poll the corresponding bit in this register. If it is set, then software can go ahead with the register access. Software does not have to repeat this exercise for every access, provided that the OTGClkCtrl bits are not disturbed.

Table 261. OTG clock status register (OTGClkSt - address 0x5000 CFF8) bit description

Bit	Symbol	Value	Description	Reset Value
0	HOST_CLK_ON		Host clock status.	0
		0	Host clock is not available.	
		1	Host clock is available.	
1	DEV_CLK_ON		Device clock status.	0
		0	Device clock is not available.	
		1	Device clock is available.	
2	I2C_CLK_ON		I^2C clock status.	0
		0	I^2C clock is not available.	
		1	I^2C clock is available.	
3	OTG_CLK_ON		OTG clock status.	0
		0	OTG clock is not available.	
		1	OTG clock is available.	
4	AHB_CLK_ON		AHB master clock status.	0
		0	AHB clock is not available.	
		1	AHB clock is available.	
31:5	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.10 I^2C Receive Register (I2C_RX - 0x5000 C300)

This register is the top byte of the receive FIFO. The receive FIFO is 4 bytes deep. The Rx FIFO is flushed by a hard reset or by a soft reset (I2C_CTL bit 7). Reading an empty FIFO gives unpredictable data results.

© NXP B.V. 2010. All rights reserved.

Table 262. I²C Receive register (I2C_RX - address 0x5000 C300) bit description

Bit	Symbol	Description	Reset Value
7:0	RX Data	Receive data.	-

8.11 I²C Transmit Register (I2C_TX - 0x5000 C300)

This register is the top byte of the transmit FIFO. The transmit FIFO is 4 bytes deep.

The Tx FIFO is flushed by a hard reset, soft reset (I2C_CTL bit 7) or if an arbitration failure occurs (I2C_STS bit 3). Data writes to a full FIFO are ignored.

I2C_TX must be written for both write and read operations to transfer each byte. Bits [7:0] are ignored for master-receive operations. The master-receiver must write a dummy byte to the TX FIFO for each byte it expects to receive in the RX FIFO. When the STOP bit is set or the START bit is set to cause a RESTART condition on a byte written to the TX FIFO (master-receiver), then the byte read from the slave is not acknowledged. That is, the last byte of a master-receive operation is not acknowledged.

Table 263. I²C Transmit register (I2C_TX - address 0x5000 C300) bit description

Bit	Symbol	Description	Reset Value
7:0	TX Data	Transmit data.	-
8	START	When 1, issue a START condition before transmitting this byte.	-
9	STOP	When 1, issue a STOP condition after transmitting this byte.	-
31:10	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

8.12 I²C Status Register (I2C_STS - 0x5000 C304)

The I2C_STS register provides status information on the TX and RX blocks as well as the current state of the external buses. Individual bits are enabled as interrupts by the I2C_CTL register and routed to the I2C_USB_INT bit in USBIntSt.

Table 264. I²C status register (I2C_STS - address 0x5000 C304) bit description

Bit	Symbol	Value	Description	Reset Value
0	TDI		Transaction Done Interrupt. This flag is set if a transaction completes successfully. It is cleared by writing a one to bit 0 of the status register. It is unaffected by slave transactions.	0
		0	Transaction has not completed.	
		1	Transaction completed.	
1	AFI		Arbitration Failure Interrupt. When transmitting, if the SDA is low when SDAOUT is high, then this I²C has lost the arbitration to another device on the bus. The Arbitration Failure bit is set when this happens. It is cleared by writing a one to bit 1 of the status register.	0
		0	No arbitration failure on last transmission.	
		1	Arbitration failure occurred on last transmission.	

© NXP B.V. 2010. All rights reserved.

Table 264. I^2C status register (I2C_STS - address 0x5000 C304) bit description

Bit	Symbol	Value	Description	Reset Value
2	NAI		No Acknowledge Interrupt. After every byte of data is sent, the transmitter expects an acknowledge from the receiver. This bit is set if the acknowledge is not received. It is cleared when a byte is written to the master TX FIFO.	0
		0	Last transmission received an acknowledge.	
		1	Last transmission did not receive an acknowledge.	
3	DRMI		Master Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a stop condition or it will hold SCL low until more data is available. The Master Data Request bit is set when the master transmitter is data-starved. If the master TX FIFO is empty and the last byte did not have a STOP condition flag, then SCL is held low until the CPU writes another byte to transmit. This bit is cleared when a byte is written to the master TX FIFO.	0
		0	Master transmitter does not need data.	
		1	Master transmitter needs data.	
4	DRSI		Slave Data Request Interrupt. Once a transmission is started, the transmitter must have data to transmit as long as it isn't followed by a STOP condition or it will hold SCL low until more data is available. The Slave Data Request bit is set when the slave transmitter is data-starved. If the slave TX FIFO is empty and the last byte transmitted was acknowledged, then SCL is held low until the CPU writes another byte to transmit. This bit is cleared when a byte is written to the slave Tx FIFO.	0
		0	Slave transmitter does not need data.	
		1	Slave transmitter needs data.	
5	Active		Indicates whether the bus is busy. This bit is set when a START condition has been seen. It is cleared when a STOP condition is seen..	0
6	SCL		The current value of the SCL signal.	-
7	SDA		The current value of the SDA signal.	-
8	RFF		Receive FIFO Full (RFF). This bit is set when the RX FIFO is full and cannot accept any more data. It is cleared when the RX FIFO is not full. If a byte arrives when the Receive FIFO is full, the SCL is held low until the CPU reads the RX FIFO and makes room for it.	0
		0	RX FIFO is not full	
		1	RX FIFO is full	
9	RFE		Receive FIFO Empty. RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data.	1
		0	RX FIFO contains data.	
		1	RX FIFO is empty	
10	TFF		Transmit FIFO Full. TFF is set when the TX FIFO is full and is cleared when the TX FIFO is not full.	0
		0	TX FIFO is not full.	
		1	TX FIFO is full	

© NXP B.V. 2010. All rights reserved.

Table 264. I²C status register (I2C_STS - address 0x5000 C304) bit description

Bit	Symbol	Value	Description	Reset Value
11	TFE		Transmit FIFO Empty. TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data.	1
		0	TX FIFO contains valid data.	
		1	TX FIFO is empty	
31:12	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.13 I²C Control Register (I2C_CTL - 0x5000 C308)

The I2C_CTL register is used to enable interrupts and reset the I²C state machine. Enabled interrupts cause the USB_I2C_INT interrupt output line to be asserted when set.

Table 265. I²C Control register (I2C_CTL - address 0x5000 C308) bit description

Bit	Symbol	Value	Description	Reset Value
0	TDIE		Transmit Done Interrupt Enable. This enables the TDI interrupt signalling that this I²C issued a STOP condition.	0
		0	Disable the TDI interrupt.	
		1	Enable the TDI interrupt.	
1	AFIE		Transmitter Arbitration Failure Interrupt Enable. This enables the AFI interrupt which is asserted during transmission when trying to set SDA high, but the bus is driven low by another device.	0
		0	Disable the AFI.	
		1	Enable the AFI.	
2	NAIE		Transmitter No Acknowledge Interrupt Enable. This enables the NAI interrupt signalling that transmitted byte was not acknowledged.	0
		0	Disable the NAI.	
		1	Enable the NAI.	
3	DRMIE		Master Transmitter Data Request Interrupt Enable. This enables the DRMI interrupt which signals that the master transmitter has run out of data, has not issued a STOP, and is holding the SCL line low.	0
		0	Disable the DRMI interrupt.	
		1	Enable the DRMI interrupt.	
4	DRSIE		Slave Transmitter Data Request Interrupt Enable. This enables the DRSI interrupt which signals that the slave transmitter has run out of data and the last byte was acknowledged, so the SCL line is being held low.	0
		0	Disable the DRSI interrupt.	
		1	Enable the DRSI interrupt.	
5	REFIE		Receive FIFO Full Interrupt Enable. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data.	0
		0	Disable the RFFI.	
		1	Enable the RFFI.	

© NXP B.V. 2010. All rights reserved.

Table 265. I²C Control register (I2C_CTL - address 0x5000 C308) bit description

Bit	Symbol	Value	Description	Reset Value
6	RFDAIE		Receive Data Available Interrupt Enable. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty).	0
		0	Disable the DAI.	
		1	Enable the DAI.	
7	TFFIE		Transmit FIFO Not Full Interrupt Enable. This enables the Transmit FIFO Not Full interrupt to indicate that the more data can be written to the transmit FIFO. Note that this is not full. It is intended help the CPU to write to the I²C block only when there is room in the FIFO and do this without polling the status register.	0
		0	Disable the TFFI.	
		1	Enable the TFFI.	
8	SRST		Soft reset. This is only needed in unusual circumstances. If a device issues a start condition without issuing a stop condition. A system timer may be used to reset the I²C if the bus remains busy longer than the time-out period. On a soft reset, the Tx and Rx FIFOs are flushed, I2C_STS register is cleared, and all internal state machines are reset to appear idle. The I2C_CLKHI, I2C_CLKLO and I2C_CTL (except Soft Reset Bit) are NOT modified by a soft reset.	0
		0	See the text.	
		1	Reset the I²C to idle state. Self clearing.	
31:9	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.14 I²C Clock High Register (I2C_CLKHI - 0x5000 C30C)

The CLK register holds a terminal count for counting 48 MHz clock cycles to create the high period of the slower I²C serial clock, SCL.

Table 266. I²C_CLKHI register (I2C_CLKHI - address 0x5000 C30C) bit description

Bit	Symbol	Description	Reset Value
7:0	CDHI	Clock divisor high. This value is the number of 48 MHz clocks the serial clock (SCL) will be high.	0xB9

8.15 I²C Clock Low Register (I2C_CLKLO - 0x5000 C310)

The CLK register holds a terminal count for counting 48 MHz clock cycles to create the low period of the slower I²C serial clock, SCL.

Table 267. I²C_CLKLO register (I2C_CLKLO - address 0x5000 C310) bit description

Bit	Symbol	Description	Reset Value
7:0	CDLO	Clock divisor low. This value is the number of 48 MHz clocks the serial clock (SCL) will be low.	0xB9

8.16 Interrupt handling

The interrupts set in the OTGIntSt register are set and cleared during HNP switching. All OTG related interrupts, if enabled, are routed to the USB_OTG_INT bit in the USBIntSt register.

© NXP B.V. 2010. All rights reserved.

I2C related interrupts are set in the I2C_STS register and routed, if enabled by I2C_CTL, to the USB_I2C_INT bit.

For more details on the interrupts created by device controller, see the USB device chapter. For interrupts created by the host controllers, see the OHCI specification.

The EN_USB_INTS bit in the USBIntSt register enables the routing of any of the USB related interrupts to the NVIC controller (see Figure 13–37).

Remark: During the HNP switching between host and device with the OTG stack active, an action may raise several levels of interrupts. It is advised to let the OTG stack initiate any actions based on interrupts and ignore device and host level interrupts. This means that during HNP switching, the OTG stack provides the communication to the host and device controllers.

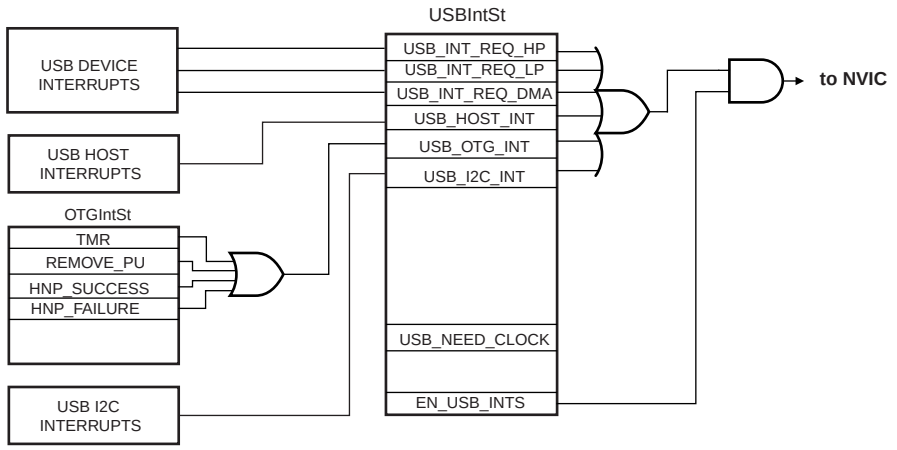


Fig 37. USB OTG interrupt handling

9. HNP support

This section describes the hardware support for the Host Negotiation Protocol (HNP) provided by the OTG controller.

When two dual-role OTG devices are connected to each other, the plug inserted into the mini-AB receptacle determines the default role of each device. The device with the mini-A plug inserted becomes the default Host (A-device), and the device with the mini-B plug inserted becomes the default Peripheral (B-device).

Once connected, the default Host (A-device) and the default Peripheral (B-device) can switch Host and Peripheral roles using HNP.

The context of the OTG controller operation is shown in Figure 13–38. Each controller (Host, Device, or OTG) communicates with its software stack through a set of status and control registers and interrupts. In addition, the OTG software stack communicates with the external OTG transceiver through the I2C interface and the external transceiver interrupt signal.

© NXP B.V. 2010. All rights reserved.

The OTG software stack is responsible for implementing the HNP state machines as described in the On-The-Go Supplement to the USB 2.0 Specification.

The OTG controller hardware provides support for some of the state transitions in the HNP state machines as described in the following subsections.

The USB state machines, the HNP switching, and the communications between the USB controllers are described in more detail in the following documentation:

- USB OHCI specification
- USB OTG supplement, version 1.2
- USB 2.0 specification
- ISP1302 data sheet and user manual

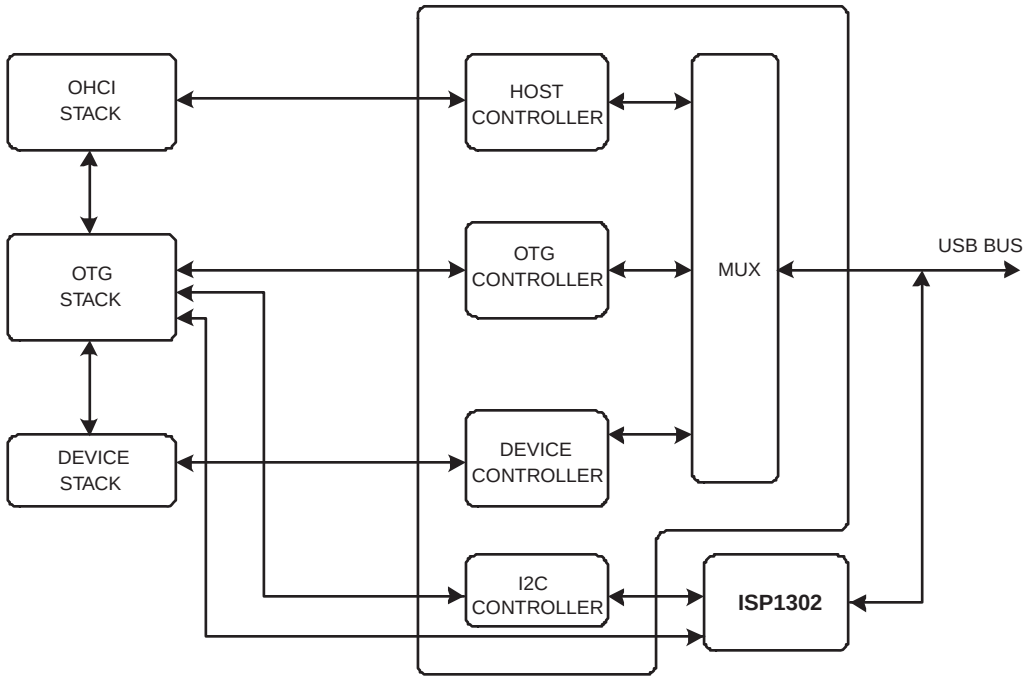


Fig 38. USB OTG controller with software stack

9.1 B-device: peripheral to host switching

In this case, the default role of the OTG controller is peripheral (B-device), and it switches roles from Peripheral to Host.

The On-The-Go Supplement defines the behavior of a dual-role B-device during HNP using a state machine diagram. The OTG software stack is responsible for implementing all of the states in the Dual-Role B-Device State Diagram.

The OTG controller hardware provides support for the state transitions between the states b_peripheral, b_wait_acon, and b_host in the Dual-Role B-Device state diagram. Setting B_HNP_TRACK in the OTGStCtrl register enables hardware support for the B-device switching from peripheral to host. The hardware actions after setting this bit are shown in Figure 13–39.

© NXP B.V. 2010. All rights reserved.

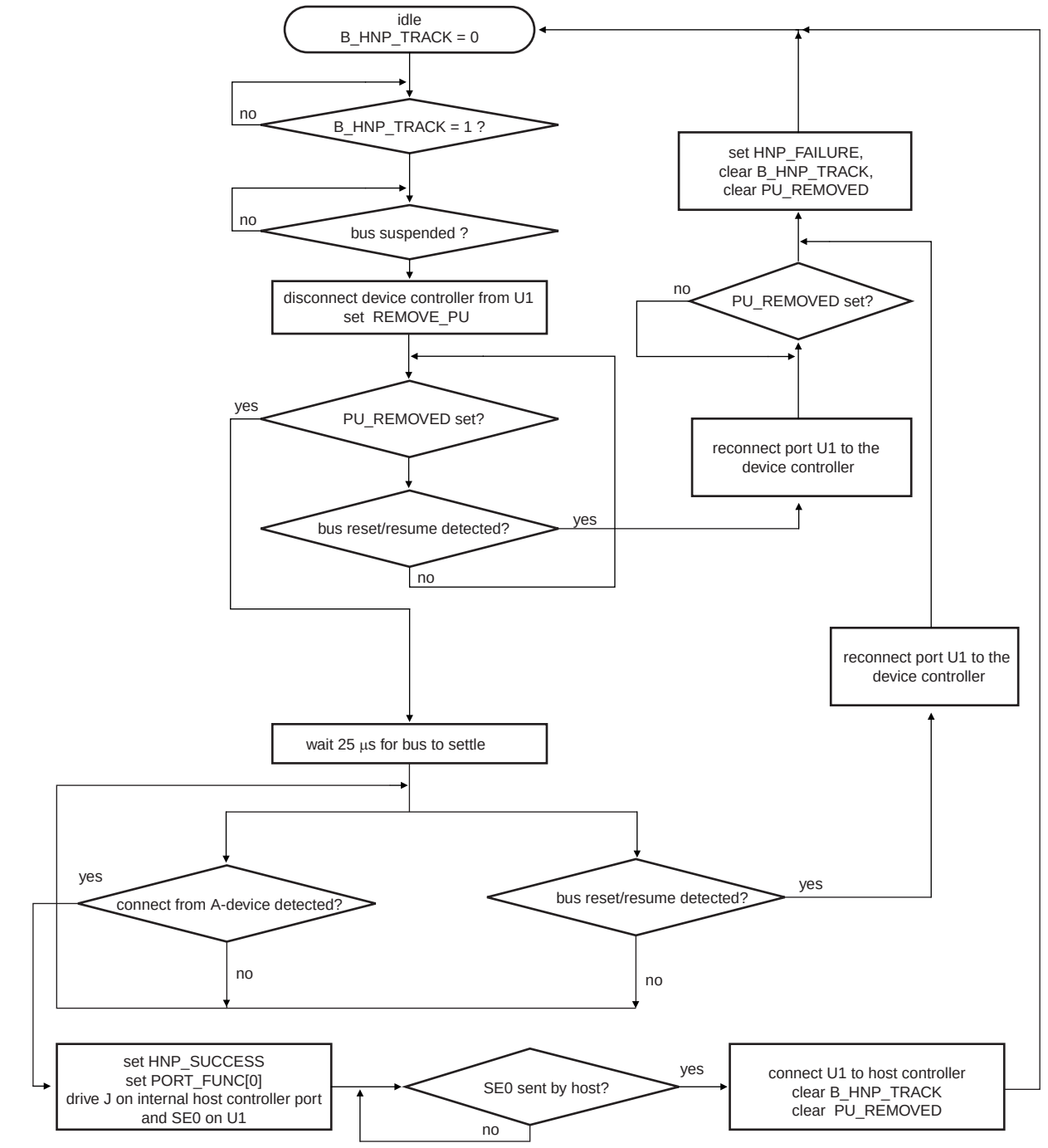


Fig 39. Hardware support for B-device switching from peripheral state to host state

Figure 13–40 shows the actions that the OTG software stack should take in response to the hardware actions setting REMOVE_PU, HNP_SUCCESS, AND HNP_FAILURE. The relationship of the software actions to the Dual-Role B-Device states is also shown. B-device states are in bold font with a circle around them.

© NXP B.V. 2010. All rights reserved.

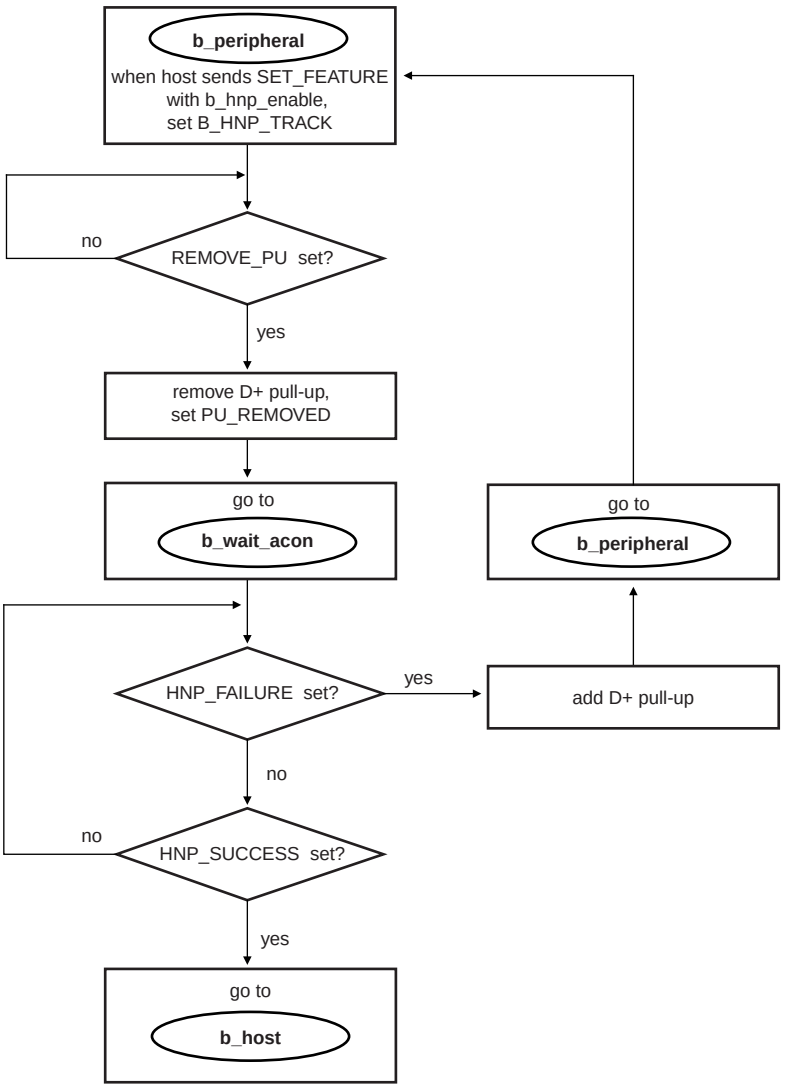


Fig 40. State transitions implemented in software during B-device switching from peripheral to host

Note that only the subset of B-device HNP states and state transitions supported by hardware are shown. Software is responsible for implementing all of the HNP states.

Figure 13–40 may appear to imply that the interrupt bits such as REMOVE_PU should be polled, but this is not necessary if the corresponding interrupt is enabled.

Following are code examples that show how the actions in Figure 13–40 are accomplished. The examples assume that ISP1302 is being used as the external OTG transceiver.

Remove D+ pull-up

```
/* Remove D+ pull-up through ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x201; // Clear DP_PULLUP bit, send STOP condition
```


© NXP B.V. 2010. All rights reserved.

```
/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Add D+ pull-up

```
/* Add D+ pull-up through ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x201; // Set DP_PULLUP bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

9.2 A-device: host to peripheral HNP switching

In this case, the role of the OTG controller is host (A-device), and the A-device switches roles from host to peripheral.

The On-The-Go Supplement defines the behavior of a dual-role A-device during HNP using a state machine diagram. The OTG software stack is responsible for implementing all of the states in the Dual-Role A-Device State Diagram.

The OTG controller hardware provides support for the state transitions between a_host, a_suspend, a_wait_vfall, and a_peripheral in the Dual-Role A-Device state diagram. Setting A_HNP_TRACK in the OTGStCtrl register enables hardware support for switching the A-device from the host state to the device state. The hardware actions after setting this bit are shown in Figure 13–41.

© NXP B.V. 2010. All rights reserved.

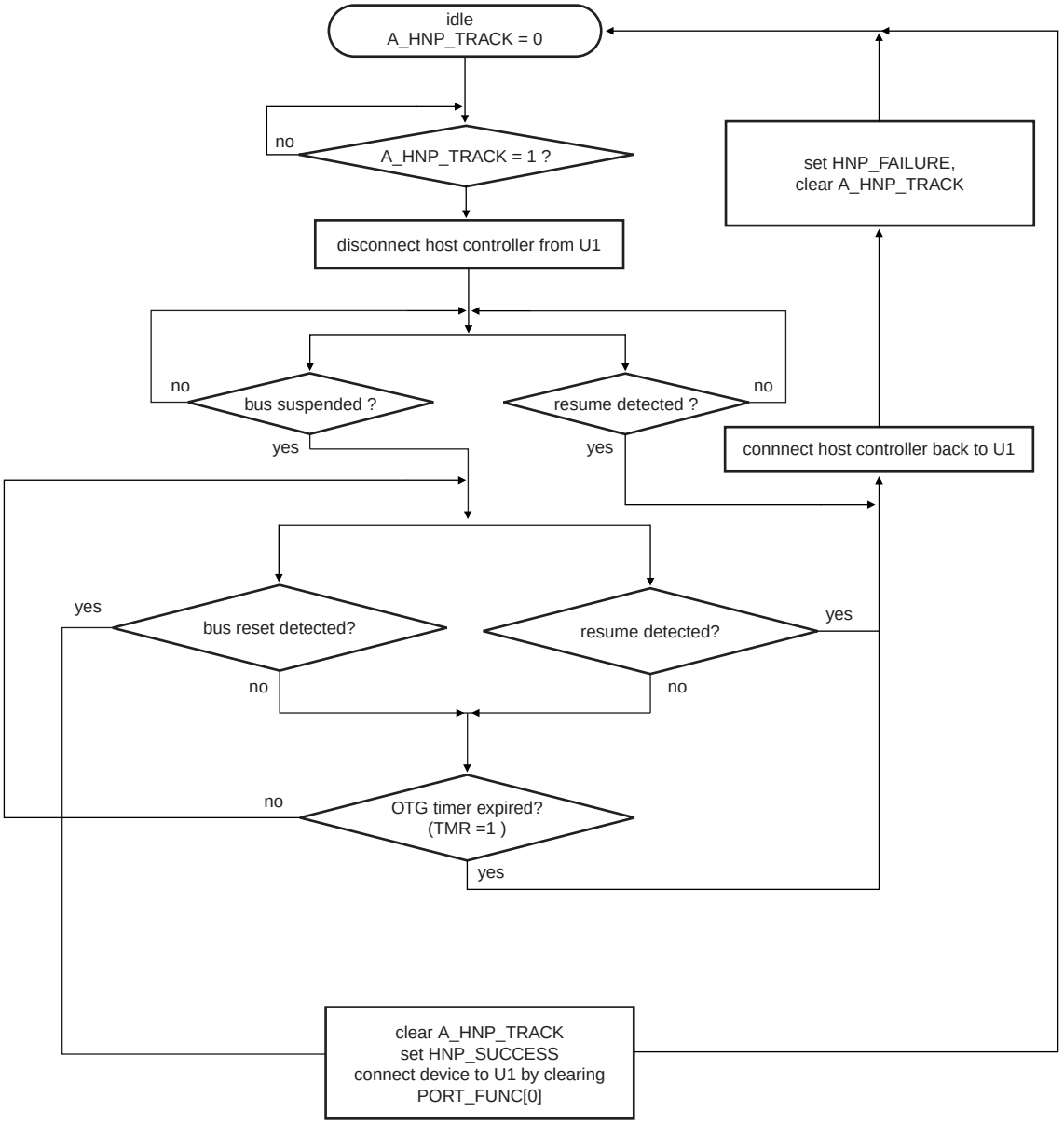


Fig 41. Hardware support for A-device switching from host state to peripheral state

Figure 13–42 shows the actions that the OTG software stack should take in response to the hardware actions setting TMR, HNP_SUCCESS, and HNP_FAILURE. The relationship of the software actions to the Dual-Role A-Device states is also shown. A-device states are shown in bold font with a circle around them.

© NXP B.V. 2010. All rights reserved.

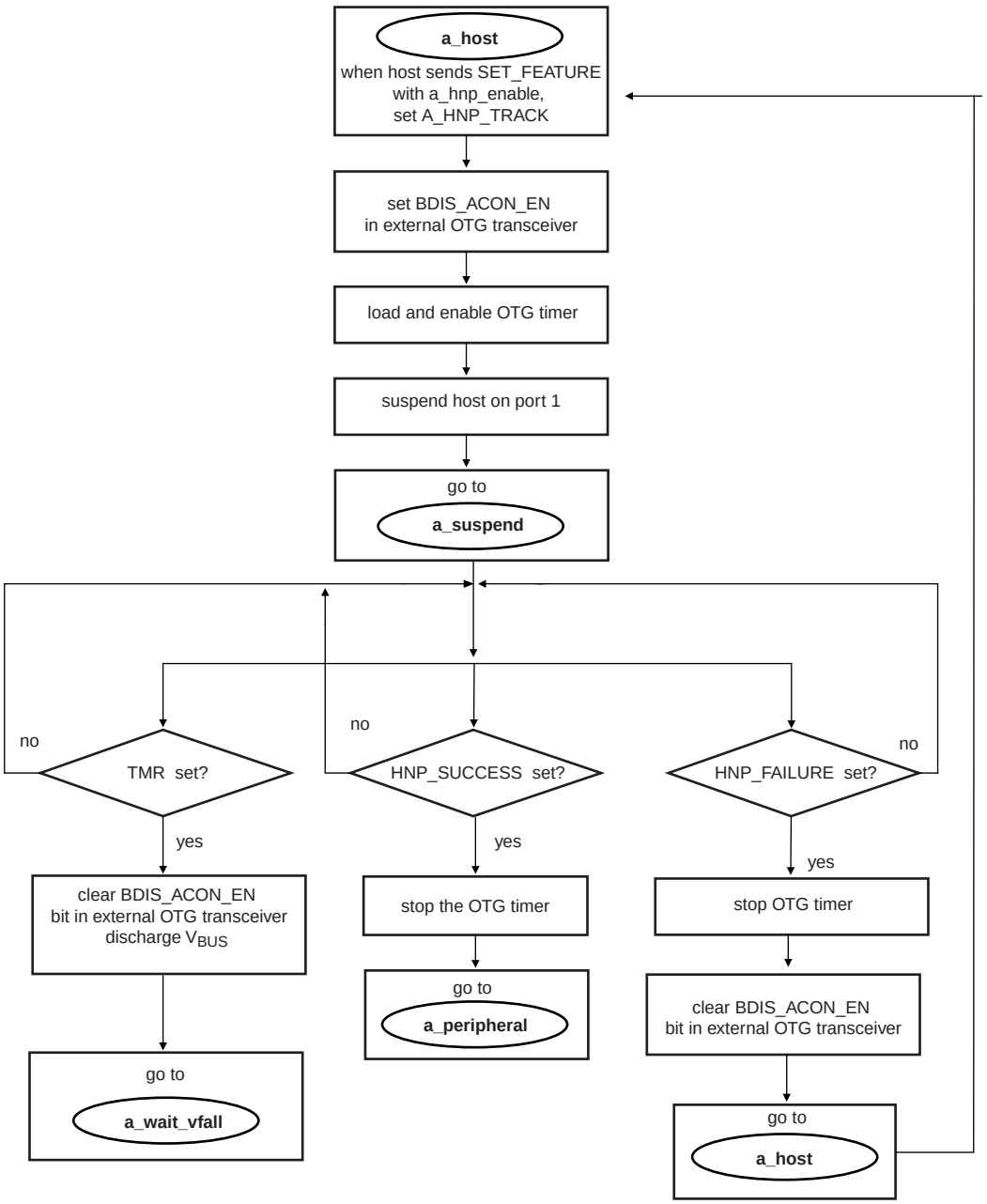


Fig 42. State transitions implemented in software during A-device switching from host to peripheral

Note that only the subset of A-device HNP states and state transitions supported by hardware are shown. Software is responsible for implementing all of the HNP states.

Figure 13–42 may appear to imply that the interrupt bits such as TMR should be polled, but this is not necessary if the corresponding interrupt is enabled.

Following are code examples that show how the actions in Figure 13–42 are accomplished. The examples assume that ISP1302 is being used as the external OTG transceiver.

© NXP B.V. 2010. All rights reserved.

Set BDIS_ACON_EN in external OTG transceiver

```
/* Set BDIS_ACON_EN in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x004; // Send Mode Control 1 (Set) register address
OTG_I2C_TX = 0x210; // Set BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Clear BDIS_ACON_EN in external OTG transceiver

```
/* Set BDIS_ACON_EN in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x005; // Send Mode Control 1 (Clear) register address
OTG_I2C_TX = 0x210; // Clear BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

Discharge V_{BUS}

```
/* Clear the VBUS_DRV bit in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x220; // Clear VBUS_DRV bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;

/* Set the VBUS_DISCHRG bit in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x240; // Set VBUS_DISCHRG bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```


© NXP B.V. 2010. All rights reserved.

Load and enable OTG timer

```
/* The following assumes that the OTG timer has previously been  */

/* configured for a time scale of 1 ms (TMR_SCALE = “10”)        */
/* and monoshot mode (TMR_MODE = 0)                              */

/* Load the timeout value to implement the a_aidl_bdis_tmr timer */
/*     the minimum value is 200 ms                               */
OTG_TIMER = 200;

/* Enable the timer */
OTG_STAT_CTRL |= TMR_EN;
```

Stop OTG timer

```
/* Disable the timer - causes TMR_CNT to be reset to 0  */
OTG_STAT_CTRL &= ~TMR_EN;

/* Clear TMR interrupt */
OTG_INT_CLR = TMR;
```

Suspend host on port 1

```
/* Write to PortSuspendStatus bit to suspend host port 1 -               */
/* this example demonstrates the low-level action software needs to take. */
/* The host stack code where this is done will be somewhat more involved. */
HC_RH_PORT_STAT1 = PSS;
```

10. Clocking and power management

The OTG controller clocking is shown in Figure 13–43.

A clock switch controls each clock with the exception of ahb_slave_clk. When the enable of the clock switch is asserted, its clock output is turned on and its CLK_ON output is asserted. The CLK_ON signals are observable in the OTGClkSt register.

To conserve power, the clocks to the Device, Host, OTG, and I^2C controllers can be disabled when not in use by clearing the respective CLK_EN bit in the OTGClkCtrl register. When the entire USB block is not in use, all of its clocks can be disabled by clearing the PCUSB bit in the PCONP register.

When software wishes to access registers in one of the controllers, it should first ensure that the respective controller’s 48 MHz clock is enabled by setting its CLK_EN bit in the OTGClkCtrl register and then poll the corresponding CLK_ON bit in OTGClkSt until set. Once set, the controller’s clock will remain enabled until CLK_EN is cleared by software. Accessing the register of a controller when its 48 MHz clock is not enabled will result in a data abort exception.

© NXP B.V. 2010. All rights reserved.

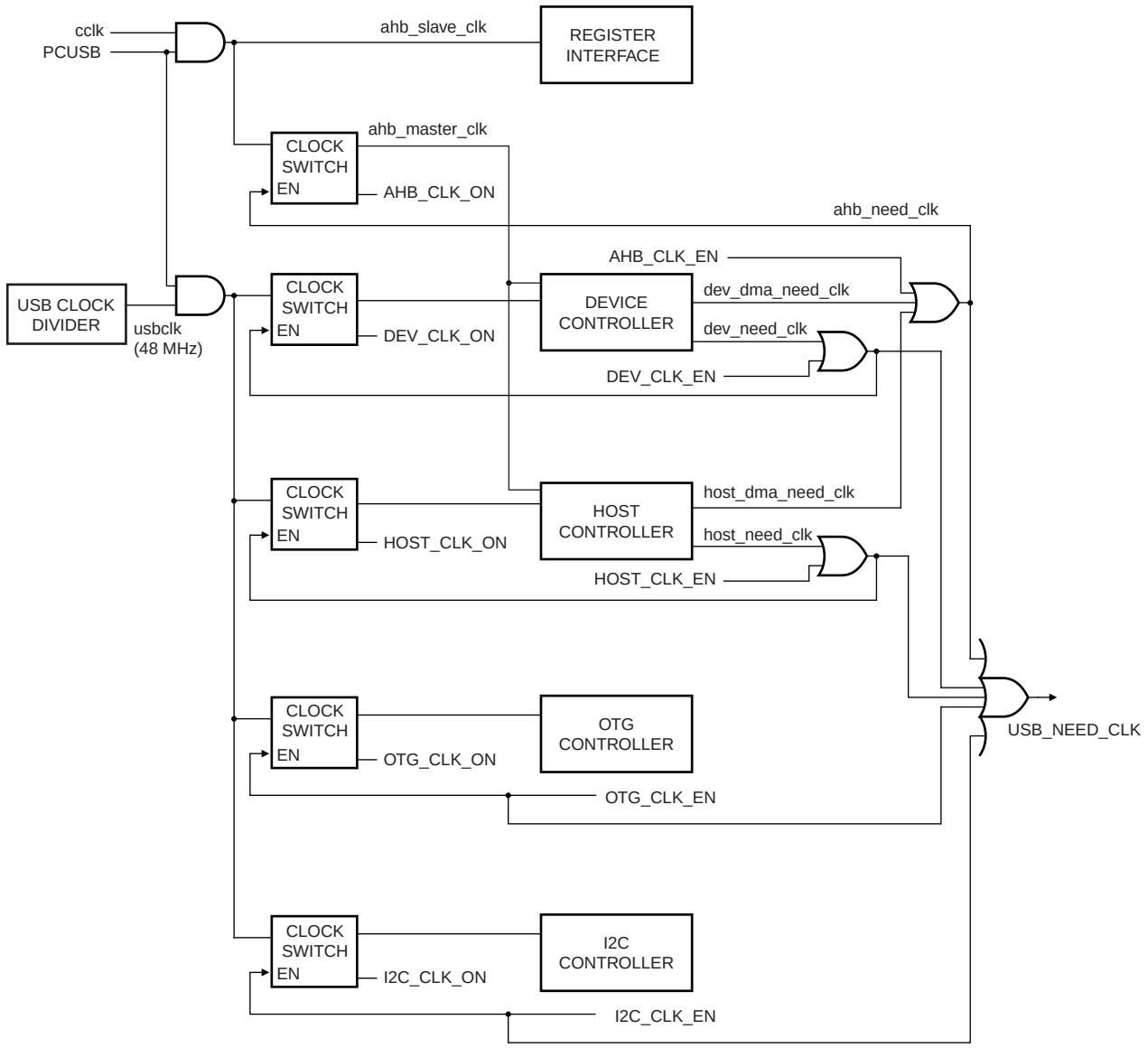


Fig 43. Clocking and power control

10.1 Device clock request signals

The Device controller has two clock request signals, dev_need_clk and dev_dma_need_clk. When asserted, these signals turn on the device's 48 MHz clock and ahb_master_clk respectively.

The dev_need_clk signal is asserted while the device is not in the suspend state, or if the device is in the suspend state and activity is detected on the USB bus. The dev_need_clk signal is de-asserted if a disconnect is detected (CON bit is cleared in the SIE Get Device Status register – Section 11–10.6). This signal allows DEV_CLK_EN to be cleared during normal operation when software does not need to access the Device controller registers – the Device will continue to function normally and automatically shut off its clock when it is suspended or disconnected.

© NXP B.V. 2010. All rights reserved.

The dev_dma_need_clk signal is asserted on any Device controller DMA access to memory. Once asserted, it remains active for 2 ms (2 frames), to help assure that DMA throughput is not affected by any latency associated with re-enabling ahb_master_clk. 2 ms after the last DMA access, dev_dma_need_clk is de-asserted to help conserve power. This signal allows AHB_CLK_EN to be cleared during normal operation.

10.1.1 Host clock request signals

The Host controller has two clock request signals, host_need_clk and host_dma_need_clk. When asserted, these signals turn on the host's 48 MHz clock and ahb_master_clk respectively.

The host_need_clk signal is asserted while the Host controller functional state is not UsbSuspend, or if the functional state is UsbSuspend and resume signaling or a disconnect is detected on the USB bus. This signal allows HOST_CLK_EN to be cleared during normal operation when software does not need to access the Host controller registers – the Host will continue to function normally and automatically shut off its clock when it goes into the UsbSuspend state.

The host_dma_need_clk signal is asserted on any Host controller DMA access to memory. Once asserted, it remains active for 2 ms (2 frames), to help assure that DMA throughput is not affected by any latency associated with re-enabling ahb_master_clk. 2 ms after the last DMA access, host_dma_need_clk is de-asserted to help conserve power. This signal allows AHB_CLK_EN to be cleared during normal operation.

10.2 Power-down mode support

The LPC17xx can be configured to wake up from Power-down mode on any USB bus activity. When the chip is in Power-down mode and the USB interrupt is enabled, the assertion of USB_NEED_CLK causes the chip to wake up from Power-down mode.

Before Power-down mode can be entered when the USB activity interrupt is enabled, USB_NEED_CLK must be de-asserted. This is accomplished by clearing all of the CLK_EN bits in OTGClkCtrl and putting the Host controller into the UsbSuspend functional state. If it is necessary to wait for either of the dma_need_clk signals or the dev_need_clk to be de-asserted, the status of USB_NEED_CLK can be polled in the USBIntSt register to determine when they have all been de-asserted.

11. USB OTG controller initialization

The LPC17xx OTG device controller initialization includes the following steps:

1. Enable the device controller by setting the PCUSB bit of PCONP.
2. Configure and enable the USB PLL (PLL1) or Main PLL (PLL0) to provide 48 MHz for usbclk and the desired frequency for cclk. For correct operation of synchronization logic in the device controller, the minimum cclk frequency is 18 MHz. For the procedure for determining the PLL setting and configuration, see Section 4–5.11 "Procedure for determining PLL0 settings" or Section 4–6.9 "Procedure for determining PLL1 settings".
3. Enable the desired controller clocks by setting their respective CLK_EN bits in the USBClkCtrl register. Poll the corresponding CLK_ON bits in the USBClkSt register until they are set.

© NXP B.V. 2010. All rights reserved.

4. Enable the desired USB pin functions by writing to the corresponding PINSEL registers.
5. Follow the appropriate steps in Section 11–13 “USB device controller initialization” to initialize the device controller.
6. Follow the guidelines given in the OpenHCI specification for initializing the host controller.

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The UART0/2/3 peripherals are configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bits PCUART0/2/3.

   Remark: On reset, UART0 is enabled (PCUART0 = 1), and UART2/3 are disabled (PCUART2/3 = 0).
2. Peripheral clock: In the PCLKSEL0 register (Table 4–40), select PCLK_UART0; in the PCLKSEL1 register (Table 4–41), select PCLK_UART2/3.
3. Baud rate: In register U0/2/3LCR (Table 14–278), set bit DLAB =1. This enables access to registers DLL (Table 14–272) and DLM (Table 14–273) for setting the baud rate. Also, if needed, set the fractional baud rate in the fractional divider register (Table 14–284).
4. UART FIFO: Use bit FIFO enable (bit 0) in register U0/2/3FCR (Table 14–277) to enable FIFO.
5. Pins: Select UART pins through the PINSEL registers and pin modes through the PINMODE registers (Section 8–5).

   Remark: UART receive pins should not have pull-down resistors enabled.
6. Interrupts: To enable UART interrupts set bit DLAB =0 in register U0/2/3LCR (Table 14–278). This enables access to U0/2/3IER (Table 14–274). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
7. DMA: UART0/2/3 transmit and receive functions can operate with the GPDMA controller (see Table 31–544).

2. Features

- Data sizes of 5, 6, 7, and 8 bits.
- Parity generation and checking: odd, even mark, space or none.
- One or two stop bits.
- 16 byte Receive and Transmit FIFOs.
- Built-in baud rate generator, including a fractional rate divider for great versatility.
- Supports DMA for both transmit and receive.
- Auto-baud capability
- Break generation and detection.
- Multiprocessor addressing mode.
- IrDA mode to support infrared communication.
- Support for software flow control.

© NXP B.V. 2010. All rights reserved.

3. Pin description

Table 268: UARTn Pin description

Pin	Type	Description
RXD0, RXD2, RXD3	Input	**Serial Input.** Serial receive data.
TXD0, TXD2, TXD3	Output	**Serial Output.** Serial transmit data.

4. Register description

Each UART contains registers as shown in Table 14–269. The Divisor Latch Access Bit (DLAB) is contained in UnLCR7 and enables access to the Divisor Latches.

© NXP B.V. 2010. All rights reserved.

Table 269. UART0/2/3 Register Map

Generic Name	Description	Access	Reset value[1]	UARTn Register Name & Address
RBR (DLAB =0)	Receiver Buffer Register. Contains the next received character to be read.	RO	NA	U0RBR - 0x4000 C000 U2RBR - 0x4009 8000 U3RBR - 0x4009 C000
THR (DLAB =0)	Transmit Holding Register. The next character to be transmitted is written here.	WO	NA	U0THR - 0x4000 C000 U2THR - 0x4009 8000 U3THR - 0x4009 C000
DLL (DLAB =1)	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	R/W	0x01	U0DLL - 0x4000 C000 U2DLL - 0x4009 8000 U3DLL - 0x4009 C000
DLM (DLAB =1)	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	R/W	0x00	U0DLM - 0x4000 C004 U2DLM - 0x4009 8004 U3DLM - 0x4009 C004
IER (DLAB =0)	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts.	R/W	0x00	U0IER - 0x4000 C004 U2IER - 0x4009 8004 U3IER - 0x4009 C004
IIR	Interrupt ID Register. Identifies which interrupt(s) are pending.	RO	0x01	U0IIR - 0x4000 C008 U2IIR - 0x4009 8008 U3IIR - 0x4009 C008
FCR	FIFO Control Register. Controls UART FIFO usage and modes.	WO	0x00	U0FCR - 0x4000 C008 U2FCR - 0x4009 8008 U3FCR - 0x4009 C008
LCR	Line Control Register. Contains controls for frame formatting and break generation.	R/W	0x00	U0LCR - 0x4000 C00C U2LCR - 0x4009 800C U3LCR - 0x4009 C00C
LSR	Line Status Register. Contains flags for transmit and receive status, including line errors.	RO	0x60	U0LSR - 0x4000 C014 U2LSR - 0x4009 8014 U3LSR - 0x4009 C014
SCR	Scratch Pad Register. 8-bit temporary storage for software.	R/W	0x00	U0SCR - 0x4000 C01C U2SCR - 0x4009 801C U3SCR - 0x4009 C01C
ACR	Auto-baud Control Register. Contains controls for the auto-baud feature.	R/W	0x00	U0ACR - 0x4000 C020 U2ACR - 0x4009 8020 U3ACR - 0x4009 C020
ICR	IrDA Control Register. Enables and configures the IrDA mode.	R/W	0x00	U0ICR - 0x4000 C024 U12CR - 0x4009 8024 U3ICR - 0x4009 C024
FDR	Fractional Divider Register. Generates a clock input for the baud rate divider.	R/W	0x10	U0FDR - 0x4000 C028 U2FDR - 0x4009 8028 U3FDR - 0x4009 C028
TER	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	R/W	0x80	U0TER - 0x4000 C030 U2TER - 0x4009 8030 U3TER - 0x4009 C030
FIFOLVL	FIFO Level register. Provides the current fill levels of the transmit and receive FIFOs.	RO	0x00	U0FIFOLVL - 0x4000 C058 U2FIFOLVL - 0x4009 8058 U3FIFOLVL - 0x4009 C058

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

© NXP B.V. 2010. All rights reserved.

14.4.1 UARTn Receiver Buffer Register (U0RBR - 0x4000 C000, U2RBR - 0x4009 8000, U3RBR - 0x4009 C000 when DLAB = 0)

The UnRBR is the top byte of the UARTn Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the UnRBR. The UnRBR is always read-only.

Since PE, FE and BI bits correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the U0LSR register, and then to read a byte from the UnRBR.

Table 270: UARTn Receiver Buffer Register (U0RBR - address 0x4000 C000, U2RBR - 0x4009 8000, U3RBR - 04009 C000 when DLAB = 0) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UARTn Receiver Buffer Register contains the oldest received byte in the UARTn Rx FIFO.	Undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

4.2 UARTn Transmit Holding Register (U0THR - 0x4000 C000, U2THR - 0x4009 8000, U3THR - 0x4009 C000 when DLAB = 0)

The UnTHR is the top byte of the UARTn TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in UnLCR must be zero in order to access the UnTHR. The UnTHR is always write-only.

Table 271: UARTn Transmit Holding Register (U0THR - address 0x4000 C000, U2THR - 0x4009 8000, U3THR - 0x4009 C000 when DLAB = 0) bit description

Bit	Symbol	Description	Reset Value
7:0	THR	Writing to the UARTn Transmit Holding Register causes the data to be stored in the UARTn transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA
31:8	-	Reserved, user software should not write ones to reserved bits.	NA

4.3 UARTn Divisor Latch LSB register (U0DLL - 0x4000 C000, U2DLL - 0x4009 8000, U3DLL - 0x4009 C000 when DLAB = 1) and UARTn Divisor Latch MSB register (U0DLM - 0x4000 C004, U2DLL - 0x4009 8004, U3DLL - 0x4009 C004 when DLAB = 1)

The UARTn Divisor Latch is part of the UARTn Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16× the desired baud rate. The UnDLL and UnDLM registers together form a 16-bit divisor where UnDLL contains the lower 8 bits of the divisor and UnDLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed. The Divisor Latch Access Bit (DLAB) in

© NXP B.V. 2010. All rights reserved.

UnLCR must be one in order to access the UARTn Divisor Latches. Details on how to select the right value for U1DLL and U1DLM can be found later in this chapter, see Section 14–4.12.

Table 272: UARTn Divisor Latch LSB register (U0DLL - address 0x4000 C000, U2DLL - 0x4009 8000, U3DLL - 0x4009 C000 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.	0x01
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 273: UARTn Divisor Latch MSB register (U0DLM - address 0x4000 C004, U2DLM - 0x4009 8004, U3DLM - 0x4009 C004 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UARTn Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UARTn.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.4 UARTn Interrupt Enable Register (U0IER - 0x4000 C004, U2IER - 0x4009 8004, U3IER - 0x4009 C004 when DLAB = 0)

The UnIER is used to enable the three UARTn interrupt sources.

Table 274: UARTn Interrupt Enable Register (U0IER - address 0x4000 C004, U2IER - 0x4009 8004, U3IER - 0x4009 C004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBR Interrupt Enable		Enables the Receive Data Available interrupt for UARTn. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THRE Interrupt Enable		Enables the THRE interrupt for UARTn. The status of this can be read from UnLSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RX Line Status Interrupt Enable		Enables the UARTn RX line status interrupts. The status of this interrupt can be read from UnLSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	ABEOIntEn		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	

© NXP B.V. 2010. All rights reserved.

Table 274: UARTn Interrupt Enable Register (U0IER - address 0x4000 C004, U2IER - 0x4009 8004, U3IER - 0x4009 C004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
9	ABTOIntEn		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.5 UARTn Interrupt Identification Register (U0IIR - 0x4000 C008, U2IIR - 0x4009 8008, U3IIR - 0x4009 C008)

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

Table 275: UARTn Interrupt Identification Register (U0IIR - address 0x4000 C008, U2IIR - 0x4009 8008, U3IIR - 0x4009 C008) bit description

Bit	Symbol	Value	Description	Reset Value
0	IntStatus		Interrupt status. Note that U1IIR[0] is active low. The pending interrupt can be determined by evaluating UnIIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	IntId		Interrupt identification. UnIER[3:1] identifies an interrupt corresponding to the UARTn Rx or TX FIFO. All other combinations of UnIER[3:1] not listed below are reserved (000,100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		Copies of UnFCR[0].	0
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Bit UnIIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 14–276. Given the status of UnIIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The UnIIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

© NXP B.V. 2010. All rights reserved.

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

The UARTn RDA interrupt (UnIIR[3:1] = 010) shares the second level priority with the CTI interrupt (UnIIR[3:1] = 110). The RDA is activated when the UARTn Rx FIFO reaches the trigger level defined in UnFCR[7:6] and is reset when the UARTn Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (UnIIR[3:1] = 110) is a second level interrupt and is set when the UARTn Rx FIFO contains at least one character and no UARTn Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UARTn Rx FIFO activity (read or write of UARTn RSR) will clear the interrupt. This interrupt is intended to flush the UARTn RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 276: UARTn Interrupt Handling

U0IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	UnLSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (UnFCR0=1)	UnRBR Read[3] or UARTn FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	UnRBR Read[3]
0010	Third	THRE	THRE[2]	UnIIR Read (if source of interrupt) or THR write[4]

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 14–4.8 “UARTn Line Status Register (U0LSR - 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014)”

[3] For details see Section 14–14.4.1 “UARTn Receiver Buffer Register (U0RBR - 0x4000 C000, U2RBR - 0x4009 8000, U3RBR - 0x4009 C000 when DLAB = 0)”

[4] For details see Section 14–4.5 “UARTn Interrupt Identification Register (U0IIR - 0x4000 C008, U2IIR - 0x4009 8008, U3IIR - 0x4009 C008)” and Section 14–4.2 “UARTn Transmit Holding Register (U0THR - 0x4000 C000, U2THR - 0x4009 8000, U3THR - 0x4009 C000 when DLAB = 0)”

The UARTn THRE interrupt (UnIIR[3:1] = 001) is a third level interrupt and is activated when the UARTn THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UARTn THR FIFO a chance to

© NXP B.V. 2010. All rights reserved.

fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the UnTHR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to UnTHR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UARTn THR FIFO has held two or more characters at one time and currently, the UnTHR is empty. The THRE interrupt is reset when a UnTHR write occurs or a read of the UnIIR occurs and the THRE is the highest interrupt (UnIIR[3:1] = 001).

4.6 UARTn FIFO Control Register (U0FCR - 0x4000 C008, U2FCR - 0x4009 8008, U3FCR - 0x4009 C008)

The write-only UnFCR controls the operation of the UARTn Rx and TX FIFOs.

Table 277: UARTn FIFO Control Register (U0FCR - address 0x4000 C008, U2FCR - 0x4009 8008, U3FCR - 0x4007 C008) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFO Enable	0	UARTn FIFOs are disabled. Must not be used in the application.	0
		1	Active high enable for both UARTn Rx and TX FIFOs and UnFCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the related UART FIFOs.	
1	RX FIFO Reset	0	No impact on either of UARTn FIFOs.	0
		1	Writing a logic 1 to UnFCR[1] will clear all bytes in UARTn Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TX FIFO Reset	0	No impact on either of UARTn FIFOs.	0
		1	Writing a logic 1 to UnFCR[2] will clear all bytes in UARTn TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMA Mode Select		When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 14–4.6.1.	0
5:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RX Trigger Level		These two bits determine how many receiver UARTn FIFO characters must be written before an interrupt or DMA request is activated.	0
		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)	
31:8	-		Reserved, user software should not write ones to reserved bits.	NA

4.6.1 DMA Operation

The user can optionally operate the UART transmit and/or receive using DMA. The DMA mode is determined by the DMA Mode Select bit in the FCR register. This bit only has an affect when the FIFOs are enabled via the FIFO Enable bit in the FCR register.

UART receiver DMA

In DMA mode, the receiver DMA request is asserted on the event of the receiver FIFO level becoming equal to or greater than trigger level, or if a character timeout occurs. See the description of the RX Trigger Level above. The receiver DMA request is cleared by the DMA controller.

© NXP B.V. 2010. All rights reserved.

UART transmitter DMA

In DMA mode, the transmitter DMA request is asserted on the event of the transmitter FIFO transitioning to not full. The transmitter DMA request is cleared by the DMA controller.

4.7 UARTn Line Control Register (U0LCR - 0x4000 C00C, U2LCR - 0x4009 800C, U3LCR - 0x4009 C00C)

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

Table 278: UARTn Line Control Register (U0LCR - address 0x4000 C00C, U2LCR - 0x4009 800C, U3LCR - 0x4009 C00C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Word Length Select	00	5-bit character length	0
		01	6-bit character length	
		10	7-bit character length	
		11	8-bit character length	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if UnLCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UARTn TXD is forced to logic 0 when UnLCR[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.8 UARTn Line Status Register (U0LSR - 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014)

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

© NXP B.V. 2010. All rights reserved.

Table 279: UARTn Line Status Register (U0LSR - address 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014) bit description

Bit	Symbol	Value	Description	Reset Value
0	Receiver Data Ready (RDR)		UnLSR0 is set when the UnRBR holds an unread character and is cleared when the UARTn RBR FIFO is empty.	0
		0	The UARTn receiver FIFO is empty.	
		1	The UARTn receiver FIFO is not empty.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An UnLSR read clears UnLSR1. UnLSR1 is set when UARTn RSR has a new character assembled and the UARTn RBR FIFO is full. In this case, the UARTn RBR FIFO will not be overwritten and the character in the UARTn RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An UnLSR read clears UnLSR[2]. Time of parity error detection is dependent on UnFCR[0]. **Note:** A parity error is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An UnLSR read clears UnLSR[3]. The time of the framing error detection is dependent on UnFCR0. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXDn is held in the spacing state (all zeroes) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXDn goes to marking state (all ones). An UnLSR read clears this status bit. The time of break detection is dependent on UnFCR[0]. **Note:** The break interrupt is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE))		THRE is set immediately upon detection of an empty UARTn THR and is cleared on a UnTHR write.	1
		0	UnTHR contains valid data.	
		1	UnTHR is empty.	
6	Transmitter Empty (TEMT)		TEMT is set when both UnTHR and UnTSR are empty; TEMT is cleared when either the UnTSR or the UnTHR contain valid data.	1
		0	UnTHR and/or the UnTSR contains valid data.	
		1	UnTHR and the UnTSR are empty.	

© NXP B.V. 2010. All rights reserved.

Table 279: UARTn Line Status Register (U0LSR - address 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014) bit description

Bit	Symbol	Value	Description	Reset Value
7	Error in RX FIFO (RXFE)		UnLSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the UnRBR. This bit is cleared when the UnLSR register is read and there are no subsequent errors in the UARTn FIFO.	0
		0	UnRBR contains no UARTn RX errors or UnFCR[0]=0.	
		1	UARTn RBR contains at least one UARTn RX error.	
31:8	-		Reserved, the value read from a reserved bit is not defined.	NA

4.9 UARTn Scratch Pad Register (U0SCR - 0x4000 C01C, U2SCR - 0x4009 801C U3SCR - 0x4009 C01C)

The UnSCR has no effect on the UARTn operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the UnSCR has occurred.

Table 280: UARTn Scratch Pad Register (U0SCR - address 0x4000 C01C, U2SCR - 0x4009 801C, U3SCR - 0x4009 C01C) bit description

Bit	Symbol	Description	Reset Value
7:0	Pad	A readable, writable byte.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.10 UARTn Auto-baud Control Register (U0ACR - 0x4000 C020, U2ACR - 0x4009 8020, U3ACR - 0x4009 C020)

The UARTn Auto-baud Control Register (UnACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 281: UARTn Auto-baud Control Register (U0ACR - address 0x4000 C020, U2ACR - 0x4009 8020, U3ACR - 0x4009 C020) bit description

Bit	Symbol	Value	Description	Reset value
0	Start		This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	Mode		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AutoRestart	0	No restart.	0
		1	Restart in case of time-out (counter restarts at next UARTn Rx falling edge)	0
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

Table 281: UARTn Auto-baud Control Register (U0ACR - address 0x4000 C020, U2ACR - 0x4009 8020, U3ACR - 0x4009 C020) bit description

Bit	Symbol	Value	Description	Reset value
8	ABEOIntClr		End of auto-baud interrupt clear bit (write-only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
9	ABTOIntClr		Auto-baud time-out interrupt clear bit (write-only accessible). Writing a 1 will clear the corresponding interrupt in the UnIIR. Writing a 0 has no impact.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.4.10.1 Auto-baud

The UARTn auto-baud function can be used to measure the incoming baud-rate based on the “AT” protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers UnDLM and UnDLL accordingly.

Remark: the fractional rate divider is not connected during auto-baud operations, and therefore should not be used when the auto-baud feature is needed.

Auto-baud is started by setting the UnACR Start bit. Auto-baud can be stopped by clearing the UnACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the UnACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UARTn Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UARTn Rx pin (the length of the start bit).

The UnACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UARTn Rx pin.

The auto-baud function can generate two interrupts.

- The UnIIR ABTOInt interrupt will get set if the interrupt is enabled (UnIER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The UnIIR ABEOInt interrupt will get set if the interrupt is enabled (UnIER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding UnACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UARTn Rx pin baud-rate, but the value of the UnFDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to UnDLM and UnDLL registers should be done before UnACR register write. The minimum and the maximum baud rates supported by UARTn are function of pclk, number of data bits, stop bits and parity bits.

© NXP B.V. 2010. All rights reserved.

(1)

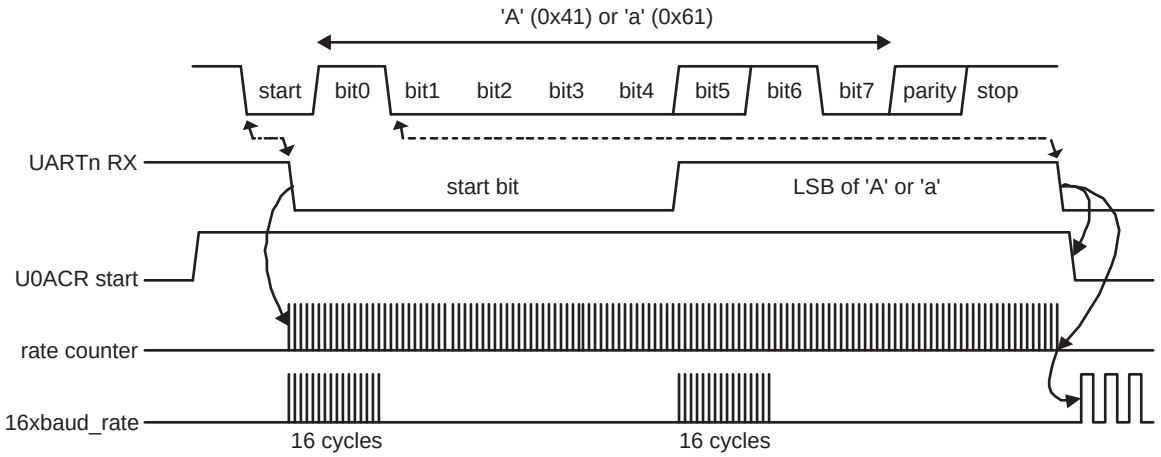
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UARTn_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

14.4.10.2 Auto-baud modes

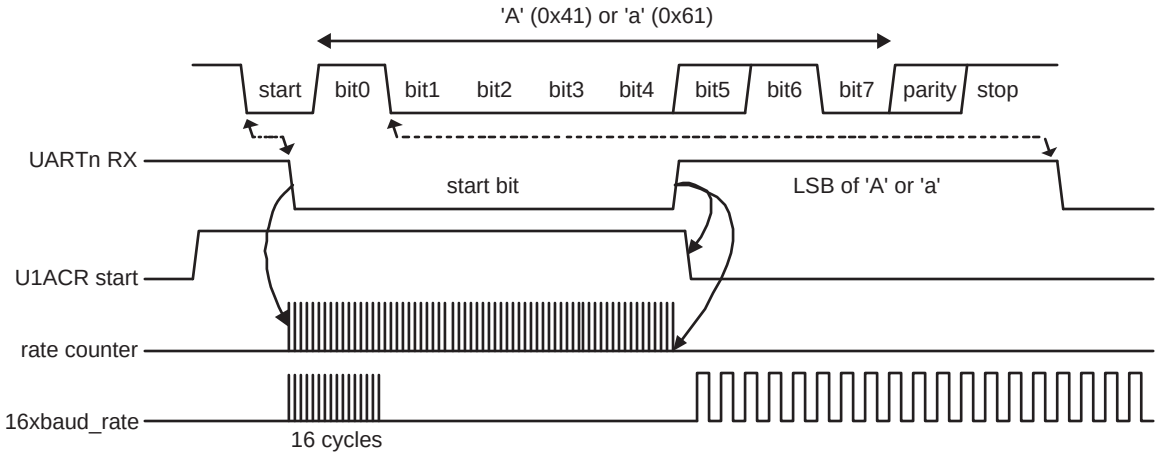
When the software is expecting an “AT” command, it configures the UARTn with the expected character format and sets the UnACR Start bit. The initial values in the divisor latches UnDLM and UnDLM don‘t care. Because of the “A” or “a” ASCII coding (”A" = 0x41, “a” = 0x61), the UARTn Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the UnACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On UnACR Start bit setting, the baud rate measurement counter is reset and the UARTn UnRSR is reset. The UnRSR baud rate is switch to the highest rate.
2. A falling edge on UARTn Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UARTn input clock, guaranteeing the start bit is stored in the UnRSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UARTn input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UARTn Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UARTn Rx pin.
6. The rate counter is loaded into UnDLM/UnDLL and the baud-rate will be switched to normal operation. After setting the UnDLM/UnDLL the end of auto-baud interrupt UnIIR ABEOInt will be set, if enabled. The UnRSR will now continue receiving the remaining bits of the “A/a” character.

© NXP B.V. 2010. All rights reserved.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 44. Auto-baud a) mode 0 and b) mode 1 waveform

4.11 UARTn IrDA Control Register (U0ICR - 0x4000 C024, U2ICR - 0x4009 8024, U3ICR - 0x4009 C024)

The IrDA Control Register enables and configures the IrDA mode on each UART. The value of UnICR should not be changed while transmitting or receiving data, or data loss or corruption may occur.

Table 282: UARTn IrDA Control Register (U0ICR - 0x4000 C024, U2ICR - 0x4009 8024, U3ICR - 0x4009 C024) bit description

Bit	Symbol	Value	Description	Reset value
0	IrDAEn	0	IrDA mode on UARTn is disabled, UARTn acts as a standard UART.	0
		1	IrDA mode on UARTn is enabled.	
1	IrDAInv		When 1, the serial input is inverted. This has no effect on the serial output. When 0, the serial input is not inverted.	0

© NXP B.V. 2010. All rights reserved.

Table 282: UARTn IrDA Control Register (U0ICR - 0x4000 C024, U2ICR - 0x4009 8024, U3ICR - 0x4009 C024) bit description

Bit	Symbol	Value	Description	Reset value
2	FixPulseEn		When 1, enabled IrDA fixed pulse width mode.	0
5:3	PulseDiv		Configures the pulse when FixPulseEn = 1. See text below for details.	0
31:6	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

The PulseDiv bits in UnICR are used to select the pulse width when the fixed pulse width mode is used in IrDA mode (IrDAEn = 1 and FixPulseEn = 1). The value of these bits should be set so that the resulting pulse width is at least 1.63 µs. Table 14–283 shows the possible pulse widths.

Table 283: IrDA Pulse Width

FixPulseEn	PulseDiv	IrDA Transmitter Pulse width (µs)
0	x	3 / (16 × baud rate)
1	0	$2 \times T_{PCLK}$
1	1	$4 \times T_{PCLK}$
1	2	$8 \times T_{PCLK}$
1	3	$16 \times T_{PCLK}$
1	4	$32 \times T_{PCLK}$
1	5	$64 \times T_{PCLK}$
1	6	$128 \times T_{PCLK}$
1	7	$256 \times T_{PCLK}$

4.12 UARTn Fractional Divider Register (U0FDR - 0x4000 C028, U2FDR - 0x4009 8028, U3FDR - 0x4009 C028)

The UART0/2/3 Fractional Divider Register (U0/2/3FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be greater than 2.

Table 284: UARTn Fractional Divider Register (U0FDR - address 0x4000 C028, U2FDR - 0x4009 8028, U3FDR - 0x4009 C028) bit description

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

© NXP B.V. 2010. All rights reserved.

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of UART0/2/3 disabled making sure that UART0/2/3 is fully software and hardware compatible with UARTs not equipped with this feature.

UART0/2/3 baud rate can be calculated as (n = 0/2/3):

(2)

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where PCLK is the peripheral clock, U0/2/3DLM and U0/2/3DLL are the standard UART0/2/3 baud rate divider registers, and DIVADDVAL and MULVAL are UART0/2/3 fractional baud rate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 14$
3. $DIVADDVAL < MULVAL$

The value of the U0/2/3FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the U0/2/3FDR 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.

4.12.1 Baud rate calculation

UARTn can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baud rate with a relative error of less than 1.1% from the desired one.

© NXP B.V. 2010. All rights reserved.

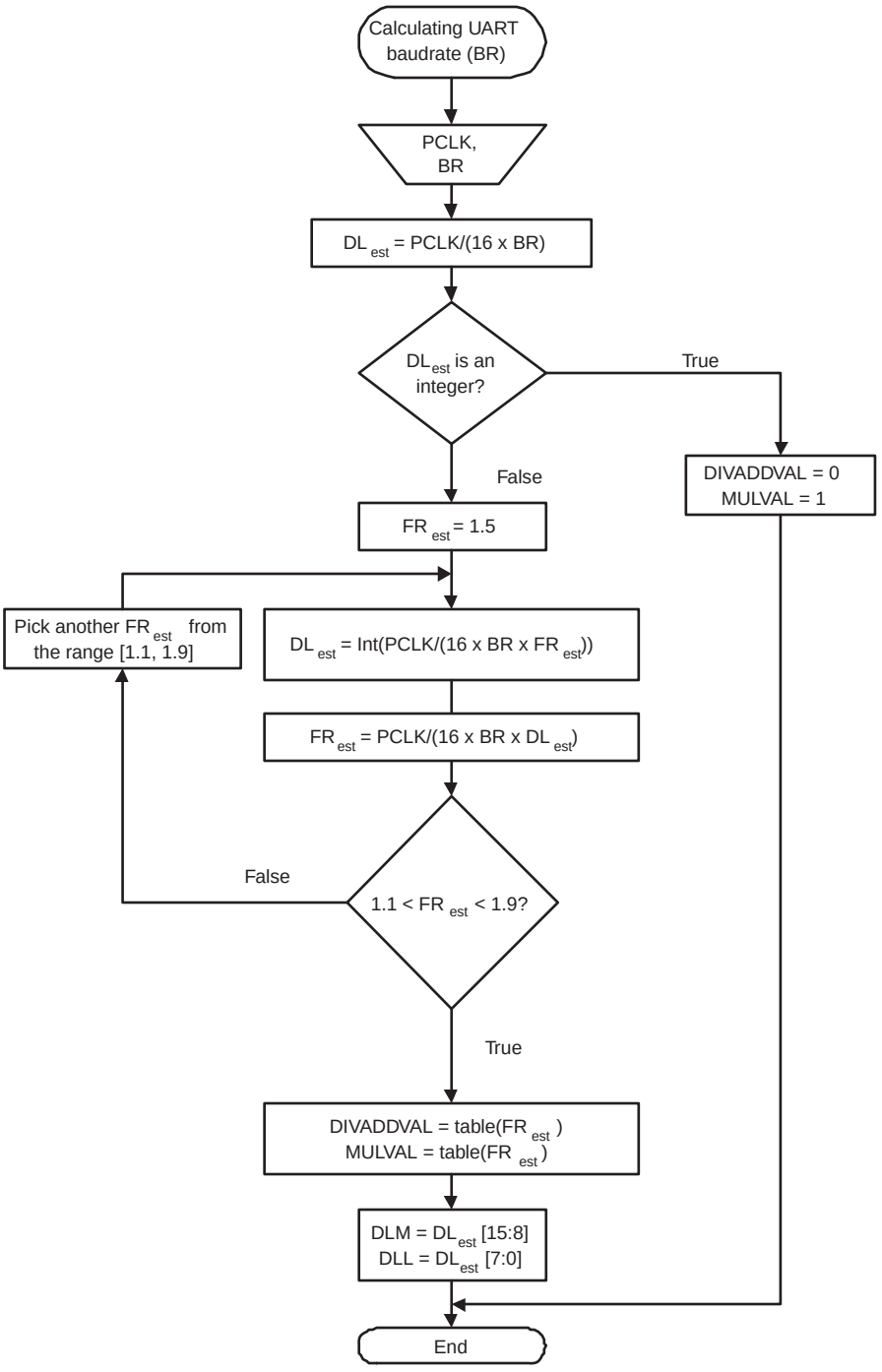


Fig 45. Algorithm for setting UART dividers

© NXP B.V. 2010. All rights reserved.

Table 285. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

4.12.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

4.12.1.2 Example 2: PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 14–285 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 14–2 the UART rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

4.13 UARTn Transmit Enable Register (U0TER - 0x4000 C030, U2TER - 0x4009 8030, U3TER - 0x4009 C030)

The UnTER register enables implementation of software flow control. When TXEn=1, UARTn transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UARTn transmission will stop.

© NXP B.V. 2010. All rights reserved.

Table 14–286 describes how to use TXEn bit in order to achieve software flow control.

Table 286: UARTn Transmit Enable Register (U0TER - address 0x4000 C030, U2TER - 0x4009 8030, U3TER - 0x4009 C030) bit description

Bit	Symbol	Description	Reset Value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit is cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software implementing software-handshaking can clear this bit when it receives an XOFF character (DC3). Software can set this bit again when it receives an XON (DC1) character.	1

4.14 UARTn FIFO Level register (U0FIFOLVL - 0x4000 C058, U2FIFOLVL - 0x4009 8058, U3FIFOLVL - 0x4009 C058)

UnFIFOLVL register is a read-only register that allows software to read the current FIFO level status. Both the transmit and receive FIFO levels are present in this register.

Table 287. UARTn FIFO Level register (U0FIFOLVL - 0x4000 C058, U2FIFOLVL - 0x4009 8058, U3FIFOLVL - 0x4009 C058) bit description

Bit	Symbol	Description	Reset value
3:0	RXFIFILVL	Reflects the current level of the UART receiver FIFO. 0 = empty, 0xF = FIFO full.	0x00
7:4	-	Reserved. The value read from a reserved bit is not defined.	NA
11:8	TXFIFOLVL	Reflects the current level of the UART transmitter FIFO. 0 = empty, 0xF = FIFO full.	0x00
31:12	-	Reserved. The value read from a reserved bit is not defined.	NA

5. Architecture

The architecture of the UARTs 0, 2 and 3 are shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART.

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

The UARTn transmitter block, UnTX, accepts data written by the CPU or host and buffers the data in the UARTn TX Holding Register FIFO (UnTHR). The UARTn TX Shift Register (UnTSR) reads the data stored in the UnTHR and assembles the data to transmit via the serial output pin, TXDn.

The UARTn Baud Rate Generator block, UnBRG, generates the timing enables used by the UARTn TX block. The UnBRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the UnDLL and UnDLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

© NXP B.V. 2010. All rights reserved.

The interrupt interface contains registers UnIER and UnIIR. The interrupt interface receives several one clock wide enables from the UnTX and UnRX blocks.

Status information from the UnTX and UnRX is stored in the UnLSR. Control information for the UnTX and UnRX is stored in the UnLCR.

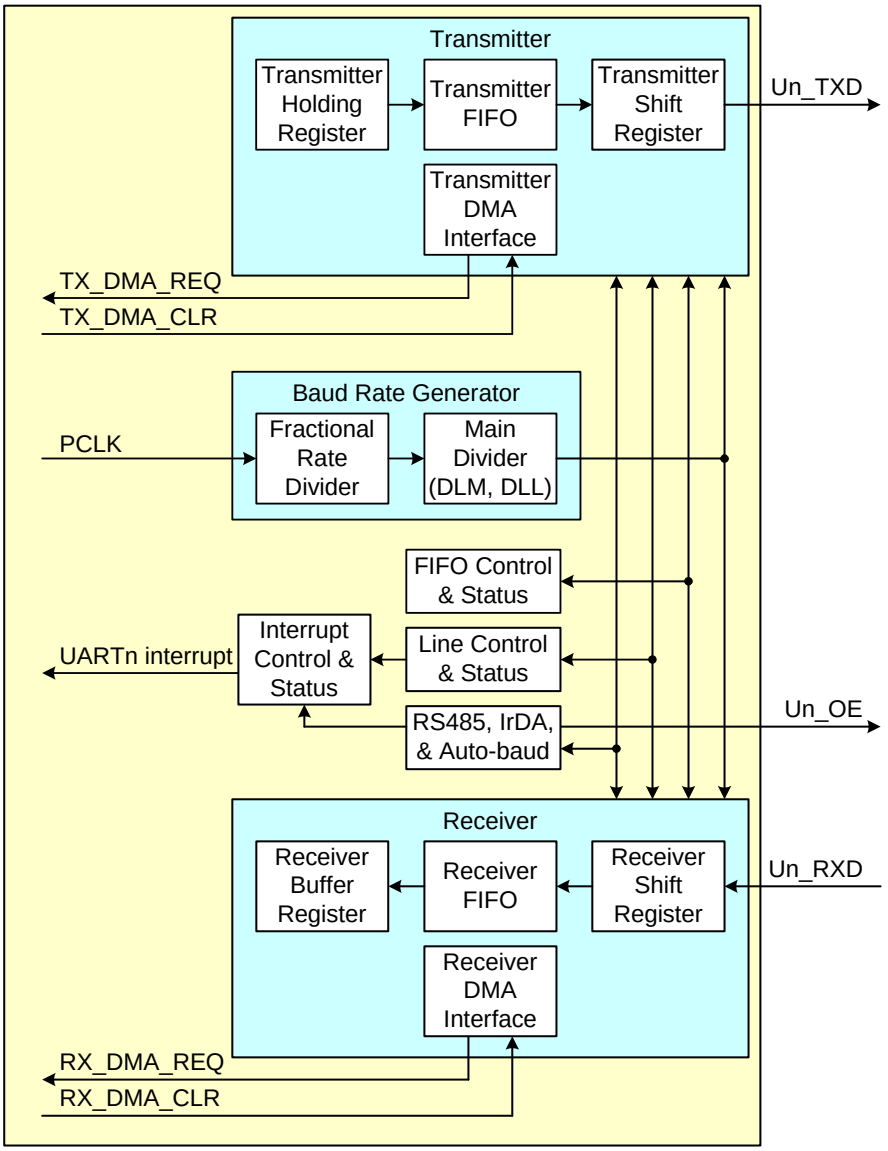


Fig 46. UART0, 2 and 3 block diagram

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The UART1 peripheral is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bits PCUART1.

   Remark: On reset, UART1 is enabled (PCUART1 = 1).
2. Peripheral clock: In the PCLKSEL0 register (Table 4–40), select PCLK_UART1.
3. Baud rate: In register U1LCR (Table 15–298), set bit DLAB =1. This enables access to registers DLL (Table 15–292) and DLM (Table 15–293) for setting the baud rate. Also, if needed, set the fractional baud rate in the fractional divider register (Table 15–305).
4. UART FIFO: Use bit FIFO enable (bit 0) in register U0FCR (Table 15–297) to enable FIFO.
5. Pins: Select UART pins through PINSEL registers and pin modes through the PINMODE registers (Section 8–5).

   Remark: UART receive pins should not have pull-down resistors enabled.
6. Interrupts: To enable UART interrupts set bit DLAB =0 in register U1LCR (Table 15–298). This enables access to U1IER (Table 15–294). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
7. DMA: UART1 transmit and receive functions can operated with the GPDMA controller (see Table 31–544).

2. Features

- Full modem control handshaking available
- Data sizes of 5, 6, 7, and 8 bits.
- Parity generation and checking: odd, even mark, space or none.
- One or two stop bits.
- 16 byte Receive and Transmit FIFOs.
- Built-in baud rate generator, including a fractional rate divider for great versatility.
- Supports DMA for both transmit and receive.
- Auto-baud capability
- Break generation and detection.
- Multiprocessor addressing mode.
- RS-485 support.

© NXP B.V. 2010. All rights reserved.

3. Pin description

Table 288: UART1 Pin Description

Pin	Type	Description
RXD1	Input	**Serial Input.** Serial receive data.
TXD1	Output	**Serial Output.** Serial transmit data.
CTS1	Input	**Clear To Send.** Active low signal indicates if the external modem is ready to accept transmitted data via TXD1 from the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[4]. State change information is stored in U1MSR[0] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1). Clear to send. CTS1 is an asynchronous, active low modem status signal. Its condition can be checked by reading bit 4 (CTS) of the modem status register. Bit 0 (DCTS) of the Modem Status Register (MSR) indicates that CTS1 has changed states since the last read from the MSR. If the modem status interrupt is enabled when CTS1 changes levels and the auto-cts mode is not enabled, an interrupt is generated. CTS1 is also used in the auto-cts mode to control the transmitter.
DCD1	Input	**Data Carrier Detect.** Active low signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged. In normal operation of the modem interface (U1MCR[4]=0), the complement value of this signal is stored in U1MSR[7]. State change information is stored in U1MSR3 and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
DSR1	Input	**Data Set Ready.** Active low signal indicates if the external modem is ready to establish a communications link with the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[5]. State change information is stored in U1MSR[1] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
DTR1	Output	Data Terminal Ready. Active low signal indicates that the UART1 is ready to establish connection with external modem. The complement value of this signal is stored in U1MCR[0]. The DTR pin can also be used as an RS-485/EIA-485 output enable signal.
RI1	Input	**Ring Indicator.** Active low signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[6]. State change information is stored in U1MSR[2] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
RTS1	Output	**Request To Send.** Active low signal indicates that the UART1 would like to transmit data to the external modem. The complement value of this signal is stored in U1MCR[1]. In auto-rts mode, RTS1 is used to control the transmitter FIFO threshold logic. Request to send. RTS1 is an active low signal informing the modem or data set that the UART is ready to receive data. RTS1 is set to the active (low) level by setting the RTS modem control register bit and is set to the inactive (high) level either as a result of a system reset or during loop-back mode operations or by clearing bit 1 (RTS) of the MCR. In the auto-rts mode, RTS1 is controlled by the transmitter FIFO threshold logic. The RTS pin can also be used as an RS-485/EIA-485 output enable signal.

© NXP B.V. 2010. All rights reserved.

4. Register description

UART1 contains registers organized as shown in Table 15–289. The Divisor Latch Access Bit (DLAB) is contained in U1LCR[7] and enables access to the Divisor Latches.

Table 289: UART1 register map

Name	Description	Access	Reset Value[1]	Address
U1RBR (when DLAB =0)	Receiver Buffer Register. Contains the next received character to be read.	RO	NA	0x4001 0000 (when DLAB=0)
U1THR (when DLAB =0)	Transmit Holding Register. The next character to be transmitted is written here.	WO	NA	0x4001 0000 (when DLAB=0)
U1DLL (when DLAB =1)	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	R/W	0x01	0x4001 0000 (when DLAB=1)
U1DLM (when DLAB =1)	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	R/W	0x00	0x4001 0004 (when DLAB=1)
U1IER (when DLAB =0)	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART1 interrupts.	R/W	0x00	0x4001 0004 (when DLAB=0)
U1IIR	Interrupt ID Register. Identifies which interrupt(s) are pending.	RO	0x01	0x4001 0008
U1FCR	FIFO Control Register. Controls UART1 FIFO usage and modes.	WO	0x00	0x4001 0008
U1LCR	Line Control Register. Contains controls for frame formatting and break generation.	R/W	0x00	0x4001 000C
U1MCR	Modem Control Register. Contains controls for flow control handshaking and loopback mode.	R/W	0x00	0x4001 0010
U1LSR	Line Status Register. Contains flags for transmit and receive status, including line errors.	RO	0x60	0x4001 0014
U1MSR	Modem Status Register. Contains handshake signal status flags.	RO	0x00	0x4001 0018
U1SCR	Scratch Pad Register. 8-bit temporary storage for software.	R/W	0x00	0x4001 001C
U1ACR	Auto-baud Control Register. Contains controls for the auto-baud feature.	R/W	0x00	0x4001 0020
U1FDR	Fractional Divider Register. Generates a clock input for the baud rate divider.	R/W	0x10	0x4001 0028
U1TER	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	R/W	0x80	0x4001 0030
U1RS485CTRL	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	R/W	0x00	0x4001 004C
U1ADRMATCH	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	R/W	0x00	0x4001 0050
U1RS485DLY	RS-485/EIA-485 direction control delay.	R/W	0x00	0x4001 0054
U1FIFOLVL	FIFO Level register. Provides the current fill levels of the transmit and receive FIFOs.	RO	0x00	0x4001 0058

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

© NXP B.V. 2010. All rights reserved.

4.1 UART1 Receiver Buffer Register (U1RBR - 0x4001 0000, when DLAB = 0)

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 290: UART1 Receiver Buffer Register (U1RBR - address 0x4001 0000 when DLAB = 0) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UART1 Receiver Buffer Register contains the oldest received byte in the UART1 RX FIFO.	undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

4.2 UART1 Transmitter Holding Register (U1THR - 0x4001 0000 when DLAB = 0)

The write-only U1THR is the top byte of the UART1 TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1THR. The U1THR is write-only.

Table 291: UART1 Transmitter Holding Register (U1THR - address 0x4001 0000 when DLAB = 0) 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
31:8	-	Reserved, user software should not write ones to reserved bits.	NA

4.3 UART1 Divisor Latch LSB and MSB Registers (U1DLL - 0x4001 0000 and U1DLM - 0x4001 0004, when DLAB = 1)

The UART1 Divisor Latch is part of the UART1 Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the APB clock (PCLK) in order to produce the baud rate clock, which must be 16x the desired baud rate. The U1DLL and U1DLM registers together form a 16-bit divisor where U1DLL contains the lower 8 bits of the divisor and U1DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U1LCR must be one in order to access the UART1 Divisor Latches. Details on how to select the right value for U1DLL and U1DLM can be found later in this chapter, see Section 15–4.16.

© NXP B.V. 2010. All rights reserved.

Table 292: UART1 Divisor Latch LSB Register (U1DLL - address 0x4001 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
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 293: UART1 Divisor Latch MSB Register (U1DLM - address 0x4001 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
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.4 UART1 Interrupt Enable Register (U1IER - 0x4001 0004, when DLAB = 0)

The U1IER is used to enable the four UART1 interrupt sources.

Table 294: UART1 Interrupt Enable Register (U1IER - address 0x4001 0004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBR Interrupt Enable		enables the Receive Data Available interrupt for UART1. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THRE Interrupt Enable		enables the THRE interrupt for UART1. The status of this interrupt can be read from U1LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RX Line Interrupt Enable		enables the UART1 RX line status interrupts. The status of this interrupt can be read from U1LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
3	Modem Status Interrupt Enable		enables the modem interrupt. The status of this interrupt can be read from U1MSR[3:0].	0
		0	Disable the modem interrupt.	
		1	Enable the modem interrupt.	
6:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	CTS Interrupt Enable		If auto-cts mode is enabled this bit enables/disables the modem status interrupt generation on a CTS1 signal transition. If auto-cts mode is disabled a CTS1 transition will generate an interrupt if Modem Status Interrupt Enable (U1IER[3]) is set. In normal operation a CTS1 signal transition will generate a Modem Status Interrupt unless the interrupt has been disabled by clearing the U1IER[3] bit in the U1IER register. In auto-cts mode a transition on the CTS1 bit will trigger an interrupt only if both the U1IER[3] and U1IER[7] bits are set.	0
		0	Disable the CTS interrupt.	
		1	Enable the CTS interrupt.	

© NXP B.V. 2010. All rights reserved.

Table 294: UART1 Interrupt Enable Register (U1IER - address 0x4001 0004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
8	ABEOIntEn		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	
9	ABTOIntEn		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.5 UART1 Interrupt Identification Register (U1IIR - 0x4001 0008)

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 295: UART1 Interrupt Identification Register (U1IIR - address 0x4001 0008) bit description

Bit	Symbol	Value	Description	Reset Value
0	IntStatus		Interrupt status. Note that U1IIR[0] is active low. The pending interrupt can be determined by evaluating U1IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	IntId		Interrupt identification. U1IER[3:1] identifies an interrupt corresponding to the UART1 Rx or TX FIFO. All other combinations of U1IER[3:1] not listed below are reserved (100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt.	
		000	4 - Modem Interrupt.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		Copies of 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, the value read from a reserved bit is not defined.	NA

Bit U1IIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 15–296. Given the status of U1IIR[3:0], an

© NXP B.V. 2010. All rights reserved.

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 296: UART1 Interrupt Handling

U1IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	U1LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U1FCR0=1)	U1RBR Read[3] or UART1 FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the RX FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	U1RBR Read[3]
0010	Third	THRE	THRE[2]	U1IIR Read[4] (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 15–4.10 “UART1 Line Status Register (U1LSR - 0x4001 0014)”

[3] For details see Section 15–4.1 “UART1 Receiver Buffer Register (U1RBR - 0x4001 0000, when DLAB = 0)”

[4] For details see Section 15–4.5 “UART1 Interrupt Identification Register (U1IIR - 0x4001 0008)” and Section 15–4.2 “UART1 Transmitter Holding Register (U1THR - 0x4001 0000 when DLAB = 0)”

© NXP B.V. 2010. All rights reserved.

The UART1 THRE interrupt (U1IIR[3:1] = 001) is a third level interrupt and is activated when the UART1 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART1 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the U1THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U1THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART1 THR FIFO has held two or more characters at one time and currently, the U1THR is empty. The THRE interrupt is reset when a U1THR write occurs or a read of the U1IIR occurs and the THRE is the highest interrupt (U1IIR[3:1] = 001).

It is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a low to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining U1MSR[3:0]. A U1MSR read will clear the modem interrupt.

4.6 UART1 FIFO Control Register (U1FCR - 0x4001 0008)

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

Table 297: UART1 FIFO Control Register (U1FCR - address 0x4001 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, 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, reset the pointer logic. This bit is self-clearing.	
3	DMA Mode Select		When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 15–4.6.1.	0
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	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).	
31:8	-		Reserved, user software should not write ones to reserved bits.	NA

4.6.1 DMA Operation

The user can optionally operate the UART transmit and/or receive using DMA. The DMA mode is determined by the DMA Mode Select bit in the FCR register. This bit only has an affect when the FIFOs are enabled via the FIFO Enable bit in the FCR register.

© NXP B.V. 2010. All rights reserved.

UART receiver DMA

In DMA mode, the receiver DMA request is asserted on the event of the receiver FIFO level becoming equal to or greater than trigger level, or if a character timeout occurs. See the description of the RX Trigger Level above. The receiver DMA request is cleared by the DMA controller.

UART transmitter DMA

In DMA mode, the transmitter DMA request is asserted on the event of the transmitter FIFO transitioning to not full. The transmitter DMA request is cleared by the DMA controller.

4.7 UART1 Line Control Register (U1LCR - 0x4001 000C)

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

Table 298: UART1 Line Control Register (U1LCR - address 0x4001 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.	
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.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.8 UART1 Modem Control Register (U1MCR - 0x4001 0010)

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

© NXP B.V. 2010. All rights reserved.

Table 299: UART1 Modem Control Register (U1MCR - address 0x4001 0010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTR Control		Source for modem output pin, DTR. This bit reads as 0 when modem loopback mode is active.	0
1	RTS Control		Source for modem output pin RTS. This bit reads as 0 when modem loopback mode is active.	0
3-2	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
4	Loopback Mode Select		The modem loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RXD1, has no effect on loopback and output pin, TXD1 is held in marking state. The 4 modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the 4 modem outputs are connected to the 4 modem inputs. As a result of these connections, the upper 4 bits of the U1MSR will be driven by the lower 4 bits of the U1MCR rather than the 4 modem inputs in normal mode. This permits modem status interrupts to be generated in loopback mode by writing the lower 4 bits of U1MCR.	0
		0	Disable modem loopback mode.	
		1	Enable modem loopback mode.	
5	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
6	RTSen	0	Disable auto-rts flow control.	0
		1	Enable auto-rts flow control.	
7	CTSen	0	Disable auto-cts flow control.	0
		1	Enable auto-cts flow control.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.9 Auto-flow control

If auto-RTS mode is enabled the UART1‘s receiver FIFO hardware controls the RTS1 output of the UART1. If the auto-CTS mode is enabled the UART1‘s U1TSR hardware will only start transmitting if the CTS1 input signal is asserted.

15.4.9.1 Auto-RTS

The auto-RTS function is enabled by setting the RTSen bit. Auto-RTS data flow control originates in the U1RBR module and is linked to the programmed receiver FIFO trigger level. If auto-RTS is enabled, the data-flow is controlled as follows:

When the receiver FIFO level reaches the programmed trigger level, RTS1 is de-asserted (to a high value). It is possible that the sending UART sends an additional byte after the trigger level is reached (assuming the sending UART has another byte to send) because it might not recognize the de-assertion of RTS1 until after it has begun sending the additional byte. RTS1 is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The re-assertion of RTS1 signals to the sending UART to continue transmitting data.

© NXP B.V. 2010. All rights reserved.

If Auto-RTS mode is disabled, the RTSen bit controls the RTS1 output of the UART1. If Auto-RTS mode is enabled, hardware controls the RTS1 output, and the actual value of RTS1 will be copied in the RTS Control bit of the UART1. As long as Auto-RTS is enabled, the value of the RTS Control bit is read-only for software.

Example: Suppose the UART1 operating in '550 mode has trigger level in U1FCR set to 0x2 then if Auto-RTS is enabled the UART1 will de-assert the RTS1 output as soon as the receive FIFO contains 8 bytes (Table 15–297 on page 323). The RTS1 output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

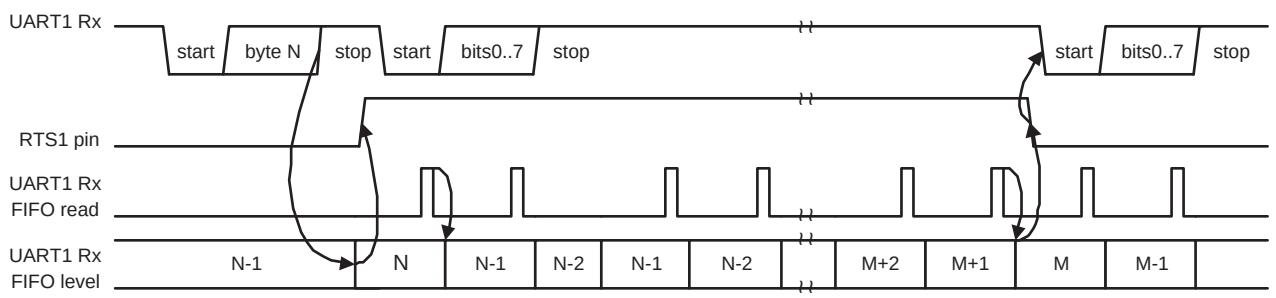


Fig 47. Auto-RTS Functional Timing

15.4.9.2 Auto-CTS

The Auto-CTS function is enabled by setting the CTSen bit. If Auto-CTS is enabled the transmitter circuitry in the U1TSR module checks CTS1 input before sending the next data byte. When CTS1 is active (low), the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS1 must be released before the middle of the last stop bit that is currently being sent. In Auto-CTS mode a change of the CTS1 signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, Delta CTS bit in the U1MSR will be set though. Table 15–300 lists the conditions for generating a Modem Status interrupt.

Table 300: Modem status interrupt generation

Enable Modem Status Interrupt (U1ER[3])	CTSen (U1MCR[7])	CTS Interrupt Enable (U1IER[7])	Delta CTS (U1MSR[0])	Delta DCD or Trailing Edge RI or Delta DSR (U1MSR[3] or U1MSR[2] or U1MSR[1])	Modem Status Interrupt
0	x	x	x	x	No
1	0	x	0	0	No
1	0	x	1	x	Yes
1	0	x	x	1	Yes
1	1	0	x	0	No
1	1	0	x	1	Yes
1	1	1	0	0	No
1	1	1	1	x	Yes
1	1	1	x	1	Yes

© NXP B.V. 2010. All rights reserved.

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 15–48 illustrates the Auto-CTS functional timing.

Fig 48. Auto-CTS Functional Timing

While starting transmission of the initial character the CTS1 signal is asserted. Transmission will stall as soon as the pending transmission has completed. The UART will continue transmitting a 1 bit as long as CTS1 is de-asserted (high). As soon as CTS1 gets de-asserted transmission resumes and a start bit is sent followed by the data bits of the next character.

4.10 UART1 Line Status Register (U1LSR - 0x4001 0014)

The U1LSR is a read-only register that provides status information on the UART1 TX and RX blocks.

Table 301: UART1 Line Status Register (U1LSR - address 0x4001 0014) 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	The UART1 receiver FIFO is empty.	
		1	The UART1 receiver FIFO is not empty.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An U1LSR read clears U1LSR[1]. U1LSR[1] is set when UART1 RSR has a new character assembled and the UART1 RBR FIFO is full. In this case, the UART1 RBR FIFO will not be overwritten and the character in the UART1 RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An U1LSR read clears U1LSR[2]. Time of parity error detection is dependent on U1FCR[0]. **Note:** A parity error is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	

© NXP B.V. 2010. All rights reserved.

Table 301: UART1 Line Status Register (U1LSR - address 0x4001 0014) 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 U1LSR read clears U1LSR[3]. The time of the framing error detection is dependent on U1FCR0. Upon detection of a framing error, the RX will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXD1 is held in the spacing state (all zeroes) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all ones). An U1LSR read clears this status bit. The time of break detection is dependent on U1FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE)		THRE is set immediately upon detection of an empty UART1 THR and is cleared on a U1THR write.	1
		0	U1THR contains valid data.	
		1	U1THR is empty.	
6	Transmitter Empty (TEMT)		TEMT is set when both U1THR and U1TSR are empty; TEMT is cleared when either the U1TSR or the U1THR contain valid data.	1
		0	U1THR and/or the U1TSR contains valid data.	
		1	U1THR and the U1TSR are empty.	
7	Error in RX FIFO (RXFE)		U1LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the U1RBR. This bit is cleared when the U1LSR register is read and there are no subsequent errors in the UART1 FIFO.	0
		0	U1RBR contains no UART1 RX errors or U1FCR[0]=0.	
		1	UART1 RBR contains at least one UART1 RX error.	
31:8	-		Reserved, the value read from a reserved bit is not defined.	NA

4.11 UART1 Modem Status Register (U1MSR - 0x4001 0018)

The U1MSR is a read-only register that provides status information on the modem input signals. U1MSR[3:0] is cleared on U1MSR read. Note that modem signals have no direct effect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 302: UART1 Modem Status Register (U1MSR - address 0x4001 0018) bit description

Bit	Symbol	Value	Description	Reset Value
0	Delta CTS		Set upon state change of input CTS. Cleared on an U1MSR read.	0
		0	No change detected on modem input, CTS.	
		1	State change detected on modem input, CTS.	

© NXP B.V. 2010. All rights reserved.

Table 302: UART1 Modem Status Register (U1MSR - address 0x4001 0018) bit description

Bit	Symbol	Value	Description	Reset Value
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
31:8	-		Reserved, the value read from a reserved bit is not defined.	NA

4.12 UART1 Scratch Pad Register (U1SCR - 0x4001 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 303: UART1 Scratch Pad Register (U1SCR - address 0x4001 0014) bit description

Bit	Symbol	Description	Reset Value
7:0	Pad	A readable, writable byte.	0x00

4.13 UART1 Auto-baud Control Register (U1ACR - 0x4001 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 304: Auto-baud Control Register (U1ACR - address 0x4001 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.	

© NXP B.V. 2010. All rights reserved.

Table 304: Auto-baud Control Register (U1ACR - address 0x4001 0020) bit description

Bit	Symbol	Value	Description	Reset value
2	AutoRestart	0	No restart	0
		1	Restart in case of time-out (counter restarts at next UART1 Rx falling edge)	0
7:3	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOIntClr		End of auto-baud interrupt clear bit (write-only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the U1IIR.	
9	ABTOIntClr		Auto-baud time-out interrupt clear bit (write-only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the U1IIR.	
31:10	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

4.14 Auto-baud

The UART1 auto-baud function can be used to measure the incoming baud-rate based on the “AT” protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers U1DLM and U1DLL accordingly.

Remark: the fractional rate divider is not connected during auto-baud operations, and therefore should not be used when the auto-baud feature is needed.

Auto-baud is started by setting the U1ACR Start bit. Auto-baud can be stopped by clearing the U1ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the U1ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART1 Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UART1 Rx pin (the length of the start bit).

The U1ACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART1 Rx pin.

The auto-baud function can generate two interrupts.

- The U1IIR ABTOInt interrupt will get set if the interrupt is enabled (U1IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U1IIR ABEOInt interrupt will get set if the interrupt is enabled (U1IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U1ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART1 Rx pin baud-rate, but the value of the U1FDR

© NXP B.V. 2010. All rights reserved.

register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U1DLM and U1DLL registers should be done before U1ACR register write. The minimum and the maximum baud rates supported by UART1 are function of pclk, number of data bits, stop bits and parity bits.

(3)

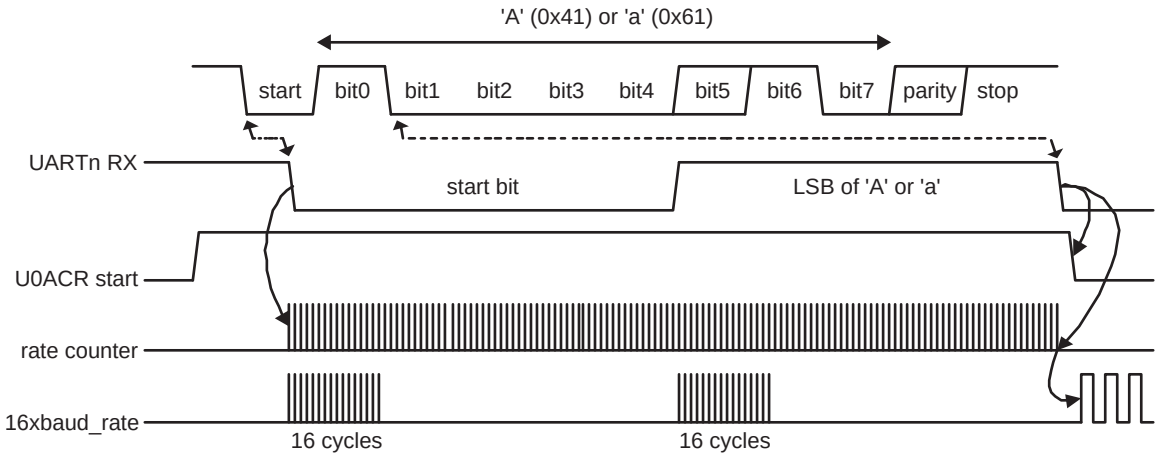
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART1_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

4.15 Auto-baud modes

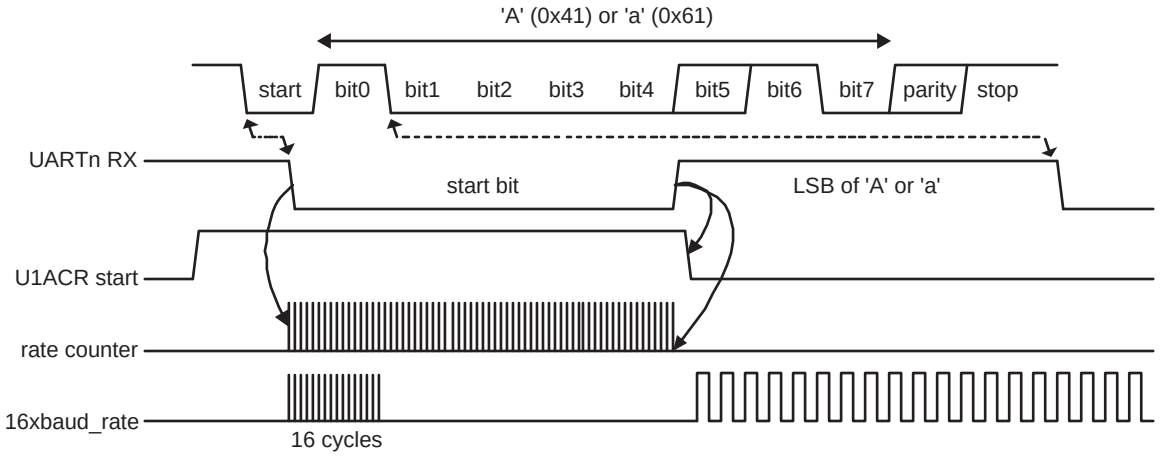
When the software is expecting an “AT” command, it configures the UART1 with the expected character format and sets the U1ACR Start bit. The initial values in the divisor latches U1DLM and U1DLM don‘t care. Because of the “A” or “a” ASCII coding (”A" = 0x41, “a” = 0x61), the UART1 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U1ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U1ACR Start bit setting, the baud-rate measurement counter is reset and the UART1 U1RSR is reset. The U1RSR baud rate is switch to the highest rate.
2. A falling edge on UART1 Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting pclk cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UART1 input clock, guaranteeing the start bit is stored in the U1RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART1 input clock (pclk).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART1 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART1 Rx pin.
6. The rate counter is loaded into U1DLM/U1DLL and the baud-rate will be switched to normal operation. After setting the U1DLM/U1DLL the end of auto-baud interrupt U1IIR ABEOInt will be set, if enabled. The U1RSR will now continue receiving the remaining bits of the “A/a” character.

© NXP B.V. 2010. All rights reserved.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 49. Auto-baud a) mode 0 and b) mode 1 waveform

4.16 UART1 Fractional Divider Register (U1FDR - 0x4001 0028)

The UART1 Fractional Divider Register (U1FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be greater than 2.

© NXP B.V. 2010. All rights reserved.

Table 305: UART1 Fractional Divider Register (U1FDR - address 0x4001 0028) bit description

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

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

UART1 baud rate can be calculated as (n = 1):

(4)

$$UART1_{baudrate} = \frac{PCLK}{16 \times (256 \times U1DLM + U1DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

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

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 14$
3. $DIVADDVAL < MULVAL$

The value of the U1FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the U1FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

4.16.1 Baud rate calculation

UART1 can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baud rate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baud rate with a relative error of less than 1.1% from the desired one.

© NXP B.V. 2010. All rights reserved.

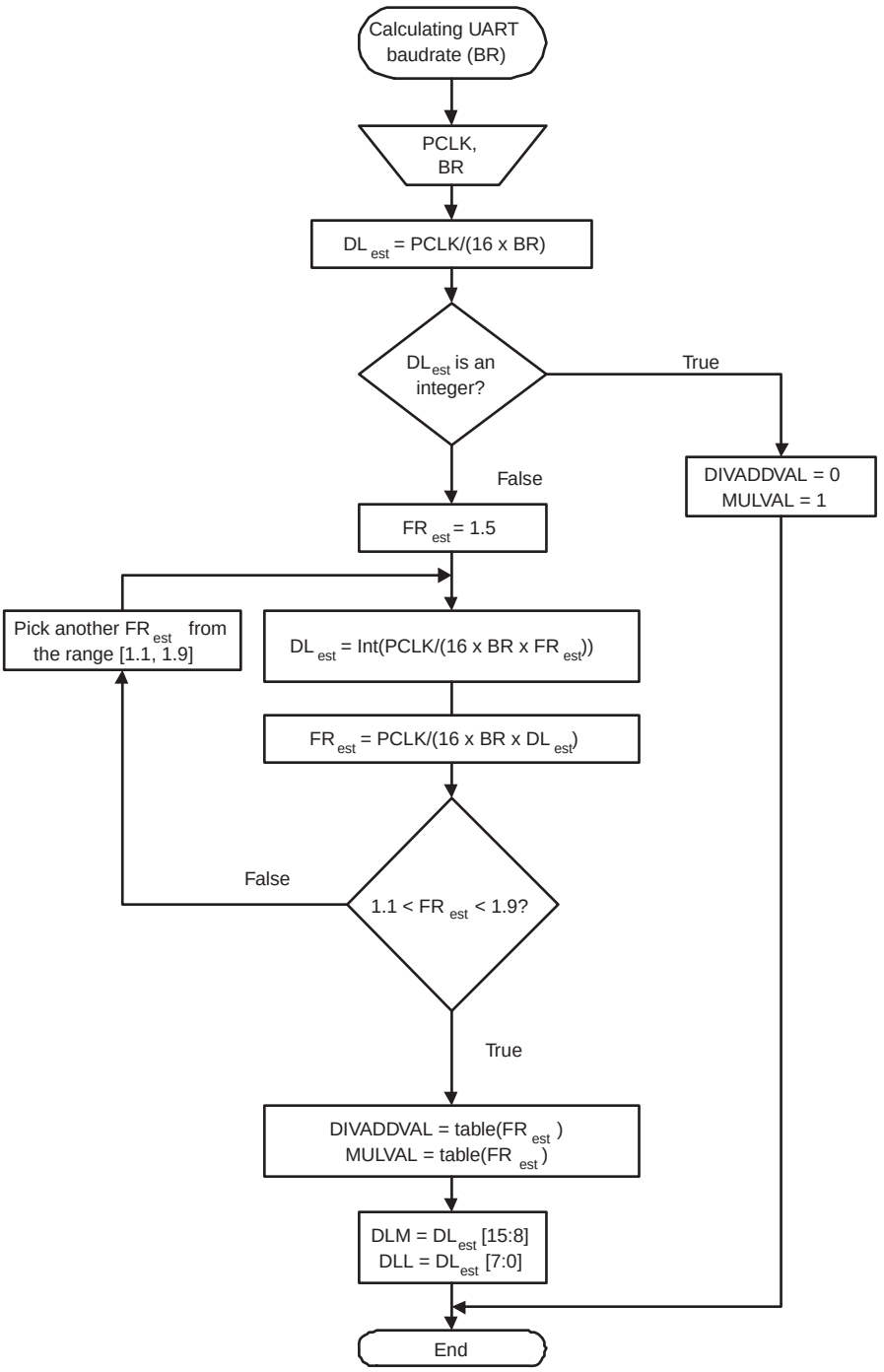


Fig 50. Algorithm for setting UART dividers

© NXP B.V. 2010. All rights reserved.

Table 306. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

4.16.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

4.16.1.2 Example 2: PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 15–306 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 15–4 the UART rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

4.17 UART1 Transmit Enable Register (U1TER - 0x4001 0030)

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), U1TER enables implementation of software flow control, too. When TxEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TxEn becomes 0, UART1 transmission will stop.

© NXP B.V. 2010. All rights reserved.

Although Table 15–307 describes how to use TxEn bit in order to achieve hardware flow control, it is strongly suggested to let UART1 hardware implemented auto flow control features take care of this, and limit the scope of TxEn to software flow control.

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 15–307 describes how to use TXEn bit in order to achieve software flow control.

Table 307: UART1 Transmit Enable Register (U1TER - address 0x4001 0030) bit description

Bit	Symbol	Description	Reset Value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software can clear this bit when it detects that the a hardware-handshaking TX-permit signal (CTS) has gone false, or with software handshaking, when it receives an XOFF character (DC3). Software can set this bit again when it detects that the TX-permit signal has gone true, or when it receives an XON (DC1) character.	1
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.18 UART1 RS485 Control register (U1RS485CTRL - 0x4001 004C)

The U1RS485CTRL register controls the configuration of the UART in RS-485/EIA-485 mode.

Table 308: UART1 RS485 Control register (U1RS485CTRL - address 0x4001 004C) bit description

Bit	Symbol	Value	Description	Reset value
0	NMMEN	0	RS-485/EIA-485 Normal Multidrop Mode (NMM) is disabled.	0
		1	RS-485/EIA-485 Normal Multidrop Mode (NMM) is enabled. In this mode, an address is detected when a received byte causes the UART to set the parity error and generate an interrupt.	
1	RXDIS	0	The receiver is enabled.	0
		1	The receiver is disabled.	
2	AADEN	0	Auto Address Detect (AAD) is disabled.	0
		1	Auto Address Detect (AAD) is enabled.	
3	SEL	0	If direction control is enabled (bit DCTRL = 1), pin $\overline{\text{RTS}}$ is used for direction control.	0
		1	If direction control is enabled (bit DCTRL = 1), pin DTR is used for direction control.	
4	DCTRL	0	Disable Auto Direction Control.	0
		1	Enable Auto Direction Control.	

© NXP B.V. 2010. All rights reserved.

Table 308: UART1 RS485 Control register (U1RS485CTRL - address 0x4001 004C) bit description

Bit	Symbol	Value	Description	Reset value
5	OINV		This bit reverses the polarity of the direction control signal on the RTS (or DTR) pin.	0
		0	The direction control pin will be driven to logic ‘0’ when the transmitter has data to be sent. It will be driven to logic ‘1’ after the last bit of data has been transmitted.	
		1	The direction control pin will be driven to logic ‘1’ when the transmitter has data to be sent. It will be driven to logic ‘0’ after the last bit of data has been transmitted.	
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.19 UART1 RS-485 Address Match register (U1RS485ADRMATCH - 0x4001 0050)

The U1RS485ADRMATCH register contains the address match value for RS-485/EIA-485 mode.

Table 309. UART1 RS-485 Address Match register (U1RS485ADRMATCH - address 0x4001 0050) bit description

Bit	Symbol	Description	Reset value
7:0	ADRMATCH	Contains the address match value.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.20 UART1 RS-485 Delay value register (U1RS485DLY - 0x4001 0054)

The user may program the 8-bit RS485DLY register with a delay between the last stop bit leaving the TXFIFO and the de-assertion of RTS (or DTR). This delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

Table 310. UART1 RS-485 Delay value register (U1RS485DLY - address 0x4001 0054) bit description

Bit	Symbol	Description	Reset value
7:0	DLY	Contains the direction control (RTS or DTR) delay value. This register works in conjunction with an 8-bit counter.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.21 RS-485/EIA-485 modes of operation

The RS-485/EIA-485 feature allows the UART to be configured as an addressable slave. The addressable slave is one of multiple slaves controlled by a single master.

The UART master transmitter will identify an address character by setting the parity (9th) bit to ‘1’. For data characters, the parity bit is set to ‘0’.

Each UART slave receiver can be assigned a unique address. The slave can be programmed to either manually or automatically reject data following an address which is not theirs.

RS-485/EIA-485 Normal Multidrop Mode (NMM)

Setting the RS485CTRL bit 0 enables this mode. In this mode, an address is detected when a received byte causes the UART to set the parity error and generate an interrupt.

© NXP B.V. 2010. All rights reserved.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received data bytes will be ignored and will not be stored in the RXFIFO. When an address byte is detected (parity bit = ‘1’) it will be placed into the RXFIFO and an Rx Data Ready Interrupt will be generated. The processor can then read the address byte and decide whether or not to enable the receiver to accept the following data.

While the receiver is ENABLED (RS485CTRL bit 1 =’0’) all received bytes will be accepted and stored in the RXFIFO regardless of whether they are data or address. When an address character is received a parity error interrupt will be generated and the processor can decide whether or not to disable the receiver.

RS-485/EIA-485 Auto Address Detection (AAD) mode

When both RS485CTRL register bits 0 (9-bit mode enable) and 2 (AAD mode enable) are set, the UART is in auto address detect mode.

In this mode, the receiver will compare any address byte received (parity = ‘1’) to the 8-bit value programmed into the RS485ADRMATCH register.

If the receiver is DISABLED (RS485CTRL bit 1 = ‘1’) any received byte will be discarded if it is either a data byte OR an address byte which fails to match the RS485ADRMATCH value.

When a matching address character is detected it will be pushed onto the RXFIFO along with the parity bit, and the receiver will be automatically enabled (RS485CTRL bit 1 will be cleared by hardware). The receiver will also generate n Rx Data Ready Interrupt.

While the receiver is ENABLED (RS485CTRL bit 1 = ‘0’) all bytes received will be accepted and stored in the RXFIFO until an address byte which does not match the RS485ADRMATCH value is received. When this occurs, the receiver will be automatically disabled in hardware (RS485CTRL bit 1 will be set), The received non-matching address character will not be stored in the RXFIFO.

RS-485/EIA-485 Auto Direction Control

RS485/EIA-485 Mode includes the option of allowing the transmitter to automatically control the state of either the $\overline{\text{RTS}}$ pin or the $\overline{\text{DTR}}$ pin as a direction control output signal.

Setting RS485CTRL bit 4 = ‘1’ enables this feature.

Direction control, if enabled, will use the $\overline{\text{RTS}}$ pin when RS485CTRL bit 3 = ‘0’. It will use the $\overline{\text{DTR}}$ pin when RS485CTRL bit 3 = ‘1’.

When Auto Direction Control is enabled, the selected pin will be asserted (driven low) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven high) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

The RS485CTRL bit 4 takes precedence over all other mechanisms controlling $\overline{\text{RTS}}$ (or $\overline{\text{DTR}}$) with the exception of loopback mode.

RS485/EIA-485 driver delay time

The driver delay time is the delay between the last stop bit leaving the TXFIFO and the de-assertion of RTS (or DTR). This delay time can be programmed in the 8-bit RS485DLY register. The delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

© NXP B.V. 2010. All rights reserved.

RS485/EIA-485 output inversion

The polarity of the direction control signal on the $\overline{RTS}$ (or $\overline{DTR}$) pins can be reversed by programming bit 5 in the U1RS485CTRL register. When this bit is set, the direction control pin will be driven to logic 1 when the transmitter has data waiting to be sent. The direction control pin will be driven to logic 0 after the last bit of data has been transmitted.

4.22 UART1 FIFO Level register (U1FIFOLVL - 0x4001 0058)

U1FIFOLVL register is a read-only register that allows software to read the current FIFO level status. Both the transmit and receive FIFO levels are present in this register.

Table 311. UART1 FIFO Level register (U1FIFOLVL - address 0x4001 0058) bit description

Bit	Symbol	Description	Reset value
3:0	RXFIFILVL	Reflects the current level of the UART1 receiver FIFO. 0 = empty, 0xF = FIFO full.	0x00
7:4	-	Reserved. The value read from a reserved bit is not defined.	NA
11:8	TXFIFOLVL	Reflects the current level of the UART1 transmitter FIFO. 0 = empty, 0xF = FIFO full.	0x00
31:12	-	Reserved, the value read from a reserved bit is not defined.	NA

5. Architecture

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

The APB interface provides a communications link between the CPU or host and the UART1.

The UART1 receiver block, U1RX, monitors the serial input line, RXD1, for valid input. The UART1 RX Shift Register (U1RSR) accepts valid characters via RXD1. After a valid character is assembled in the U1RSR, it is passed to the UART1 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

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

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 TX block. The U1BRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the U1DLL and U1DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The modem interface contains registers U1MCR and U1MSR. This interface is responsible for handshaking between a modem peripheral and the UART1.

The interrupt interface contains registers U1IER and U1IIR. The interrupt interface receives several one clock wide enables from the U1TX and U1RX blocks.

Status information from the U1TX and U1RX is stored in the U1LSR. Control information for the U1TX and U1RX is stored in the U1LCR.

© NXP B.V. 2010. All rights reserved.

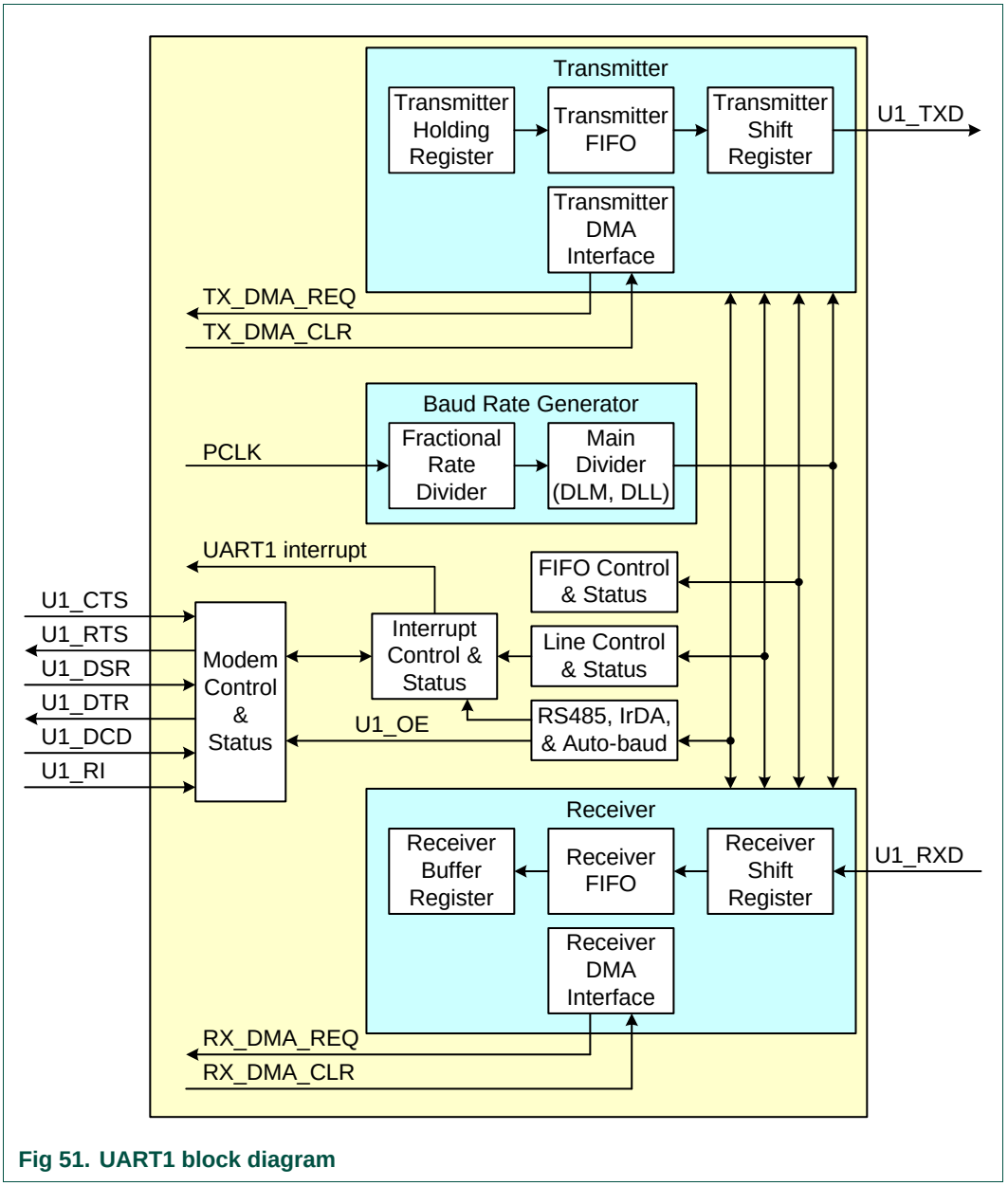


Fig 51. UART1 block diagram

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The CAN1/2 peripherals are configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bits PCAN1/2.

   Remark: On reset, the CAN1/2 blocks are disabled (PCAN1/2 = 0).

2. Peripheral clock: In the PCLKSEL0 register (Table 4–40), select PCLK_CAN1, PCLK_CAN2, and, for the acceptance filter, PCLK_ACF. Note that these must all be the same value.

   Remark: If CAN baud rates above 100 kbit/s (see Table 16–323) are needed, do not select the IRC as the clock source (see Table 4–17).

3. Wake-up: CAN controllers are able to wake up the microcontroller from Power-down mode, see Section 4–8.8.

4. Pins: Select CAN1/2 pins through the PINSEL registers and their pin modes through the PINMODE registers (Section 8–5).

5. Interrupts: CAN interrupts are enabled using the CAN1/2IER registers (Table 16–322). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

6. CAN controller initialization: see CANMOD register (Section 16–7.1).

2. CAN controllers

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The CAN Controller is designed to provide a full implementation of the CAN-Protocol according to the CAN Specification Version 2.0B. Microcontrollers with this on-chip CAN controller are used to build powerful local networks by supporting distributed real-time control with a very high level of security. The applications are automotive, industrial environments, and high speed networks as well as low cost multiplex wiring. The result is a strongly reduced wiring harness and enhanced diagnostic and supervisory capabilities.

The CAN block is intended to support multiple CAN buses simultaneously, allowing the device to be used as a gateway, switch, or router among a number of CAN buses in various applications.

The CAN module consists of two elements: the controller and the Acceptance Filter. All registers and the RAM are accessed as 32-bit words.

3. Features

3.1 General CAN features

- Compatible with *CAN specification 2.0B, ISO 11898-1*.
- Multi-master architecture with non destructive bit-wise arbitration.

© NXP B.V. 2010. All rights reserved.

- Bus access priority determined by the message identifier (11-bit or 29-bit).
- Guaranteed latency time for high priority messages.
- Programmable transfer rate (up to 1 Mbit/s).
- Multicast and broadcast message facility.
- Data length from 0 up to 8 bytes.
- Powerful error handling capability.
- Non-return-to-zero (NRZ) encoding/decoding with bit stuffing.

3.2 CAN controller features

- 2 CAN controllers and buses.
- Supports 11-bit identifier as well as 29-bit identifier.
- Double Receive Buffer and Triple Transmit Buffer.
- Programmable Error Warning Limit and Error Counters with read/write access.
- Arbitration Lost Capture and Error Code Capture with detailed bit position.
- Single Shot Transmission (no re-transmission).
- Listen Only Mode (no acknowledge, no active error flags).
- Reception of "own" messages (Self Reception Request).

3.3 Acceptance filter features

- Fast hardware implemented search algorithm supporting a large number of CAN identifiers.
- Global Acceptance Filter recognizes 11-bit and 29-bit Rx Identifiers for all CAN buses.
- Allows definition of explicit and groups for 11-bit and 29-bit CAN identifiers.
- Acceptance Filter can provide FullCAN-style automatic reception for selected Standard Identifiers.

4. Pin description

Table 312. CAN Pin descriptions

Pin Name	Type	Description
RD1, RD2	Input	**Serial Inputs.** From CAN transceivers.
TD1, TD2	Output	**Serial Outputs.** To CAN transceivers.

5. CAN controller architecture

The CAN Controller is a complete serial interface with both Transmit and Receive Buffers but without Acceptance Filter. CAN Identifier filtering is done for all CAN channels in a separate block (Acceptance Filter). Except for message buffering and acceptance filtering the functionality is similar to the PeliCAN concept.

The CAN Controller Block includes interfaces to the following blocks:

- APB Interface

© NXP B.V. 2010. All rights reserved.

- Acceptance Filter
- Nested Vectored Interrupt Controller (NVIC)
- CAN Transceiver
- Common Status Registers

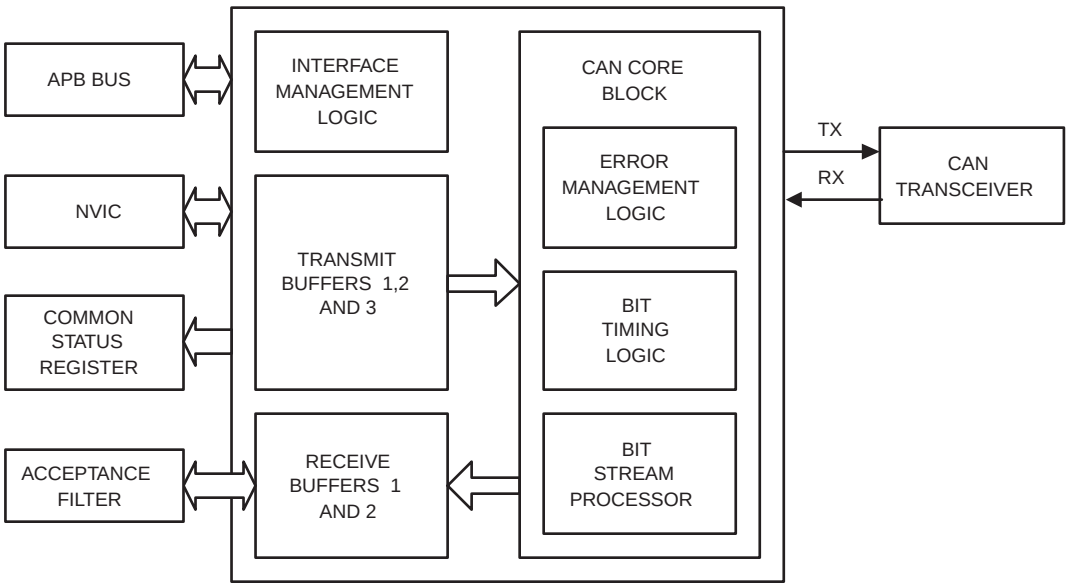


Fig 52. CAN controller block diagram

5.1 APB Interface Block (AIB)

The APB Interface Block provides access to all CAN Controller registers.

5.2 Interface Management Logic (IML)

The Interface Management Logic interprets commands from the CPU, controls internal addressing of the CAN Registers and provides interrupts and status information to the CPU.

5.3 Transmit Buffers (TXB)

The TXB represents a Triple Transmit Buffer, which is the interface between the Interface Management Logic (IML) and the Bit Stream Processor (BSP). Each Transmit Buffer is able to store a complete message which can be transmitted over the CAN network. This buffer is written by the CPU and read out by the BSP.

© NXP B.V. 2010. All rights reserved.

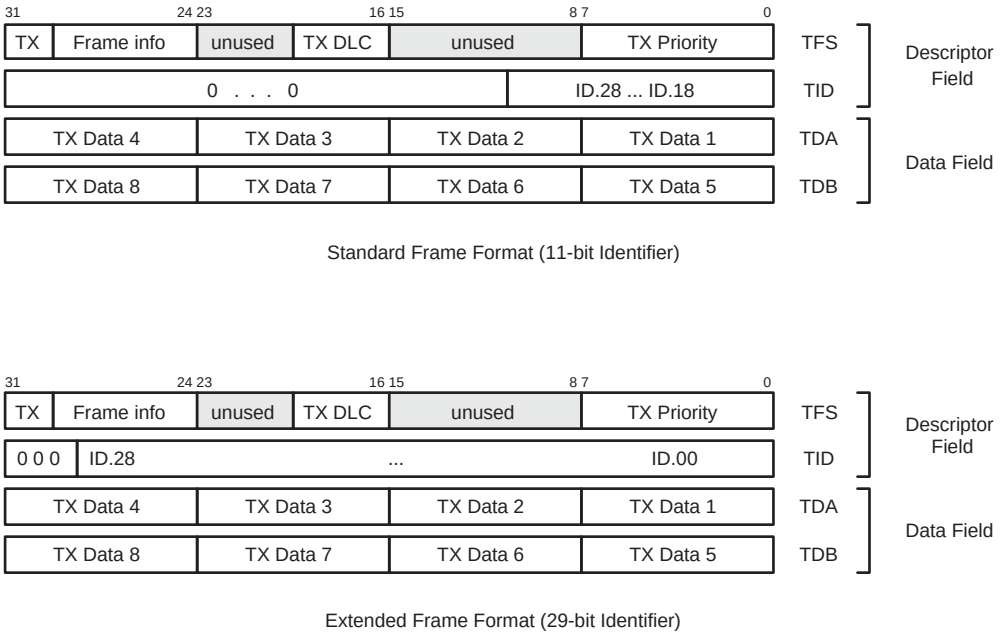


Fig 53. Transmit buffer layout for standard and extended frame format configurations

5.4 Receive Buffer (RXB)

The Receive Buffer (RXB) represents a CPU accessible Double Receive Buffer. It is located between the CAN Controller Core Block and APB Interface Block and stores all received messages from the CAN Bus line. With the help of this Double Receive Buffer concept the CPU is able to process one message while another message is being received.

The global layout of the Receive Buffer is very similar to the Transmit Buffer described earlier. Identifier, Frame Format, Remote Transmission Request bit and Data Length Code have the same meaning as described for the Transmit Buffer. In addition, the Receive Buffer includes an ID Index field (see Section 16–7.9.1 “ID index field”).

The received Data Length Code represents the real transmitted Data Length Code, which may be greater than 8 depending on transmitting CAN node. Nevertheless, the maximum number of received data bytes is 8. This should be taken into account by reading a message from the Receive Buffer. If there is not enough space for a new message within the Receive Buffer, the CAN Controller generates a Data Overrun condition when this message becomes valid and the acceptance test was positive. A message that is partly written into the Receive Buffer (when the Data Overrun situation occurs) is deleted. This situation is signalled to the CPU via the Status Register and the Data Overrun Interrupt, if enabled.

© NXP B.V. 2010. All rights reserved.

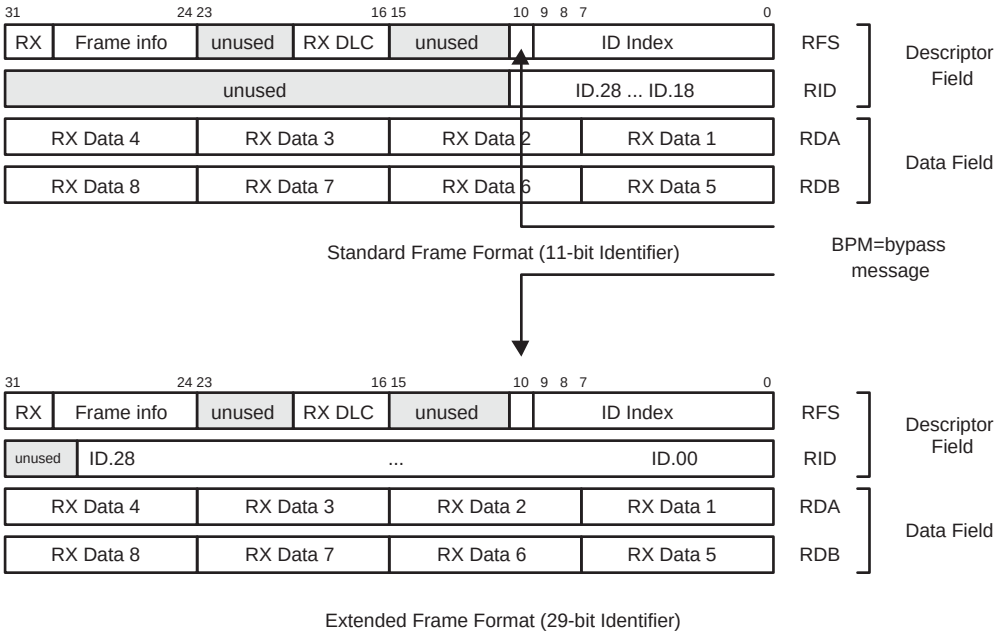


Fig 54. Receive buffer layout for standard and extended frame format configurations

5.5 Error Management Logic (EML)

The EML is responsible for the error confinement. It gets error announcements from the BSP and then informs the BSP and IML about error statistics.

5.6 Bit Timing Logic (BTL)

The Bit Timing Logic monitors the serial CAN Bus line and handles the Bus line related bit timing. It synchronizes to the bit stream on the CAN Bus on a "recessive" to "dominant" Bus line transition at the beginning of a message (hard synchronization) and re-synchronizes on further transitions during the reception of a message (soft synchronization). The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g. due to oscillator drifts) and to define the sample point and the number of samples to be taken within a bit time.

5.7 Bit Stream Processor (BSP)

The Bit Stream Processor is a sequencer, controlling the data stream between the Transmit Buffer, Receive Buffers and the CAN Bus. It also performs the error detection, arbitration, stuffing and error handling on the CAN Bus.

5.8 CAN controller self-tests

The CAN controller supports two different options for self-tests:

- Global Self-Test (setting the self reception request bit in normal Operating Mode)
- Local Self-Test (setting the self reception request bit in Self Test Mode)

© NXP B.V. 2010. All rights reserved.

Both self-tests are using the ‘Self Reception’ feature of the CAN Controller. With the Self Reception Request, the transmitted message is also received and stored in the receive buffer. Therefore the acceptance filter has to be configured accordingly. As soon as the CAN message is transmitted, a transmit and a receive interrupt are generated, if enabled.

Global self test

A Global Self-Test can for example be used to verify the chosen configuration of the CAN Controller in a given CAN system. As shown in Figure 16–55, at least one other CAN node, which is acknowledging each CAN message has to be connected to the CAN bus.

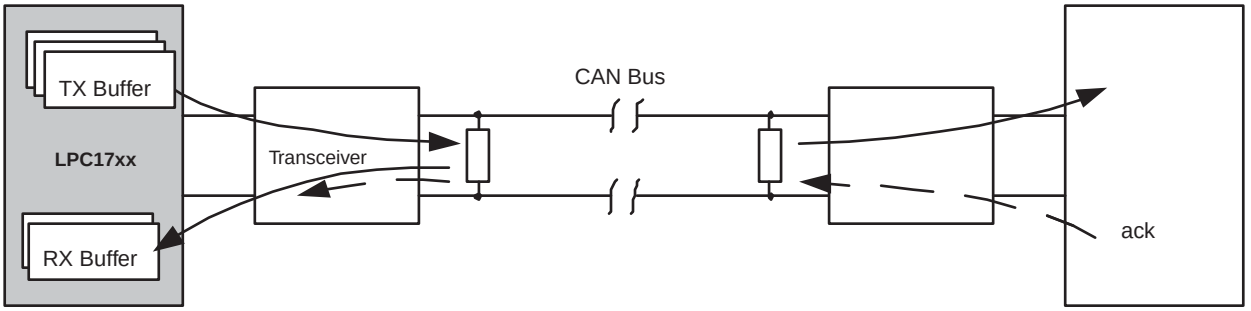


Fig 55. Global Self-Test (high-speed CAN Bus example)

Initiating a Global Self-Test is similar to a normal CAN transmission. In this case the transmission of a CAN message(s) is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer bits (STB3, STB2, STB1) in the CAN Controller Command register (CANCMR).

Local self test

The Local Self-Test perfectly fits for single node tests. In this case an acknowledge from other nodes is not needed. As shown in the Figure below, a CAN transceiver with an appropriate CAN bus termination has to be connected to the LPC17xx. The CAN Controller has to be put into the 'Self Test Mode' by setting the STM bit in the CAN Controller Mode register (CANMOD). Hint: Setting the Self Test Mode bit (STM) is possible only when the CAN Controller is in Reset Mode.

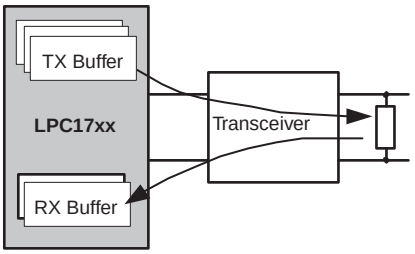


Fig 56. Local self test (high-speed CAN Bus example)

A message transmission is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer(s) (STB3, STB2, STB1).

© NXP B.V. 2010. All rights reserved.

6. Memory map of the CAN block

The CAN Controllers and Acceptance Filter occupy a number of APB slots, as follows:

Table 313. Memory map of the CAN block

Address Range	Used for
0x4003 8000 - 0x4003 87FF	Acceptance Filter RAM.
0x4003 C000 - 0x4003 C017	Acceptance Filter Registers.
0x4004 0000 - 0x4004 000B	Central CAN Registers.
0x4004 4000 - 0x4004 405F	CAN Controller 1 Registers.
0x4004 8000 - 0x4004 805F	CAN Controller 2 Registers.
0x400F C110 - 0x400F C114	CAN Wake and Sleep Registers.

7. CAN controller registers

CAN block implements the registers shown in Table 16–314 and Table 16–315. More detailed descriptions follow.

Table 314. CAN acceptance filter and central CAN registers

Name	Description	Access	Reset Value	Address
AFMR	Acceptance Filter Register	R/W	1	0x4003 C000
SFF_sa	Standard Frame Individual Start Address Register	R/W	0	0x4003 C004
SFF_GRP_sa	Standard Frame Group Start Address Register	R/W	0	0x4003 C008
EFF_sa	Extended Frame Start Address Register	R/W	0	0x4003 C00C
EFF_GRP_sa	Extended Frame Group Start Address Register	R/W	0	0x4003 C010
ENDofTable	End of AF Tables register	R/W	0	0x4003 C014
LUTerrAd	LUT Error Address register	RO	0	0x4003 C018
LUTerr	LUT Error Register	RO	0	0x4003 C01C
CANTxSR	CAN Central Transmit Status Register	RO	0x0003 0300	0x4004 0000
CANRxSR	CAN Central Receive Status Register	RO	0	0x4004 0004
CANMSR	CAN Central Miscellaneous Register	RO	0	0x4004 0008

Table 315. CAN1 and CAN2 controller register map

Generic Name	Description	Access	Reset value	CAN1 & 2 Register Name & Address
MOD	Controls the operating mode of the CAN Controller.	R/W	1	CAN1MOD - 0x4004 4000 CAN2MOD - 0x4004 8000
CMR	Command bits that affect the state of the CAN Controller	WO	0	CAN1CMR - 0x4004 4004 CAN2CMR - 0x4004 8004
GSR	Global Controller Status and Error Counters	RO[1]	0x3C	CAN1GSR - 0x4004 4008 CAN2GSR - 0x4004 8008
ICR	Interrupt status, Arbitration Lost Capture, Error Code Capture	RO	0	CAN1ICR - 0x4004 400C CAN2ICR - 0x4004 800C
IER	Interrupt Enable	R/W	0	CAN1IER - 0x4004 4010 CAN2IER - 0x4004 8010

© NXP B.V. 2010. All rights reserved.

Table 315. CAN1 and CAN2 controller register map

Generic Name	Description	Access	Reset value	CAN1 & 2 Register Name & Address
BTR	Bus Timing	R/W[2]	0x1C0000	CAN1BTR - 0x4004 4014 CAN2BTR - 0x4004 8014
EWL	Error Warning Limit	R/W[2]	0x60	CAN1EWL - 0x4004 4018 CAN2EWL - 0x4004 8018
SR	Status Register	RO	0x3C3C3C	CAN1SR - 0x4004 401C CAN2SR - 0x4004 801C
RFS	Receive frame status	R/W[2]	0	CAN1RFS - 0x4004 4020 CAN2RFS - 0x4004 8020
RID	Received Identifier	R/W[2]	0	CAN1RID - 0x4004 4024 CAN2RID - 0x4004 8024
RDA	Received data bytes 1-4	R/W[2]	0	CAN1RDA - 0x4004 4028 CAN2RDA - 0x4004 8028
RDB	Received data bytes 5-8	R/W[2]	0	CAN1RDB - 0x4004 402C CAN2RDB - 0x4004 802C
TFI1	Transmit frame info (Tx Buffer 1)	R/W	0	CAN1TFI1 - 0x4004 4030 CAN2TFI1 - 0x4004 8030
TID1	Transmit Identifier (Tx Buffer 1)	R/W	0	CAN1TID1 - 0x4004 4034 CAN2TID1 - 0x4004 8034
TDA1	Transmit data bytes 1-4 (Tx Buffer 1)	R/W	0	CAN1TDA1 - 0x4004 4038 CAN2TDA1 - 0x4004 8038
TDB1	Transmit data bytes 5-8 (Tx Buffer 1)	R/W	0	CAN1TDB1- 0x4004 403C CAN2TDB1 - 0x4004 803C
TFI2	Transmit frame info (Tx Buffer 2)	R/W	0	CAN1TFI2 - 0x4004 4040 CAN2TFI2 - 0x4004 8040
TID2	Transmit Identifier (Tx Buffer 2)	R/W	0	CAN1TID2 - 0x4004 4044 CAN2TID2 - 0x4004 8044
TDA2	Transmit data bytes 1-4 (Tx Buffer 2)	R/W	0	CAN1TDA2 - 0x4004 4048 CAN2TDA2 - 0x4004 8048
TDB2	Transmit data bytes 5-8 (Tx Buffer 2)	R/W	0	CAN1TDB2 - 0x4004 404C CAN2TDB2 - 0x4004 804C
TFI3	Transmit frame info (Tx Buffer 3)	R/W	0	CAN1TFI3 - 0x4004 4050 CAN2TFI3 - 0x4004 8050
TID3	Transmit Identifier (Tx Buffer 3)	R/W	0	CAN1TID3 - 0x4004 4054 CAN2TID3 - 0x4004 8054
TDA3	Transmit data bytes 1-4 (Tx Buffer 3)	R/W	0	CAN1TDA3 - 0x4004 4058 CAN2TDA3 - 0x4004 8058
TDB3	Transmit data bytes 5-8 (Tx Buffer 3)	R/W	0	CAN1TDB3 - 0x4004 405C CAN2TDB3 - 0x4004 805C

[1] The error counters can only be written when RM in CANMOD is 1.

[2] These registers can only be written when RM in CANMOD is 1.

© NXP B.V. 2010. All rights reserved.

The internal registers of each CAN Controller appear to the CPU as on-chip memory mapped peripheral registers. Because the CAN Controller can operate in different modes (Operating/Reset, see also Section 16–7.1 “CAN Mode register (CAN1MOD - 0x4004 4000, CAN2MOD - 0x4004 8000)”), one has to distinguish between different internal address definitions. Note that write access to some registers is only allowed in Reset Mode.

Table 316. CAN1 and CAN2 controller register summary

Generic Name	Operating Mode		Reset Mode	
	Read	Write	Read	Write
MOD	Mode	Mode	Mode	Mode
CMR	0x00	Command	0x00	Command
GSR	Global Status and Error Counters	-	Global Status and Error Counters	Error Counters only
ICR	Interrupt and Capture	-	Interrupt and Capture	-
IER	Interrupt Enable	Interrupt Enable	Interrupt Enable	Interrupt Enable
BTR	Bus Timing	-	Bus Timing	Bus Timing
EWL	Error Warning Limit	-	Error Warning Limit	Error Warning Limit
SR	Status	-	Status	-
RFS	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
RID	Rx Identifier	-	Rx Identifier	Rx Identifier
RDA	Rx Data	-	Rx Data	Rx Data
RDB	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
TFI1	Tx Info1	Tx Info	Tx Info	Tx Info
TID1	Tx Identifier	Tx Identifier	Tx Identifier	Tx Identifier
TDA1	Tx Data	Tx Data	Tx Data	Tx Data
TDB1	Tx Data	Tx Data	Tx Data	Tx Data

Table 317. CAN Wake and Sleep registers

Name	Description	Access	Reset Value	Address
CANSLEEPCLR	Allows clearing the current CAN channel sleep state as well as reading that state.	R/W	0	0x400F C110
CANWAKEFLAGS	Allows reading the wake-up state of the CAN channels.	R/W	0	0x400F C114

In the following register tables, the column “Reset Value” shows how a hardware reset affects each bit or field, while the column “RM Set” indicates how each bit or field is affected if software sets the RM bit, or RM is set because of a Bus-Off condition. Note that while hardware reset sets RM, in this case the setting noted in the “Reset Value” column prevails over that shown in the “RM Set” column, in the few bits where they differ. In both columns, X indicates the bit or field is unchanged.

7.1 CAN Mode register (CAN1MOD - 0x4004 4000, CAN2MOD - 0x4004 8000)

The contents of the Mode Register are used to change the behavior of the CAN Controller. Bits may be set or reset by the CPU that uses the Mode Register as a read/write memory.

© NXP B.V. 2010. All rights reserved.

Table 318. CAN Mode register (CAN1MOD - address 0x4004 4000, CAN2MOD - address 0x4004 8000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RM[1][6]		Reset Mode.	1	1
		0 (normal)	The CAN Controller is in the Operating Mode, and certain registers can not be written.		
		1 (reset)	CAN operation is disabled, writable registers can be written and the current transmission/reception of a message is aborted.		
1	LOM[3][2][6]		Listen Only Mode.	0	x
		0 (normal)	The CAN controller acknowledges a successfully received message on the CAN bus. The error counters are stopped at the current value.		
		1 (listen only)	The controller gives no acknowledgment, even if a message is successfully received. Messages cannot be sent, and the controller operates in “error passive” mode. This mode is intended for software bit rate detection and “hot plugging”.		
2	STM[3][6]		Self Test Mode.	0	x
		0 (normal)	A transmitted message must be acknowledged to be considered successful.		
		1 (self test)	The controller will consider a Tx message successful even if there is no acknowledgment received. In this mode a full node test is possible without any other active node on the bus using the SRR bit in CANxCMR.		
3	TPM[4]		Transmit Priority Mode.	0	x
		0 (CAN ID)	The transmit priority for 3 Transmit Buffers depends on the CAN Identifier.		
		1 (local prio)	The transmit priority for 3 Transmit Buffers depends on the contents of the Tx Priority register within the Transmit Buffer.		
4	SM[5]		Sleep Mode.	0	0
		0 (wake-up)	Normal operation.		
		1 (sleep)	The CAN controller enters Sleep Mode if no CAN interrupt is pending and there is no bus activity. See the Sleep Mode description Section 16–8.2 on page 369.		
5	RPM		Receive Polarity Mode.	0	x
		0 (low active)	RD input is active Low (dominant bit = 0).		
		1 (high active)	RD input is active High (dominant bit = 1) -- reverse polarity.		
6	-	-	Reserved, user software should not write ones to reserved bits.	0	0
7	TM		Test Mode.	0	x
		0 (disabled)	Normal operation.		
		1 (enabled)	The TD pin will reflect the bit, detected on RD pin, with the next positive edge of the system clock.		
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

[1] During a Hardware reset or when the Bus Status bit is set '1' (Bus-Off), the Reset Mode bit is set '1' (present). After the Reset Mode bit is set '0' the CAN Controller will wait for:

- one occurrence of Bus-Free signal (11 recessive bits), if the preceding reset has been caused by a Hardware reset or a CPU-initiated reset.

- 128 occurrences of Bus-Free, if the preceding reset has been caused by a CAN Controller initiated Bus-Off, before re-entering the Bus-On mode.

[2] This mode of operation forces the CAN Controller to be error passive. Message Transmission is not possible. The Listen Only Mode can be used e.g. for software driven bit rate detection and "hot plugging".

© NXP B.V. 2010. All rights reserved.

[3] A write access to the bits MOD.1 and MOD.2 is possible only if the Reset Mode is entered previously.

[4] Transmit Priority Mode is explained in more detail in Section 16–5.3 “Transmit Buffers (TXB)”.

[5] The CAN Controller will enter Sleep Mode, if the Sleep Mode bit is set '1' (sleep), there is no bus activity, and none of the CAN interrupts is pending. Setting of SM with at least one of the previously mentioned exceptions valid will result in a wake-up interrupt. The CAN Controller will wake up if SM is set LOW (wake-up) or there is bus activity. On wake-up, a Wake-up Interrupt is generated. A sleeping CAN Controller which wakes up due to bus activity will not be able to receive this message until it detects 11 consecutive recessive bits (Bus-Free sequence). Note that setting of SM is not possible in Reset Mode. After clearing of Reset Mode, setting of SM is possible only when Bus-Free is detected again.

[6] The LOM and STM bits can only be written if the RM bit is 1 prior to the write operation.

7.2 CAN Command Register (CAN1CMR - 0x4004 x004, CAN2CMR - 0x4004 8004)

Writing to this write-only register initiates an action within the transfer layer of the CAN Controller. Reading this register yields zeroes.

At least one internal clock cycle is needed for processing between two commands.

Table 319. CAN Command Register (CAN1CMR - address 0x4004 4004, CAN2CMR - address 0x4004 8004) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0[1][2]	TR		Transmission Request.	0	0
		0 (absent)	No transmission request.		
		1 (present)	The message, previously written to the CANxTFI, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer. If at two or all three of STB1, STB2 and STB3 bits are selected when TR=1 is written, Transmit Buffer will be selected based on the chosen priority scheme (for details see Section 16–5.3 “Transmit Buffers (TXB)”)		
1[1][3]	AT		Abort Transmission.	0	0
		0 (no action)	Do not abort the transmission.		
		1 (present)	if not already in progress, a pending Transmission Request for the selected Transmit Buffer is cancelled.		
2[4]	RRB		Release Receive Buffer.	0	0
		0 (no action)	Do not release the receive buffer.		
		1 (released)	The information in the Receive Buffer (consisting of CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers) is released, and becomes eligible for replacement by the next received frame. If the next received frame is not available, writing this command clears the RBS bit in the Status Register(s).		
3[5]	CDO		Clear Data Overrun.	0	0
		0 (no action)	Do not clear the data overrun bit.		
		1 (clear)	The Data Overrun bit in Status Register(s) is cleared.		
4[1][6]	SRR		Self Reception Request.	0	0
		0 (absent)	No self reception request.		
		1 (present)	The message, previously written to the CANxTFS, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer and received simultaneously. This differs from the TR bit above in that the receiver is not disabled during the transmission, so that it receives the message if its Identifier is recognized by the Acceptance Filter.		

© NXP B.V. 2010. All rights reserved.

Table 319. CAN Command Register (CAN1CMR - address 0x4004 4004, CAN2CMR - address 0x4004 8004) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	STB1		Select Tx Buffer 1.	0	0
		0 (not selected)	Tx Buffer 1 is not selected for transmission.		
		1 (selected)	Tx Buffer 1 is selected for transmission.		
6	STB2		Select Tx Buffer 2.	0	0
		0 (not selected)	Tx Buffer 2 is not selected for transmission.		
		1 (selected)	Tx Buffer 2 is selected for transmission.		
7	STB3		Select Tx Buffer 3.	0	0
		0 (not selected)	Tx Buffer 3 is not selected for transmission.		
		1 (selected)	Tx Buffer 3 is selected for transmission.		
31:8	-		Reserved, user software should not write ones to reserved bits.	NA	

[1] - Setting the command bits TR and AT simultaneously results in transmitting a message once. No re-transmission will be performed in case of an error or arbitration lost (single shot transmission).
- Setting the command bits SRR and TR simultaneously results in sending the transmit message once using the self-reception feature. No re-transmission will be performed in case of an error or arbitration lost.
- Setting the command bits TR, AT and SRR simultaneously results in transmitting a message once as described for TR and AT. The moment the Transmit Status bit is set within the Status Register, the internal Transmission Request Bit is cleared automatically.
- Setting TR and SRR simultaneously will ignore the set SRR bit.

[2] If the Transmission Request or the Self-Reception Request bit was set '1' in a previous command, it cannot be cancelled by resetting the bits. The requested transmission may only be cancelled by setting the Abort Transmission bit.

[3] The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit a more urgent message before. A transmission already in progress is not stopped. In order to see if the original message has been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. This should be done after the Transmit Buffer Status bit has been set to '1' or a Transmit Interrupt has been generated.

[4] After reading the contents of the Receive Buffer, the CPU can release this memory space by setting the Release Receive Buffer bit '1'. This may result in another message becoming immediately available. If there is no other message available, the Receive Interrupt bit is reset. If the RRB command is given, it will take at least 2 internal clock cycles before a new interrupt is generated.

[5] This command bit is used to clear the Data Overrun condition signalled by the Data Overrun Status bit. As long as the Data Overrun Status bit is set no further Data Overrun Interrupt is generated.

[6] Upon Self Reception Request, a message is transmitted and simultaneously received if the Acceptance Filter is set to the corresponding identifier. A receive and a transmit interrupt will indicate correct self reception (see also Self Test Mode in Section 16–7.1 “CAN Mode register (CAN1MOD - 0x4004 4000, CAN2MOD - 0x4004 8000)”).

7.3 CAN Global Status Register (CAN1GSR - 0x4004 x008, CAN2GSR - 0x4004 8008)

The content of the Global Status Register reflects the status of the CAN Controller. This register is read-only, except that the Error Counters can be written when the RM bit in the CANMOD register is 1. Bits not listed read as 0 and should be written as 0.

© NXP B.V. 2010. All rights reserved.

Table 320. CAN Global Status Register (CAN1GSR - address 0x4004 4008, CAN2GSR - address 0x4004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS[1]		Receive Buffer Status.	0	0
		0 (empty)	No message is available.		
		1 (full)	At least one complete message is received by the Double Receive Buffer and available in the CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers. This bit is cleared by the Release Receive Buffer command in CANxCMR, if no subsequent received message is available.		
1	DOS[2]		Data Overrun Status.	0	0
		0 (absent)	No data overrun has occurred since the last Clear Data Overrun command was given/written to CANxCMR (or since Reset).		
		1 (overrun)	A message was lost because the preceding message to this CAN controller was not read and released quickly enough (there was not enough space for a new message in the Double Receive Buffer).		
2	TBS		Transmit Buffer Status.	1	1
		0 (locked)	At least one of the Transmit Buffers is not available for the CPU, i.e. at least one previously queued message for this CAN controller has not yet been sent, and therefore software should not write to the CANxTFI, CANxTID, CANxTDA, nor CANxTDB registers of that (those) Tx buffer(s).		
		1 (released)	All three Transmit Buffers are available for the CPU. No transmit message is pending for this CAN controller (in any of the 3 Tx buffers), and software may write to any of the CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS[3]		Transmit Complete Status.	1	x
		0 (incomplete)	At least one requested transmission has not been successfully completed yet.		
		1 (complete)	All requested transmission(s) has (have) been successfully completed.		
4	RS[4]		Receive Status.	1	0
		0 (idle)	The CAN controller is idle.		
		1 (receive)	The CAN controller is receiving a message.		
5	TS[4]		Transmit Status.	1	0
		0 (idle)	The CAN controller is idle.		
		1 (transmit)	The CAN controller is sending a message.		
6	ES[5]		Error Status.	0	0
		0 (ok)	Both error counters are below the Error Warning Limit.		
		1 (error)	One or both of the Transmit and Receive Error Counters has reached the limit set in the Error Warning Limit register.		
7	BS[6]		Bus Status.	0	0
		0 (Bus-On)	The CAN Controller is involved in bus activities		
		1 (Bus-Off)	The CAN controller is currently not involved/prohibited from bus activity because the Transmit Error Counter reached its limiting value of 255.		
15:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
23:16	RXERR	-	The current value of the Rx Error Counter (an 8-bit value).	0	X
31:24	TXERR	-	The current value of the Tx Error Counter (an 8-bit value).	0	X

[1] After reading all messages and releasing their memory space with the command 'Release Receive Buffer,' this bit is cleared.

© NXP B.V. 2010. All rights reserved.

[2] If there is not enough space to store the message within the Receive Buffer, that message is dropped and the Data Overrun condition is signalled to the CPU in the moment this message becomes valid. If this message is not completed successfully (e.g. because of an error), no overrun condition is signalled.

[3] The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' at least for one of the three Transmit Buffers. The Transmission Complete Status bit will remain '0' until all messages are transmitted successfully.

[4] If both the Receive Status and the Transmit Status bits are '0' (idle), the CAN-Bus is idle. If both bits are set, the controller is waiting to become idle again. After hardware reset 11 consecutive recessive bits have to be detected until idle status is reached. After Bus-off this will take 128 times of 11 consecutive recessive bits.

[5] Errors detected during reception or transmission will effect the error counters according to the CAN specification. The Error Status bit is set when at least one of the error counters has reached or exceeded the Error Warning Limit. An Error Warning Interrupt is generated, if enabled. The default value of the Error Warning Limit after hardware reset is 96 decimal, see also Section 16–7.7 “CAN Error Warning Limit register (CAN1EWL - 0x4004 4018, CAN2EWL - 0x4004 8018)”.

[6] Mode bit '1' (present) and an Error Warning Interrupt is generated, if enabled. Afterwards the Transmit Error Counter is set to '127', and the Receive Error Counter is cleared. It will stay in this mode until the CPU clears the Reset Mode bit. Once this is completed the CAN Controller will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) counting down the Transmit Error Counter. After that, the Bus Status bit is cleared (Bus-On), the Error Status bit is set '0' (ok), the Error Counters are reset, and an Error Warning Interrupt is generated, if enabled. Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery.

RX error counter

The RX Error Counter Register, which is part of the Status Register, reflects the current value of the Receive Error Counter. After hardware reset this register is initialized to 0. In Operating Mode this register appears to the CPU as a read-only memory. A write access to this register is possible only in Reset Mode. If a Bus Off event occurs, the RX Error Counter is initialized to 0. As long as Bus Off is valid, writing to this register has no effect.The Rx Error Counter is determined as follows:

RX Error Counter = (CANxGSR AND 0x00FF0000) / 0x00010000

Note that a CPU-forced content change of the RX Error Counter is possible only if the Reset Mode was entered previously. An Error Status change (Status Register), an Error Warning or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

TX error counter

The TX Error Counter Register, which is part of the Status Register, reflects the current value of the Transmit Error Counter. In Operating Mode this register appears to the CPU as a read-only memory. After hardware reset this register is initialized to 0. A write access to this register is possible only in Reset Mode. If a bus-off event occurs, the TX Error Counter is initialized to 127 to count the minimum protocol-defined time (128 occurrences of the Bus-Free signal). Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery. If Bus Off is active, a write access to TXERR in the range of 0 to 254 clears the Bus Off Flag and the controller will wait for one occurrence of 11 consecutive recessive bits (bus free) after clearing of Reset Mode. The Tx error counter is determined as follows:

TX Error Counter = (CANxGSR AND 0xFF000000) / 0x01000000

Writing 255 to TXERR allows initiation of a CPU-driven Bus Off event. Note that a CPU-forced content change of the TX Error Counter is possible only if the Reset Mode was entered previously. An Error or Bus Status change (Status Register), an Error

© NXP B.V. 2010. All rights reserved.

Warning, or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again. After leaving the Reset Mode, the new TX Counter content is interpreted and the Bus Off event is performed in the same way as if it was forced by a bus error event. That means, that the Reset Mode is entered again, the TX Error Counter is initialized to 127, the RX Counter is cleared, and all concerned Status and Interrupt Register bits are set. Clearing of Reset Mode now will perform the protocol defined Bus Off recovery sequence (waiting for 128 occurrences of the Bus-Free signal). If the Reset Mode is entered again before the end of Bus Off recovery (TXERR>0), Bus Off keeps active and TXERR is frozen.

7.4 CAN Interrupt and Capture Register (CAN1ICR - 0x4004 400C, CAN2ICR - 0x4004 800C)

Bits in this register indicate information about events on the CAN bus. This register is read-only.

The Interrupt flags of the Interrupt and Capture Register allow the identification of an interrupt source. When one or more bits are set, a CAN interrupt will be indicated to the CPU. After this register is read from the CPU all interrupt bits are reset **except** of the Receive Interrupt bit. The Interrupt Register appears to the CPU as a read-only memory.

Bits 1 through 10 clear when they are read.

Bits 16-23 are captured when a bus error occurs. At the same time, if the BEIE bit in CANIER is 1, the BEI bit in this register is set, and a CAN interrupt can occur.

Bits 24-31 are captured when CAN arbitration is lost. At the same time, if the ALIE bit in CANIER is 1, the ALI bit in this register is set, and a CAN interrupt can occur. Once either of these bytes is captured, its value will remain the same until it is read, at which time it is released to capture a new value.

The clearing of bits 1 to 10 and the releasing of bits 16-23 and 24-31 all occur on any read from CANxICR, regardless of whether part or all of the register is read. This means that software should always read CANxICR as a word, and process and deal with all bits of the register as appropriate for the application.

Table 321. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RI[1]	0 (reset) 1 (set)	Receive Interrupt. This bit is set whenever the RBS bit in CANxSR and the RIE bit in CANxIER are both 1, indicating that a new message was received and stored in the Receive Buffer.	0	0
1	TI1	0 (reset) 1 (set)	Transmit Interrupt 1. This bit is set when the TBS1 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB1 was successfully transmitted or aborted), indicating that Transmit buffer 1 is available, and the TIE1 bit in CANxIER is 1.	0	0
2	EI	0 (reset) 1 (set)	Error Warning Interrupt. This bit is set on every change (set or clear) of either the Error Status or Bus Status bit in CANxSR and the EIE bit bit is set within the Interrupt Enable Register at the time of the change.	0	X
3	DOI	0 (reset) 1 (set)	Data Overrun Interrupt. This bit is set when the DOS bit in CANxSR goes from 0 to 1 and the DOIE bit in CANxIER is 1.	0	0
4	WUI[2]	0 (reset) 1 (set)	Wake-Up Interrupt. This bit is set if the CAN controller is sleeping and bus activity is detected and the WUIE bit in CANxIER is 1.	0	0

© NXP B.V. 2010. All rights reserved.

Table 321. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	EPI	0 (reset) 1 (set)	Error Passive Interrupt. This bit is set if the EPIE bit in CANxIER is 1, and the CAN controller switches between Error Passive and Error Active mode in either direction. This is the case when the CAN Controller has reached the Error Passive Status (at least one error counter exceeds the CAN protocol defined level of 127) or if the CAN Controller is in Error Passive Status and enters the Error Active Status again.	0	0
6	ALI	0 (reset) 1 (set)	Arbitration Lost Interrupt. This bit is set if the ALIE bit in CANxIER is 1, and the CAN controller loses arbitration while attempting to transmit. In this case the CAN node becomes a receiver.	0	0
7	BEI	0 (reset) 1 (set)	Bus Error Interrupt -- this bit is set if the BEIE bit in CANxIER is 1, and the CAN controller detects an error on the bus.	0	X
8	IDI	0 (reset) 1 (set)	ID Ready Interrupt -- this bit is set if the IDIE bit in CANxIER is 1, and a CAN Identifier has been received (a message was successfully transmitted or aborted). This bit is set whenever a message was successfully transmitted or aborted and the IDIE bit is set in the IER register.	0	0
9	TI2	0 (reset) 1 (set)	Transmit Interrupt 2. This bit is set when the TBS2 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB2 was successfully transmitted or aborted), indicating that Transmit buffer 2 is available, and the TIE2 bit in CANxIER is 1.	0	0
10	TI3	0 (reset) 1 (set)	Transmit Interrupt 3. This bit is set when the TBS3 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB3 was successfully transmitted or aborted), indicating that Transmit buffer 3 is available, and the TIE3 bit in CANxIER is 1.	0	0
15:11	-	-	Reserved, user software should not write ones to reserved bits.	0	0

© NXP B.V. 2010. All rights reserved.

Table 321. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
20:16	ERRBIT 4:0[3]		Error Code Capture: when the CAN controller detects a bus error, the location of the error within the frame is captured in this field. The value reflects an internal state variable, and as a result is not very linear:	0	X
		00011	Start of Frame		
		00010	ID28 ... ID21		
		00110	ID20 ... ID18		
		00100	SRTR Bit		
		00101	IDE bit		
		00111	ID17 ... 13		
		01111	ID12 ... ID5		
		01110	ID4 ... ID0		
		01100	RTR Bit		
		01101	Reserved Bit 1		
		01001	Reserved Bit 0		
		01011	Data Length Code		
		01010	Data Field		
		01000	CRC Sequence		
		11000	CRC Delimiter		
		11001	Acknowledge Slot		
		11011	Acknowledge Delimiter		
		11010	End of Frame		
		10010	Intermission		
		10001	Active Error Flag		
		10110	Passive Error Flag		
		10011	Tolerate Dominant Bits		
		10111	Error Delimiter		
		11100	Overload flag		
21	ERRDIR		When the CAN controller detects a bus error, the direction of the current bit is captured in this bit.	0	X
		0	Error occurred during transmitting.		
		1	Error occurred during receiving.		
23:22	ERRC1:0		When the CAN controller detects a bus error, the type of error is captured in this field:	0	X
		00	Bit error		
		01	Form error		
		10	Stuff error		
		11	Other error		

© NXP B.V. 2010. All rights reserved.

Table 321. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
31:24	ALCBIT[4]	-	Each time arbitration is lost while trying to send on the CAN, the bit number within the frame is captured into this field. After the content of ALCBIT is read, the ALI bit is cleared and a new Arbitration Lost interrupt can occur.	0	X
		00	arbitration lost in the first bit (MS) of identifier		
		...			
		11	arbitration lost in SRTS bit (RTR bit for standard frame messages)		
		12	arbitration lost in IDE bit		
		13	arbitration lost in 12th bit of identifier (extended frame only)		
		...			
		30	arbitration lost in last bit of identifier (extended frame only)		
		31	arbitration lost in RTR bit (extended frame only)		

[1] The Receive Interrupt Bit is not cleared upon a read access to the Interrupt Register. Giving the Command “Release Receive Buffer” will clear RI temporarily. If there is another message available within the Receive Buffer after the release command, RI is set again. Otherwise RI remains cleared.

[2] A Wake-Up Interrupt is also generated if the CPU tries to set the Sleep bit while the CAN controller is involved in bus activities or a CAN Interrupt is pending. The WUI flag can also get asserted when the according enable bit WUIE is not set. In this case a Wake-Up Interrupt does not get asserted.

[3] Whenever a bus error occurs, the corresponding bus error interrupt is forced, if enabled. At the same time, the current position of the Bit Stream Processor is captured into the Error Code Capture Register. The content within this register is fixed until the user software has read out its content once. From now on, the capture mechanism is activated again, i.e. reading the CANxICR enables another Bus Error Interrupt.

[4] On arbitration lost, the corresponding arbitration lost interrupt is forced, if enabled. At that time, the current bit position of the Bit Stream Processor is captured into the Arbitration Lost Capture Register. The content within this register is fixed until the user application has read out its contents once. From now on, the capture mechanism is activated again.

7.5 CAN Interrupt Enable Register (CAN1IER - 0x4004 4010, CAN2IER - 0x4004 8010)

This read/write register controls whether various events on the CAN controller will result in an interrupt or not. Bits 10:0 in this register correspond 1-to-1 with bits 10:0 in the CANxICR register. If a bit in the CANxIER register is 0 the corresponding interrupt is disabled; if a bit in the CANxIER register is 1 the corresponding source is enabled to trigger an interrupt.

Table 322. CAN Interrupt Enable Register (CAN1IER - address 0x4004 4010, CAN2IER - address 0x4004 8010) bit description

Bit	Symbol	Function	Reset Value	RM Set
0	RIE	Receiver Interrupt Enable. When the Receive Buffer Status is 'full', the CAN Controller requests the respective interrupt.	0	X
1	TIE1	Transmit Interrupt Enable for Buffer1. When a message has been successfully transmitted out of TXB1 or Transmit Buffer 1 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
2	EIE	Error Warning Interrupt Enable. If the Error or Bus Status change (see Status Register), the CAN Controller requests the respective interrupt.	0	X

© NXP B.V. 2010. All rights reserved.

Table 322. CAN Interrupt Enable Register (CAN1IER - address 0x4004 4010, CAN2IER - address 0x4004 8010) bit description

Bit	Symbol	Function	Reset Value	RM Set
3	DOIE	Data Overrun Interrupt Enable. If the Data Overrun Status bit is set (see Status Register), the CAN Controller requests the respective interrupt.	0	X
4	WUIE	Wake-Up Interrupt Enable. If the sleeping CAN controller wakes up, the respective interrupt is requested.	0	X
5	EPIE	Error Passive Interrupt Enable. If the error status of the CAN Controller changes from error active to error passive or vice versa, the respective interrupt is requested.	0	X
6	ALIE	Arbitration Lost Interrupt Enable. If the CAN Controller has lost arbitration, the respective interrupt is requested.	0	X
7	BEIE	Bus Error Interrupt Enable. If a bus error has been detected, the CAN Controller requests the respective interrupt.	0	X
8	IDIE	ID Ready Interrupt Enable. When a CAN identifier has been received, the CAN Controller requests the respective interrupt.	0	X
9	TIE2	Transmit Interrupt Enable for Buffer2. When a message has been successfully transmitted out of TXB2 or Transmit Buffer 2 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
10	TIE3	Transmit Interrupt Enable for Buffer3. When a message has been successfully transmitted out of TXB3 or Transmit Buffer 3 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

7.6 CAN Bus Timing Register (CAN1BTR - 0x4004 4014, CAN2BTR - 0x4004 8014)

This register controls how various CAN timings are derived from the APB clock. It defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). Furthermore, it defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. It can be read at any time but can only be written if the RM bit in CANmod is 1.

Table 323. CAN Bus Timing Register (CAN1BTR - address 0x4004 4014, CAN2BTR - address 0x4004 8014) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
9:0	BRP		Baud Rate Prescaler. The APB clock is divided by (this value plus one) to produce the CAN clock.	0	X
13:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
15:14	SJW		The Synchronization Jump Width is (this value plus one) CAN clocks.	0	X
19:16	TESG1		The delay from the nominal Sync point to the sample point is (this value plus one) CAN clocks.	1100	X
22:20	TESG2		The delay from the sample point to the next nominal sync point is (this value plus one) CAN clocks. The nominal CAN bit time is (this value plus the value in TSEG1 plus 3) CAN clocks.	001	X

© NXP B.V. 2010. All rights reserved.

Table 323. CAN Bus Timing Register (CAN1BTR - address 0x4004 4014, CAN2BTR - address 0x4004 8014) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
23	SAM		Sampling		
		0	The bus is sampled once (recommended for high speed buses)	0	X
		1	The bus is sampled 3 times (recommended for low to medium speed buses to filter spikes on the bus-line)		
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Baud rate prescaler

The period of the CAN system clock t_{SCL} is programmable and determines the individual bit timing. The CAN system clock t_{SCL} is calculated using the following equation:

(5)

$$t_{SCL} = t_{CANsuppliedCLK} \times (BRP + 1)$$

Synchronization jump width

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width t_{SJW} defines the maximum number of clock cycles a certain bit period may be shortened or lengthened by one re-synchronization:

(6)

$$t_{SJW} = t_{SCL} \times (SJW + 1)$$

Time segment 1 and time segment 2

Time segments TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

(7)

$$t_{SYNCSEG} = t_{SCL}$$

(8)

$$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$$

(9)

$$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$$

7.7 CAN Error Warning Limit register (CAN1EWL - 0x4004 4018, CAN2EWL - 0x4004 8018)

This register sets a limit on Tx or Rx errors at which an interrupt can occur. It can be read at any time but can only be written if the RM bit in CANmod is 1.

© NXP B.V. 2010. All rights reserved.

Table 324. CAN Error Warning Limit register (CAN1EWL - address 0x4004 4018, CAN2EWL - address 0x4004 8018) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	EWL	During CAN operation, this value is compared to both the Tx and Rx Error Counters. If either of these counter matches this value, the Error Status (ES) bit in CANSR is set.	96_{10} = 0x60	X
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Note that a content change of the Error Warning Limit Register is possible only if the Reset Mode was entered previously. An Error Status change (Status Register) and an Error Warning Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

7.8 CAN Status Register (CAN1SR - 0x4004 401C, CAN2SR - 0x4004 801C)

This read-only register contains three status bytes in which the bits not related to transmission are identical to the corresponding bits in the Global Status Register, while those relating to transmission reflect the status of each of the 3 Tx Buffers.

Table 325. CAN Status Register (CAN1SR - address 0x4004 401C, CAN2SR - address 0x4004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
1	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
2	TBS1[1]		Transmit Buffer Status 1.	1	1
		0(locked)	Software cannot access the Tx Buffer 1 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 1 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS1[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 1 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 1 has been successfully completed.		
4	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
5	TS1		Transmit Status 1.	1	0
		0(idle)	There is no transmission from Tx Buffer 1.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 1.		
6	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
7	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
8	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
9	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0

© NXP B.V. 2010. All rights reserved.

Table 325. CAN Status Register (CAN1SR - address 0x4004 401C, CAN2SR - address 0x4004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
10	TBS2[1]		Transmit Buffer Status 2.	1	1
		0(locked)	Software cannot access the Tx Buffer 2 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 2 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
11	TCS2[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 2 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 2 has been successfully completed.		
12	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
13	TS2		Transmit Status 2.	1	0
		0(idle)	There is no transmission from Tx Buffer 2.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 2.		
14	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
15	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
16	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
17	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
18	TBS3[1]		Transmit Buffer Status 3.	1	1
		0(locked)	Software cannot access the Tx Buffer 3 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 3 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
19	TCS3[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 3 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 3 has been successfully completed.		
20	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
21	TS3		Transmit Status 3.	1	0
		0(idle)	There is no transmission from Tx Buffer 3.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 3.		
22	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
23	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
31:24	-		Reserved, the value read from a reserved bit is not defined.	NA	

[1] If the CPU tries to write to this Transmit Buffer when the Transmit Buffer Status bit is '0' (locked), the written byte is not accepted and is lost without this being signalled.

[2] The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' for this TX buffer. The Transmission Complete Status bit remains '0' until a message is transmitted successfully.

© NXP B.V. 2010. All rights reserved.

7.9 CAN Receive Frame Status register (CAN1RFS - 0x4004 4020, CAN2RFS - 0x4004 8020)

This register defines the characteristics of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANxMOD is 1.

Table 326. CAN Receive Frame Status register (CAN1RFS - address 0x4004 4020, CAN2RFS - address 0x4004 8020) bit description

Bit	Symbol	Function	Reset Value	RM Set
9:0	ID Index	If the BP bit (below) is 0, this value is the zero-based number of the Lookup Table RAM entry at which the Acceptance Filter matched the received Identifier. Disabled entries in the Standard tables are included in this numbering, but will not be matched. See Section 16–17 "Examples of acceptance filter tables and ID index values" on page 391 for examples of ID Index values.	0	X
10	BP	If this bit is 1, the current message was received in AF Bypass mode, and the ID Index field (above) is meaningless.	0	X
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
19:16	DLC	The field contains the Data Length Code (DLC) field of the current received message. When RTR = 0, this is related to the number of data bytes available in the CANRDA and CANRDB registers as follows: 0000-0111 = 0 to 7 bytes1000-1111 = 8 bytes With RTR = 1, this value indicates the number of data bytes requested to be sent back, with the same encoding.	0	X
29:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
30	RTR	This bit contains the Remote Transmission Request bit of the current received message. 0 indicates a Data Frame, in which (if DLC is non-zero) data can be read from the CANRDA and possibly the CANRDB registers. 1 indicates a Remote frame, in which case the DLC value identifies the number of data bytes requested to be sent using the same Identifier.	0	X
31	FF	A 0 in this bit indicates that the current received message included an 11-bit Identifier, while a 1 indicates a 29-bit Identifier. This affects the contents of the CANid register described below.	0	X

7.9.1 ID index field

The ID Index is a 10-bit field in the Info Register that contains the table position of the ID Look-up Table if the currently received message was accepted. The software can use this index to simplify message transfers from the Receive Buffer into the Shared Message Memory. Whenever bit 10 (BP) of the ID Index in the CANRFS register is 1, the current CAN message was received in acceptance filter bypass mode.

7.10 CAN Receive Identifier register (CAN1RID - 0x4004 4024, CAN2RID - 0x4004 8024)

This register contains the Identifier field of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANmod is 1. It has two different formats depending on the FF bit in CANRFS. See Table 16–314 for details on specific CAN channel register address.

© NXP B.V. 2010. All rights reserved.

Table 327. CAN Receive Identifier register (CAN1RID - address 0x4004 4024, CAN2RID - address 0x4004 8024) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11-bit Identifier field of the current received message. In CAN 2.0A, these bits are called ID10-0, while in CAN 2.0B they're called ID29-18.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 328. RX Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29-bit Identifier field of the current received message. In CAN 2.0B these bits are called ID29-0.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

7.11 CAN Receive Data register A (CAN1RDA - 0x4004 4028, CAN2RDA - 0x4004 8028)

This register contains the first 1-4 Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 16–314 for details on specific CAN channel register address.

Table 329. CAN Receive Data register A (CAN1RDA - address 0x4004 4028, CAN2RDA - address 0x4004 8028) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If the DLC field in CANRFS ≥ 0001, this contains the first Data byte of the current received message.	0	X
15:8	Data 2	If the DLC field in CANRFS ≥ 0010, this contains the first Data byte of the current received message.	0	X
23:16	Data 3	If the DLC field in CANRFS ≥ 0011, this contains the first Data byte of the current received message.	0	X
31:24	Data 4	If the DLC field in CANRFS ≥ 0100, this contains the first Data byte of the current received message.	0	X

7.12 CAN Receive Data register B (CAN1RDB - 0x4004 402C, CAN2RDB - 0x4004 802C)

This register contains the 5th through 8th Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 16–314 for details on specific CAN channel register address.

Table 330. CAN Receive Data register B (CAN1RDB - address 0x4004 402C, CAN2RDB - address 0x4004 802C) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If the DLC field in CANRFS ≥ 0101, this contains the first Data byte of the current received message.	0	X

© NXP B.V. 2010. All rights reserved.

Table 330. CAN Receive Data register B (CAN1RDB - address 0x4004 402C, CAN2RDB - address 0x4004 802C) bit description

Bit	Symbol	Function	Reset Value	RM Set
15:8	Data 6	If the DLC field in CANRFS ≥ 0110, this contains the first Data byte of the current received message.	0	X
23:16	Data 7	If the DLC field in CANRFS ≥ 0111, this contains the first Data byte of the current received message.	0	X
31:24	Data 8	If the DLC field in CANRFS ≥ 1000, this contains the first Data byte of the current received message.	0	X

7.13 CAN Transmit Frame Information register (CAN1TFI[1/2/3] - 0x4004 40[30/ 40/50], CAN2TFI[1/2/3] - 0x4004 80[30/40/50])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the format of the next transmit message for that Tx buffer. Bits not listed read as 0 and should be written as 0.

The values for the reserved bits of the CANxTFI register in the Transmit Buffer should be set to the values expected in the Receive Buffer for an easy comparison, when using the Self Reception facility (self test), otherwise they are not defined.

The CAN Controller consist of three Transmit Buffers. Each of them has a length of 4 words and is able to store one complete CAN message as shown in Figure 16–53.

The buffer layout is subdivided into Descriptor and Data Field where the first word of the Descriptor Field includes the TX Frame Info that describes the Frame Format, the Data Length and whether it is a Remote or Data Frame. In addition, a TX Priority register allows the definition of a certain priority for each transmit message. Depending on the chosen Frame Format, an 11-bit identifier for Standard Frame Format (SFF) or an 29-bit identifier for Extended Frame Format (EFF) follows. Note that unused bits in the TID field have to be defined as 0. The Data Field in TDA and TDB contains up to eight data bytes.

Table 331. CAN Transmit Frame Information register (CAN1TFI[1/2/3] - address 0x4004 40[30/40/50], CAN2TFI[1/2/3] - 0x4004 80[30/40/50]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	PRIO	If the TPM (Transmit Priority Mode) bit in the CANxMOD register is set to 1, enabled Tx Buffers contend for the right to send their messages based on this field. The buffer with the lowest TX Priority value wins the prioritization and is sent first.		x
15:8	-	Reserved.	0	
19:16	DLC	Data Length Code. This value is sent in the DLC field of the next transmit message. In addition, if RTR = 0, this value controls the number of Data bytes sent in the next transmit message, from the CANxTDA and CANxTDB registers: 0000-0111 = 0-7 bytes 1xxx = 8 bytes	0	X
29:20	-	Reserved.	0	
30	RTR	This value is sent in the RTR bit of the next transmit message. If this bit is 0, the number of data bytes called out by the DLC field are sent from the CANxTDA and CANxTDB registers. If this bit is 1, a Remote Frame is sent, containing a request for that number of bytes.	0	X
31	FF	If this bit is 0, the next transmit message will be sent with an 11-bit Identifier (standard frame format), while if it's 1, the message will be sent with a 29-bit Identifier (extended frame format).	0	X

© NXP B.V. 2010. All rights reserved.

Automatic transmit priority detection

To allow uninterrupted streams of transmit messages, the CAN Controller provides Automatic Transmit Priority Detection for all Transmit Buffers. Depending on the selected Transmit Priority Mode, internal prioritization is based on the CAN Identifier or a user defined "local priority". If more than one message is enabled for transmission (TR=1) the internal transmit message queue is organized such as that the transmit buffer with the lowest CAN Identifier (TID) or the lowest "local priority" (TX Priority) wins the prioritization and is sent first. The result of the internal scheduling process is taken into account short before a new CAN message is sent on the bus. This is also true after the occurrence of a transmission error and right before a re-transmission.

Tx DLC

The number of bytes in the Data Field of a message is coded with the Data Length Code (DLC). At the start of a Remote Frame transmission the DLC is not considered due to the RTR bit being '1 ' (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the DLC must be specified correctly to avoid bus errors, if two CAN Controllers start a Remote Frame transmission with the same identifier simultaneously. For reasons of compatibility no DLC > 8 should be used. If a value greater than 8 is selected, 8 bytes are transmitted in the data frame with the Data Length Code specified in DLC. The range of the Data Byte Count is 0 to 8 bytes and is coded as follows:

$$DataByteCount = DLC \quad (10)$$

7.14 CAN Transmit Identifier register (CAN1TID[1/2/3] - 0x4004 40[34/44/54], CAN2TID[1/2/3] - 0x4004 80[34/44/54])

When the corresponding TBS bit in CANxSR is 1, software can write to one of these registers to define the Identifier field of the next transmit message. Bits not listed read as 0 and should be written as 0. The register assumes two different formats depending on the FF bit in CANTFI.

In Standard Frame Format messages, the CAN Identifier consists of 11 bits (ID.28 to ID.18), and in Extended Frame Format messages, the CAN identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit, and it is transmitted first on the bus during the arbitration process. The Identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority during the arbitration process.

Table 332. CAN Transfer Identifier register (CAN1TID[1/2/3] - address 0x4004 40[34/44/54], CAN2TID[1/2/3] - address 0x4004 80[34/44/54]) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11-bit Identifier to be sent in the next transmit message.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

© NXP B.V. 2010. All rights reserved.

Table 333. Transfer Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29-bit Identifier to be sent in the next transmit message.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

7.15 CAN Transmit Data register A (CAN1TDA[1/2/3] - 0x4004 40[38/48/58], CAN2TDA[1/2/3] - 0x4004 80[38/48/58])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the first 1 - 4 data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 334. CAN Transmit Data register A (CAN1TDA[1/2/3] - address 0x4004 40[38/48/58], CAN2TDA[1/2/3] - address 0x4004 80[38/48/58]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If RTR = 0 and DLC ≥ 0001 in the corresponding CANxTFI, this byte is sent as the first Data byte of the next transmit message.	0	X
15;8	Data 2	If RTR = 0 and DLC ≥ 0010 in the corresponding CANxTFI, this byte is sent as the 2nd Data byte of the next transmit message.	0	X
23:16	Data 3	If RTR = 0 and DLC ≥ 0011 in the corresponding CANxTFI, this byte is sent as the 3rd Data byte of the next transmit message.	0	X
31:24	Data 4	If RTR = 0 and DLC ≥ 0100 in the corresponding CANxTFI, this byte is sent as the 4th Data byte of the next transmit message.	0	X

7.16 CAN Transmit Data register B (CAN1TDB[1/2/3] - 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3] - 0x4004 80[3C/4C/5C])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the 5th through 8th data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 335. CAN Transmit Data register B (CAN1TDB[1/2/3] - address 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3] - address 0x4004 80[3C/4C/5C]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If RTR = 0 and DLC ≥ 0101 in the corresponding CANTFI, this byte is sent as the 5th Data byte of the next transmit message.	0	X
15;8	Data 6	If RTR = 0 and DLC ≥ 0110 in the corresponding CANTFI, this byte is sent as the 6th Data byte of the next transmit message.	0	X
23:16	Data 7	If RTR = 0 and DLC ≥ 0111 in the corresponding CANTFI, this byte is sent as the 7th Data byte of the next transmit message.	0	X
31:24	Data 8	If RTR = 0 and DLC ≥ 1000 in the corresponding CANTFI, this byte is sent as the 8th Data byte of the next transmit message.	0	X

7.17 CAN Sleep Clear register (CANSLEEPCLR - 0x400F C110)

This register provides the current sleep state of the two CAN channels and provides a means to restore the clocks to that channel following wake-up. Refer to Section 16–8.2 "Sleep mode" for more information on the CAN sleep feature.

© NXP B.V. 2010. All rights reserved.

Table 336. CAN Sleep Clear register (CANSLEEPCLR - address 0x400F C110) bit description

Bit	Symbol	Function	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	CAN1SLEEP	Sleep status and control for CAN channel 1. Read: when 1, indicates that CAN channel 1 is in the sleep mode. Write: writing a 1 causes clocks to be restored to CAN channel 1.	0
2	CAN2SLEEP	Sleep status and control for CAN channel 2. Read: when 1, indicates that CAN channel 2 is in the sleep mode. Write: writing a 1 causes clocks to be restored to CAN channel 2.	0
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.18 CAN Wake-up Flags register (CANWAKEFLAGS - 0x400F C114)

This register provides the wake-up status for the two CAN channels and allows clearing wake-up events. Refer to Section 16–8.2 “Sleep mode” for more information on the CAN sleep feature.

Table 337. CAN Wake-up Flags register (CANWAKEFLAGS - address 0x400F C114) bit description

Bit	Symbol	Function	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	CAN1WAKE	Wake-up status for CAN channel 1. Read: when 1, indicates that a falling edge has occurred on the receive data line of CAN channel 1. Write: writing a 1 clears this bit.	0
2	CAN2WAKE	Wake-up status for CAN channel 2. Read: when 1, indicates that a falling edge has occurred on the receive data line of CAN channel 2. Write: writing a 1 clears this bit.	0
31:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8. CAN controller operation

8.1 Error handling

The CAN Controllers count and handle transmit and receive errors as specified in CAN Spec 2.0B. The Transmit and Receive Error Counters are incriminated for each detected error and are decremented when operation is error-free. If the Transmit Error counter contains 255 and another error occurs, the CAN Controller is forced into a state called Bus-Off. In this state, the following register bits are set: BS in CANxSR, BEI and EI in CANxIR if these are enabled, and RM in CANxMOD. RM resets and disables much of the CAN Controller. Also at this time the Transmit Error Counter is set to 127 and the Receive Error Counter is cleared. Software must next clear the RM bit. Thereafter the Transmit Error Counter will count down 128 occurrences of the Bus Free condition (11 consecutive

© NXP B.V. 2010. All rights reserved.

recessive bits). Software can monitor this countdown by reading the Tx Error Counter. When this countdown is complete, the CAN Controller clears BS and ES in CANxSR, and sets EI in CANxSR if EIE in IER is 1.

The Tx and Rx error counters can be written if RM in CANxMOD is 1. Writing 255 to the Tx Error Counter forces the CAN Controller to Bus-Off state. If Bus-Off (BS in CANxSR) is 1, writing any value 0 through 254 to the Tx Error Counter clears Bus-Off. When software clears RM in CANxMOD thereafter, only one Bus Free condition (11 consecutive recessive bits) is needed before operation resumes.

8.2 Sleep mode

The CAN Controller will enter sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. Software can only set SM when RM in the CAN Mode register is 0; it can also set the WUIE bit in the CAN Interrupt Enable register to enable an interrupt on any wake-up condition.

The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1) in response to a) a dominant bit on the CAN bus, or b) software clearing SM in the CAN Mode register. A sleeping CAN Controller that wakes up in response to bus activity is not able to receive an initial message until after it detects Bus_Free (11 consecutive recessive bits). If an interrupt is pending or the CAN bus is active when software sets SM, the wake-up is immediate.

Upon wake-up, software needs to do the following things:

1. Write a 1 to the relevant bit(s) in the CANSLEEPCLR register.
2. Write a 0 to the SM bit in the CAN1MOD and/or CAN2MOD register.
3. Write a 1 to the relevant bit(s) in the CANWAKEFLAGS register. Failure to perform this step will prevent subsequent entry into Power-down mode.

If the LPC17xx is in Deep Sleep or Power-down mode, CAN activity will wake up the device if the CAN activity interrupt is enabled. See Section 4–8 “Power control”.

8.3 Interrupts

Each CAN Controller produces 3 interrupt requests, Receive, Transmit, and “other status”. The Transmit interrupt is the OR of the Transmit interrupts from the three Tx Buffers. Each Receive and Transmit interrupt request from each controller is assigned its own channel in the NVIC, and can have its own interrupt service routine. The “other status” interrupts from all of the CAN controllers, and the Acceptance Filter LUTerr condition, are ORed into one NVIC channel.

8.4 Transmit priority

If the TPM bit in the CANxMOD register is 0, multiple enabled Tx Buffers contend for the right to send their messages based on the value of their CAN Identifier (TID). If TPM is 1, they contend based on the PRIO fields in bits 7:0 of their CANxTFS registers. In both cases the smallest binary value has priority. If two (or three) transmit-enabled buffers have the same smallest value, the lowest-numbered buffer sends first.

The CAN controller selects among multiple enabled Tx Buffers dynamically, just before it sends each message.

© NXP B.V. 2010. All rights reserved.

9. Centralized CAN registers

For easy and fast access, all CAN Controller Status bits from each CAN Controller Status register are bundled together. Each defined byte of the following registers contains one particular status bit from each of the CAN controllers, in its LS bits.

All Status registers are read-only and allow byte, half word and word access.

9.1 Central Transmit Status Register (CANTxSR - 0x4004 0000)

Table 338. Central Transit Status Register (CANTxSR - address 0x4004 0000) bit description

Bit	Symbol	Description	Reset Value
0	TS1	When 1, the CAN controller 1 is sending a message (same as TS in the CAN1GSR).	0
1	TS2	When 1, the CAN controller 2 is sending a message (same as TS in the CAN2GSR)	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	TBS1	When 1, all 3 Tx Buffers of the CAN1 controller are available to the CPU (same as TBS in CAN1GSR).	1
9	TBS2	When 1, all 3 Tx Buffers of the CAN2 controller are available to the CPU (same as TBS in CAN2GSR).	1
15:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	TCS1	When 1, all requested transmissions have been completed successfully by the CAN1 controller (same as TCS in CAN1GSR).	1
17:16	TCS2	When 1, all requested transmissions have been completed successfully by the CAN2 controller (same as TCS in CAN2GSR).	1
31:18	-	Reserved, the value read from a reserved bit is not defined.	NA

9.2 Central Receive Status Register (CANRxSR - 0x4004 0004)

Table 339. Central Receive Status Register (CANRxSR - address 0x4004 0004) bit description

Bit	Symbol	Description	Reset Value
0	RS1	When 1, CAN1 is receiving a message (same as RS in CAN1GSR).	0
1	RS2	When 1, CAN2 is receiving a message (same as RS in CAN2GSR).	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	RB1	When 1, a received message is available in the CAN1 controller (same as RBS in CAN1GSR).	0
9	RB2	When 1, a received message is available in the CAN2 controller (same as RBS in CAN2GSR).	0
15:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	DOS1	When 1, a message was lost because the preceding message to CAN1 controller was not read out quickly enough (same as DOS in CAN1GSR).	0
17:16	DOS2	When 1, a message was lost because the preceding message to CAN2 controller was not read out quickly enough (same as DOS in CAN2GSR).	0
31:18	-	Reserved, the value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

9.3 Central Miscellaneous Status Register (CANMSR - 0x4004 0008)

Table 340. Central Miscellaneous Status Register (CANMSR - address 0x4004 0008) bit description

Bit	Symbol	Description	Reset Value
0	E1	When 1, one or both of the CAN1 Tx and Rx Error Counters has reached the limit set in the CAN1EWL register (same as ES in CAN1GSR)	0
1	E2	When 1, one or both of the CAN2 Tx and Rx Error Counters has reached the limit set in the CAN2EWL register (same as ES in CAN2GSR)	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	BS1	When 1, the CAN1 controller is currently involved in bus activities (same as BS in CAN1GSR).	0
9	BS2	When 1, the CAN2 controller is currently involved in bus activities (same as BS in CAN2GSR).	0
31:10	-	Reserved, the value read from a reserved bit is not defined.	NA

10. Global acceptance filter

This block provides lookup for received Identifiers (called Acceptance Filtering in CAN terminology) for all the CAN Controllers. It includes a 512 × 32 (2 kB) RAM in which software maintains one to five tables of Identifiers. This RAM can contain up to 1024 Standard Identifiers or 512 Extended Identifiers, or a mixture of both types.

11. Acceptance filter modes

The Acceptance Filter can be put into different modes by setting the according AccOff, AccBP, and eFCAN bits in the Acceptance Filter Mode Register (Section 16–14.1 “Acceptance Filter Mode Register (AFMR - 0x4003 C000)”). During each mode the access to the Configuration Register and the ID Look-up table is handled differently.

Table 341. Acceptance filter modes and access control

Acceptance filter mode	Bit AccOff	Bit AccBP	Acceptance filter state	ID Look-up table RAM[1]	Acceptance filter config. registers	CAN controller message receive interrupt
Off Mode	1	0	reset & halted	r/w access from CPU	r/w access from CPU	no messages accepted
Bypass Mode	X	1	reset & halted	r/w access from CPU	r/w access from CPU	all messages accepted
Operating Mode and FullCAN Mode	0	0	running	read-only from CPU[2]	access from Acceptance filter only	hardware acceptance filtering

[1] The whole ID Look-up Table RAM is only word accessible.

[2] During the Operating Mode of the Acceptance Filter the Look-up Table can be accessed only to disable or enable Messages.

A write access to all section configuration registers is only possible during the Acceptance Filter Off and Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

© NXP B.V. 2010. All rights reserved.

11.1 Acceptance filter Off mode

The Acceptance Filter Off Mode is typically used during initialization. During this mode an unconditional access to all registers and to the Look-up Table RAM is possible. With the Acceptance Filter Off Mode, CAN messages are not accepted and therefore not stored in the Receive Buffers of active CAN Controllers.

11.2 Acceptance filter Bypass mode

The Acceptance Filter Bypass Mode can be used for example to change the acceptance filter configuration during a running system, e.g. change of identifiers in the ID-Look-up Table memory. During this re-configuration, software acceptance filtering has to be used.

It is recommended to use the ID ready Interrupt (ID Index) and the Receive Interrupt (RI). In this mode all CAN message are accepted and stored in the Receive Buffers of active CAN Controllers.

11.3 Acceptance filter Operating mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 0.

11.4 FullCAN mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 1. More details on FullCAN mode are available in Section 16–16 “FullCAN mode”.

12. Sections of the ID look-up table RAM

Four 12-bit section configuration registers (SFF_sa, SFF_GRP_sa, EFF_sa, EFF_GRP_sa) are used to define the boundaries of the different identifier sections in the ID-Look-up Table Memory. The fifth 12-bit section configuration register, the End of Table address register (ENDofTable) is used to define the end of all identifier sections. The End of Table address is also used to assign the start address of the section where FullCAN Message Objects, if enabled are stored.

Table 342. Section configuration register settings

ID-Look up Table Section	Register	Value	Section status
FullCAN (Standard Frame Format) Identifier Section	SFF_sa	= 0x000	disabled
		> 0x000	enabled
Explicit Standard Frame Format Identifier Section	SFF_GRP_sa	= SFF_sa	disabled
		> SFF_sa	enabled
Group of Standard Frame Format Identifier Section	EFF_sa	= SFF_GRP_sa	disabled
		> SFF_GRP_sa	enabled
Explicit Extended Frame Format Identifier Section	EFF_GRP_sa	= EFF_sa	disabled
		> EFF_sa	enabled
Group of Extended Frame Format Identifier Section	ENDofTable	= EFF_GRP_sa	disabled
		> EFF_GRP_sa	enabled

© NXP B.V. 2010. All rights reserved.

13. ID look-up table RAM

The Whole ID Look-up Table RAM is only word accessible. A write access is only possible during the Acceptance Filter Off or Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

If Standard (11-bit) Identifiers are used in the application, at least one of 3 tables in Acceptance Filter RAM must not be empty. If the optional “FullCAN mode” is enabled, the first table contains Standard identifiers for which reception is to be handled in this mode. The next table contains individual Standard Identifiers and the third contains ranges of Standard Identifiers, for which messages are to be received via the CAN Controllers. The tables of FullCAN and individual Standard Identifiers must be arranged in ascending numerical order, one per halfword, two per word. Since each CAN bus has its own address map, each entry also contains the number of the CAN Controller (001-010) to which it applies.

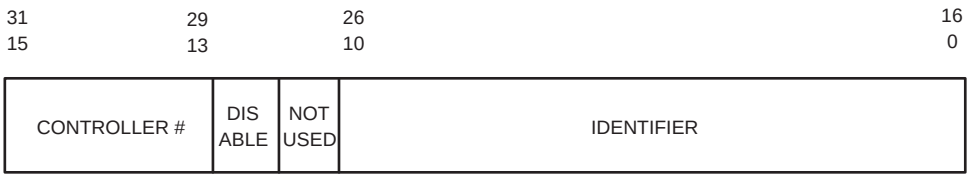


Fig 57. Entry in FullCAN and individual standard identifier tables

The table of Standard Identifier Ranges contains paired upper and lower (inclusive) bounds, one pair per word. These must also be arranged in ascending numerical order.

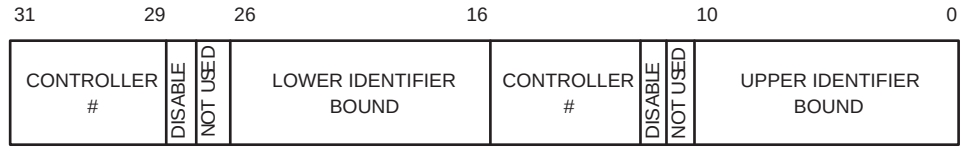


Fig 58. Entry in standard identifier range table

The disable bits in Standard entries provide a means to turn response, to particular CAN Identifiers or ranges of Identifiers, on and off dynamically. When the Acceptance Filter function is enabled, only the disable bits in Acceptance Filter RAM can be changed by software. Response to a range of Standard addresses can be enabled by writing 32 zero bits to its word in RAM, and turned off by writing 32 one bits (0xFFFF FFFF) to its word in RAM. Only the disable bits are actually changed. Disabled entries must maintain the ascending sequence of Identifiers.

If Extended (29-bit) Identifiers are used in the application, at least one of the other two tables in Acceptance Filter RAM must not be empty, one for individual Extended Identifiers and one for ranges of Extended Identifiers. The table of individual Extended Identifiers must be arranged in ascending numerical order.

© NXP B.V. 2010. All rights reserved.

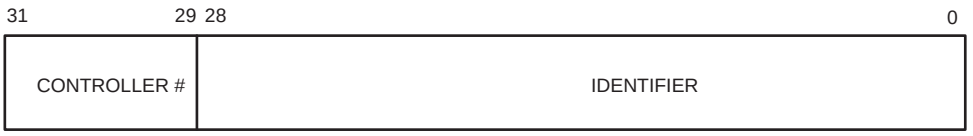


Fig 59. Entry in either extended identifier table

The table of ranges of Extended Identifiers must contain an even number of entries, of the same form as in the individual Extended Identifier table. Like the Individual Extended table, the Extended Range must be arranged in ascending numerical order. The first and second (3rd and 4th …) entries in the table are implicitly paired as an inclusive range of Extended addresses, such that any received address that falls in the inclusive range is received (accepted). Software must maintain the table to consist of such word pairs.

There is no facility to receive messages to Extended identifiers using the FullCAN method.

Five address registers point to the boundaries between the tables in Acceptance Filter RAM: FullCAN Standard addresses, Standard Individual addresses, Standard address ranges, Extended Individual addresses, and Extended address ranges. These tables must be consecutive in memory. The start of each of the latter four tables is implicitly the end of the preceding table. The end of the Extended range table is given in an End of Tables register. If the start address of a table equals the start of the next table or the End Of Tables register, that table is empty.

When the Receive side of a CAN controller has received a complete Identifier, it signals the Acceptance Filter of this fact. The Acceptance Filter responds to this signal, and reads the Controller number, the size of the Identifier, and the Identifier itself from the Controller. It then proceeds to search its RAM to determine whether the message should be received or ignored.

If FullCAN mode is enabled and the CAN controller signals that the current message contains a Standard identifier, the Acceptance Filter first searches the table of identifiers for which reception is to be done in FullCAN mode. Otherwise, or if the AF doesn't find a match in the FullCAN table, it searches its individual Identifier table for the size of Identifier signalled by the CAN controller. If it finds an equal match, the AF signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register.

If the Acceptance Filter does not find a match in the appropriate individual Identifier table, it then searches the Identifier Range table for the size of Identifier signalled by the CAN controller. If the AF finds a match to a range in the table, it similarly signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register. If the Acceptance Filter does not find a match in either the individual or Range table for the size of Identifier received, it signals the CAN controller to discard/ignore the received message.

© NXP B.V. 2010. All rights reserved.

14. Acceptance filter registers

14.1 Acceptance Filter Mode Register (AFMR - 0x4003 C000)

The AccBP and AccOff bits of the acceptance filter mode register are used for putting the acceptance filter into the Bypass and Off mode. The eFCAN bit of the mode register can be used to activate a FullCAN mode enhancement for received 11-bit CAN ID messages.

Table 343. Acceptance Filter Mode Register (AFMR - address 0x4003 C000) bit description

Bit	Symbol	Value	Description	Reset Value
0	AccOff[2]	1	if AccBP is 0, the Acceptance Filter is not operational. All Rx messages on all CAN buses are ignored.	1
1	AccBP[1]	1	All Rx messages are accepted on enabled CAN controllers. Software must set this bit before modifying the contents of any of the registers described below, and before modifying the contents of Lookup Table RAM in any way other than setting or clearing Disable bits in Standard Identifier entries. When both this bit and AccOff are 0, the Acceptance filter operates to screen received CAN Identifiers.	0
2	eFCAN[3]	0	Software must read all messages for all enabled IDs on all enabled CAN buses, from the receiving CAN controllers.	0
		1	The Acceptance Filter itself will take care of receiving and storing messages for selected Standard ID values on selected CAN buses. See Section 16–16 “FullCAN mode” on page 380.	
31:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Acceptance Filter Bypass Mode (AccBP): By setting the AccBP bit in the Acceptance Filter Mode Register, the Acceptance filter is put into the Acceptance Filter Bypass mode. During bypass mode, the internal state machine of the Acceptance Filter is reset and halted. All received CAN messages are accepted, and acceptance filtering can be done by software.

[2] Acceptance Filter Off mode (AccOff): After power-up or hardware reset, the Acceptance filter will be in Off mode, the AccOff bit in the Acceptance filter Mode register 0 will be set to 1. The internal state machine of the acceptance filter is reset and halted. If not in Off mode, setting the AccOff bit, either by hardware or by software, will force the acceptance filter into Off mode.

[3] FullCAN Mode Enhancements: A FullCAN mode for received CAN messages can be enabled by setting the eFCAN bit in the acceptance filter mode register.

14.2 Section configuration registers

The 10-bit section configuration registers are used for the ID look-up table RAM to indicate the boundaries of the different sections for explicit and group of CAN identifiers for 11-bit CAN and 29-bit CAN identifiers, respectively. The 10-bit wide section configuration registers allow the use of a 512x32 (2 kB) look-up table RAM. The whole ID Look-up Table RAM is only word accessible. All five section configuration registers contain APB addresses for the acceptance filter RAM and do not include the APB base address. A write access to all section configuration registers is only possible during the Acceptance filter off and Bypass modes. Read access is allowed in all acceptance filter modes.

© NXP B.V. 2010. All rights reserved.

14.3 Standard Frame Individual Start Address register (SFF_sa - 0x4003 C004)

Table 344. Standard Frame Individual Start Address register (SFF_sa - address 0x4003 C004) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	SFF_sa[1]	The start address of the table of individual Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the SFF_GRP_sa register described below. For compatibility with possible future devices, write zeroes in bits 31:11 and 1:0 of this register. If the eFCAN bit in the AFMR is 1, this value also indicates the size of the table of Standard IDs which the Acceptance Filter will search and (if found) automatically store received messages in Acceptance Filter RAM.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

14.4 Standard Frame Group Start Address register (SFF_GRP_sa - 0x4003 C008)

Table 345. Standard Frame Group Start Address register (SFF_GRP_sa - address 0x4003 C008) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	SFF_GRP_sa[1]	The start address of the table of grouped Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_sa register described below. The largest value that should be written to this register is 0x800, when only the Standard Individual table is used, and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

14.5 Extended Frame Start Address register (EFF_sa - 0x4003 C00C)

Table 346. Extended Frame Start Address register (EFF_sa - address 0x4003 C00C) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	EFF_sa[1]	The start address of the table of individual Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_GRP_sa register described below. The largest value that should be written to this register is 0x800, when both Extended Tables are empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:11 and 1:0 of this register.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

14.6 Extended Frame Group Start Address register (EFF_GRP_sa - 0x4003 C010)

Table 347. Extended Frame Group Start Address register (EFF_GRP_sa - address 0x4003 C010) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	Eff_GRP_sa[1]	The start address of the table of grouped Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the ENDofTable register described below. The largest value that should be written to this register is 0x800, when this table is empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

14.7 End of AF Tables register (ENDofTable - 0x4003 C014)

Table 348. End of AF Tables register (ENDofTable - address 0x4003 C014) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	EndofTable[1]	The address above the last active address in the last active AF table. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register. If the eFCAN bit in the AFMR is 0, the largest value that should be written to this register is 0x800, which allows the last word (address 0x7FC) in AF Lookup Table RAM to be used. If the eFCAN bit in the AFMR is 1, this value marks the start of the area of Acceptance Filter RAM, into which the Acceptance Filter will automatically receive messages for selected IDs on selected CAN buses. In this case, the maximum value that should be written to this register is 0x800 minus 6 times the value in SFF_sa. This allows 12 bytes of message storage between this address and the end of Acceptance Filter RAM, for each Standard ID that is specified between the start of Acceptance Filter RAM, and the next active AF table.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

14.8 Status registers

The look-up table error status registers, the error addresses, and the flag register provide information if a programming error in the look-up table RAM during the ID screening was encountered. The look-up table error address and flag register have only read access. If an error is detected, the LUTerror flag is set, and the LUTerrorAddr register provides the

© NXP B.V. 2010. All rights reserved.

information under which address during an ID screening an error in the look-up table was encountered. Any read of the LUTerrorAddr Filter block can be used for a look-up table interrupt.

14.9 LUT Error Address register (LUTerrAd - 0x4003 C018)

Table 349. LUT Error Address register (LUTerrAd - address 0x4003 C018) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	LUTerrAd	It the LUT Error bit (below) is 1, this read-only field contains the address in AF Lookup Table RAM, at which the Acceptance Filter encountered an error in the content of the tables.	0
31:11	-	Reserved, the value read from a reserved bit is not defined.	NA

14.10 LUT Error register (LUTerr - 0x4003 C01C)

Table 350. LUT Error register (LUTerr - address 0x4003 C01C) bit description

Bit	Symbol	Description	Reset Value
0	LUTerr	This read-only bit is set to 1 if the Acceptance Filter encounters an error in the content of the tables in AF RAM. It is cleared when software reads the LUTerrAd register. This condition is ORed with the “other CAN” interrupts from the CAN controllers, to produce the request that is connected to the NVIC.	0
31:1	-	Reserved, the value read from a reserved bit is not defined.	NA

14.11 Global FullCANInterrupt Enable register (FCANIE - 0x4003 C020)

A write access to the Global FullCAN Interrupt Enable register is only possible when the Acceptance Filter is in the off mode.

Table 351. Global FullCAN Enable register (FCANIE - address 0x4003 C020) bit description

Bit	Symbol	Description	Reset Value
0	FCANIE	Global FullCAN Interrupt Enable. When 1, this interrupt is enabled.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.12 FullCAN Interrupt and Capture registers (FCANIC0 - 0x4003 C024 and FCANIC1 - 0x4003 C028)

For detailed description on these two registers, see Section 16–16.2 “FullCAN interrupts”.

Table 352. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0x4003 C024) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd0	FullCan Interrupt Pending bit 0.	0
...	IntPndx (0<x<31)	FullCan Interrupt Pending bit x.	0
31	IntPnd31	FullCan Interrupt Pending bit 31.	0

© NXP B.V. 2010. All rights reserved.

Table 353. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0x4003 C028) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd32	FullCan Interrupt Pending bit 32.	0
...	IntPndx (32<x<63)	FullCan Interrupt Pending bit x.	0
31	IntPnd63	FullCan Interrupt Pending bit 63.	0

15. Configuration and search algorithm

The CAN Identifier Look-up Table Memory can contain explicit identifiers and groups of CAN identifiers for Standard and Extended CAN Frame Formats. They are organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

Every CAN identifier is linked to an ID Index number. In case of a CAN Identifier match, the matching ID Index is stored in the Identifier Index of the Frame Status Register (CANRFS) of the according CAN Controller.

15.1 Acceptance filter search algorithm

The identifier screening process of the acceptance filter starts in the following order:

1. FullCAN (Standard Frame Format) Identifier Section
2. Explicit Standard Frame Format Identifier Section
3. Group of Standard Frame Format Identifier Section
4. Explicit Extended Frame Format Identifier Section
5. Group of Extended Frame Format Identifier Section

Note: Only activated sections will take part in the screening process.

In cases where equal message identifiers of same frame format are defined in more than one section, the first match will end the screening process for this identifier.

For example, if the same Source CAN Channel in conjunction with the identifier is defined in the FullCAN, the Explicit Standard Frame Format and the Group of Standard Frame Format Identifier Sections, the screening will already be finished with the match in the FullCAN section.

In the example of Figure 16–60, Identifiers with their Source CAN Channel have been defined in the FullCAN, Explicit and Group of Standard Frame Format Identifier Sections. This example corresponds part with 6 CAN controllers.

© NXP B.V. 2010. All rights reserved.

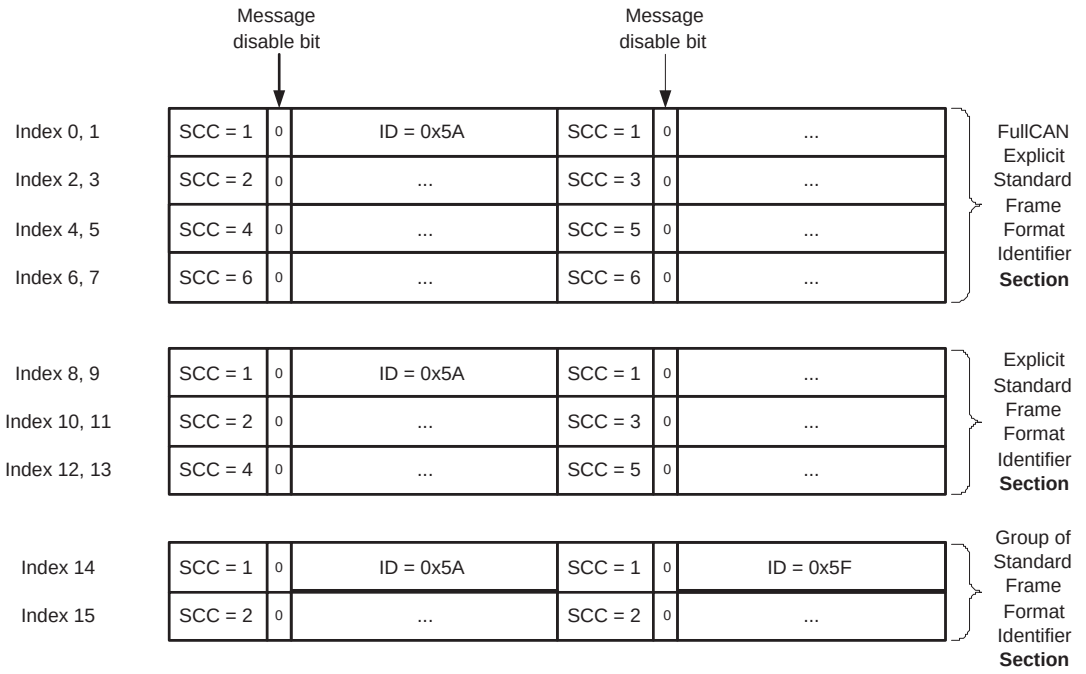


Fig 60. ID Look-up table example explaining the search algorithm

The identifier 0x5A of the CAN Controller 1 with the Source CAN Channel SCC = 1, is defined in all three sections. With this configuration incoming CAN messages on CAN Controller 1 with a 0x5A identifier will find a match in the FullCAN section.

It is possible to disable the '0x5A identifier' in the FullCAN section. With that, the screening process would be finished with the match in the Explicit Identifier Section.

The first group in the Group Identifier Section has been defined in that way, that incoming CAN messages with identifiers of 0x5A up to 0x5F are accepted on CAN Controller 1 with the Source CAN Channel SCC = 1. As stated above, the identifier 0x5A would find a match already in the FullCAN or in the Explicit Identifier section if enabled. The rest of the defined identifiers of this group (0x5B to 0x5F) will find a match in this Group Identifier Section.

This way the user can switch dynamically between different filter modes for same identifiers.

16. FullCAN mode

The FullCAN mode is based on capabilities provided by the CAN Gateway module used in the LPC2000 family of products. This block uses the Acceptance Filter to provide filtering for both CAN channels.

The concept of the CAN Gateway block is mainly based on a BasicCAN functionality. This concept fits perfectly in systems where a gateway is used to transfer messages or message data between different CAN channels. A BasicCAN device is generating a

© NXP B.V. 2010. All rights reserved.

receive interrupt whenever a CAN message is accepted and received. Software has to move the received message out of the receive buffer from the according CAN controller into the user RAM.

To cover dashboard like applications where the controller typically receives data from several CAN channels for further processing, the CAN Gateway block was extended by a so-called FullCAN receive function. This additional feature uses an internal message handler to move received FullCAN messages from the receive buffer of the according CAN controller into the FullCAN message object data space of Look-up Table RAM.

When FullCAN mode is enabled, the Acceptance Filter itself takes care of receiving and storing messages for selected Standard ID values on selected CAN buses, in the style of “FullCAN” controllers.

In order to set this bit and use this mode, two other conditions must be met with respect to the contents of Acceptance Filter RAM and the pointers into it:

- The Standard Frame Individual Start Address Register (SFF_sa) must be greater than or equal to the number of IDs for which automatic receive storage is to be done, times two. SFF_sa must be rounded up to a multiple of 4 if necessary.
- The EndOfTable register must be less than or equal to 0x800 minus 6 times the SFF_sa value, to allow 12 bytes of message storage for each ID for which automatic receive storage will be done.

When these conditions are met and eFCAN is set:

- The area between the start of Acceptance Filter RAM and the SFF_sa address, is used for a table of individual Standard IDs and CAN Controller/bus identification, sorted in ascending order and in the same format as in the Individual Standard ID table (see Figure 16–57 “Entry in FullCAN and individual standard identifier tables” on page 373). Entries can be marked as “disabled” as in the other Standard tables. If there are an odd number of “FullCAN” ID’s, at least one entry in this table must be so marked.
- The first (SFF_sa)/2 IDindex values are assigned to these automatically-stored ID’s. That is, IDindex values stored in the Rx Frame Status Register, for IDs not handled in this way, are increased by (SFF_sa)/2 compared to the values they would have when eFCAN is 0.
- When a Standard ID is received, the Acceptance Filter searches this table before the Standard Individual and Group tables.
- When a message is received for a controller and ID in this table, the Acceptance filter reads the received message out of the CAN controller and stores it in Acceptance Filter RAM, starting at (EndOfTable) + its IDindex*12.
- The format of such messages is shown in Table 16–354.

© NXP B.V. 2010. All rights reserved.

16.1 FullCAN message layout

Table 354. Format of automatically stored Rx messages

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FF	RTR	0000				SEM [1:0]		0000				DLC				00000					ID.28 ... ID.18										
+4	Rx Data 4								Rx Data 3								Rx Data 2								Rx Data 1							
+8	Rx Data 8								Rx Data 7								Rx Data 6								Rx Data 5							

The FF, RTR, and DLC fields are as described in Table 16–326.

Since the FullCAN message object section of the Look-up table RAM can be accessed both by the Acceptance Filter and the CPU, there is a method for insuring that no CPU reads from FullCAN message object occurs while the Acceptance Filter hardware is writing to that object.

For this purpose the Acceptance Filter uses a 3-state semaphore, encoded with the two semaphore bits SEM1 and SEM0 (see Table 16–354 “Format of automatically stored Rx messages”) for each message object. This mechanism provides the CPU with information about the current state of the Acceptance Filter activity in the FullCAN message object section.

The semaphore operates in the following manner:

Table 355. FullCAN semaphore operation

SEM1	SEM0	activity
0	1	Acceptance Filter is updating the content
1	1	Acceptance Filter has finished updating the content
0	0	CPU is in process of reading from the Acceptance Filter

Prior to writing the first data byte into a message object, the Acceptance Filter will write the FrameInfo byte into the according buffer location with SEM[1:0] = 01.

After having written the last data byte into the message object, the Acceptance Filter will update the semaphore bits by setting SEM[1:0] = 11.

Before reading a message object, the CPU should read SEM[1:0] to determine the current state of the Acceptance Filter activity therein. If SEM[1:0] = 01, then the Acceptance Filter is currently active in this message object. If SEM[1:0] = 11, then the message object is available to be read.

Before the CPU begins reading from the message object, it should clear SEM[1:0] = 00.

When the CPU is finished reading, it can check SEM[1:0] again. At the time of this final check, if SEM[1:0] = 01 or 11, then the Acceptance Filter has updated the message object during the time when the CPU reads were taking place, and the CPU should discard the data. If, on the other hand, SEM[1:0] = 00 as expected, then valid data has been successfully read by the CPU.

Figure 16–61 shows how software should use the SEM field to ensure that all three words read from the message are all from the same received message.

© NXP B.V. 2010. All rights reserved.

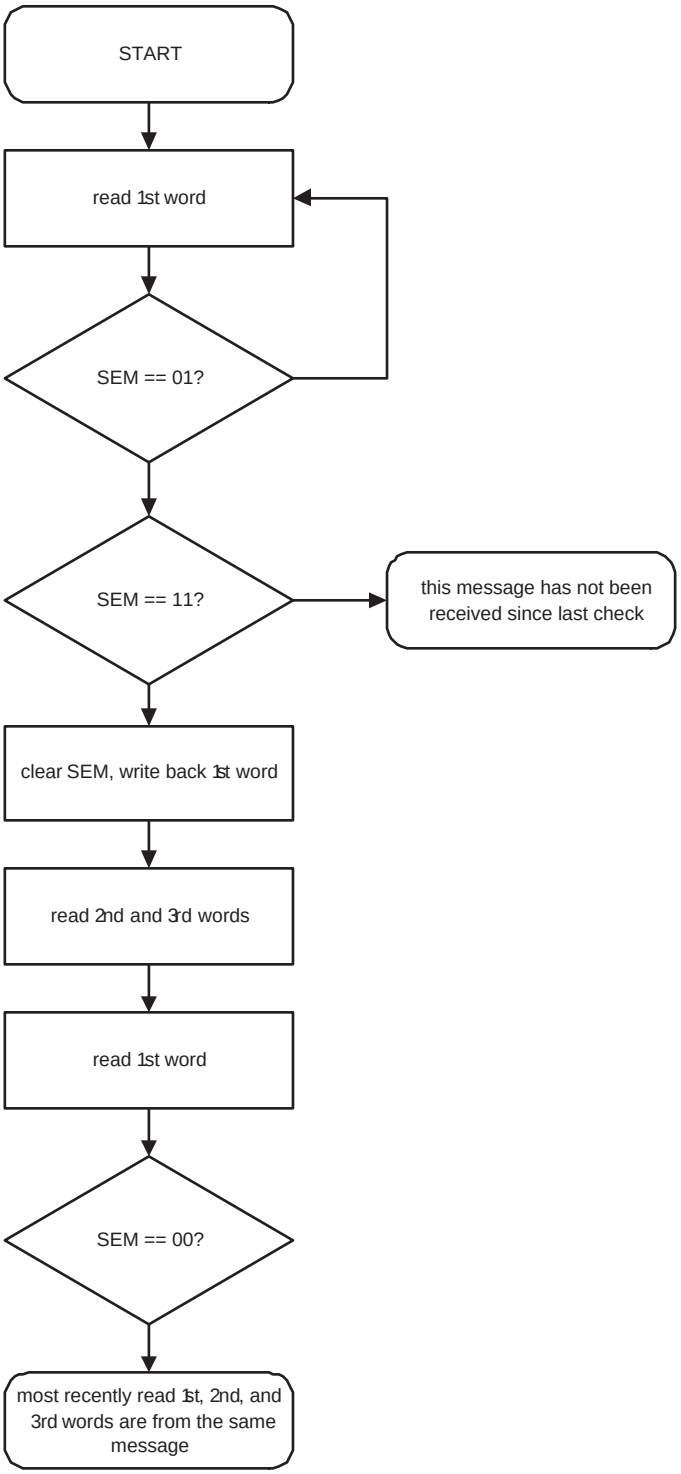


Fig 61. Semaphore procedure for reading an auto-stored message

© NXP B.V. 2010. All rights reserved.

16.2 FullCAN interrupts

The CAN Gateway Block contains a 2 kB ID Look-up Table RAM. With this size a maximum number of 146 FullCAN objects can be defined if the whole Look-up Table RAM is used for FullCAN objects only. Only the first 64 FullCAN objects can be configured to participate in the interrupt scheme. It is still possible to define more than 64 FullCAN objects. The only difference is, that the remaining FullCAN objects will not provide a FullCAN interrupt.

The FullCAN Interrupt Register-set contains interrupt flags (IntPndx) for (pending) FullCAN receive interrupts. As soon as a FullCAN message is received, the according interrupt bit (IntPndx) in the FCAN Interrupt Register gets asserted. In case that the Global FullCAN Interrupt Enable bit is set, the FullCAN Receive Interrupt is passed to the Vectored Interrupt Controller.

Application Software has to solve the following:

1. Index/Object number calculation based on the bit position in the FCANIC Interrupt Register for more than one pending interrupt.
2. Interrupt priority handling if more than one FullCAN receive interrupt is pending.

The software that covers the interrupt priority handling has to assign a receive interrupt priority to every FullCAN object. If more than one interrupt is pending, then the software has to decide, which received FullCAN object has to be served next.

To each FullCAN object a new FullCAN Interrupt Enable bit (FCANIntxEn) is added, so that it is possible to enable or disable FullCAN interrupts for each object individually. The new Message Lost flag (MsgLstx) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read by the CPU. The Interrupt Enable and the Message Lost bits reside in the existing Look-up Table RAM.

16.2.1 FullCAN message interrupt enable bit

In Figure 16–62 8 FullCAN Identifiers with their Source CAN Channel are defined in the FullCAN, Section. The new introduced FullCAN Message Interrupt enable bit can be used to enable for each FullCAN message an Interrupt.

© NXP B.V. 2010. All rights reserved.

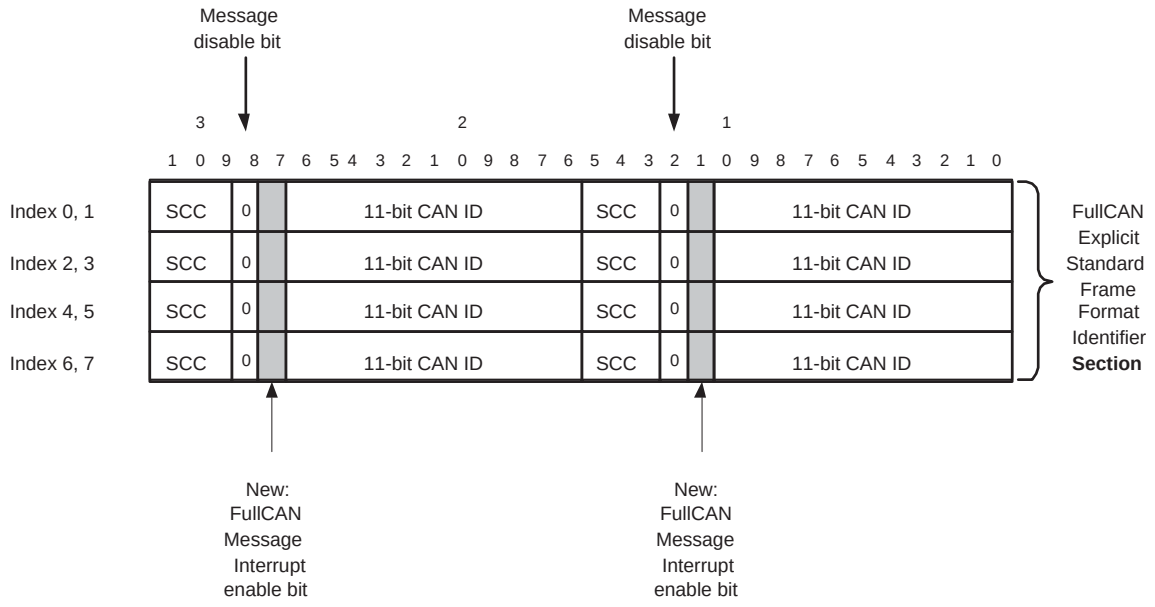


Fig 62. FullCAN section example of the ID look-up table

16.2.2 Message lost bit and CAN channel number

Figure 16–63 is the detailed layout structure of one FullCAN message stored in the FullCAN message object section of the Look-up Table.

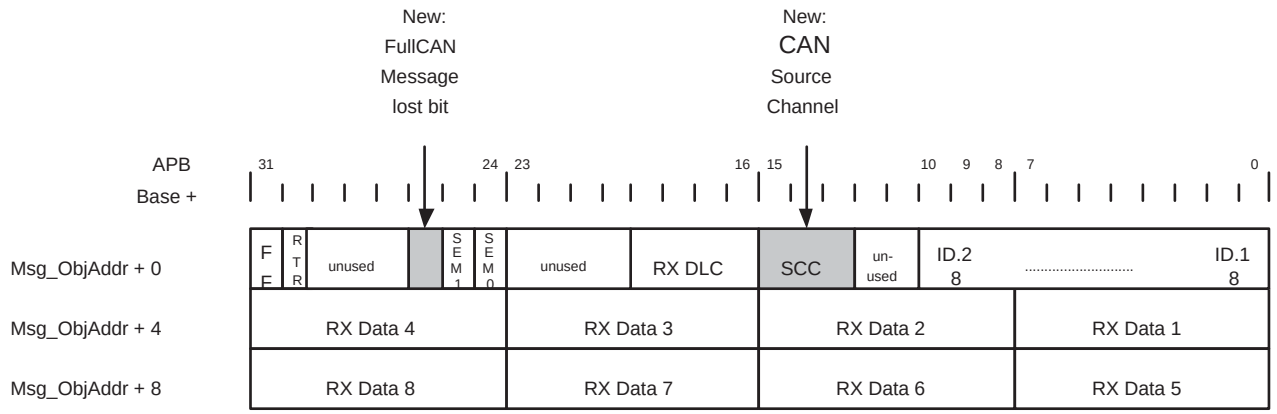


Fig 63. FullCAN message object layout

The new message lost bit (MsgLst) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read. For more information the CAN Source Channel (SCC) of the received FullCAN message is added to Message Object.

© NXP B.V. 2010. All rights reserved.

16.2.3 Setting the interrupt pending bits (IntPnd 63 to 0)

The interrupt pending bit (IntPndx) gets asserted in case of an accepted FullCAN message and if the interrupt of the according FullCAN Object is enabled (enable bit FCANIntxEn) is set).

During the **last write access** from the data storage of a FullCAN message object the interrupt pending bit of a FullCAN object (IntPndx) gets asserted.

16.2.4 Clearing the interrupt pending bits (IntPnd 63 to 0)

Each of the FullCAN Interrupt Pending requests gets cleared when the semaphore bits of a message object are cleared by Software (ARM CPU).

16.2.5 Setting the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets asserted in case of an accepted FullCAN message and when the FullCAN Interrupt of the same object is asserted already.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets asserted if the interrupt pending bit is set already.

16.2.6 Clearing the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets cleared when the FullCAN Interrupt of the same object is not asserted.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets cleared if the interrupt pending bit is not set.

16.3 Set and clear mechanism of the FullCAN interrupt

Special precaution is needed for the built-in set and clear mechanism of the FullCAN Interrupts. The following text illustrates how the already existing Semaphore Bits (see Section 16–16.1 “FullCAN message layout” for more details) and how the new introduced features (IntPndx, MsgLstx) will behave.

16.3.1 Scenario 1: Normal case, no message lost

Figure 16–64 below shows a typical “normal” scenario in which an accepted FullCAN message is stored in the FullCAN Message Object Section. After storage the message is read out by Software (ARM CPU).

© NXP B.V. 2010. All rights reserved.

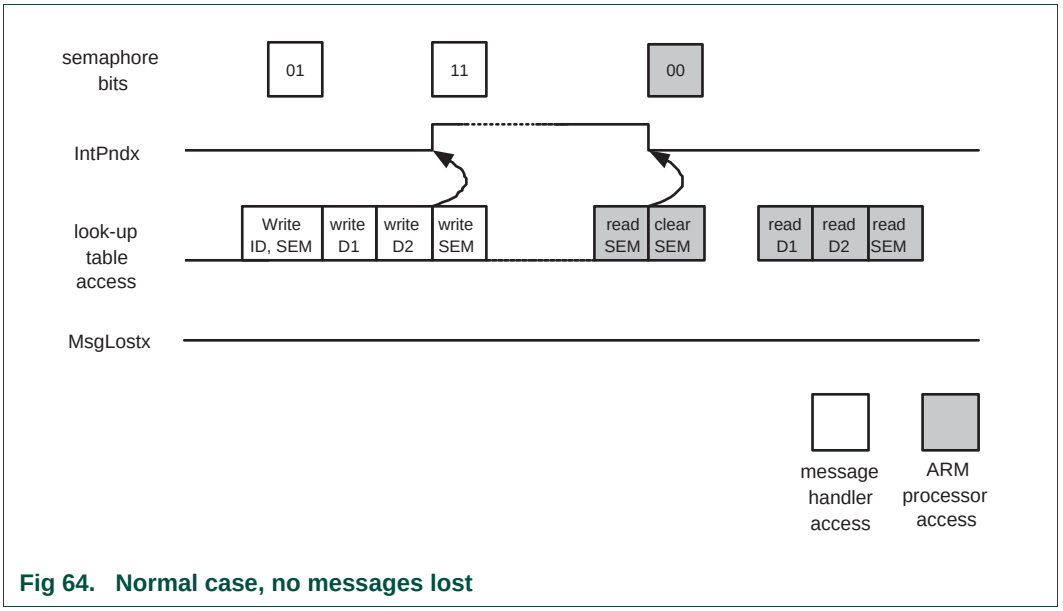


Fig 64. Normal case, no messages lost

16.3.2 Scenario 2: Message lost

In this scenario a first FullCAN Message is stored and read out by Software (1st Object write and read). In a second course a second message is stored (2nd Object write) but not read out before a third message gets stored (3rd Object write). Since the FullCAN Interrupt of that Object (IntPndx) is already asserted, the Message Lost Signal gets asserted.

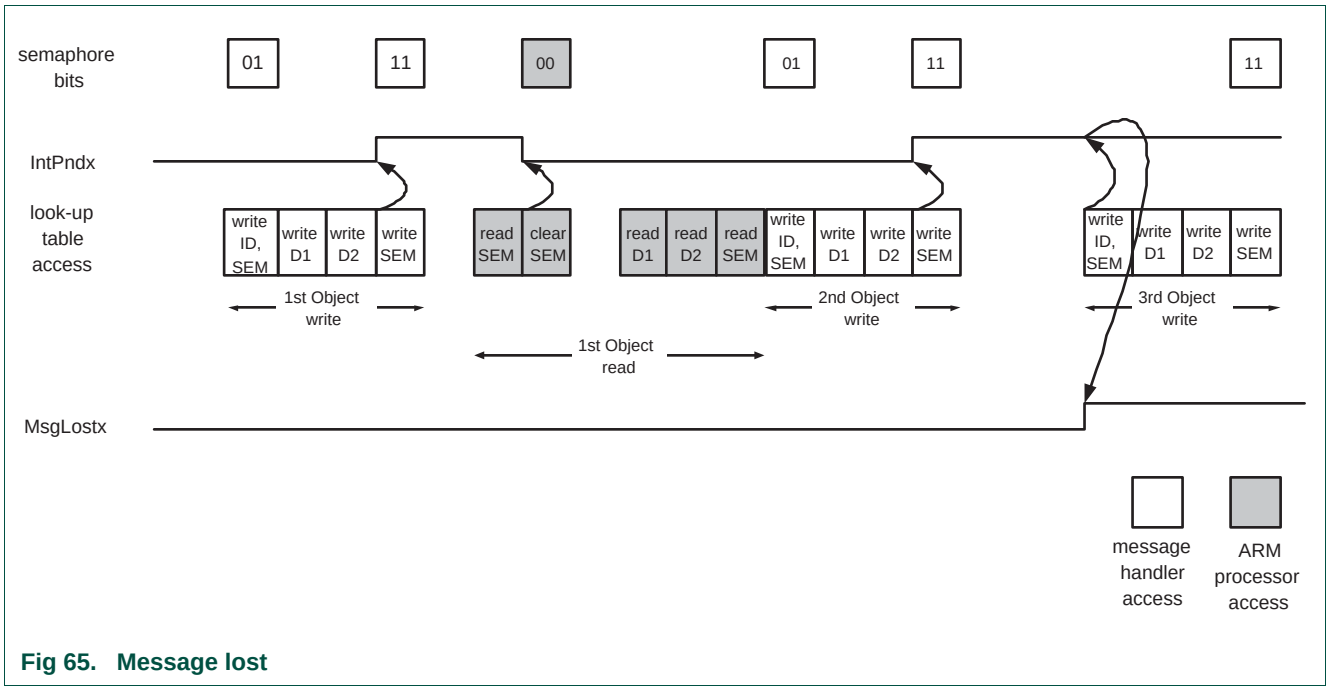


Fig 65. Message lost

© NXP B.V. 2010. All rights reserved.

16.3.3 Scenario 3: Message gets overwritten indicated by Semaphore bits

This scenario is a special case in which the lost message is indicated by the existing semaphore bits. The scenario is entered, if during a Software read of a message object another new message gets stored by the message handler. In this case, the FullCAN Interrupt bit gets set for a second time with the 2nd Object write.

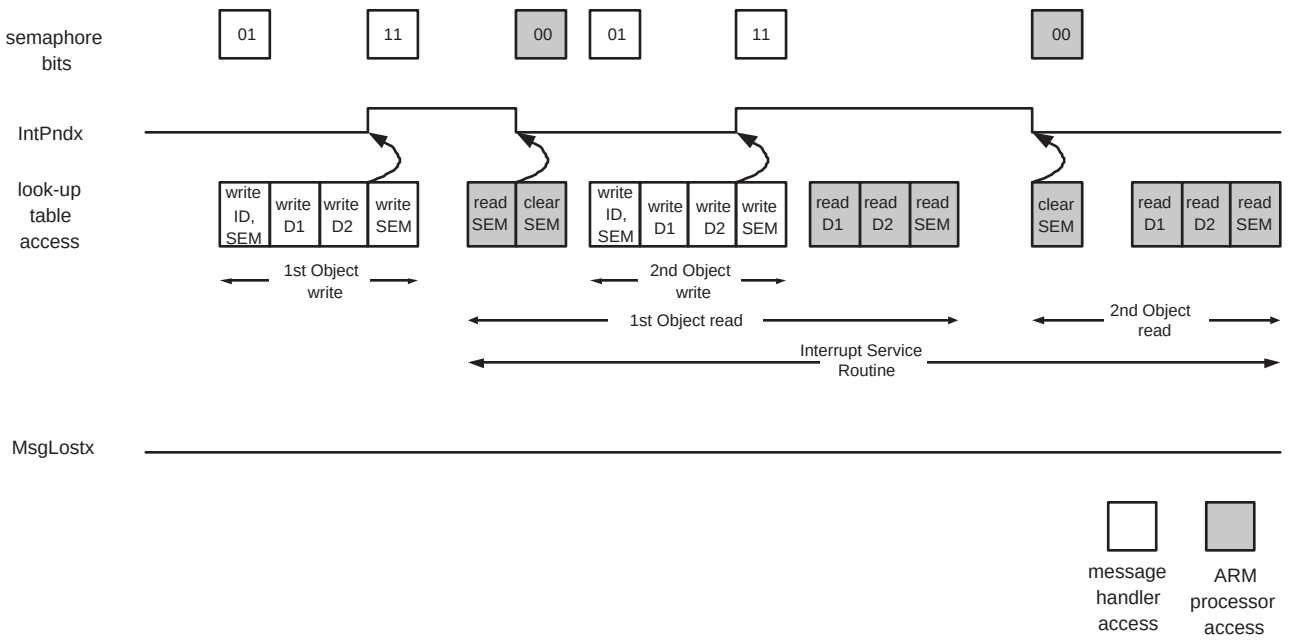


Fig 66. Message gets overwritten

16.3.4 Scenario 3.1: Message gets overwritten indicated by Semaphore bits and Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by the existing semaphore bits and by Message Lost.

© NXP B.V. 2010. All rights reserved.

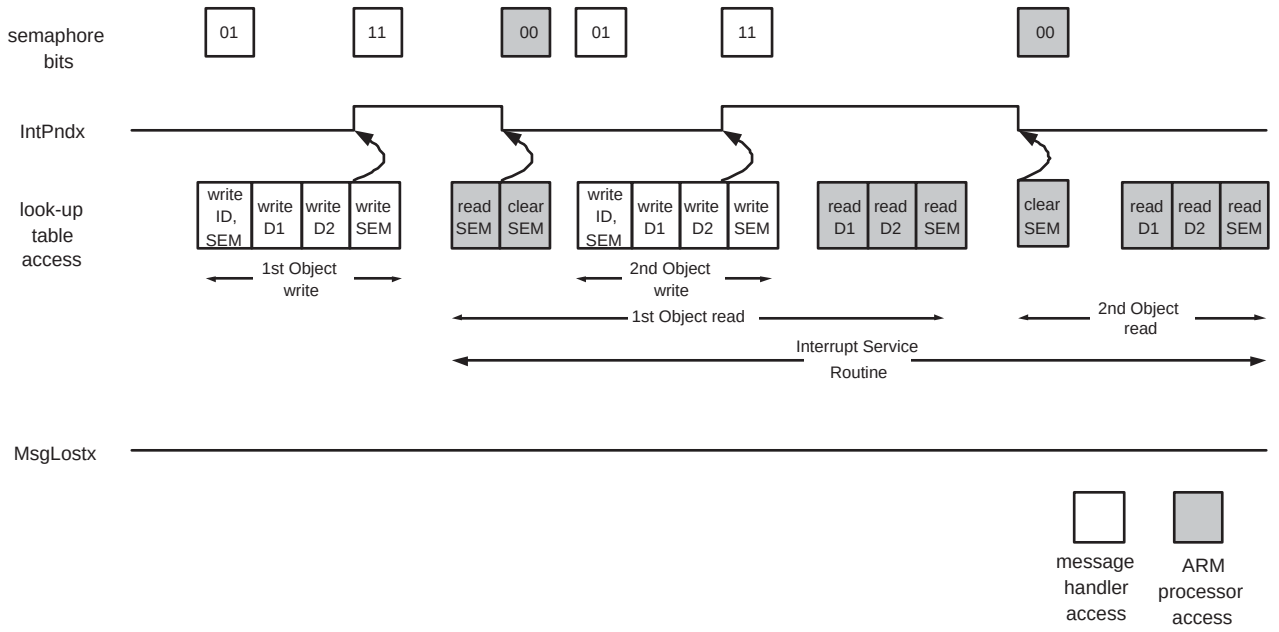


Fig 67. Message overwritten indicated by semaphore bits and message lost

16.3.5 Scenario 3.2: Message gets overwritten indicated by Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by Message Lost.

© NXP B.V. 2010. All rights reserved.

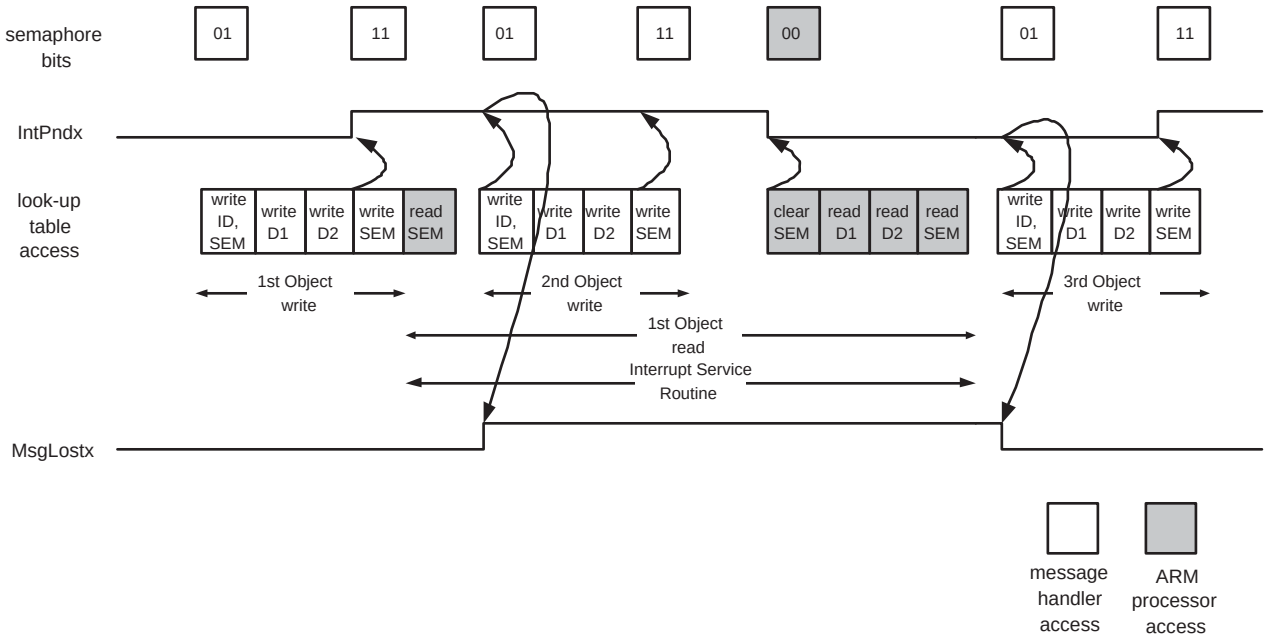


Fig 68. Message overwritten indicated by message lost

16.3.6 Scenario 4: Clearing Message Lost bit

This scenario is a special case in which the lost message bit of an object gets set during an overwrite of a none read message object (2nd Object write). The subsequent read out of that object by Software (1st Object read) clears the pending Interrupt. The 3rd Object write clears the Message Lost bit. Every “write ID, SEM” clears Message Lost bit if no pending Interrupt of that object is set.

© NXP B.V. 2010. All rights reserved.

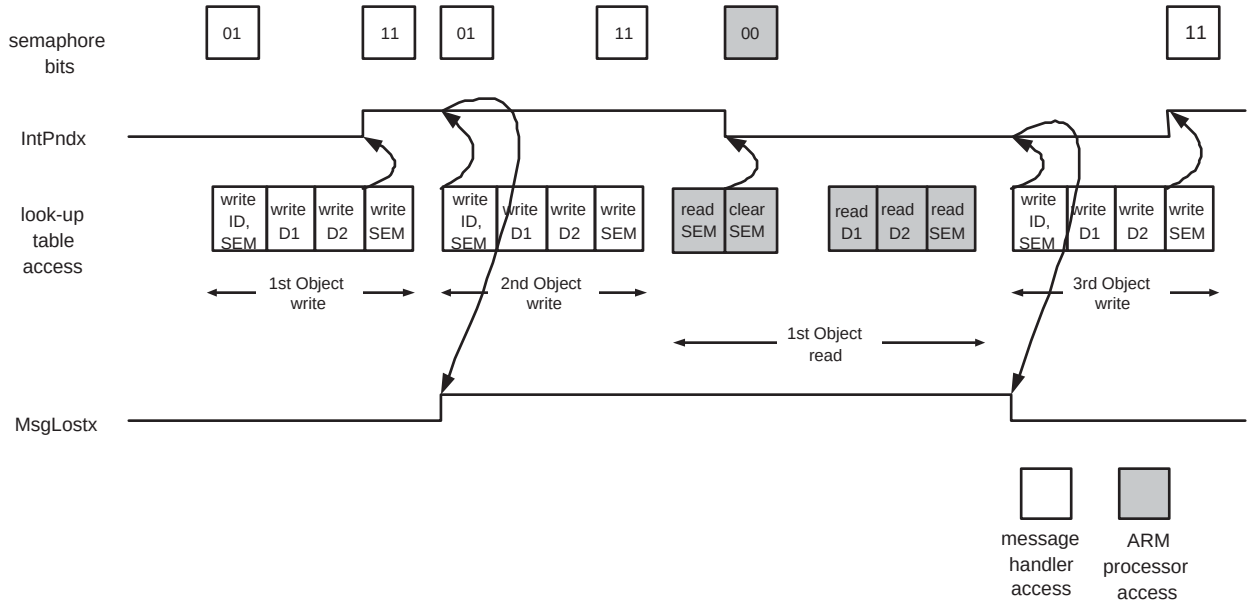


Fig 69. Clearing message lost

17. Examples of acceptance filter tables and ID index values

17.1 Example 1: only one section is used

```
SFF_sa     <    ENDofTable      OR
SFF_GRP_sa <    ENDofTable      OR
EFF_sa     <    ENDofTable      OR
EFF_GRP_sa <    ENDofTable
```

The start address of a section is lower than the end address of all programmed CAN identifiers.

17.2 Example 2: all sections are used

```
SFF_sa     <    SFF_GRP_sa      AND
SFF_GRP_sa <    EFF_sa          AND
EFF_sa     <    EFF_GRP_sa      AND
EFF_GRP_sa <    ENDofTable
```

In cases of a section not being used, the start address has to be set onto the value of the next section start address.

17.3 Example 3: more than one but not all sections are used

If the SFF group is not used, the start address of the SFF Group Section (SFF_GRP_sa register) has to be set to the same value of the next section start address, in this case the start address of the Explicit SFF Section (SFF_sa register).

© NXP B.V. 2010. All rights reserved.

In cases where explicit identifiers as well as groups of the identifiers are programmed, a CAN identifier search has to start in the explicit identifier section first. If no match is found, it continues the search in the group of identifier section. By this order it can be guaranteed that in case where an explicit identifier match is found, the succeeding software can directly proceed on this certain message whereas in case of a group of identifier match the succeeding software needs more steps to identify the message.

17.4 Configuration example 4

Suppose that the five Acceptance Filter address registers contain the values shown in the third column below. In this case each table contains the decimal number of words and entries shown in the next two columns, and the ID Index field of the CANRFS register can return the decimal values shown in the column ID Indexes for CAN messages whose Identifiers match the entries in that table.

Table 356. Example of Acceptance Filter Tables and ID index Values

Table	Register	Value	# Words	# Entire	ID Indexes
Standard Individual	SFF_sa	0x040	8_{10}	16_{10}	$0\text{-}15_{10}$
Standard Group	SFF_GRP_sa	0x060	4_{10}	4_{10}	$16\text{-}19_{10}$
Extended Individual	EFF_sa	0x070	8_{10}	16_{10}	$20\text{-}55_{10}$
Extended Group	EFF_GRP_sa	0x100	8_{10}	16_{10}	$56\text{-}57_{10}$
	ENDofTable	0x110			

17.5 Configuration example 5

Figure 16–70 below is a more detailed and graphic example of the address registers, table layout, and ID Index values. It shows:

- A Standard Individual table starting at the start of Acceptance Filter RAM and containing 26 Identifiers, followed by:
- A Standard Group table containing 12 ranges of Identifiers, followed by:
- An Extended Individual table containing 3 Identifiers, followed by:
- An Extended Group table containing 2 ranges of Identifiers.

© NXP B.V. 2010. All rights reserved.

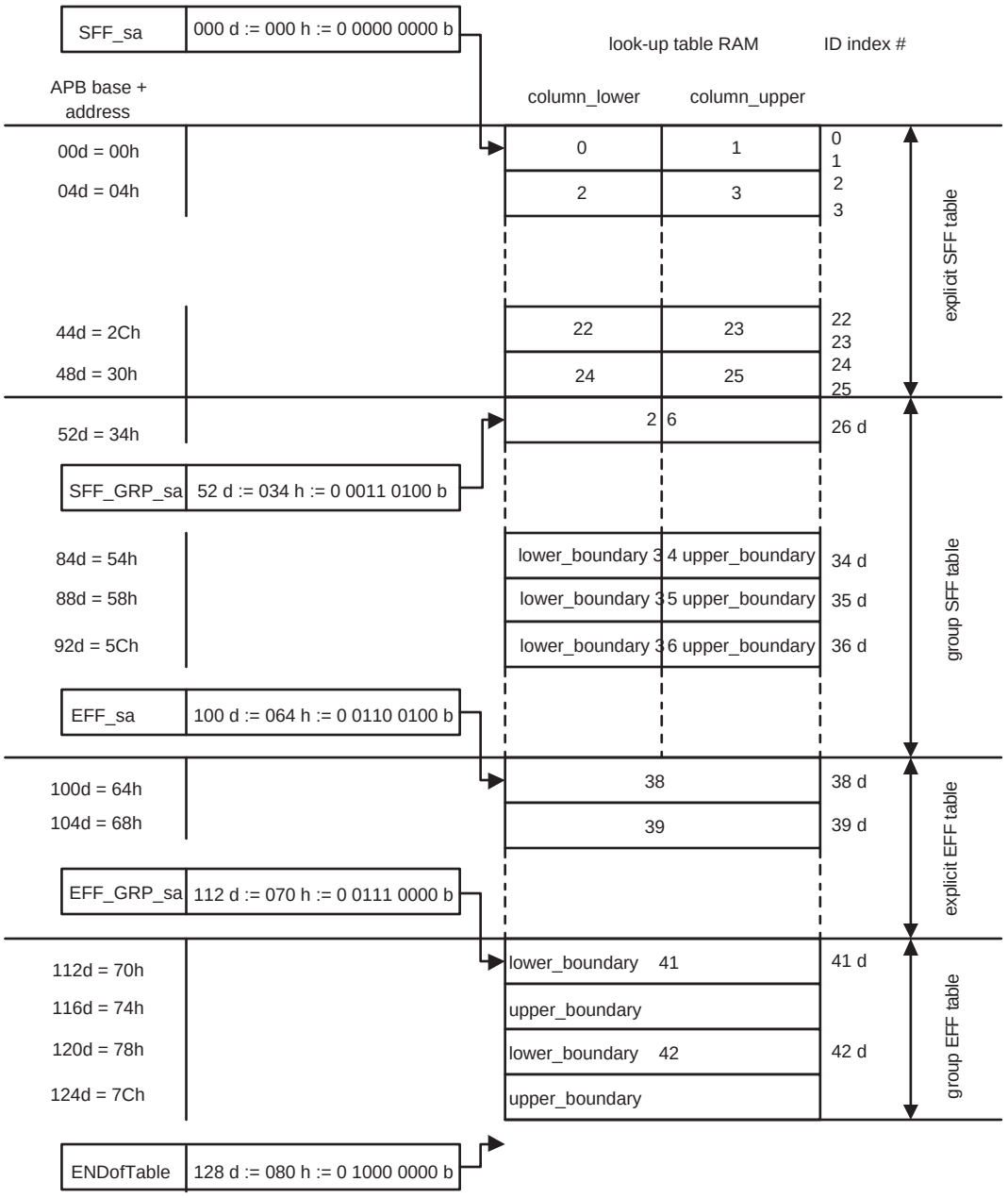


Fig 70. Detailed example of acceptance filter tables and ID index values

17.6 Configuration example 6

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 16–71.

© NXP B.V. 2010. All rights reserved.

Table 357. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	not activated
Explicit Standard Frame Format	activated
Group of Standard Frame Format	activated
Explicit Extended Frame Format	activated
Group of Extended Frame Format	activated

Explicit standard frame format identifier section (11-bit CAN ID):

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x00. The end of this section is defined in the SFF_GRP_sa register. In the Explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit.

Group of standard frame format identifier section (11-bit CAN ID):

The start address of the Group of Standard Frame Format section is defined with the SFF_GRP_sa register with the value of 0x10. The end of this section is defined with the EFF_sa register. In the Group of Standard Frame Format section two CAN Identifiers with the same Source CAN Channel (SCC) share one 32-bit word and represent a range of CAN Identifiers to be accepted. Bit 31 down to 16 represents the lower boundary and bit 15 down to 0 represents the upper boundary of the range of CAN Identifiers. All Identifiers within this range (including the boundary identifiers) will be accepted. A whole group can be disabled and not used by the acceptance filter by setting the message disable bit in the upper and lower boundary identifier. To provide memory space for four Groups of Standard Frame Format identifiers, the EFF_sa register value is set to 0x20. The identifier group with the Index 9 of this section is not used and therefore disabled.

Explicit extended frame format identifier section (29-bit CAN ID, Figure 16–71)

The start address of the Explicit Extended Frame Format section is defined with the EFF_sa register with the value of 0x20. The end of this section is defined with the EFF_GRP_sa register. In the explicit Extended Frame Format section only one CAN Identifier with its Source CAN Channel (SCC) is programmed per address line. To provide memory space for four Explicit Extended Frame Format identifiers, the EFF_GRP_sa register value is set to 0x30.

Group of extended frame format identifier section (29-bit CAN ID, Figure 16–71)

The start address of the Group of Extended Frame Format is defined with the EFF_GRP_sa register with the value of 0x30. The end of this section is defined with the End of Table address register (ENDofTable). In the Group of Extended Frame Format section the boundaries are programmed with a pair of address lines; the first is the lower boundary, the second the upper boundary. To provide memory space for two Groups of Extended Frame Format Identifiers, the ENDofTable register value is set to 0x40.

© NXP B.V. 2010. All rights reserved.

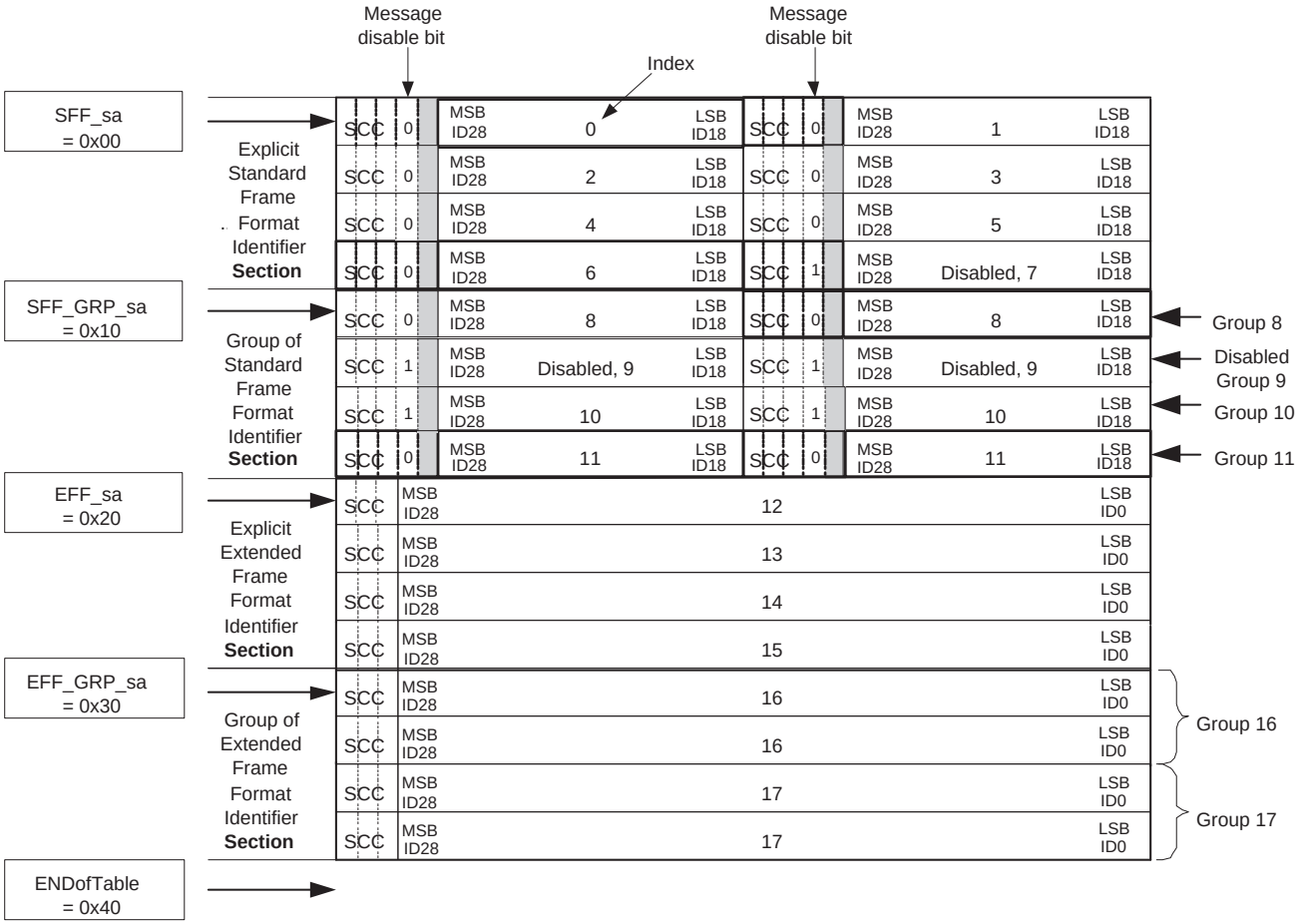


Fig 71. ID Look-up table configuration example (no FullCAN)

17.7 Configuration example 7

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 16–72.

This example uses a typical configuration in which FullCAN as well as Explicit Standard Frame Format messages are defined. As described in Section 16–15.1 “Acceptance filter search algorithm”, acceptance filtering takes place in a certain order. With the enabled FullCAN section, the identifier screening process of the acceptance filter starts always in the FullCAN section first, before it continues with the rest of enabled sections.e disabled.

Table 358. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	activated and enabled
Explicit Standard Frame Format	activated
Group of Standard Frame Format	not activated
Explicit Extended Frame Format	not activated
Group of Extended Frame Format	not activated

© NXP B.V. 2010. All rights reserved.

FullCAN explicit standard frame format identifier section (11-bit CAN ID)

The start address of the FullCAN Explicit Standard Frame Format Identifier section is (automatically) set to 0x00. The end of this section is defined in the SFF_sa register. In the FullCAN ID section only identifiers of FullCAN Object are stored for acceptance filtering. In this section two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. The FullCAN Object data for each defined identifier can be found in the FullCAN Message Object section. In case of an identifier match during the acceptance filter process, the received FullCAN message object data is moved from the Receive Buffer of the appropriate CAN Controller into the FullCAN Message Object section. To provide memory space for eight FullCAN, Explicit Standard Frame Format identifiers, the SFF_sa register value is set to 0x10. The identifier with the Index 1 of this section is not used and therefore disabled.

Explicit standard frame format identifier section (11-bit CAN ID)

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x10. The end of this section is defined in the End of Table address register (ENDofTable). In the explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channel (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. To provide memory space for eight Explicit Standard Frame Format identifiers, the ENDofTable register value is set to 0x20.

FullCAN message object data section

The start address of the FullCAN Message Object Data section is defined with the ENDofTable register. The number of enabled FullCAN identifiers is limited to the available memory space in the FullCAN Message Object Data section. Each defined FullCAN Message needs three address lines for the Message Data in the FullCAN Message Object Data section. The FullCAN Message Object section is organized in that way, that each Index number of the FullCAN Identifier section corresponds to a Message Object Number in the FullCAN Message Object section.

© NXP B.V. 2010. All rights reserved.

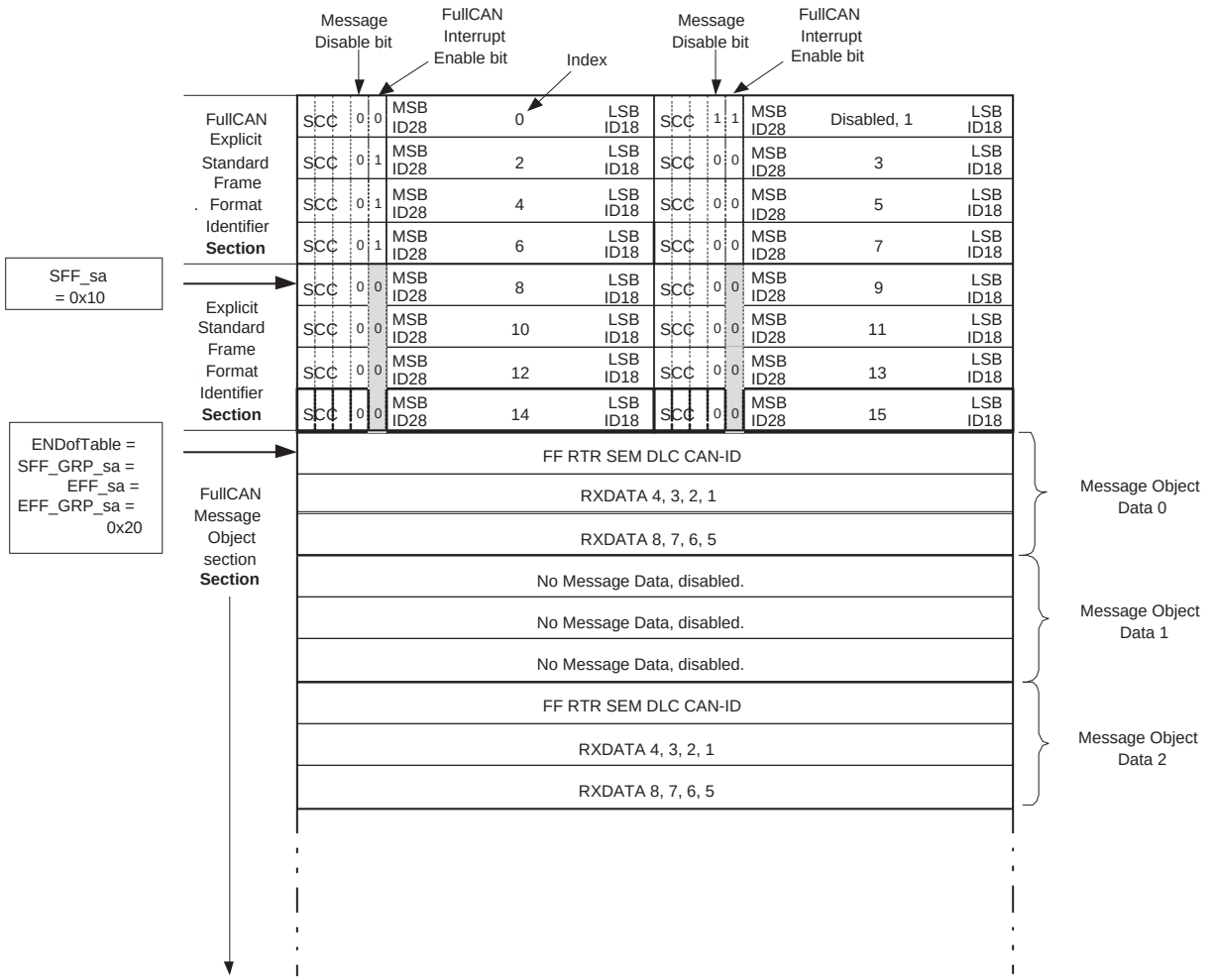


Fig 72. ID Look-up table configuration example (FullCAN activated and enabled)

17.8 Look-up table programming guidelines

All identifier sections of the ID Look-up Table have to be programmed in such a way, that each active section is organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

In cases, where a syntax error in the ID Look-up Table is encountered, the Look-up Table address of the incorrect line is made available in the Look-up Table Error Address Register (LUTerrAd).

The reporting process in the Look-up Table Error Address Register (LUTerrAd) is a “run-time” process. Only those address lines with syntax error are reported, which were passed through the acceptance filtering process.

The following general rules for programming the Look-up Table apply:

© NXP B.V. 2010. All rights reserved.

- Each section has to be organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) in conjunction with the CAN Identifier (there is no exception for disabled identifiers).
- The upper and lower bound in a Group of Identifiers definition has to be from the same Source CAN Channel.
- To disable a Group of Identifiers the message disable bit has to be set for both, the upper and lower bound.

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The SPI is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCSPI.

   Remark: On reset, the SPI is enabled (PCSPI = 1).

2. Clock: In the PCLKSEL0 register (Table 4–40), set bit PCLK_SPI. In master mode, the clock must be an even number greater than or equal to 8 (see Section 17–7.4).

3. Pins: The SPI pins are configured using both PINSEL0 (Table 8–78) and PINSEL1 (Table 8–79), as well as the PINMODE (Section 8–4) register. PINSEL0[31:30] is used to configure the SPI CLK pin. PINSEL1[1:0], PINSEL1[3:2] and PINSEL1[5:4] are used to configure the pins SSEL, MISO and MOSI, respectively.

4. Interrupts: The SPI interrupt flag is enabled using the S0SPINT[0] bit (Section 17–7.7). The SPI interrupt flag must be enabled in the NVIC, see Table 6–50.

Remark: SSP0 is intended to be used as an alternative for the SPI interface, which is included as a legacy peripheral. Only one of these peripherals can be used at the any one time.

2. Features

- Compliant with Serial Peripheral Interface (SPI) specification.
- Synchronous, Serial, Full Duplex Communication.
- SPI master or slave.
- Maximum data bit rate of one eighth of the peripheral clock rate.
- 8 to 16 bits per transfer.

3. SPI overview

SPI is a full duplex serial interface. 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.

© NXP B.V. 2010. All rights reserved.

4. Pin description

Table 359. SPI pin description

Pin Name	Type	Pin Description
SCK	Input/ Output	**Serial Clock.** The SPI clock signal (SCK) is used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL	Input	**Slave Select.** The SPI slave select signal (SSEL) 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.
MISO	Input/ Output	**Master In Slave Out.** The SPI Master In Slave Out signal (MISO) is a unidirectional signal used to transfer serial data from an SPI slave to an SPI master. When a device is a slave, serial data is output on this pin. When a device is a master, serial data is input on this pin. When a slave device is not selected, the slave drives the signal high-impedance.
MOSI	Input/ Output	**Master Out Slave In.** The SPI Master Out Slave In signal (MOSI) is a unidirectional signal used to transfer serial data from an SPI master to an SPI slave. When a device is a master, serial data is output on this pin. When a device is a slave, serial data is input on this pin.

5. SPI data transfers

Figure 17–73 is a timing diagram that illustrates the four different data transfer formats that are available with the SPI port. 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 Clock Phase control bit (CPHA) in the SPI Control Register 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 the Clock Polarity control bit (CPOL) in the SPI Control Register set to both 0 and 1. The second point to note is the activation and de-activation of the SSEL signal. When CPHA = 0, the SSEL signal will always go inactive between data transfers. This is not guaranteed when CPHA = 1 (the signal can remain active).

© NXP B.V. 2010. All rights reserved.

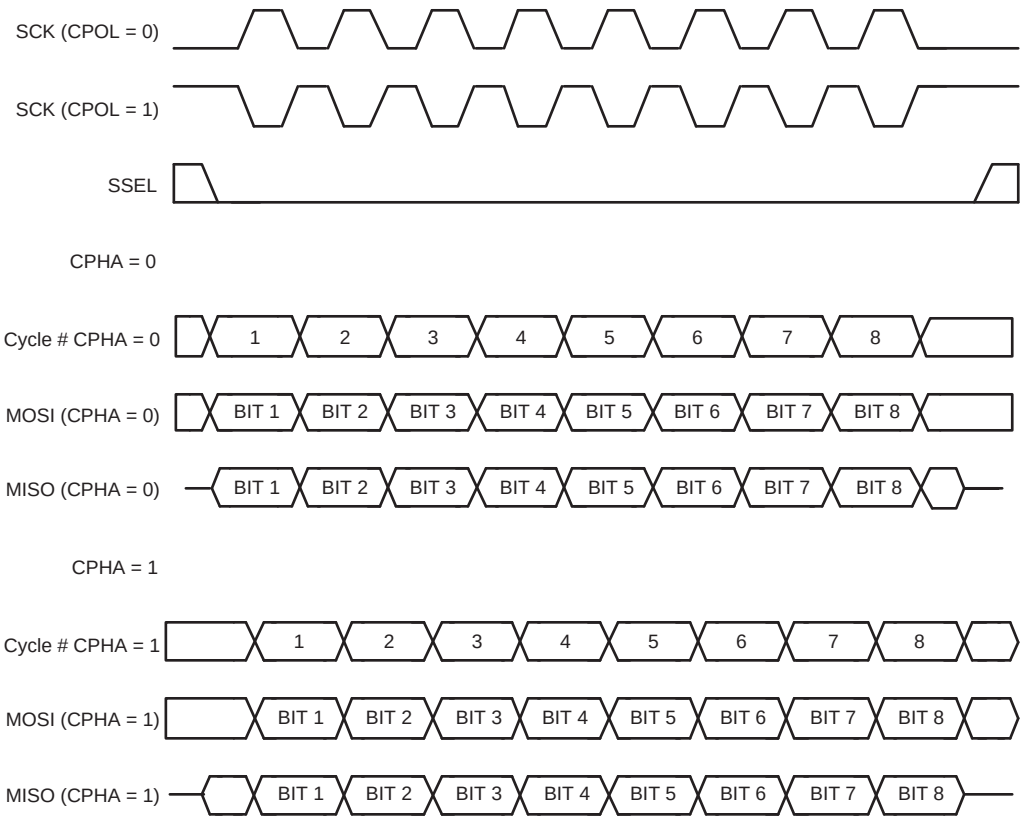


Fig 73. SPI data transfer format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in Table 17–360.

Table 360. SPI Data To Clock Phase Relationship

CPOL and CPHA settings	When the first data bit is driven	When all other data bits are driven	When data is 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 a 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.

© NXP B.V. 2010. All rights reserved.

6. SPI peripheral details

6.1 General information

There are five control and status registers for the SPI port. They are described in detail in Section 17–7 “Register description” on page 404.

The SPI Control Register (S0SPCR) 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 (S0SPSR) 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 SPI Interrupt Flag (SPIF) in the S0SPINT register. The remaining bits in the register are exception condition indicators. These exceptions will be described later in this section.

The SPI Data Register (S0SPDR) 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 (S0SPCCR) 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.

Prior to use, SPI configurations such as the master/slave settings, clock polarity, clock rate, etc. must be set up in the SPI Control Register and SPI Clock Counter Register.

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.

6.2 Master operation

The following sequence can be followed to set up the SPI prior to its first use as a master. This is typically done during program initialization.

1. Set the SPI Clock Counter Register to the desired clock rate.
2. Set the SPI Control Register to the desired settings for master mode.

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. Optionally, verify the SPI setup before starting the transfer.
2. Write the data to transmitted to the SPI Data Register. This write starts the SPI data transfer.

© NXP B.V. 2010. All rights reserved.

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 cycle 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 to be transmitted.

Note: A read or write of the SPI Data Register is required in order to clear the SPIF status bit. Therefore, if the optional read of the SPI Data Register does not take place, a write to this register is required in order to clear the SPIF status bit.

6.3 Slave operation

The following sequence can be followed to set up the SPI prior to its first use as a slave. This is typically done during program initialization.

1. Set the SPI Control Register to the desired settings for slave mode.

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. Optionally, verify the SPI setup before starting the transfer.
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 to be transferred.

Note: A read or write of the SPI Data Register is required in order to clear the SPIF status bit. Therefore, at least one of the optional reads or writes of the SPI Data Register must take place, in order to clear the SPIF status bit.

6.4 Exception conditions

Read Overrun

A read overrun occurs when the SPI block internal read buffer contains data that has not been read by the processor, and a new transfer has completed. The read buffer containing valid data is indicated by the SPIF bit in the SPI Interrupt 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 SPI Status Register will be activated.

Write Collision

As stated previously, there is no write buffer between the SPI block bus interface, and the internal shift register. As a result, data must not be written to the SPI Data Register when a SPI data transfer is currently in progress. The time frame where data cannot be written to the SPI Data Register is from when the transfer starts, until after the SPI Status

© NXP B.V. 2010. All rights reserved.

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 SPI Status Register will be activated.

Mode Fault

If the SSEL signal goes active when the SPI block is a master, this indicates another master has selected the device to be a slave. This condition is known as a mode fault. When a mode fault is detected, the mode fault (MODF) bit in the SPI Status Register will be activated, the SPI signal drivers will be de-activated, and the SPI mode will be changed to be a slave.

If the SSEL function is assigned to its related pin in the relevant Pin Function Select Register, the SSEL signal must always be inactive when the SPI controller is a master.

Slave Abort

A slave transfer is considered to be aborted if the SSEL signal goes inactive before the transfer is complete. In the event of a slave abort, the transmit and receive data for the transfer that was in progress are lost, and the slave abort (ABRT) bit in the SPI Status Register will be activated.

7. Register description

The SPI contains 5 registers as shown in Table 17–361. All registers are byte, half word and word accessible.

Table 361. 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	0x4002 0000
S0SPSR	SPI Status Register. This register shows the status of the SPI.	RO	0x00	0x4002 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	0x4002 0008
S0SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK0.	R/W	0x00	0x4002 000C
S0SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	R/W	0x00	0x4002 001C

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

7.1 SPI Control Register (S0SPCR - 0x4002 0000)

The S0SPCR register controls the operation of SPI0 as per the configuration bits setting shown in Table 17–362.

© NXP B.V. 2010. All rights reserved.

Table 362: SPI Control Register (S0SPCR - address 0x4002 0000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	BitEnable	0	The SPI controller sends and receives 8 bits of data per transfer.	0
		1	The SPI controller sends and receives the number of bits selected by bits 11:8.	
3	CPHA		Clock phase control determines the relationship between the data and the clock on SPI transfers, and controls when a slave transfer is defined as starting and ending.	0
		0	Data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	
		1	Data is sampled on the second clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active.	
4	CPOL		Clock polarity control.	0
		0	SCK is active high.	
		1	SCK is active low.	
5	MSTR		Master mode select.	0
		0	The SPI operates in Slave mode.	
		1	The SPI operates in Master mode.	
6	LSBF		LSB First controls which direction each byte is shifted when transferred.	0
		0	SPI data is transferred MSB (bit 7) first.	
		1	SPI data is transferred LSB (bit 0) first.	
7	SPIE		Serial peripheral interrupt enable.	0
		0	SPI interrupts are inhibited.	
		1	A hardware interrupt is generated each time the SPIF or MODF bits are activated.	
11:8	BITS		When bit 2 of this register is 1, this field controls the number of bits per transfer:	0000
		1000	8 bits per transfer	
		1001	9 bits per transfer	
		1010	10 bits per transfer	
		1011	11 bits per transfer	
		1100	12 bits per transfer	
		1101	13 bits per transfer	
		1110	14 bits per transfer	
		1111	15 bits per transfer	
		0000	16 bits per transfer	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

7.2 SPI Status Register (S0SPSR - 0x4002 0004)

The S0SPSR register controls the operation of SPI0 as per the configuration bits setting shown in Table 17–363.

Table 363: SPI Status Register (S0SPSR - address 0x4002 0004) bit description

Bit	Symbol	Description	Reset Value
2:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort. When 1, this bit indicates that a slave abort has occurred. This bit is cleared by reading this register.	0
4	MODF	Mode fault. when 1, this bit indicates that a Mode fault error has occurred. This bit is cleared by reading this register, then writing the SPI0 control register.	0
5	ROVR	Read overrun. When 1, this bit indicates that a read overrun has occurred. This bit is cleared by reading this register.	0
6	WCOL	Write collision. When 1, this bit indicates that a write collision has occurred. This bit is cleared by reading this register, then accessing the SPI Data Register.	0
7	SPIF	SPI transfer complete flag. When 1, this bit indicates when a SPI data transfer is complete. When a master, this bit is set at the end of the last cycle of the transfer. When a slave, this bit is set on the last data sampling edge of the SCK. This bit is cleared by first reading this register, then accessing the SPI Data Register. **Note:** this is not the SPI interrupt flag. This flag is found in the SPINT register.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.3 SPI Data Register (S0SPDR - 0x4002 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 used as a master, a write to this register will start an SPI data transfer. Writes to this register will be blocked when a data transfer starts, or when the SPIF status bit is set, and the SPI Status Register has not been read.

Table 364: SPI Data Register (S0SPDR - address 0x4002 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
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.4 SPI Clock Counter Register (S0SPCCR - 0x4002 000C)

This register controls the frequency of a master's SCK. The register indicates the number of SPI peripheral clock cycles that make up an SPI clock.

© NXP B.V. 2010. All rights reserved.

In Master mode, this register must be an even number greater than or equal to 8. Violations of this can result in unpredictable behavior. The SPI0 SCK rate may be calculated as: PCLK_SPI / SPCCR0 value. The SPI peripheral clock is determined by the PCLKSEL0 register contents for PCLK_SPI as described in Section 4–7.3.

In Slave mode, the SPI clock rate provided by the master must not exceed 1/8 of the SPI peripheral clock selected in Section 4–7.3. The content of the S0SPCCR register is not relevant.

Table 365: SPI Clock Counter Register (S0SPCCR - address 0x4002 000C) bit description

Bit	Symbol	Description	Reset Value
7:0	Counter	SPI0 Clock counter setting.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.5 SPI Test Control Register (SPTCR - 0x4002 0010)

Note that the bits in this register are intended for functional verification only. This register should not be used for normal operation.

Table 366: SPI Test Control Register (SPTCR - address 0x4002 0010) bit description

Bit	Symbol	Description	Reset Value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:1	Test	SPI test mode. When 0, the SPI operates normally. When 1, SCK will always be on, independent of master mode select, and data availability setting.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.6 SPI Test Status Register (SPTSR - 0x4002 0014)

Note: The bits in this register are intended for functional verification only. This register should not be used for normal operation.

This register is a replication of the SPI Status Register. The difference between the registers is that a read of this register will not start the sequence of events required to clear these status bits. A write to this register will set an interrupt if the write data for the respective bit is a 1.

Table 367: SPI Test Status Register (SPTSR - address 0x4002 0014) bit description

Bit	Symbol	Description	Reset Value
2:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort.	0
4	MODF	Mode fault.	0
5	ROVR	Read overrun.	0

© NXP B.V. 2010. All rights reserved.

Table 367: SPI Test Status Register (SPTSR - address 0x4002 0014) bit description

Bit	Symbol	Description	Reset Value
6	WCOL	Write collision.	0
7	SPIF	SPI transfer complete flag.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

7.7 SPI Interrupt Register (S0SPINT - 0x4002 001C)

This register contains the interrupt flag for the SPI0 interface.

Table 368: SPI Interrupt Register (S0SPINT - address 0x4002 001C) bit description

Bit	Symbol	Description	Reset Value
0	SPIF	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 NVIC, 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
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

8. Architecture

The block diagram of the SPI solution implemented in SPI0 interface is shown in the Figure 17–74.

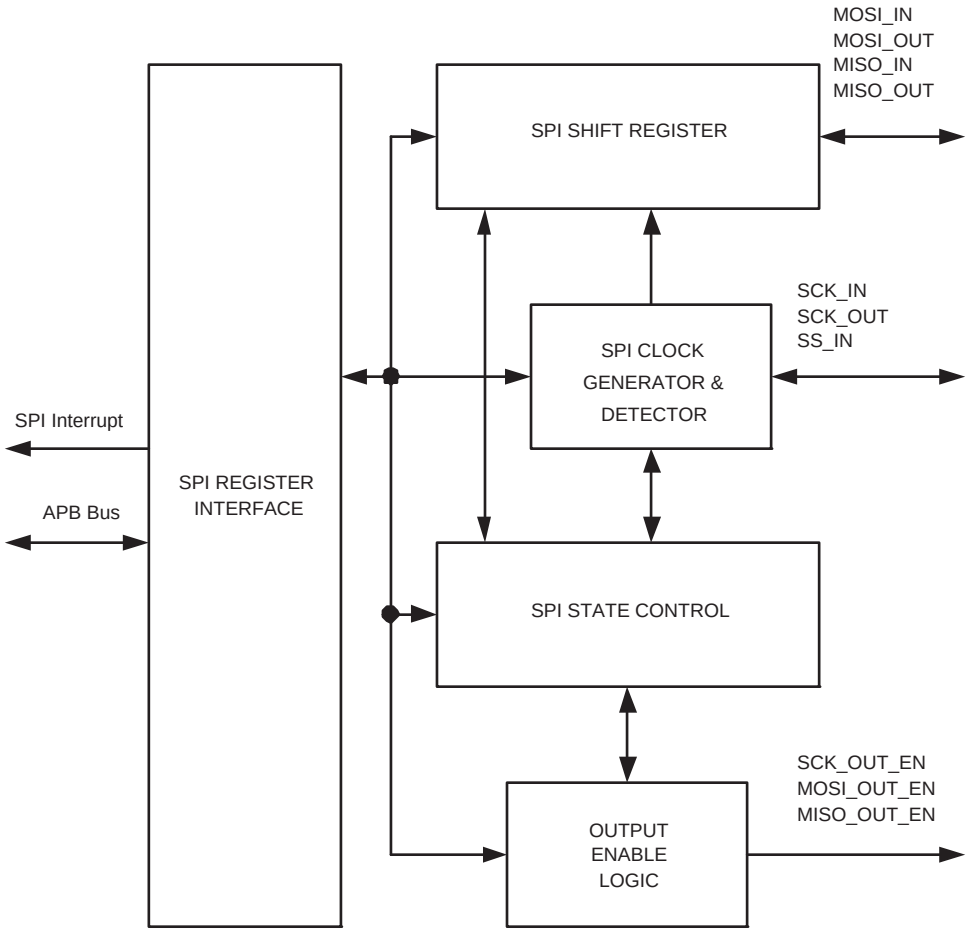


Fig 74. SPI block diagram

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The two SSP interfaces, SSP0 and SSP1 are configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCSSP0 to enable SSP0 and bit PCSSP1 to enable SSP1.

   Remark: On reset, both SSP interfaces are enabled (PCSSP0/1 = 1).
2. Clock: In PCLKSEL0 select PCLK_SSP1; in PCLKSEL1 select PCLK_SSP0 (see Section 4–7.3. In master mode, the clock must be scaled down (see Section 18–6.5).
3. Pins: Select the SSP pins through the PINSEL registers (Section 8–5) and pin modes through the PINMODE registers (Section 8–4).
4. Interrupts: Interrupts are enabled in the SSP0IMSC register for SSP0 and SSP1IMSC register for SSP1 Table 18–376. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register, see Table 6–50.
5. Initialization: There are two control registers for each of the SSP ports to be configured: SSP0CR0 and SSP0CR1 for SSP0, SSP1CR0 and SSP1CR1 for SSP1. See Section 18–6.1 and Section 18–6.2.
6. DMA: The Rx and Tx FIFOs of the SSP interfaces can be connected to the GPDMA controller (see Section 18–6.10). For GPDMA system connections, see Table 31–544.

Remark: SSP0 is intended to be used as an alternative for the SPI interface, which is included as a legacy peripheral. Only one of these peripherals can be used at the any one time.

2. Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses.
- Synchronous Serial Communication.
- Master or slave operation.
- 8 frame FIFOs for both transmit and receive.
- 4 to 16 bit data frame.
- DMA transfers supported by GPDMA.

3. Description

The SSP is a Synchronous Serial Port (SSP) controller capable of operation on a SPI, 4-wire SSI, or Microwire bus. It can interact with multiple masters and slaves on the bus. Only a single master and a single slave can communicate on the bus during a given data transfer. Data transfers are in principle full duplex, with frames of 4 to 16 bits of data flowing from the master to the slave and from the slave to the master. In practice it is often the case that only one of these data flows carries meaningful data.

© NXP B.V. 2010. All rights reserved.

The LPC17xx has two Synchronous Serial Port controllers -- SSP0 and SSP1.

4. Pin descriptions

Table 369. SSP pin descriptions

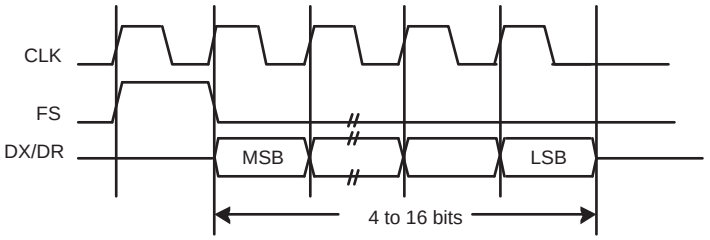
Pin Name	Type	Interface pin name/function			Pin Description
		SPI	SSI	Microwire	
SCK0/1	I/O	SCK	CLK	SK	**Serial Clock.** SCK/CLK/SK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low, otherwise it is always active-high. SCK1 only switches during a data transfer. Any other time, the SSPn interface either holds it in its inactive state, or does not drive it (leaves it in high-impedance state).
SSEL0/1	I/O	SSEL	FS	CS	**Frame Sync/Slave Select.** When the SSPn interface is a bus master, it drives this signal to an active state before the start of serial data, and then releases it to an inactive state after the serial data has been sent. The active state of this signal can be high or low depending upon the selected bus and mode. When the SSPn is a bus slave, this signal qualifies the presence of data from the Master, according to the protocol in use. When there is just one bus master and one bus slave, the Frame Sync or Slave Select signal from the Master can be connected directly to the slave's corresponding input. When there is more than one slave on the bus, further qualification of their Frame Select/Slave Select inputs will typically be necessary to prevent more than one slave from responding to a transfer.
MISO0/1	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**Master In Slave Out.** The MISO signal transfers serial data from the slave to the master. When the SSPn is a slave, serial data is output on this signal. When the SSPn is a master, it clocks in serial data from this signal. When the SSPn is a slave and is not selected by FS/SSEL, it does not drive this signal (leaves it in high-impedance state).
MOSI0/1	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**Master Out Slave In.** The MOSI signal transfers serial data from the master to the slave. When the SSPn is a master, it outputs serial data on this signal. When the SSPn is a slave, it clocks in serial data from this signal.

5. Bus description

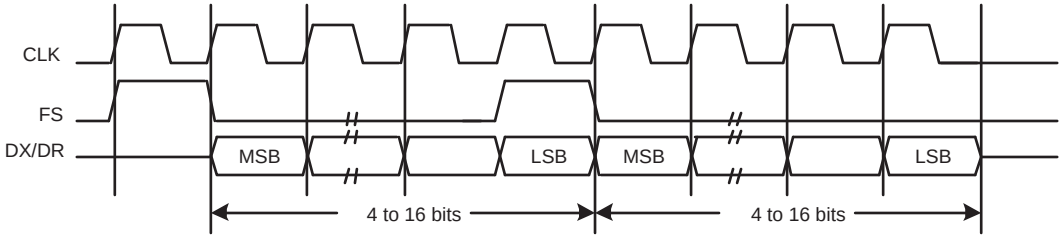
5.1 Texas Instruments synchronous serial frame format

Figure 18–75 shows the 4-wire Texas Instruments synchronous serial frame format supported by the SSP module.

© NXP B.V. 2010. All rights reserved.



a. Single frame transfer



b. Continuous/back-to-back frames transfer

Fig 75. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer

For device configured as a master in this mode, CLK and FS are forced LOW, and the transmit data line DX is tri-stated whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, FS is pulsed HIGH for one CLK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of CLK, the MSB of the 4-bit to 16-bit data frame is shifted out on the DX pin. Likewise, the MSB of the received data is shifted onto the DR pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each CLK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of CLK after the LSB has been latched.

5.2 SPI frame format

The SPI interface is a four-wire interface where the SSEL signal behaves as a slave select. The main feature of the SPI format is that the inactive state and phase of the SCK signal are programmable through the CPOL and CPHA bits within the SSPCR0 control register.

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

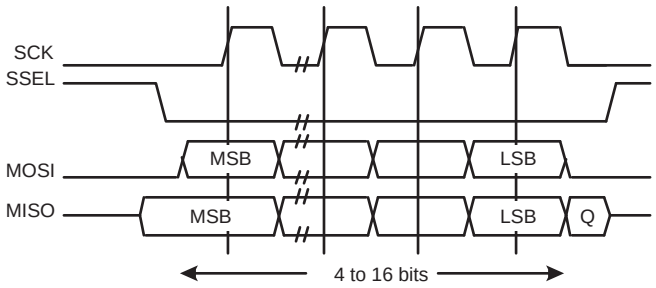
When the CPOL clock polarity control bit is 0, it produces a steady state low value on the SCK pin. If the CPOL clock polarity control bit is 1, a steady state high value is placed on the CLK pin when data is not being transferred.

© NXP B.V. 2010. All rights reserved.

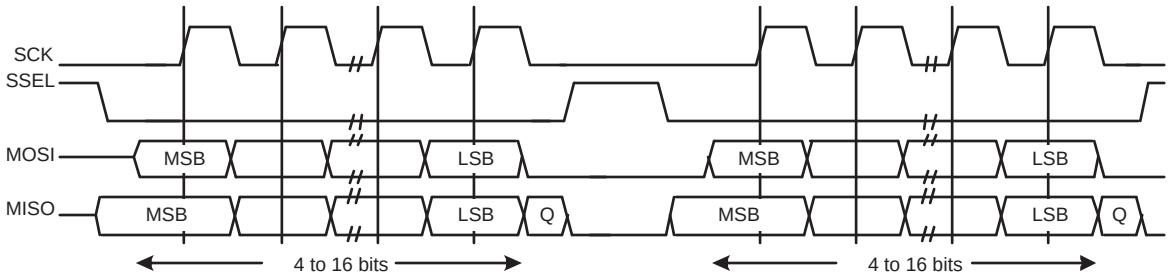
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is 0, data is captured on the first clock edge transition. If the CPHA clock phase control bit is 1, data is captured on the second clock edge transition.

5.2.2 SPI format with CPOL=0,CPHA=0

Single and continuous transmission signal sequences for SPI format with CPOL = 0, CPHA = 0 are shown in Figure 18–76.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 76. 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.

© NXP B.V. 2010. All rights reserved.

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.

5.2.3 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 18–77, which covers both single and continuous transfers.

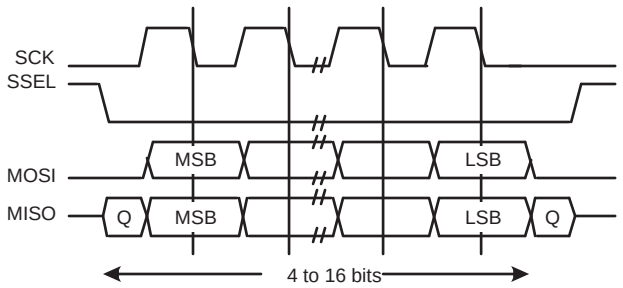


Fig 77. SPI frame format with CPOL=0 and CPHA=1

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.

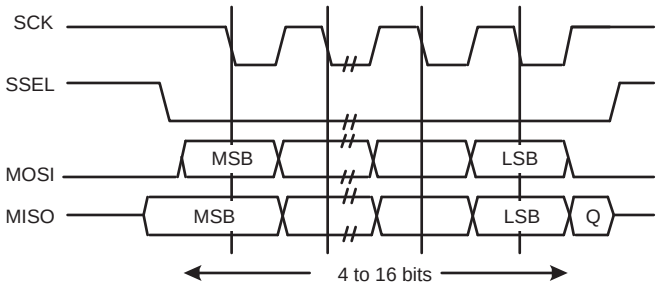
In the case of a single word transfer, after all bits have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

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.

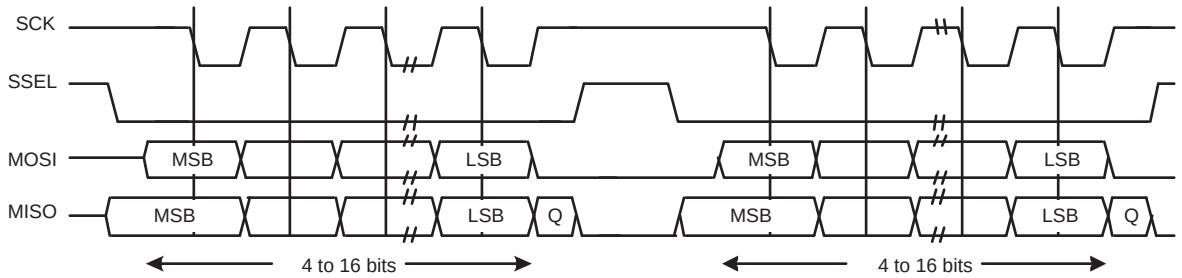
5.2.4 SPI format with CPOL = 1,CPHA = 0

Single and continuous transmission signal sequences for SPI format with CPOL=1, CPHA=0 are shown in Figure 18–78.

© NXP B.V. 2010. All rights reserved.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 78. 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.

© NXP B.V. 2010. All rights reserved.

5.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 18–79, which covers both single and continuous transfers.

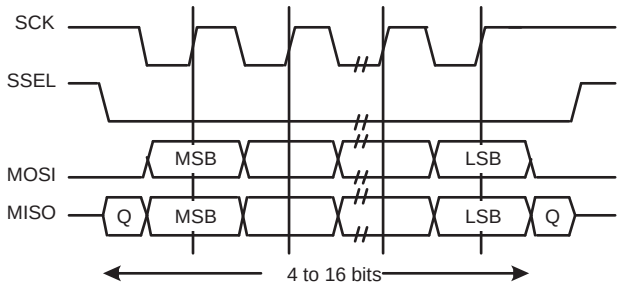


Fig 79. SPI Frame Format with CPOL = 1 and CPHA = 1

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.

5.3 National Semiconductor Microwire frame format

Figure 18–80 shows the Microwire frame format for a single frame. Figure 18–81 shows the same format when back-to-back frames are transmitted.

© NXP B.V. 2010. All rights reserved.

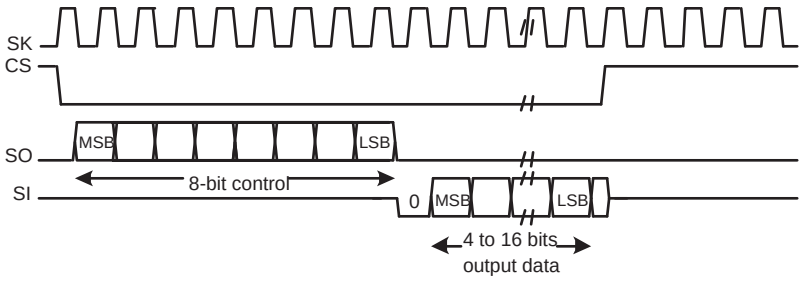


Fig 80. Microwire frame format (single transfer)

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmission, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- The SK signal is forced LOW.
- CS is forced HIGH.
- The transmit data line SO is arbitrarily forced LOW.

A transmission is triggered by writing a control byte to the transmit FIFO.The falling edge of CS causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SO pin. CS remains LOW for the duration of the frame transmission. The SI pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SK. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSP. Each bit is driven onto SI line on the falling edge of SK. The SSP in turn latches each bit on the rising edge of SK. At the end of the frame, for single transfers, the CS signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of SK after the LSB has been latched by the receive shiftier, or when the CS pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the CS line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SK, after the LSB of the frame has been latched into the SSP.

© NXP B.V. 2010. All rights reserved.

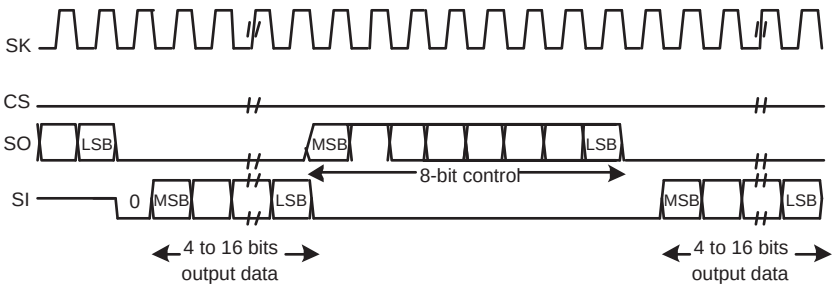


Fig 81. Microwire frame format (continuos transfers)

5.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode

In the Microwire mode, the SSP slave samples the first bit of receive data on the rising edge of SK after CS has gone LOW. Masters that drive a free-running SK must ensure that the CS signal has sufficient setup and hold margins with respect to the rising edge of SK.

Figure 18–82 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.

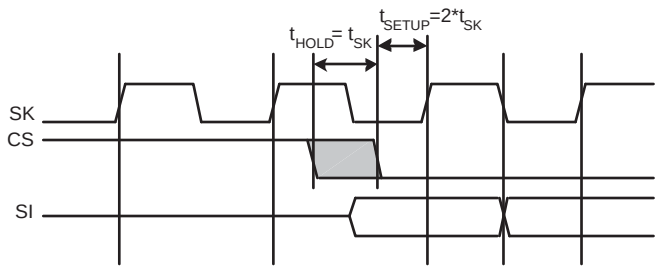


Fig 82. Microwire frame format setup and hold details

© NXP B.V. 2010. All rights reserved.

6. Register description

The register addresses of the SSP controllers addresses are shown in Table 18–370.

Table 370. SSP Register Map

Generic Name	Description	Access	Reset Value[1]	SSPn Register Name & Address
CR0	Control Register 0. Selects the serial clock rate, bus type, and data size.	R/W	0	SSP0CR0 - 0x4008 8000 SSP1CR0 - 0x4003 0000
CR1	Control Register 1. Selects master/slave and other modes.	R/W	0	SSP0CR1 - 0x4008 8004 SSP1CR1 - 0x4003 0004
DR	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	R/W	0	SSP0DR - 0x4008 8008 SSP1DR - 0x4003 0008
SR	Status Register	RO		SSP0SR - 0x4008 800C SSP1SR - 0x4003 000C
CPSR	Clock Prescale Register	R/W	0	SSP0CPSR - 0x4008 8010 SSP1CPSR - 0x4003 0010
IMSC	Interrupt Mask Set and Clear Register	R/W	0	SSP0IMSC - 0x4008 8014 SSP1IMSC - 0x4003 0014
RIS	Raw Interrupt Status Register	R/W		SSP0RIS - 0x4008 8018 SSP1RIS - 0x4003 0018
MIS	Masked Interrupt Status Register	R/W	0	SSP0MIS - 0x4008 801C SSP1MIS - 0x4003 001C
ICR	SSPICR Interrupt Clear Register	R/W	NA	SSP0ICR - 0x4008 8020 SSP1ICR - 0x4003 0020
DMACR	DMA Control Register	R/W	0	SSP0DMACR - 0x4008 8024 SSP1DMACR - 0x4003 0024

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

6.1 SSPn Control Register 0 (SSP0CR0 - 0x4008 8000, SSP1CR0 - 0x4003 0000)

This register controls the basic operation of the SSP controller.

© NXP B.V. 2010. All rights reserved.

Table 371: SSPn Control Register 0 (SSP0CR0 - address 0x4008 8000, SSP1CR0 - 0x4003 0000) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0011	4-bit transfer	
		0100	5-bit transfer	
		0101	6-bit transfer	
		0110	7-bit transfer	
		0111	8-bit transfer	
		1000	9-bit transfer	
		1001	10-bit transfer	
		1010	11-bit transfer	
		1011	12-bit transfer	
		1100	13-bit transfer	
		1101	14-bit transfer	
		1110	15-bit transfer	
		1111	16-bit transfer	
5:4	FRF		Frame Format.	00
		00	SPI	
		01	TI	
		10	Microwire	
		11	This combination is not supported and should not be used.	
6	CPOL		Clock Out Polarity. This bit is only used in SPI mode.	0
		0	SSP controller maintains the bus clock low between frames.	
		1	SSP controller maintains the bus clock high between frames.	
7	CPHA		Clock Out Phase. This bit is only used in SPI mode.	0
		0	SSP controller captures serial data on the first clock transition of the frame, that is, the transition **away from** the inter-frame state of the clock line.	
		1	SSP controller captures serial data on the second clock transition of the frame, that is, the transition **back to** the inter-frame state of the clock line.	
15:8	SCR		Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR $\times$ [SCR+1]).	0x00
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2 SSPn Control Register 1 (SSP0CR1 - 0x4008 8004, SSP1CR1 - 0x4003 0004)

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

© NXP B.V. 2010. All rights reserved.

Table 372: SSPn Control Register 1 (SSP0CR1 - address 0x4008 8004, SSP1CR1 - 0x4003 0004) bit description

Bit	Symbol	Value	Description	Reset Value
0	LBM		Loop Back Mode.	0
		0	During normal operation.	
		1	Serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively).	
1	SSE		SSP Enable.	0
		0	The SSP controller is disabled.	
		1	The SSP controller will interact with other devices on the serial bus. Software should write the appropriate control information to the other SSP registers and interrupt controller registers, before setting this bit.	
2	MS		Master/Slave Mode.This bit can only be written when the SSE bit is 0.	0
		0	The SSP controller acts as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line.	
		1	The SSP controller acts as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines.	
3	SOD		Slave Output Disable. This bit is relevant only in slave mode (MS = 1). If it is 1, this blocks this SSP controller from driving the transmit data line (MISO).	0
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.3 SSPn Data Register (SSP0DR - 0x4008 8008, SSP1DR - 0x4003 0008)

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

Table 373: SSPn Data Register (SSP0DR - address 0x4008 8008, SSP1DR - 0x4003 0008) bit description

Bit	Symbol	Description	Reset Value
15:0	DATA	**Write:** software can write data to be sent in a future frame to this register whenever the TNF bit in the Status register is 1, indicating that the Tx FIFO is not full. If the Tx FIFO was previously empty and the SSP controller is not busy on the bus, transmission of the data will begin immediately. Otherwise the data written to this register will be sent as soon as all previous data has been sent (and received). If the data length is less than 16 bits, software must right-justify the data written to this register. **Read:** software can read data from this register whenever the RNE bit in the Status register is 1, indicating that the Rx FIFO is not empty. When software reads this register, the SSP controller returns data from the least recent frame in the Rx FIFO. If the data length is less than 16 bits, the data is right-justified in this field with higher order bits filled with 0s.	0x0000
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6.4 SSPn Status Register (SSP0SR - 0x4008 800C, SSP1SR - 0x4003 000C)

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

Table 374: SSPn Status Register (SSP0SR - address 0x4008 800C, SSP1SR - 0x4003 000C) bit description

Bit	Symbol	Description	Reset Value
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
4	BSY	Busy. This bit is 0 if the SSPn controller is idle, or 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.5 SSPn Clock Prescale Register (SSP0CPSR - 0x4008 8010, SSP1CPSR - 0x4003 0010)

This register controls the factor by which the Prescaler divides the SSP peripheral clock SSP_PCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in SSPnCR0, to determine the bit clock.

Table 375: SSPn Clock Prescale Register (SSP0CPSR - address 0x4008 8010, SSP1CPSR - 0x4003 0010) bit description

Bit	Symbol	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which SSP_PCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Important: the SSPnCPSR value must be properly initialized or the SSP controller will not be able to transmit data correctly.

In Slave mode, the SSP clock rate provided by the master must not exceed 1/12 of the SSP peripheral clock selected in Section 4–7.3. The content of the SSPnCPSR register is not relevant.

In master mode, $CPSDVSR_{min}$ = 2 or larger (even numbers only).

6.6 SSPn Interrupt Mask Set/Clear Register (SSP0IMSC - 0x4008 8014, SSP1IMSC - 0x4003 0014)

This register controls whether each of the four possible interrupt conditions in the SSP controller are enabled. Note that ARM uses the word “masked” in the opposite sense from classic computer terminology, in which “masked” meant “disabled”. ARM uses the word “masked” to mean “enabled”. To avoid confusion we will not use the word “masked”.

© NXP B.V. 2010. All rights reserved.

Table 376: SSPn Interrupt Mask Set/Clear register (SSP0IMSC - address 0x4008 8014, SSP1IMSC - 0x4003 0014) bit description

Bit	Symbol	Description	Reset Value
0	RORIM	Software should set this bit to enable interrupt when a Receive Overrun occurs, that is, when the Rx FIFO is full and another frame is completely received. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTIM	Software should set this bit to enable interrupt when a Receive Timeout condition occurs. A Receive Timeout occurs when the Rx FIFO is not empty, and no has not been read for a "timeout period".	0
2	RXIM	Software should set this bit to enable interrupt when the Rx FIFO is at least half full.	0
3	TXIM	Software should set this bit to enable interrupt when the Tx FIFO is at least half empty.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.7 SSPn Raw Interrupt Status Register (SSP0RIS - 0x4008 8018, SSP1RIS - 0x4003 0018)

This read-only register contains a 1 for each interrupt condition that is asserted, regardless of whether or not the interrupt is enabled in the SSPnIMSC.

Table 377: SSPn Raw Interrupt Status register (SSP0RIS - address 0x4008 8018, SSP1RIS - 0x4003 0018) bit description

Bit	Symbol	Description	Reset Value
0	RORRIS	This bit is 1 if another frame was completely received while the RxFIFO was full. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTRIS	This bit is 1 if the Rx FIFO is not empty, and has not been read for a "timeout period".	0
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.8 SSPn Masked Interrupt Status Register (SSP0MIS - 0x4008 801C, SSP1MIS - 0x4003 001C)

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

© NXP B.V. 2010. All rights reserved.

Table 378: SSPn Masked Interrupt Status register (SSPnMIS -address 0x4008 801C, SSP1MIS - 0x4003 001C) bit description

Bit	Symbol	Description	Reset Value
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0
1	RTMIS	This bit is 1 if the Rx FIFO is not empty, has not been read for a "timeout period", and this interrupt is enabled.	0
2	RXMIS	This bit is 1 if the Rx FIFO is at least half full, and this interrupt is enabled.	0
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, and this interrupt is enabled.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.9 SSPn Interrupt Clear Register (SSP0ICR - 0x4008 8020, SSP1ICR - 0x4003 0020)

Software can write one or more one(s) to this write-only register, to clear the corresponding interrupt condition(s) in the SSP controller. Note that the other two interrupt conditions can be cleared by writing or reading the appropriate FIFO, or disabled by clearing the corresponding bit in SSPnIMSC.

Table 379: SSPn interrupt Clear Register (SSP0ICR - address 0x4008 8020, SSP1ICR - 0x4003 0020) bit description

Bit	Symbol	Description	Reset Value
0	RORIC	Writing a 1 to this bit clears the “frame was received when RxFIFO was full” interrupt.	NA
1	RTIC	Writing a 1 to this bit clears the "Rx FIFO was not empty and has not been read for a timeout period" interrupt.	NA
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.10 SSPn DMA Control Register (SSP0DMACR - 0x4008 8024, SSP1DMACR - 0x4003 0024)

The SSPnDMACR register is the DMA control register. It is a read/write register.

Table 380: SSPn DMA Control Register (SSP0DMACR - address 0x4008 8024, SSP1DMACR - 0x4003 0024) bit description

Bit	Symbol	Description	Reset Value
0	Receive DMA Enable (RXDMAE)	When this bit is set to one 1, DMA for the receive FIFO is enabled, otherwise receive DMA is disabled.	0
1	Transmit DMA Enable (TXDMAE)	When this bit is set to one 1, DMA for the transmit FIFO is enabled, otherwise transmit DMA is disabled	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The I^2C0/1/2 interfaces are configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCI2C0/1/2.

   Remark: On reset, all I^2C interfaces are enabled (PCI2C0/1/2 = 1).

2. Clock: In PCLKSEL0 select PCLK_I2C0; in PCLKSEL1 select PCLK_I2C1 or PCLK_I2C2 (see Section 4–7.3).

3. Pins: Select I^2C0, I^2C1, or I^2C2 pins through the PINSEL registers. Select the pin modes for the port pins with I^2C1 or I^2C2 functions through the PINMODE registers (no pull-up, no pull-down resistors) and the PINMODE_OD registers (open drain) (See Section 8–5).

   Remark: I^2C0 pins SDA0 and SCL0 are open-drain outputs and fully I^2C-bus compliant (see Table 7–72). I^2C0 can be further configured through the I2CPADCFG register to support Fast Mode Plus (See Table 8–98).

   Remark: I^2C0 is not available in the 80-pin package.

   Remark: I^2C1 and I^2C2 pins are not fully I^2C-bus compliant open-drain pins but can be configured to be open-drain via the PINMODE and PINMODE_OD registers. The non-compliance is in the I^2C-bus ability to turn off power to the device without pulling down the I^2C-bus itself.

4. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

5. Initialization: see Section 19–9.8.1 and Section 19–10.1.

2. Features

- Standard I^2C compliant bus interfaces may be configured as Master, Slave, or Master/Slave.
- Arbitration is handled between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock allows adjustment of I^2C transfer rates.
- Data transfer is bidirectional between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization is used as a handshake mechanism to suspend and resume serial transfer.
- Supports Fast Mode Plus (I^2C0 only).
- Optional recognition of up to 4 distinct slave addresses.
- Monitor mode allows observing all I^2C-bus traffic, regardless of slave address, without affecting the actual I^2C-bus traffic.
- The I^2C-bus can be used for test and diagnostic purposes.

© NXP B.V. 2010. All rights reserved.

- I^2C0 is a standard I^2C compliant bus interface with open-drain pins. This interface supports functions described in the I^2C specification for speeds up to 1 MHz. This includes multi-master operation and allows powering off this device in a working system while leaving the I^2C-bus functional.
- I^2C1 and I^2C2 use standard I/O pins and are intended for use with a single-master I^2C-bus and do not support powering off of this device while leaving the I^2C-bus functional, and do not support multi-master I^2C implementations.

3. Applications

Interfaces to external I^2C standard parts, such as serial RAMs, LCDs, tone generators, other microcontrollers, etc.

4. Description

A typical I^2C-bus configuration is shown in Figure 19–83. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^2C-bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte, unless the slave device is unable to accept more data.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I^2C-bus will not be released.

The LPC17xx I^2C interfaces are byte oriented and have four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

I^2C0 complies with the entire I^2C specification, supporting the ability to have the LPC17xx powered off and not interfere with other powered devices on the same I^2C-bus. I^2C1 and I^2C2 do not support the ability to have the power to the LPC17xx turned off without interfering with other powered devices on the same I^2C-bus.

Since I^2C1 and I^2C2 use standard port pins, internal pull-ups could (in theory) be enabled in order to pull I^2C-bus signals high when they are not driven low. However, these internal pull-ups are far weaker than what would normally be used for I^2C, so this practice is not recommended. Refer to the “I^2C-bus specification and user manual” for information on proper pull-up values for a specific case.

© NXP B.V. 2010. All rights reserved.

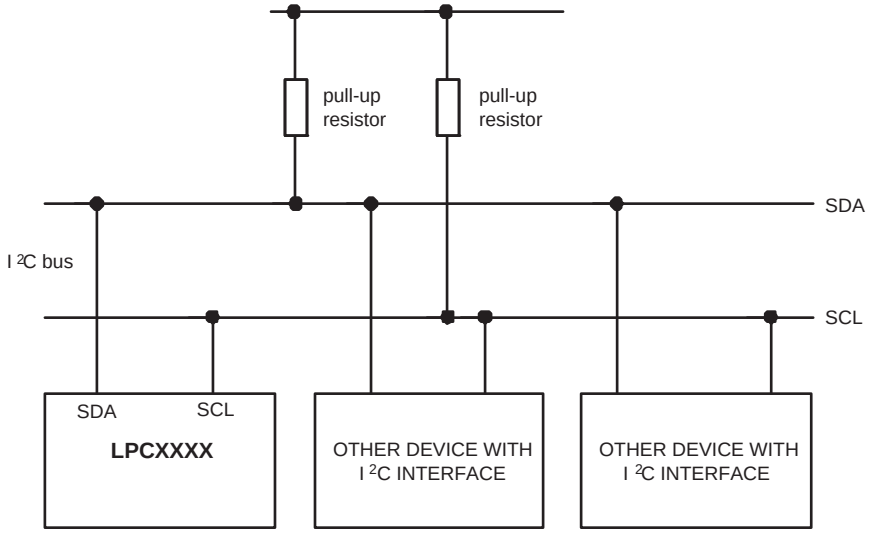


Fig 83. I²C-bus configuration

4.1 I²C FAST Mode Plus

Fast Mode Plus is a 1 Mbit/sec transfer rate to communicate with the I²C products which the NXP Semiconductors is now providing.

In order to use Fast Mode Plus, the I²C0 pins must be configured, then rates above 400 kHz and up to 1 Mhz may be selected, see Table 19–395. To configure the pins for Fast Mode Plus, the SDADRV0 and SCLDRV0 bits in the I2CPADCFG register must be set, see Section 8–5.21.

5. Pin description

Table 381. I²C Pin Description

Pin	Type	Description
SDA0[1]	Input/Output	I²C0 Serial Data
SCL0[1]	Input/Output	I²C0 Serial Clock
SDA1	Input/Output	I²C1 Serial Data
SCL1	Input/Output	I²C1 Serial Clock
SDA2	Input/Output	I²C2 Serial Data
SCL2	Input/Output	I²C2 Serial Clock

[1] I²C0 is only available in 100-pin LPC17xx devices. The SDA0 and SCL0 pins are open-drain pins to comply with I²C specifications. The pins must be configured in the I2CPADCFG register for Fast Mode Plus.

The three I²C interfaces are identical except for the pin I/O characteristics. I²C0 complies with the entire I²C specification, supporting the ability to turn power off to the device without causing a problem with other devices on the same I²C-bus (see "The I²C-bus specification" description under the heading "Fast Mode", and notes for the table titled "Characteristics of the SDA and SCL I/O stages for F/S-mode I²C-bus devices"). This is sometimes a useful capability, but intrinsically limits alternate uses for the same pins if the I²C interface is not used. Seldom is this capability needed on multiple I²C interfaces within

© NXP B.V. 2010. All rights reserved.

the same microcontroller. Therefore, I^2C1 and I^2C2 are implemented using standard port pins, and do not support the ability to turn power off to the device while leaving the I^2C-bus functioning between other devices. Standard I/Os also change I^2C-bus pull-up characteristics and do not support multi-master I^2C implementations. This difference should be considered during system design while assigning uses for the I^2C interfaces. The pins associated with I^2C1 and I^2C2 should be switched to the open drain mode when the pins are used for I^2C communications.

6. I^2C operating modes

In a given application, the I^2C block may operate as a master, a slave, or both. In the slave mode, the I^2C hardware looks for any one of its four slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I^2C block switches to the slave mode immediately and can detect any of its own configured slave addresses in the same serial transfer.

6.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 19–382. I2EN must be set to 1 to enable the I^2C function. If the AA bit is 0, the I^2C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI bit is cleared by writing 1 to the SIC bit in the I2CONCLR register. THe STA bit should be cleared after writing the slave address.

Table 382. I2C0CONSET and I2C1CONSET used to configure Master mode

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

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

The I^2C interface will enter master transmitter mode when software sets the STA bit. The I^2C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the I2STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register.

© NXP B.V. 2010. All rights reserved.

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 19–399 to Table 19–402.

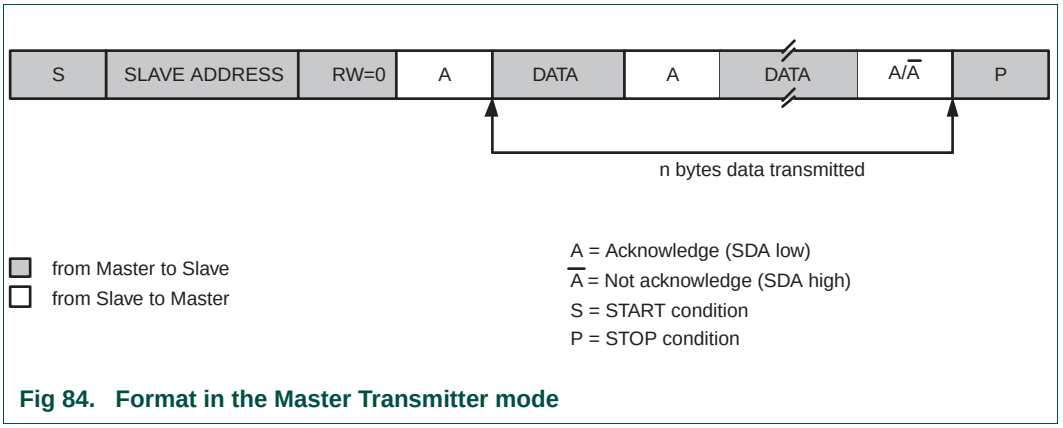


Fig 84. Format in the Master Transmitter mode

6.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^2C Data register (I2DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 19–400.

When the LPC17xx needs to acknowledge a received byte, the AA bit needs to be set accordingly prior to clearing the SI bit and initiating the byte read. When the LPC17xx needs to not acknowledge a received byte, the AA bit needs to be cleared prior to clearing the SI bit and initiating the byte read.

Note that the last received byte is always followed by a "Not Acknowledge" from the LPC17xx so that the master can signal the slave that the reading sequence is finished and that it needs to issue a STOP or repeated START Command. Once the "Not Acknowledge has been sent and the SI bit is set, the LPC17xx can send either a STOP (STO bit is set) or a repeated START (STA bit is set). Then the SI bit is cleared to initiate the requested operation.

© NXP B.V. 2010. All rights reserved.

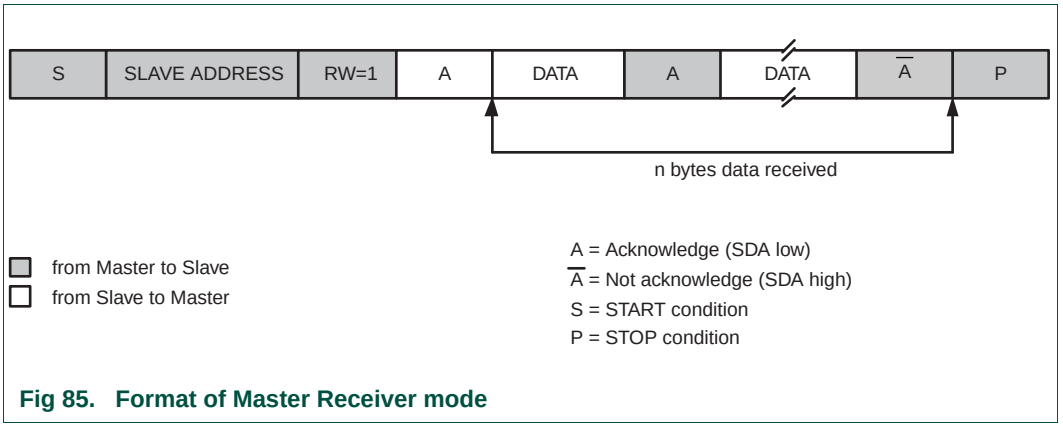


Fig 85. Format of Master Receiver mode

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

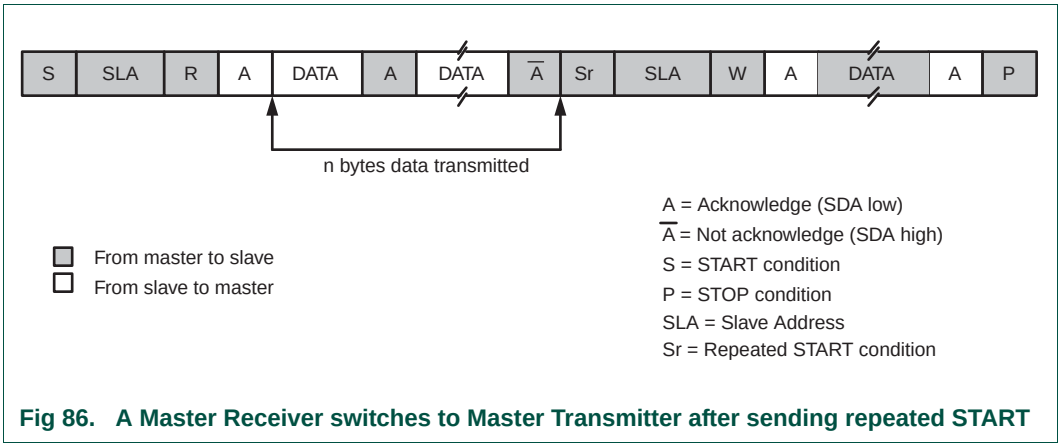


Fig 86. A Master Receiver switches to Master Transmitter after sending repeated START

6.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, write any of the Slave Address registers (I2ADR0-3) and Slave Mask registers (I2MASK0-3) and write the I²C Control Set register (I2CONSET) as shown in Table 19–383.

Table 383. 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 any of its own slave addresses or the General Call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I²C interface waits until it is addressed by its any of its own slave addresses or General Call address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it

© NXP B.V. 2010. All rights reserved.

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 19–401 for the status codes and actions.

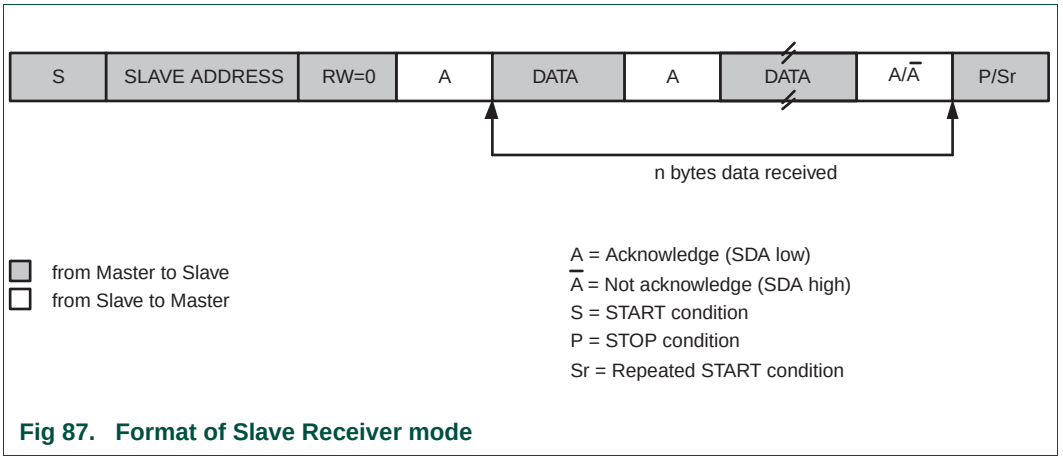


Fig 87. Format of Slave Receiver mode

6.4 Slave Transmitter mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for any of its own slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^2C interface switches to the slave mode immediately and can detect any of its own slave addresses in the same serial transfer.

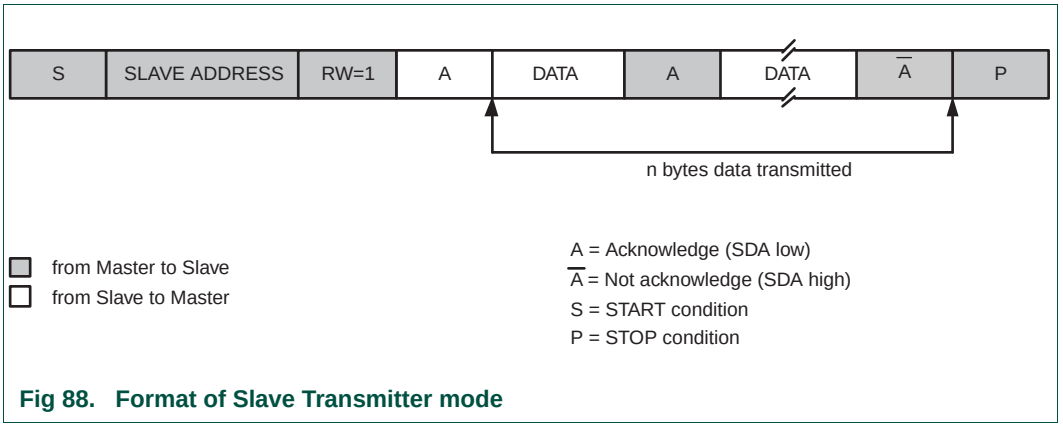


Fig 88. Format of Slave Transmitter mode

7. I^2C implementation and operation

Figure 19–89 shows how the on-chip I^2C-bus interface is implemented, and the following text describes the individual blocks.

© NXP B.V. 2010. All rights reserved.

7.1 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 I^2C is a special pad designed to conform to the I^2C specification.

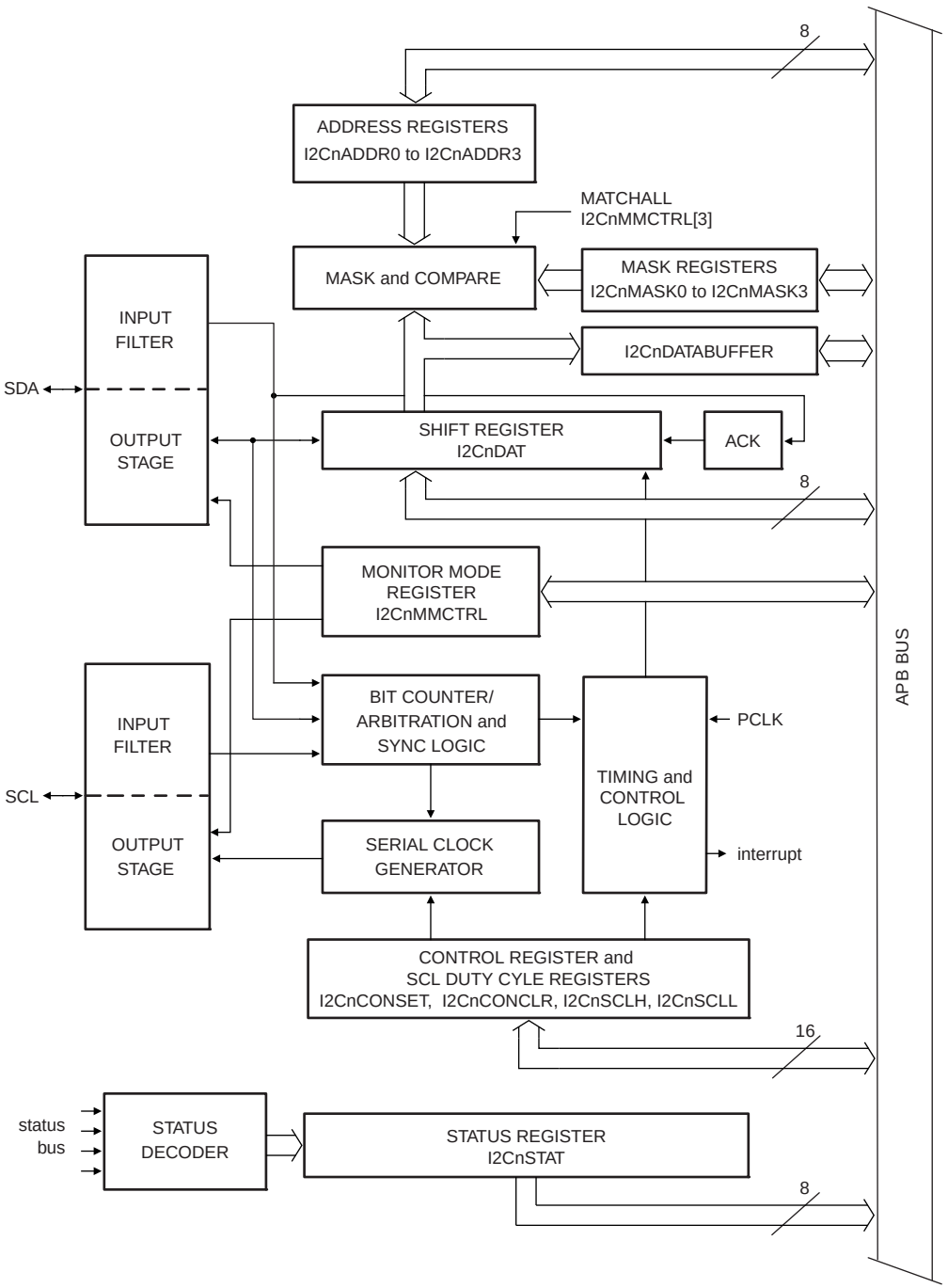


Fig 89. I^2C serial interface block diagram

© NXP B.V. 2010. All rights reserved.

7.2 Address Registers, I2ADR0 to I2ADR3

These registers may be loaded with the 7-bit slave address (7 most significant bits) to which the I^2C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable General Call address (0x00) recognition. When multiple slave addresses are enabled, the actual address received may be read from the I2DAT register at the state where the “own slave address” has just been received.

Remark: in the remainder of this chapter, when the phrase “own slave address” is used, it refers to any of the four configured slave addresses after address masking.

7.3 Address mask registers, I2MASK0 to I2MASK3

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADRn register associated with that mask register. In other words, bits in an I2ADRn register which are masked are not taken into account in determining an address match.

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine which received address actually caused the match.

7.4 Comparator

The comparator compares the received 7-bit slave address with any of the four configured slave addresses in I2ADR0 through I2ADR3 after masking. It also compares the first received 8-bit byte with the General Call address (0x00). If an a match is found, the appropriate status bits are set and an interrupt is requested.

7.5 Shift register, I2DAT

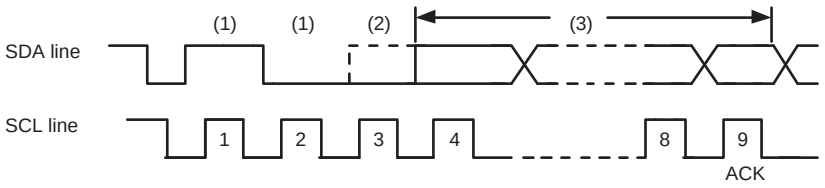
This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in I2DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in I2DAT.

7.6 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^2C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^2C block immediately changes from master transmitter to slave receiver. The I^2C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^2C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal low. Since this can occur only at the end of a serial byte, the I^2C block generates no further clock pulses. Figure 19–90 shows the arbitration procedure.

© NXP B.V. 2010. All rights reserved.



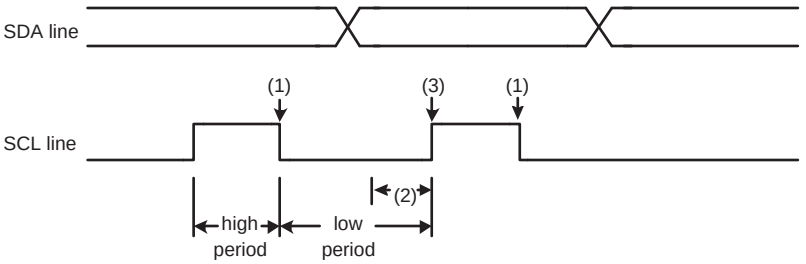
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I2C master by pulling the SDA line low. Arbitration is lost, and this I2C enters Slave Receiver mode.

(3) This I2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 90. 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 19–91 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I2C has timed a complete low time and released SCL. The I2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 91. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

7.7 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I2C block is in the master transmitter or master receiver mode. It is switched off when the I2C block is in a slave mode. The I2C output clock frequency and duty cycle is programmable

© NXP B.V. 2010. All rights reserved.

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.

7.8 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for I2DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I²C-bus status.

7.9 Control register, I2CONSET and I2CONCLR

The I²C control register contains bits used to control the following I²C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I²C control register may be read as I2CONSET. Writing to I2CONSET will set bits in the I²C control register that correspond to ones in the value written. Conversely, writing to I2CONCLR will clear bits in the I²C control register that correspond to ones in the value written.

7.10 Status decoder and status register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I²C-bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I²C block are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

© NXP B.V. 2010. All rights reserved.

8. Register description

Each I²C interface contains 16 registers as shown in Table 19–384 below.

Remark: In the LPC17xx, the following registers have been added to support response to multiple addresses in Slave mode and a new Monitor mode: I2ADR1 to 3, I2MASK0 To 3, MMCTRL, and I2DATA_BUFFER.

Table 384. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Name & Address
I2CONSET	**I²C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is set. Writing a zero has no effect on the corresponding bit in the I²C control register.	R/W	0x00	I2C0CONSET - 0x4001 C000 I2C1CONSET - 0x4005 C000 I2C2CONSET - 0x400A 0000
I2STAT	**I²C Status Register.** During I²C operation, this register provides detailed status codes that allow software to determine the next action needed.	RO	0xF8	I2C0STAT - 0x4001 C004 I2C1STAT - 0x4005 C004 I2C2STAT - 0x400A 0004
I2DAT	**I²C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	R/W	0x00	I2C0DAT - 0x4001 C008 I2C1DAT - 0x4005 C008 I2C2DAT - 0x400A 0008
I2ADR0	**I²C Slave Address Register 0.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR0 - 0x4001 C00C I2C1ADR0 - 0x4005 C00C I2C2ADR0 - 0x400A 000C
I2SCLH	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I²C clock.	R/W	0x04	I2C0SCLH - 0x4001 C010 I2C1SCLH - 0x4005 C010 I2C2SCLH - 0x400A 0010
I2SCLL	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I²C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I²C master and certain times used in slave mode.	R/W	0x04	I2C0SCLL - 0x4001 C014 I2C1SCLL - 0x4005 C014 I2C2SCLL - 0x400A 0014
I2CONCLR	**I²C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is cleared. Writing a zero has no effect on the corresponding bit in the I²C control register.	WO	NA	I2C0CONCLR - 0x4001 C018 I2C1CONCLR - 0x4005 C018 I2C2CONCLR - 0x400A 0018
MMCTRL	**Monitor mode control register.**	R/W	0x00	I2C0MMCTRL - 0x4001 C01C I2C1MMCTRL - 0x4005 C01C I2C2MMCTRL - 0x400A 001C
I2ADR1	**I²C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR1 - 0x4001 C020 I2C1ADR1 - 0x4005 C020 I2C2ADR1 - 0x400A 0020

© NXP B.V. 2010. All rights reserved.

Table 384. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Name & Address
I2ADR2	**I²C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR2 - 0x4001 C024 I2C1ADR2 - 0x4005 C024 I2C2ADR2 - 0x400A 0024
I2ADR3	**I²C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR3 - 0x4001 C028 I2C1ADR3 - 0x4005 C028 I2C2ADR3 - 0x400A 0028
I2DATA_ BUFFER	**Data buffer register.** The contents of the 8 MSBs of the I2DAT shift register will be transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus.	RO	0x00	I2C0DATA_ BUFFER - 0x4001 C02C I2C1DATA_ BUFFER - 0x4005 C02C I2C2DATA_ BUFFER - 0x400A 002C
I2MASK0	**I²C Slave address mask register 0**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK0 - 0x4001 C030 I2C1MASK0 - 0x4005 C030 I2C2MASK0 - 0x400A 0030
I2MASK1	**I²C Slave address mask register 1**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK1 - 0x4001 C034 I2C1MASK1 - 0x4005 C034 I2C2MASK1 - 0x400A 0034
I2MASK2	**I²C Slave address mask register 2**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK2 - 0x4001 C038 I2C1MASK2 - 0x4005 C038 I2C2MASK2 - 0x400A 0038
I2MASK3	**I²C Slave address mask register 3**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK3 - 0x4001 C03C I2C1MASK3 - 0x4005 C03C I2C2MASK3 - 0x400A 003C

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

8.1 I²C Control Set register (I2CONSET: I²C0, I2C0CONSET - 0x4001 C000; I²C1, I2C1CONSET - 0x4005 C000; I²C2, I2C2CONSET - 0x400A 0000)

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I²C interface. Writing a one to a bit of this register causes the corresponding bit in the I²C control register to be set. Writing a zero has no effect. Reading this register provides the current values of the control and flag bits.

© NXP B.V. 2010. All rights reserved.

Table 385. I²C Control Set register (I2CONSET: I²C0, I2C0CONSET - address 0x4001 C000, I²C1, I2C1CONSET - address 0x4005 C000, I²C2, I2C2CONSET - address 0x400A 0000) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag.	0
3	SI	I²C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I²C interface enable.	0
31:7	-	Reserved. 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.

© NXP B.V. 2010. All rights reserved.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I^2C Interrupt Flag. This bit is set when the I^2C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is HIGH, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in I2CONCLR register. The SI bit should be cleared only after the required bit(s) has (have) been set and the value in I2DAT has been loaded or read.

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A matching address defined by registers I2ADR0 through I2ADR3, masked by I2MASK0 though I2MASK3, has been received.
2. The General Call address has been received while the General Call bit (GC) in I2ADR is set.
3. A data byte has been received while the I^2C is in the master receiver mode.
4. A data byte has been received while the I^2C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (HIGH level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I^2C is in the master receiver mode.
2. A data byte has been received while the I^2C is in the addressed slave receiver mode.

8.2 I^2C Control Clear register (I2CONCLR: I^2C0, I2C0CONCLR - 0x4001 C018; I^2C1, I2C1CONCLR - 0x4005 C018; I^2C2, I2C2CONCLR - 0x400A 0018)

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I^2C interface. Writing a one to a bit of this register causes the corresponding bit in the I^2C control register to be cleared. Writing a zero has no effect. I2CONCLR is a write-only register. The value of the related bits can be read from the I2CONSET register.

Table 386. I^2C Control Clear register (I2CONCLR: I^2C0, I2C0CONCLR - 0x4001 C018; I^2C1, I2C1CONCLR - 0x4005 C018; I^2C2, I2C2CONCLR - 0x400A 0018) bit description

Bit	Symbol	Description
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.
2	AAC	Assert acknowledge Clear bit.
3	SIC	I^2C interrupt Clear bit.

© NXP B.V. 2010. All rights reserved.

Table 386. I²C Control Clear register (I2CONCLR: I²C0, I2C0CONCLR - 0x4001 C018; I²C1, I2C1CONCLR - 0x4005 C018; I²C2, I2C2CONCLR - 0x400A 0018) bit description

Bit	Symbol	Description
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.
5	STAC	START flag Clear bit.
6	I2ENC	I²C interface Disable bit.
31:7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.

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.

8.3 I²C Status register (I2STAT: I²C0, I2C0STAT - 0x4001 C004; I²C1, I2C1STAT - 0x4005 C004; I²C2, I2C2STAT - 0x400A 0004)

Each I²C Status register reflects the condition of the corresponding I²C interface. The I²C Status register is read-only.

Table 387. I²C Status register (I2STAT: I²C0, I2C0STAT - 0x4001 C004; I²C1, I2C1STAT - 0x4005 C004; I²C2, I2C2STAT - 0x400A 0004) bit description

Bit	Symbol	Description	Reset value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I²C interface.	0x1F
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I²C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 19–399 to Table 19–402.

8.4 I²C Data register (I2DAT: I²C0, I2C0DAT - 0x4001 C008; I²C1, I2C1DAT - 0x4005 C008; I²C2, I2C2DAT - 0x400A 0008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

© NXP B.V. 2010. All rights reserved.

Table 388. I²C Data register (I2DAT: I²C0, I2C0DAT - 0x4001 C008; I²C1, I2C1DAT - 0x4005 C008; I²C2, I2C2DAT - 0x400A 0008) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds data values that have been received or are to be transmitted.	0
31:8	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

8.5 I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C)

This register controls the Monitor mode which allows the I²C module to monitor traffic on the I²C-bus without actually participating in traffic or interfering with the I²C-bus.

Table 389. I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C) bit description

Bit	Symbol	Value	Description	Reset value
0	MM_ENA		Monitor mode enable.	0
		0	Monitor mode disabled.	
		1	The I²C module will enter monitor mode. In this mode the SDA output will be put in high impedance mode. This prevents the I²C module from outputting data of any kind (including ACK) onto the I²C data bus. Depending on the state of the ENA_SCL bit, the output may be also forced high, preventing the module from having control over the I²C clock line.	
1	ENA_SCL		SCL output enable.	0
		0	When this bit is cleared to '0', the SCL output will be forced high when the module is in monitor mode. As described above, this will prevent the module from having any control over the I²C clock line.	
		1	When this bit is set, the I²C module may exercise the same control over the clock line that it would in normal operation. This means that, acting as a slave peripheral, the I²C module can "stretch" the clock line (hold it low) until it has had time to respond to an I²C interrupt.[1]	

© NXP B.V. 2010. All rights reserved.

Table 389. I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C) bit description

Bit	Symbol	Value	Description	Reset value
2	MATCH_ALL		Select interrupt register match.	0
		0	When this bit is cleared, an interrupt will only be generated when a match occurs to one of the (up-to) four address registers, I2ADR0 through I2ADR3. That is, the module will respond as a normal slave as far as address-recognition is concerned.	
		1	When this bit is set to '1' and the I²C is in monitor mode, an interrupt will be generated on ANY address received. This will enable the part to monitor all traffic on the bus.	
31:3	-		Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] When the ENA_SCL bit is cleared and the I²C no longer has the ability to stretch the clock, interrupt response time becomes important. To give the part more time to respond to an I²C interrupt under these conditions, an I2DATA_BUFFER register is used (Section 19–8.6) to hold received data for a full 9-bit word transmission time.

Remark: The ENA_SCL and MATCH_ALL bits have no effect if the MM_ENA is '0' (i.e. if the module is NOT in monitor mode).

8.5.1 Interrupt in Monitor mode

All interrupts will occur as normal when the module is in monitor mode. This means that the first interrupt will occur when an address-match is detected (any address received if the MATCH_ALL bit is set, otherwise an address matching one of the four address registers).

Subsequent to an address-match detection, interrupts will be generated after each data byte is received for a slave-write transfer, or after each byte that the module believes it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

8.5.2 Loss of arbitration in Monitor mode

In monitor mode, the I²C module will not be able to respond to a request for information by the bus master or issue an ACK. Some other slave on the bus will respond instead.

Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected.

© NXP B.V. 2010. All rights reserved.

8.6 I²C Data buffer register (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER - 0x4001 C02C; I²C1, I2C1DATA_BUFFER- 0x4005 C02C; I²C2, I2C2DATA_BUFFER- 0x400A 002C)

In monitor mode, the I²C module may lose the ability to stretch the clock if the ENA_SCL bit is not set. This means that the processor will have a limited amount of time to read the contents of the data received on the bus. If the processor reads the I2DAT shift register, as it ordinarily would, it could have only one bit-time to respond to the interrupt before the received data is overwritten by new data.

To give the processor more time to respond, a new 8-bit, read-only I2DATA_BUFFER register will be added. The contents of the 8 MSBs of the I2DAT shift register will be transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus. This means that the processor will have 9 bit transmission times to respond to the interrupt and read the data before it is overwritten.

The processor will still have the ability to read I2DAT directly, as usual, and the behavior of I2DAT will not be altered in any way.

Although the I2DATA_BUFFER register is primarily intended for use in monitor mode with the ENA_SCL bit = ‘0’, it will be available for reading at any time under any mode of operation.

Table 390. I²C Data buffer register (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER - 0x4001 C02C; I²C1, I2C1DATA_BUFFER- 0x4005 C02C; I²C2, I2C2DATA_BUFFER- 0x400A 002C) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds contents of the 8 MSBs of the I2DAT shift register.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

8.7 I²C Slave Address registers (I2ADR0 to 3: I²C0, I2C0ADR[0, 1, 2, 3]- 0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3] - address 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3] - address 0x400A 00[0C, 20, 24, 28])

These registers are readable and writable and are only used when an I²C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

If these registers contain 0x00, the I²C will not acknowledge any address on the bus. All four registers will be cleared to this disabled state on reset.

Table 391. I²C Slave Address registers (I2ADR0 to 3: I²C0, I2C0ADR[0, 1, 2, 3]- 0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3] - address 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3] - address 0x400A 00[0C, 20, 24, 28]) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I²C device address for slave mode.	0x00
31:8	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

8.8 I²C Mask registers (I2MASK0 to 3: I²C0, I2C0MASK[0, 1, 2, 3] - 0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3] - address 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3] - address 0x400A 00[30, 34, 38, 3C])

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to '1' will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADRn register associated with that mask register. In other words, bits in an I2ADRn register which are masked are not taken into account in determining an address match.

The mask register has no effect on comparison to the General Call address ("0000000").

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine which received address actually caused the match.

Table 392. I²C Mask registers (I2MASK0 to 3: I²C0, I2C0MASK[0, 1, 2, 3] - 0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3] - address 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3] - address 0x400A 00[30, 34, 38, 3C]) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. User software should not write ones to reserved bits. This bit reads always back as 0.	0
7:1	MASK	Mask bits.	0x00
31:8	-	Reserved. User software should not write ones to reserved bits. These bits read always back as zeroes.	0

8.9 I²C SCL HIGH duty cycle register (I2SCLH: I²C0, I2C0SCLH - 0x4001 C010; I²C1, I2C1SCLH - 0x4005 C010; I²C2, I2C2SCLH - 0x400A 0010)

Table 393. I²C SCL HIGH Duty Cycle register (I2SCLH: I²C0, I2C0SCLH - address 0x4001 C010; I²C1, I2C1SCLH - address 0x4005 C010; I²C2, I2C2SCLH - 0x400A 0010) bit description

Bit	Symbol	Description	Reset value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	NA

8.10 I²C SCL Low duty cycle register (I2SCLL: I²C0 - I2C0SCLL: 0x4001 C014; I²C1 - I2C1SCLL: 0x4005 C014; I²C2 - I2C2SCLL: 0x400A 0014)

Table 394. I²C SCL Low duty cycle register (I2SCLL: I²C0 - I2C0SCLL: 0x4001 C014; I²C1 - I2C1SCLL: 0x4005 C014; I²C2 - I2C2SCLL: 0x400A 0014) bit description

Bit	Symbol	Description	Reset value
15:0	SCLL	Count for SCL low time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

8.11 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_I2C cycles for the SCL HIGH time, I2SCLL defines the number of PCLK_I2C cycles for the SCL low time. The frequency is determined by the following formula (PCLK_I2C is the frequency of the peripheral bus APB):

(11)

$$I^2C_{bitfrequency} = \frac{PCLKI2C}{I2CSCLH + I2CSCLL}$$

The values for I2SCLL and I2SCLH must ensure that the data rate is in the appropriate I²C data rate range. Each register value must be greater than or equal to 4. Table 19–395 gives some examples of I²C-bus rates based on PCLK_I2C frequency and I2SCLL and I2SCLH values.

Table 395. Example I²C clock rates

I²C Rate	I2SCLL + I2SCLH values at PCLK_I2C (MHz)													
	6	8	10	12	16	20	30	40	50	60	70	80	90	100
100 kHz (Standard)	60	80	100	120	160	200	300	400	500	600	700	800	900	1000
400 kHz (Fast Mode)	15	20	25	30	40	50	75	100	125	150	175	200	225	250
1 MHz (Fast Mode Plus)	-	8	10	12	16	20	30	40	50	60	70	80	90	100

I2SCLL and I2SCLH values should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I²C-bus specification defines the SCL low time and high time at different values for a Fast Mode and Fast Mode Plus I²C.

© NXP B.V. 2010. All rights reserved.

9. Details of I²C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figure 19–92, Figure 19–93, Figure 19–94, Figure 19–95, and Figure 19–96. Table 19–396 lists abbreviations used in these figures when describing the I²C operating modes.

Table 396. Abbreviations used to describe an I²C operation

Abbreviation	Explanation
S	START condition
SLA	7-bit slave address
R	Read bit (HIGH level at SDA)
W	Write bit (LOW level at SDA)
A	Acknowledge bit (LOW level at SDA)
$\overline{A}$	Not acknowledge bit (HIGH level at SDA)
Data	8-bit data byte
P	STOP condition
Sr	Repeated START condition

In Figure 19–92 to Figure 19–96, 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 19–399 to Table 19–403.

© NXP B.V. 2010. All rights reserved.

9.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 19–92). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 397. I2CONSET used to initialize Master Transmitter mode

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

The I^2C rate must also be configured in the I2SCLL and I2SCLH registers. I2EN must be set to logic 1 to enable the I^2C block. If the AA bit is reset, the I^2C block will not acknowledge its own slave address or the General Call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I^2C interface cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I^2C logic will now test the I^2C-bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (I2STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads I2DAT with the slave address and the data direction bit (SLA+W). The SI bit in I2CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 19–399. After a repeated START condition (state 0x10). The I^2C block may switch to the master receiver mode by loading I2DAT with SLA+R).

© NXP B.V. 2010. All rights reserved.

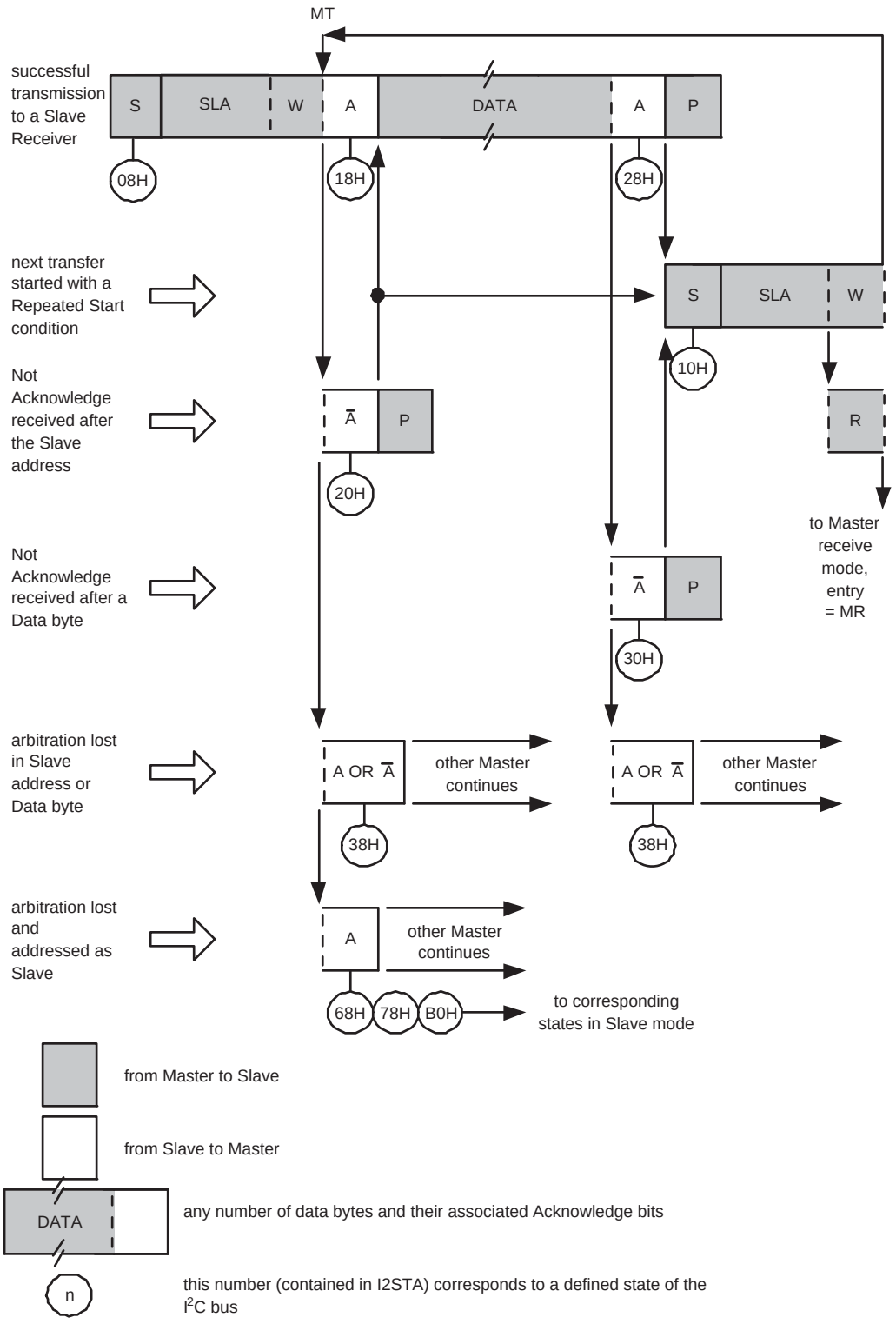


Fig 92. Format and states in the Master Transmitter mode

© NXP B.V. 2010. All rights reserved.

9.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 19–93). 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 19–400. After a repeated START condition (state 0x10), the I^2C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

© NXP B.V. 2010. All rights reserved.

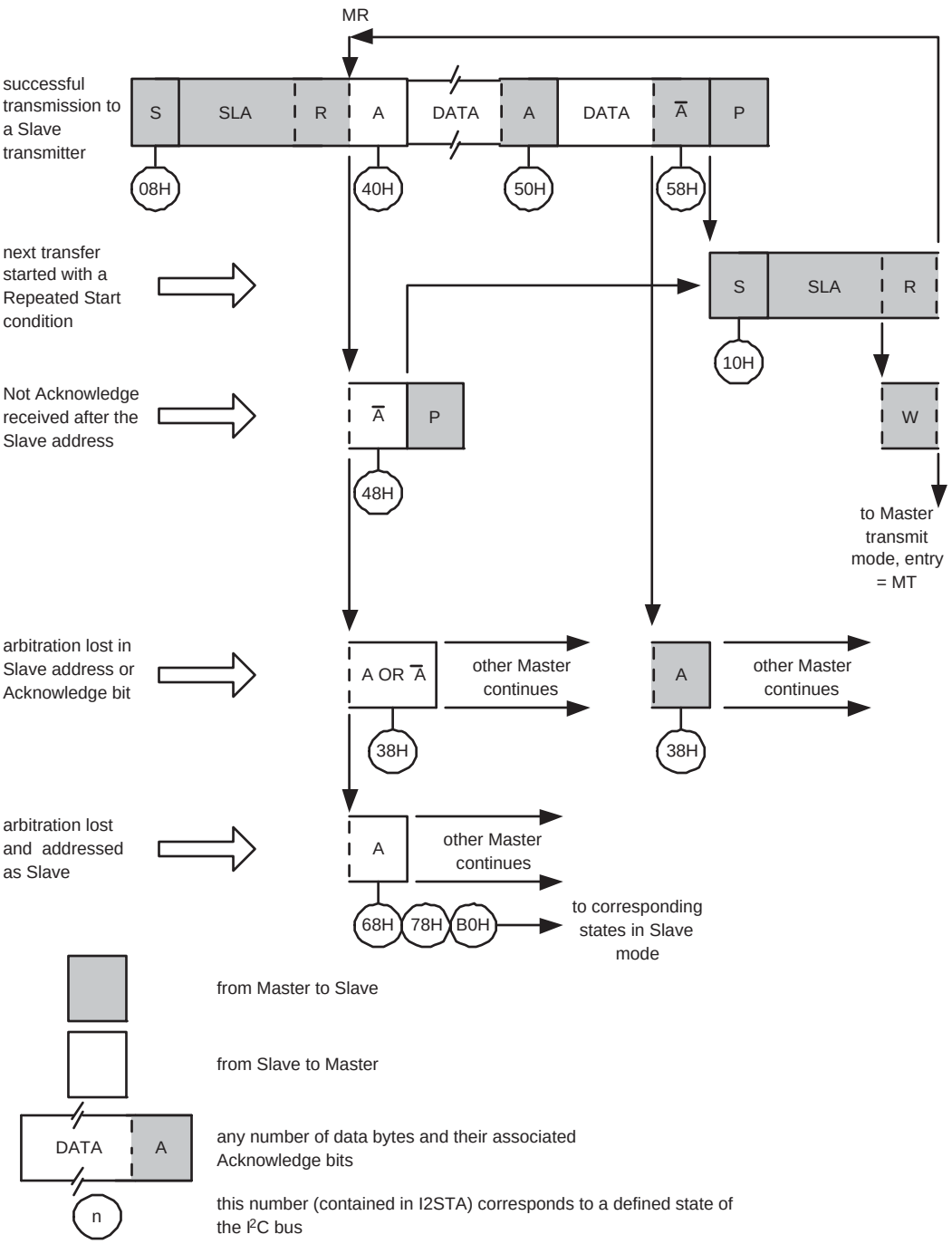


Fig 93. Format and states in the Master Receiver mode

© NXP B.V. 2010. All rights reserved.

9.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 19–94). To initiate the slave receiver mode, I2CON register, the I2ADR registers, and the I2MASK registers must be configured.

The values on the four I2ADR registers combined with the values on the four I2MASK registers determines which address(es) the I^2C block will respond to when slave functions are enabled. See sections 7.2, 7.3, 8.7, and 8.8 for details.

Table 398. I2CONSET used to initialize Slave Receiver mode

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

The I^2C-bus rate settings do not affect the I^2C block in the slave mode. I2EN must be set to logic 1 to enable the I^2C block. The AA bit must be set to enable the I^2C block to acknowledge its own slave address or the General Call address. STA, STO, and SI must be reset.

When the I2ADR, I2MASK, and I2CON registers have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^2C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 19–401. The slave receiver mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^2C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

© NXP B.V. 2010. All rights reserved.

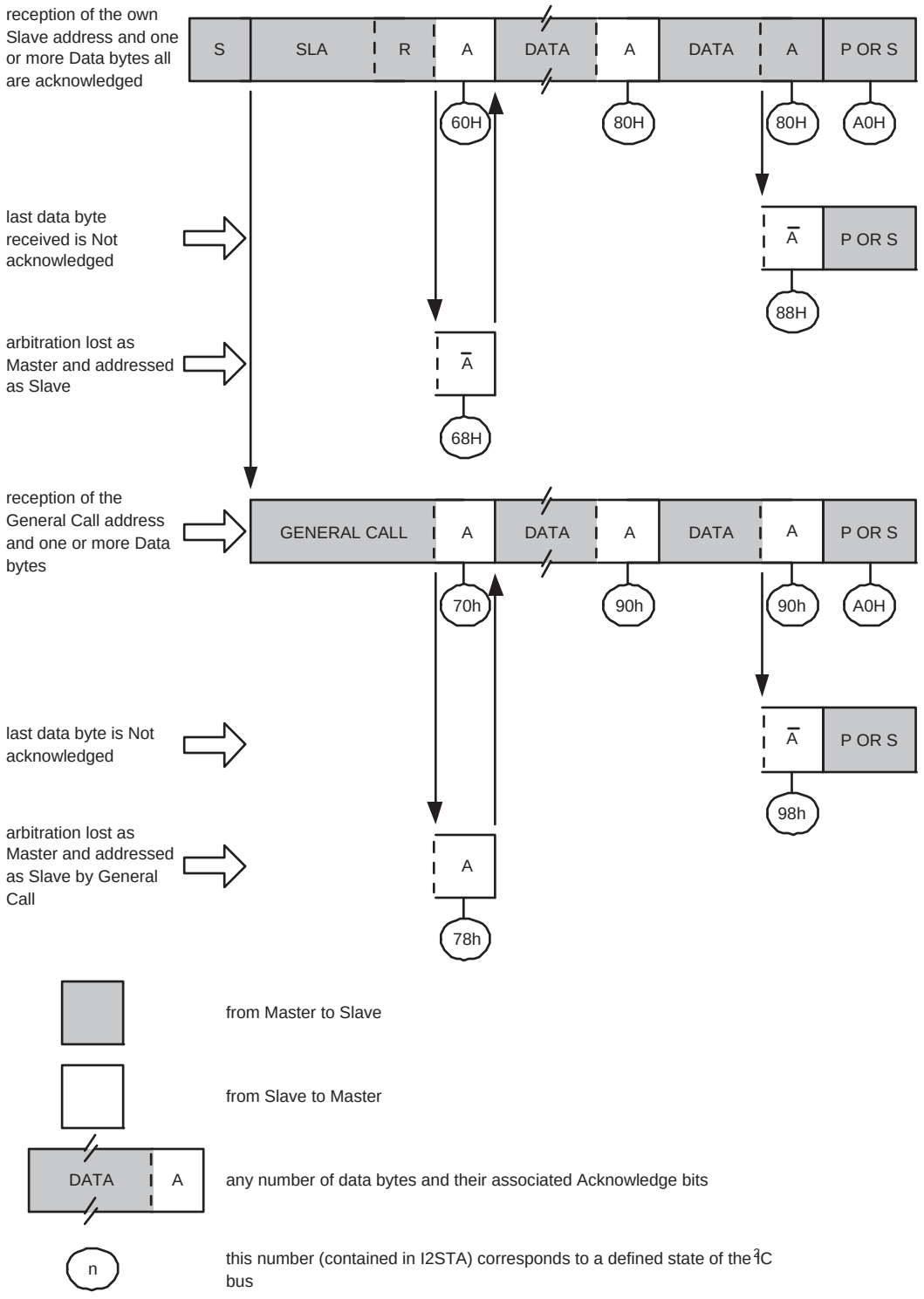


Fig 94. Format and states in the Slave Receiver mode

© NXP B.V. 2010. All rights reserved.

9.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 19–95). Data transfer is initialized as in the slave receiver mode. When I2ADR and I2CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “1” (R) for the I^2C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 19–402. The slave transmitter mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I^2C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^2C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

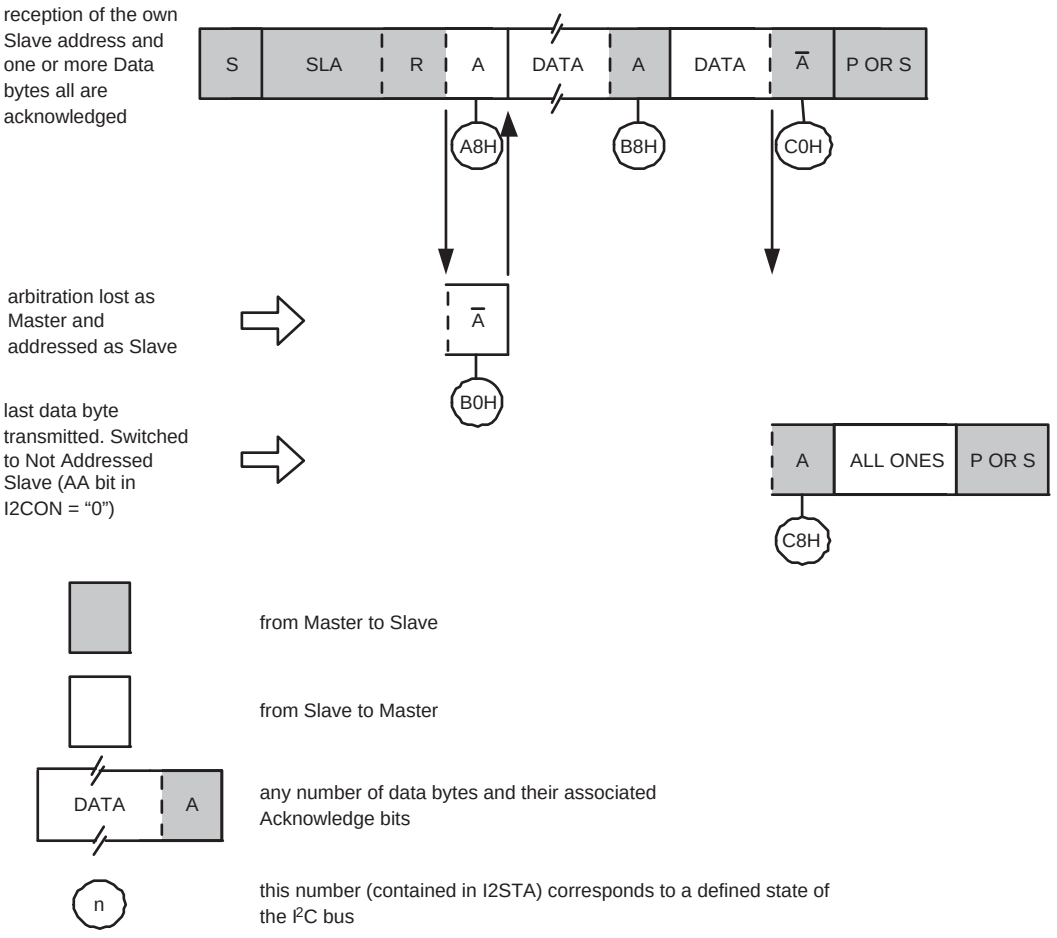


Fig 95. Format and states in the Slave Transmitter mode

© NXP B.V. 2010. All rights reserved.

9.5 Detailed state tables

The following tables show detailed state information for the four I^2C operating modes.

Table 399. Master Transmitter mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W; clear STA	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R; Clear STA	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in I2DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No I2DAT action or	0	0	0	X	I^2C-bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

© NXP B.V. 2010. All rights reserved.

Table 400. Master Receiver mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No I2DAT action or	0	0	0	X	I^2C-bus will be released; the I^2C block will enter a slave mode.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No I2DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No I2DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No I2DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

© NXP B.V. 2010. All rights reserved.

Table 401. Slave Receiver mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General Call address (0x00) has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General Call address has been received, ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLA address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

© NXP B.V. 2010. All rights reserved.

Table 401. Slave Receiver mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave Receiver or Slave Transmitter.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

© NXP B.V. 2010. All rights reserved.

Table 402. Slave Transmitter mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
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.

© NXP B.V. 2010. All rights reserved.

9.6 Miscellaneous states

There are two I2STAT codes that do not correspond to a defined I^2C hardware state (see Table 19–403). These are discussed below.

9.6.1 I2STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I^2C block is not involved in a serial transfer.

9.6.2 I2STAT = 0x00

This status code indicates that a bus error has occurred during an I^2C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I^2C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I^2C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in I2CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 403. Miscellaneous States

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

9.7 Some special cases

The I^2C hardware has facilities to handle the following special cases that may occur during a serial transfer:

9.7.1 Simultaneous repeated START conditions from two masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 19–96). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

© NXP B.V. 2010. All rights reserved.

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.

9.7.2 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 19–90). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 19–92 and Figure 19–93).

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.

9.7.3 Forced access to the I²C-bus

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

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I²C-bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I²C-bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I²C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware Figure 19–97.

9.7.4 I²C-bus obstructed by a LOW level on SCL or SDA

An I²C-bus hang-up can occur if either the SDA or SCL line is held LOW by any device on the bus. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the problem must be resolved by the device that is pulling the SCL bus line LOW.

Typically, the SDA line may be obstructed by another device on the bus that has become out of synchronization with the current bus master by either missing a clock, or by sensing a noise pulse as a clock. In this case, the problem can be solved by transmitting additional clock pulses on the SCL line Figure 19–98. The I²C interface does not include a dedicated timeout timer to detect an obstructed bus, but this can be implemented using another timer in the system. When detected, software can force clocks (up to 9 may be required) on SCL until SDA is released by the offending device. At that point, the slave may still be out of synchronization, so a START should be generated to insure that all I²C peripherals are synchronized.

9.7.5 Bus error

A bus error occurs when a START or STOP condition is detected at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

© NXP B.V. 2010. All rights reserved.

The I^2C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^2C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 19–403.

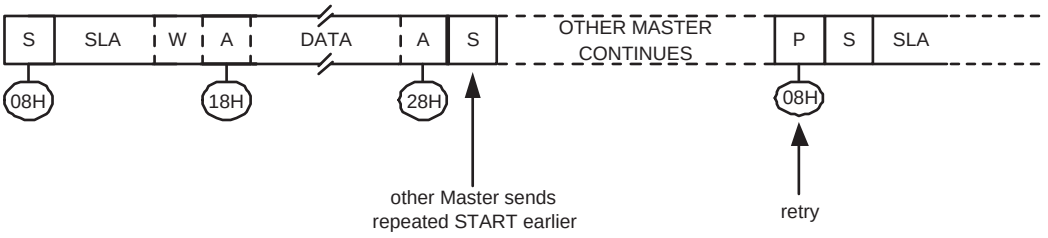


Fig 96. Simultaneous repeated START conditions from two masters

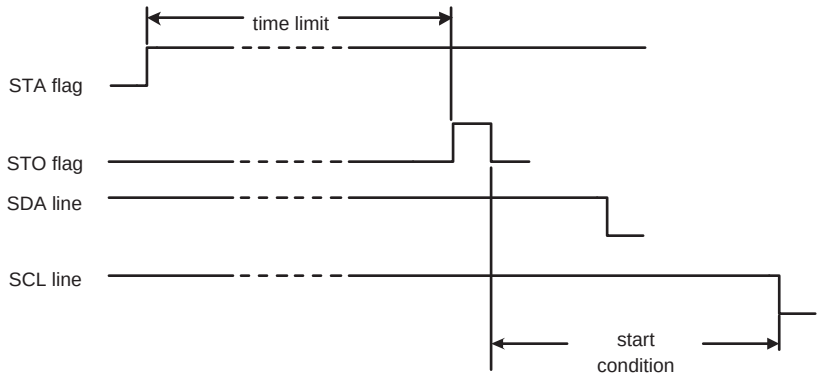
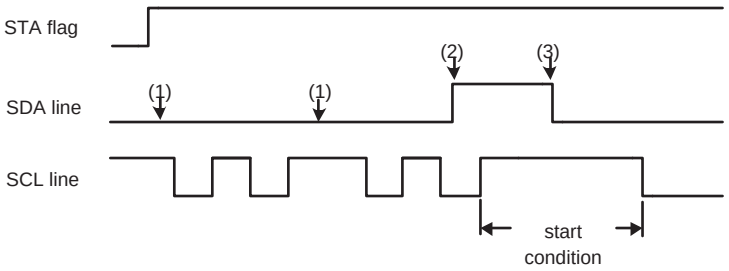


Fig 97. Forced access to a busy I^2C-bus



(1) Unsuccessful attempt to send a START condition.

(2) SDA line is released.

(3) Successful attempt to send a START condition. State 08H is entered.

Fig 98. Recovering from a bus obstruction caused by a LOW level on SDA

© NXP B.V. 2010. All rights reserved.

9.8 I²C state service routines

This section provides examples of operations that must be performed by various I²C state service routines. This includes:

- Initialization of the I²C block after a Reset.
- I²C Interrupt Service
- The 26 state service routines providing support for all four I²C operating modes.

9.8.1 Initialization

In the initialization example, the I²C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- The I2ADR registers and I2MASK registers are loaded with values to configure the part's own slave address(es) and the General Call bit (GC)
- The I²C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by loading the I2SCLH and I2SCLL registers. The master routines must be started in the main program.

The I²C hardware now begins checking the I²C-bus for its own slave address and General Call. If the General Call or the own slave address is detected, an interrupt is requested and I2STAT is loaded with the appropriate state information.

9.8.2 I²C interrupt service

When the I²C interrupt is entered, I2STAT contains a status code which identifies one of the 26 state services to be executed.

9.8.3 The state service routines

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

9.8.4 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I²C state codes. If one or more of the four I²C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of timeout during I²C operations, in order to trap an inoperative bus or a lost service routine.

© NXP B.V. 2010. All rights reserved.

10. Software example

10.1 Initialization routine

Example to initialize I^2C Interface as a Slave and/or Master.

1. Load the I2ADR registers and I2MASK registers with values to configure the own Slave Address, enable General Call recognition if needed.
2. Enable I^2C interrupt.
3. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to I2CONSET.

10.2 Start Master Transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

10.3 Start Master Receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

10.4 I^2C interrupt routine

Determine the I^2C state and which state routine will be used to handle it.

1. Read the I^2C status from I2STA.
2. Use the status value to branch to one of 26 possible state routines.

10.5 Non mode specific states

10.5.1 State: 0x00

Bus Error. Enter not addressed Slave mode and release bus.

© NXP B.V. 2010. All rights reserved.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.5.2 Master States

State 0x08 and State 0x10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

10.5.3 State: 0x08

A START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

10.5.4 State: 0x10

A repeated START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

10.6 Master Transmitter states

10.6.1 State: 0x18

Previous state was State 0x08 or State 0x10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted.

1. Load I2DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

© NXP B.V. 2010. All rights reserved.

10.6.2 State: 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.6.3 State: 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a STOP condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit
5. Load I2DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to I2CONSET to set the AA bit.
7. Write 0x08 to I2CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

10.6.4 State: 0x30

Data has been transmitted, NOT ACK received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.6.5 State: 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new START condition will be transmitted when the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.7 Master Receive states

10.7.1 State: 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.

© NXP B.V. 2010. All rights reserved.

3. Exit

10.7.2 State: 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.7.3 State: 0x50

Data has been received, ACK has been returned. Data will be read from I2DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from I2DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

10.7.4 State: 0x58

Data has been received, NOT ACK has been returned. Data will be read from I2DAT. A STOP condition will be transmitted.

1. Read data byte from I2DAT into Master Receive buffer.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit

10.8 Slave Receiver states

10.8.1 State: 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

© NXP B.V. 2010. All rights reserved.

10.8.2 State: 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

10.8.3 State: 0x70

General Call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

10.8.4 State: 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General Call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

10.8.5 State: 0x80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

© NXP B.V. 2010. All rights reserved.

10.8.6 State: 0x88

Previously addressed with own Slave Address. Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.8.7 State: 0x90

Previously addressed with General Call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
3. Exit

10.8.8 State: 0x98

Previously addressed with General Call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.8.9 State: 0xA0

A STOP condition or repeated START has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

10.9 Slave Transmitter states

10.9.1 State: 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

© NXP B.V. 2010. All rights reserved.

10.9.2 State: 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to I2CONSET to set the STA and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

10.9.3 State: 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

10.9.4 State: 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit.

10.9.5 State: 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The I^2S interface is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCI2S.

   Remark: On reset, the I^2S interface is disabled (PCI2S = 0).
2. Clock: In PCLKSEL1 select PCLK_I2S, see Table 4–41.
3. Pins: Select I^2S pins and their modes in PINSEL0 to PINSEL4 and PINMODE0 to PINMODE4 (see Section 8–5).
4. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
5. DMA: The I^2S interface supports two DMA requests, see Table 20–411 and Table 20–412, and Table 31–544.

2. Features

The I^2S bus provides a standard communication interface for digital audio applications.

The I^2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select signal. The basic I^2S connection has one master, which is always the master, and one slave. The I^2S interface on the LPC17xx provides a separate transmit and receive channel, each of which can operate as either a master or a slave.

- The I^2S input can operate in both master and slave mode.

  The I^2S output can operate in both master and slave mode, independent of the I^2S input.
- Capable of handling 8-bit, 16-bit, and 32-bit word sizes.
- Mono and stereo audio data supported.
- Versatile clocking includes independent transmit and receive fractional rate generators, and an ability to use a single clock input or output for a 4-wire mode.
- The sampling frequency (fs) can range (in practice) from 16 to 96 kHz. (16, 22.05, 32, 44.1, 48, or 96 kHz) for audio applications.
- Separate Master Clock outputs for both transmit and receive channels support a clock up to 512 times the I^2S sampling frequency.
- Word Select period in master mode is configurable (separately for I^2S input and I^2S output).
- Two 8 word (32 byte) FIFO data buffers are provided, one for transmit and one for receive.
- Generates interrupt requests when buffer levels cross a programmable boundary.
- Two DMA requests, controlled by programmable buffer levels. These are connected to the General Purpose DMA block.
- Controls include reset, stop and mute options separately for I^2S input and I^2S output.

© NXP B.V. 2010. All rights reserved.

3. Description

The I^2S performs serial data out via the transmit channel and serial data in via the receive channel. These support the NXP Inter IC Audio format for 8-bit, 16-bit and 32-bit audio data, both for stereo and mono modes. Configuration, data access and control is performed by a APB register set. Data streams are buffered by FIFOs with a depth of 8 words.

The I^2S receive and transmit stage can operate independently in either slave or master mode. Within the I^2S module the difference between these modes lies in the word select (WS) signal which determines the timing of data transmissions. Data words start on the next falling edge of the transmitting clock after a WS change. In stereo mode when WS is low left data is transmitted and right data when WS is high. In mono mode the same data is transmitted twice, once when WS is low and again when WS is high.

- In master mode, word select is generated internally with a 9-bit counter. The half period count value of this counter can be set in the control register.
- In slave mode, word select is input from the relevant bus pin.
- When an I^2S bus is active, the word select, receive clock and transmit clock signals are sent continuously by the bus master, while data is sent continuously by the transmitter.
- Disabling the I^2S can be done with the stop or mute control bits separately for the transmit and receive.
- The stop bit will disable accesses by the transmit channel or the receive channel to the FIFOs and will place the transmit channel in mute mode.
- The mute control bit will place the transmit channel in mute mode. In mute mode, the transmit channel FIFO operates normally, but the output is discarded and replaced by zeroes. This bit does not affect the receive channel, data reception can occur normally.

© NXP B.V. 2010. All rights reserved.

4. Pin descriptions

Table 404. Pin descriptions

Pin Name	Type	Description
I2SRX_CLK	Input/ Output	Receive Clock. A clock signal used to synchronize the transfer of data on the receive channel. It is driven by the master and received by the slave. Corresponds to the signal SCK in the I^2S bus specification.
I2SRX_WS	Input/ Output	Receive Word Select. Selects the channel from which data is to be received. It is driven by the master and received by the slave. Corresponds to the signal WS in the I^2S bus specification. WS = 0 indicates that data is being received by channel 1 (left channel). WS = 1 indicates that data is being received by channel 2 (right channel).
I2SRX_SDA	Input/ Output	Receive Data. Serial data, received MSB first. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the I^2S bus specification.
RX_MCLK	Output	Optional master clock output for the I^2S receive function.
I2STX_CLK	Input/ Output	Transmit Clock. A clock signal used to synchronize the transfer of data on the transmit channel. It is driven by the master and received by the slave. Corresponds to the signal SCK in the I^2S bus specification.
I2STX_WS	Input/ Output	Transmit Word Select. Selects the channel to which data is being sent. It is driven by the master and received by the slave. Corresponds to the signal WS in the I^2S bus specification. WS = 0 indicates that data is being sent to channel 1 (left channel). WS = 1 indicates that data is being sent to channel 2 (right channel).
I2STX_SDA	Input/ Output	Transmit Data. Serial data, sent MSB first. It is driven by the transmitter and read by the receiver. Corresponds to the signal SD in the I^2S bus specification.
TX_MCLK	Output	Optional master clock output for the I^2S transmit function.

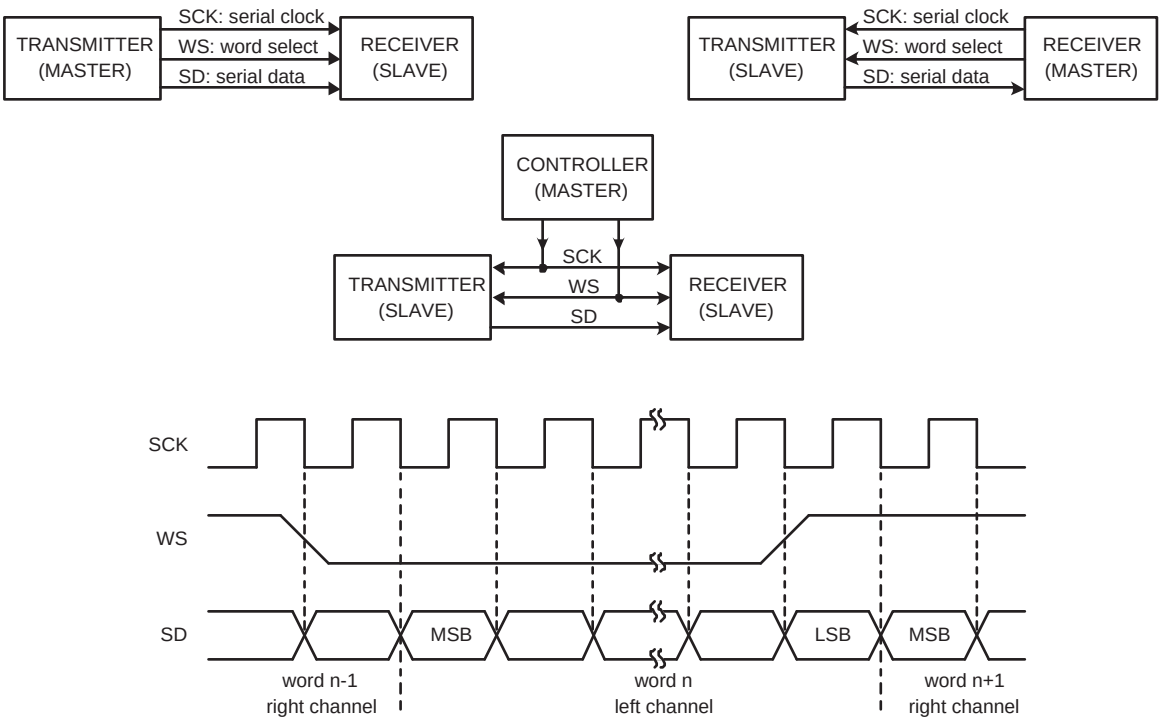


Fig 99. Simple I^2S configurations and bus timing

© NXP B.V. 2010. All rights reserved.

5. Register description

Table 20–405 shows the registers associated with the I²S interface and a summary of their functions. Following the table are details for each register.

Table 405. I²S register map

Name	Description	Access	Reset Value[1]	Address
I2SDAO	Digital Audio Output Register. Contains control bits for the I²S transmit channel.	R/W	0x87E1	0x400A 8000
I2SDAI	Digital Audio Input Register. Contains control bits for the I²S receive channel.	R/W	0x07E1	0x400A 8004
I2STXFIFO	Transmit FIFO. Access register for the 8 × 32-bit transmitter FIFO.	WO	0	0x400A 8008
I2SRXFIFO	Receive FIFO. Access register for the 8 × 32-bit receiver FIFO.	RO	0	0x400A 800C
I2SSTATE	Status Feedback Register. Contains status information about the I²S interface.	RO	0x7	0x400A 8010
I2SDMA1	DMA Configuration Register 1. Contains control information for DMA request 1.	R/W	0	0x400A 8014
I2SDMA2	DMA Configuration Register 2. Contains control information for DMA request 2.	R/W	0	0x400A 8018
I2SIRQ	Interrupt Request Control Register. Contains bits that control how the I²S interrupt request is generated.	R/W	0	0x400A 801C
I2STXRATE	Transmit MCLK divider. This register determines the I²S TX MCLK rate by specifying the value to divide PCLK by in order to produce MCLK.	R/W	0	0x400A 8020
I2SRXRATE	Receive MCLK divider. This register determines the I²S RX MCLK rate by specifying the value to divide PCLK by in order to produce MCLK.	R/W	0	0x400A 8024
I2STXBITRATE	Transmit bit rate divider. This register determines the I²S transmit bit rate by specifying the value to divide TX_MCLK by in order to produce the transmit bit clock.	R/W	0	0x400A 8028
I2SRXBITRATE	Receive bit rate divider. This register determines the I²S receive bit rate by specifying the value to divide RX_MCLK by in order to produce the receive bit clock.	R/W	0	0x400A 802C
I2STXMODE	Transmit mode control.	R/W	0	0x400A 8030
I2SRXMODE	Receive mode control.	R/W	0	0x400A 8034

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

5.1 Digital Audio Output register (I2SDAO - 0x400A 8000)

The I2SDAO register controls the operation of the I²S transmit channel. The function of bits in DAO are shown in Table 20–406.

Table 406: Digital Audio Output register (I2SDAO - address 0x400A 8000) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	wordwidth		Selects the number of bytes in data as follows:	01
		00	8-bit data	
		01	16-bit data	
		10	Reserved, do not use this setting	
		11	32-bit data	

© NXP B.V. 2010. All rights reserved.

Table 406: Digital Audio Output register (I2SDAO - address 0x400A 8000) bit description

Bit	Symbol	Value	Description	Reset Value
2	mono		When 1, data is of monaural format. When 0, the data is in stereo format.	0
3	stop		When 1, disables accesses on FIFOs, places the transmit channel in mute mode.	0
4	reset		When 1, asynchronously resets the transmit channel and FIFO.	0
5	ws_sel		When 0, the interface is in master mode. When 1, the interface is in slave mode. See Section 20–7 for a summary of useful combinations for this bit with I2STXMODE.	1
14:6	ws_halfperiod		Word select half period minus 1, i.e. WS 64clk period -> ws_halfperiod = 31.	0x1F
15	mute		When 1, the transmit channel sends only zeroes.	1
31:16	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.2 Digital Audio Input register (I2SDAI - 0x400A 8004)

The I2SDAI register controls the operation of the I^2S receive channel. The function of bits in DAI are shown in Table 20–407.

Table 407: Digital Audio Input register (I2SDAI - address 0x400A 8004) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	wordwidth		Selects the number of bytes in data as follows:	01
		00	8-bit data	
		01	16-bit data	
		10	Reserved, do not use this setting	
		11	32-bit data	
2	mono		When 1, data is of monaural format. When 0, the data is in stereo format.	0
3	stop		When 1, disables accesses on FIFOs, places the transmit channel in mute mode.	0
4	reset		When 1, asynchronously reset the transmit channel and FIFO.	0
5	ws_sel		When 0, the interface is in master mode. When 1, the interface is in slave mode. See Section 20–7 for a summary of useful combinations for this bit with I2SRXMODE.	1
14:6	ws_halfperiod		Word select half period minus 1, i.e. WS 64clk period -> ws_halfperiod = 31.	0x1F
31:15	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.3 Transmit FIFO register (I2STXFIFO - 0x400A 8008)

The I2STXFIFO register provides access to the transmit FIFO. The function of bits in I2STXFIFO are shown in Table 20–408.

Table 408: Transmit FIFO register (I2STXFIFO - address 0x400A 8008) bit description

Bit	Symbol	Description	Reset Value
31:0	I2STXFIFO	8 × 32-bit transmit FIFO.	Level = 0

5.4 Receive FIFO register (I2SRXFIFO - 0x400A 800C)

The I2SRXFIFO register provides access to the receive FIFO. The function of bits in I2SRXFIFO are shown in Table 20–409.

© NXP B.V. 2010. All rights reserved.

Table 409: Receive FIFO register (I2RXFIFO - address 0x400A 800C) bit description

Bit	Symbol	Description	Reset Value
31:0	I2SRXFIFO	8 × 32-bit transmit FIFO.	level = 0

5.5 Status Feedback register (I2SSTATE - 0x400A 8010)

The I2SSTATE register provides status information about the I^2S interface. The meaning of bits in I2SSTATE are shown in Table 20–410.

Table 410: Status Feedback register (I2SSTATE - address 0x400A 8010) bit description

Bit	Symbol	Description	Reset Value
0	irq	This bit reflects the presence of Receive Interrupt or Transmit Interrupt. This is determined by comparing the current FIFO levels to the rx_depth_irq and tx_depth_irq fields in the I2SIRQ register.	1
1	dmareq1	This bit reflects the presence of Receive or Transmit DMA Request 1. This is determined by comparing the current FIFO levels to the rx_depth_dma1 and tx_depth_dma1 fields in the I2SDMA1 register.	1
2	dmareq2	This bit reflects the presence of Receive or Transmit DMA Request 2. This is determined by comparing the current FIFO levels to the rx_depth_dma2 and tx_depth_dma2 fields in the I2SDMA2 register.	1
7:3	Unused	Unused.	0
11:8	rx_level	Reflects the current level of the Receive FIFO.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	tx_level	Reflects the current level of the Transmit FIFO.	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.6 DMA Configuration Register 1 (I2SDMA1 - 0x400A 8014)

The I2SDMA1 register controls the operation of DMA request 1. The function of bits in I2SDMA1 are shown in Table 20–411. Refer to the General Purpose DMA Controller chapter for details of DMA operation.

Table 411: DMA Configuration register 1 (I2SDMA1 - address 0x400A 8014) bit description

Bit	Symbol	Description	Reset Value
0	rx_dma1_enable	When 1, enables DMA1 for I^2S receive.	0
1	tx_dma1_enable	When 1, enables DMA1 for I^2S transmit.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
11:8	rx_depth_dma1	Set the FIFO level that triggers a receive DMA request on DMA1.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	tx_depth_dma1	Set the FIFO level that triggers a transmit DMA request on DMA1.	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

5.7 DMA Configuration Register 2 (I2SDMA2 - 0x400A 8018)

The I2SDMA2 register controls the operation of DMA request 2. The function of bits in I2SDMA2 are shown in Table 20–406.

Table 412: DMA Configuration register 2 (I2SDMA2 - address 0x400A 8018) bit description

Bit	Symbol	Description	Reset Value
0	rx_dma2_enable	When 1, enables DMA1 for I^2S receive.	0
1	tx_dma2_enable	When 1, enables DMA1 for I^2S transmit.	0
7:2	Unused	Unused.	0
11:8	rx_depth_dma2	Set the FIFO level that triggers a receive DMA request on DMA2.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	tx_depth_dma2	Set the FIFO level that triggers a transmit DMA request on DMA2.	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.8 Interrupt Request Control register (I2SIRQ - 0x400A 801C)

The I2SIRQ register controls the operation of the I^2S interrupt request. The function of bits in I2SIRQ are shown in Table 20–406.

Table 413: Interrupt Request Control register (I2SIRQ - address 0x400A 801C) bit description

Bit	Symbol	Description	Reset Value
0	rx_Irq_enable	When 1, enables I^2S receive interrupt.	0
1	tx_Irq_enable	When 1, enables I^2S transmit interrupt.	0
7:2	Unused	Unused.	0
11:8	rx_depth_irq	Set the FIFO level on which to create an irq request.	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	tx_depth_irq	Set the FIFO level on which to create an irq request.	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.9 Transmit Clock Rate register (I2STXRATE - 0x400A 8020)

The MCLK rate for the I^2S transmitter is determined by the values in the I2STXRATE register. The required I2STXRATE setting depends on the desired audio sample rate desired, the format (stereo/mono) used, and the data size.

The transmitter MCLK rate is generated using a fractional rate generator, dividing down the frequency of PCLK_I2S. Values of the numerator (X) and the denominator (Y) must be chosen to produce a frequency twice that desired for the transmitter MCLK, which must be an integer multiple of the transmitter bit clock rate. Fractional rate generators have some aspects that the user should be aware of when choosing settings. These are discussed in Section 20–5.9.1. The equation for the fractional rate generator is:

I2STXMCLK = PCLK_I2S * (X/Y) /2

© NXP B.V. 2010. All rights reserved.

Note: If the value of X or Y is 0, then no clock is generated. Also, the value of Y must be greater than or equal to X.

Table 414: Transmit Clock Rate register (I2TXRATE - address 0x400A 8020) bit description

Bit	Symbol	Description	Reset Value
7:0	Y_divider	I^2S transmit MCLK rate denominator. This value is used to divide PCLK to produce the transmit MCLK. Eight bits of fractional divide supports a wide range of possibilities. A value of 0 stops the clock.	0
15:8	X_divider	I^2S transmit MCLK rate numerator. This value is used to multiply PCLK by to produce the transmit MCLK. A value of 0 stops the clock. Eight bits of fractional divide supports a wide range of possibilities. Note: the resulting ratio X/Y is divided by 2.	0
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.9.1 Notes on fractional rate generators

The nature of a fractional rate generator is that there will be some output jitter with some divide settings. This is because the fractional rate generator is a fully digital function, so output clock transitions are synchronous with the source clock, whereas a theoretical perfect fractional rate may have edges that are not related to the source clock. So, output jitter will not be greater than plus or minus one source clock between consecutive clock edges.

For example, if X = 0x07 and Y = 0x11, the fractional rate generator will output 7 clocks for every 17 (11 hex) input clocks, distributed as evenly as it can. In this example, there is no way to distribute the output clocks in a perfectly even fashion, so some clocks will be longer than others. The output is divided by 2 in order to square it up, which also helps with the jitter. The frequency averages out to exactly (7/17) / 2, but some clocks will be a slightly different length than their neighbors. It is possible to avoid jitter entirely by choosing fractions such that X divides evenly into Y, such as 2/4, 2/6, 3/9, 1/N, etc.

5.10 Receive Clock Rate register (I2SRXRATE - 0x400A 8024)

The MCLK rate for the I^2S receiver is determined by the values in the I2SRXRATE register. The required I2SRXRATE setting depends on the peripheral clock rate (PCLK_I2S) and the desired MCLK rate (such as 256 fs).

The receiver MCLK rate is generated using a fractional rate generator, dividing down the frequency of PCLK_I2S. Values of the numerator (X) and the denominator (Y) must be chosen to produce a frequency twice that desired for the receiver MCLK, which must be an integer multiple of the receiver bit clock rate. Fractional rate generators have some aspects that the user should be aware of when choosing settings. These are discussed in Section 20–5.9.1. The equation for the fractional rate generator is:

I2SRXMCLK = PCLK_I2S * (X/Y) /2

Note: If the value of X or Y is 0, then no clock is generated. Also, the value of Y must be greater than or equal to X.

© NXP B.V. 2010. All rights reserved.

Table 415: Receive Clock Rate register (I2SRXRATE - address 0x400A 8024) bit description

Bit	Symbol	Description	Reset Value
7:0	Y_divider	I^2S receive MCLK rate denominator. This value is used to divide PCLK to produce the receive MCLK. Eight bits of fractional divide supports a wide range of possibilities. A value of 0 stops the clock.	0
15:8	X_divider	I^2S receive MCLK rate numerator. This value is used to multiply PCLK by to produce the receive MCLK. A value of 0 stops the clock. Eight bits of fractional divide supports a wide range of possibilities. Note: the resulting ratio X/Y is divided by 2.	0
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.11 Transmit Clock Bit Rate register (I2STXBITRATE - 0x400A 8028)

The bit rate for the I^2S transmitter is determined by the value of the I2STXBITRATE register. The value depends on the audio sample rate desired, and the data size and format (stereo/mono) used. For example, a 48 kHz sample rate for 16-bit stereo data requires a bit rate of 48,000×16×2 = 1.536 MHz.

Table 416: Transmit Clock Rate register (I2TXBITRATE - address 0x400A 8028) bit description

Bit	Symbol	Description	Reset Value
5:0	tx_bitrate	I^2S transmit bit rate. This value plus one is used to divide TX_MCLK to produce the transmit bit clock.	0
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.12 Receive Clock Bit Rate register (I2SRXBITRATE - 0x400A 802C)

The bit rate for the I^2S receiver is determined by the value of the I2SRXBITRATE register. The value depends on the audio sample rate, as well as the data size and format used. The calculation is the same as for I2SRXBITRATE.

Table 417: Receive Clock Rate register (I2SRXBITRATE - address 0x400A 802C) bit description

Bit	Symbol	Description	Reset Value
5:0	rx_bitrate	I^2S receive bit rate. This value plus one is used to divide RX_MCLK to produce the receive bit clock.	0
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.13 Transmit Mode Control register (I2STXMODE - 0x400A 8030)

The Transmit Mode Control register contains additional controls for transmit clock source, enabling the 4-pin mode, and how MCLK is used. See Section 20–7 for a summary of useful mode combinations.

© NXP B.V. 2010. All rights reserved.

Table 418: Transmit Mode Control register (I2STXMODE - 0x400A 8030) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	TXCLKSEL		Clock source selection for the transmit bit clock divider.	0
		00	Select the TX fractional rate divider clock output as the source	
		01	Reserved	
		10	Select the RX_MCLK signal as the TX_MCLK clock source	
		11	Reserved	
2	TX4PIN		Transmit 4-pin mode selection. When 1, enables 4-pin mode.	0
3	TXMCENA		Enable for the TX_MCLK output. When 0, output of TX_MCLK is not enabled. When 1, output of TX_MCLK is enabled.	0
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.14 Receive Mode Control register (I2SRXMODE - 0x400A 8034)

The Receive Mode Control register contains additional controls for receive clock source, enabling the 4-pin mode, and how MCLK is used. See Section 20–7 for a summary of useful mode combinations.

Table 419: Receive Mode Control register (I2SRXMODE - 0x400A 8034) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	RXCLKSEL		Clock source selection for the receive bit clock divider.	0
		00	Select the RX fractional rate divider clock output as the source	
		01	Reserved	
		10	Select the TX_MCLK signal as the RX_MCLK clock source	
		11	Reserved	
2	RX4PIN		Receive 4-pin mode selection. When 1, enables 4-pin mode.	0
3	RXMCENA		Enable for the RX_MCLK output. When 0, output of RX_MCLK is not enabled. When 1, output of RX_MCLK is enabled.	0
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6. I²S transmit and receive interfaces

The I²S interface can transmit and receive 8-bit, 16-bit or 32-bit stereo or mono audio information. Some details of I²S implementation are:

- When the FIFO is empty, the transmit channel will repeat transmitting the same data until new data is written to the FIFO.
- When mute is true, the data value 0 is transmitted.
- When mono is false, two successive data words are respectively left and right data.
- Data word length is determined by the wordwidth value in the configuration register. There is a separate wordwidth value for the receive channel and the transmit channel.
  - 0: word is considered to contain four 8-bit data words.
  - 1: word is considered to contain two 16-bit data words.
  - 3: word is considered to contain one 32-bit data word.
- When the transmit FIFO contains insufficient data the transmit channel will repeat transmitting the last data until new data is available. This can occur when the microprocessor or the DMA at some time is unable to provide new data fast enough. Because of this delay in new data there is a need to fill the gap, which is accomplished by continuing to transmit the last sample. The data is not muted as this would produce an noticeable and undesirable effect in the sound.
- The transmit channel and the receive channel only handle 32-bit aligned words, data chunks must be clipped or extended to a multiple of 32 bits.

When switching between data width or modes the I²S must be reset via the reset bit in the control register in order to ensure correct synchronization. It is advisable to set the stop bit also until sufficient data has been written in the transmit FIFO. Note that when stopped data output is muted.

All data accesses to FIFOs are 32 bits. Figure 20–112 shows the possible data sequences.

A data sample in the FIFO consists of:

- 1×32 bits in 8-bit or 16-bit stereo modes.
- 1×32 bits in mono modes.
- 2×32 bits, first left data, second right data, in 32-bit stereo modes.

Data is read from the transmit FIFO after the falling edge of WS, it will be transferred to the transmit clock domain after the rising edge of WS. On the next falling edge of WS the left data will be loaded in the shift register and transmitted and on the following rising edge of WS the right data is loaded and transmitted.

The receive channel will start receiving data after a change of WS. When word select becomes low it expects this data to be left data, when WS is high received data is expected to be right data. Reception will stop when the bit counter has reached the limit set by wordwidth. On the next change of WS the received data will be stored in the appropriate hold register. When complete data is available it will be written into the receive FIFO.

© NXP B.V. 2010. All rights reserved.

7. I²S operating modes

The clocking and WS usage of the I²S interface is configurable. In addition to master and slave modes, which are independently configurable for the transmitter and the receiver, several different clock sources are possible, including variations that share the clock and/or WS between the transmitter and receiver. This last option allows using I²S with fewer pins, typically four.

Many configurations are possible that are not considered useful, the following tables and figures give details of the configurations that are most likely to be useful.

Table 420: I²S transmit modes

I2SDAO [5]	I2STXMODE [3:0]	Description
0	0 0 0 0	Typical transmitter master mode. See Figure 20–100. The I²S transmit function operates as a master. The transmit clock source is the fractional rate divider. The WS used is the internally generated TX_WS. The TX_MCLK pin is not enabled for output.
0	0 0 1 0	Transmitter master mode sharing the receiver reference clock. See Figure 20–101. The I²S transmit function operates as a master. The transmit clock source is RX_REF. The WS used is the internally generated TX_WS. The TX_MCLK pin is not enabled for output.
0	0 1 0 0	4-wire transmitter master mode sharing the receiver bit clock and WS. See Figure 20–102. The I²S transmit function operates as a master. The transmit clock source is the RX bit clock. The WS used is the internally generated RX_WS. The TX_MCLK pin is not enabled for output.
0	1 0 0 0	Transmitter master mode with TX_MCLK output. See Figure 20–100. The I²S transmit function operates as a master. The transmit clock source is the fractional rate divider. The WS used is the internally generated TX_WS. The TX_MCLK pin is enabled for output.
1	0 0 0 0	Typical transmitter slave mode. See Figure 20–103. The I²S transmit function operates as a slave. The transmit clock source is the TX_CLK pin. The WS used is the TX_WS pin.
1	0 0 1 0	Transmitter slave mode sharing the receiver reference clock. See Figure 20–104. The I²S transmit function operates as a slave. The transmit clock source is RX_REF. The WS used is the TX_WS pin.
1	0 1 0 0	4-wire transmitter slave mode sharing the receiver bit clock and WS. See Figure 20–105. The I²S transmit function operates as a slave. The transmit clock source is the RX bit clock. The WS used is RX_WS ref.

© NXP B.V. 2010. All rights reserved.

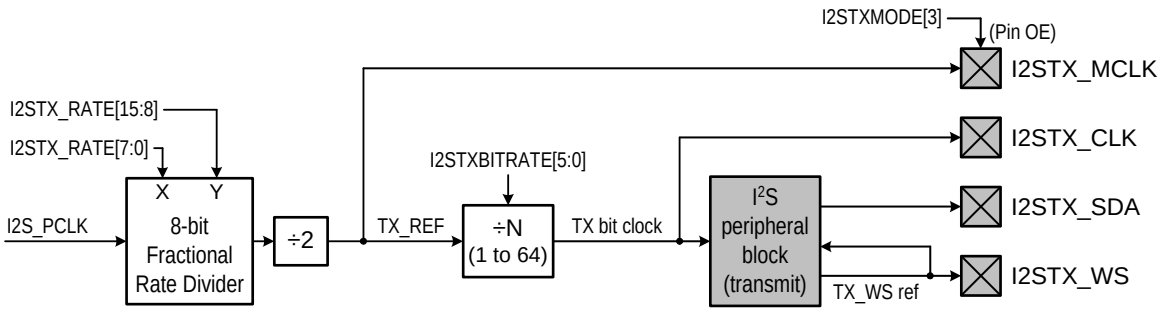


Fig 100. Typical transmitter master mode, with or without MCLK output

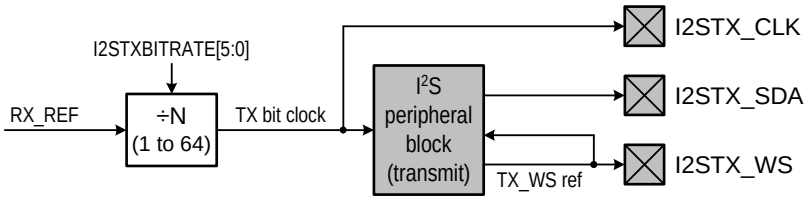


Fig 101. Transmitter master mode sharing the receiver reference clock

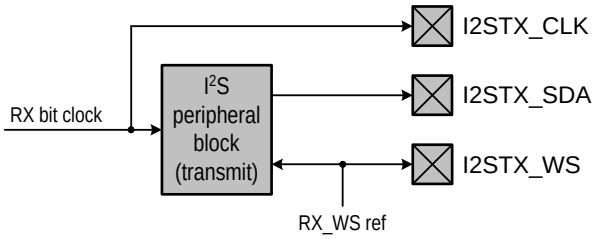


Fig 102. 4-wire transmitter master mode sharing the receiver bit clock and WS

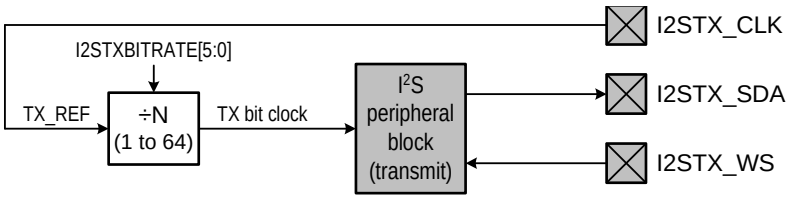


Fig 103. Typical transmitter slave mode

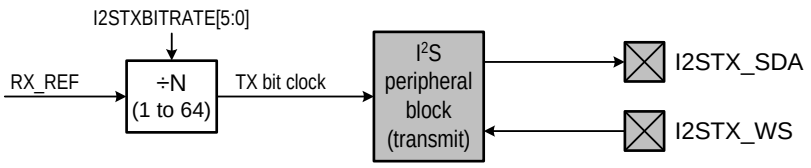


Fig 104. Transmitter slave mode sharing the receiver reference clock

© NXP B.V. 2010. All rights reserved.

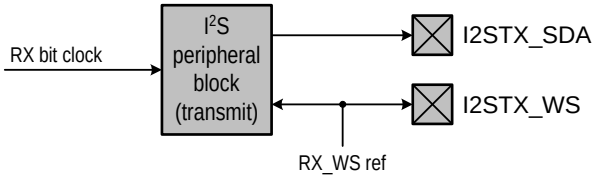


Fig 105. 4-wire transmitter slave mode sharing the receiver bit clock and WS

Table 421: I²S receive modes

I2SDAI [5]	I2SRXMODE [3:0]	Description
0	0 0 0 0	Typical receiver master mode. See Figure 20–106. The I²S receive function operates as a master. The receive clock source is the fractional rate divider. The WS used is the internally generated RX_WS. The RX_MCLK pin is not enabled for output.
0	0 0 1 0	Receiver master mode sharing the transmitter reference clock. See Figure 20–107. The I²S receive function operates as a master. The receive clock source is TX_REF. The WS used is the internally generated RX_WS. The RX_MCLK pin is not enabled for output.
0	0 1 0 0	4-wire receiver master mode sharing the transmitter bit clock and WS. See Figure 20–108. The I²S receive function operates as a master. The receive clock source is the TX bit clock. The WS used is the internally generated TX_WS. The RX_MCLK pin is not enabled for output.
0	1 0 0 0	Receiver master mode with RX_MCLK output. See Figure 20–106. The I²S receive function operates as a master. The receive clock source is the fractional rate divider. The WS used is the internally generated RX_WS. The RX_MCLK pin is enabled for output.
1	0 0 0 0	Typical receiver slave mode. See Figure 20–109. The I²S receive function operates as a slave. The receive clock source is the RX_CLK pin. The WS used is the RX_WS pin.
1	0 0 1 0	Receiver slave mode sharing the transmitter reference clock. See Figure 20–110. The I²S receive function operates as a slave. The receive clock source is TX_REF. The WS used is the RX_WS pin.
1	0 1 0 0	This is a 4-wire receiver slave mode sharing the transmitter bit clock and WS. See Figure 20–111. The I²S receive function operates as a slave. The receive clock source is the TX bit clock. The WS used is TX_WS ref.

© NXP B.V. 2010. All rights reserved.

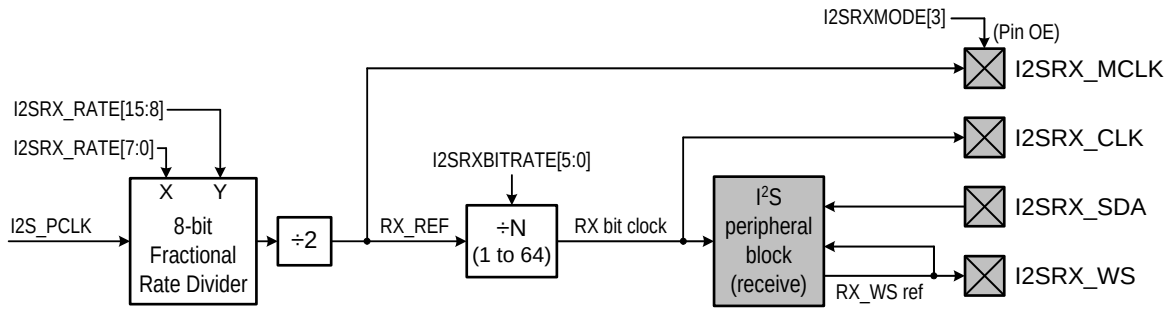


Fig 106. Typical receiver master mode, with or without MCLK output

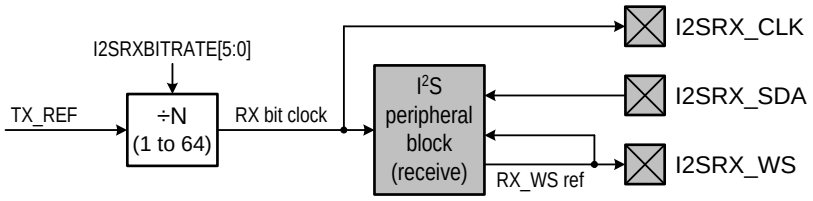


Fig 107. Receiver master mode sharing the transmitter reference clock

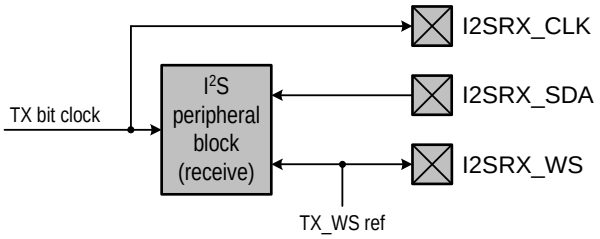


Fig 108. 4-wire receiver master mode sharing the transmitter bit clock and WS

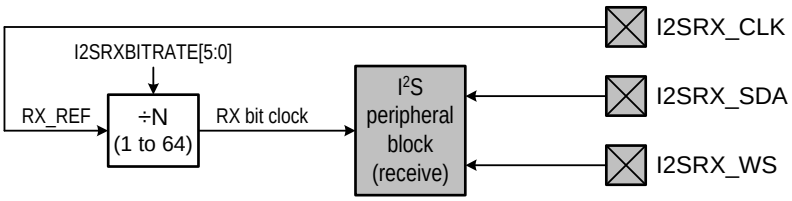


Fig 109. Typical receiver slave mode

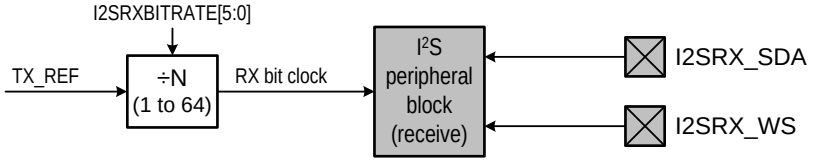


Fig 110. Receiver slave mode sharing the transmitter reference clock

© NXP B.V. 2010. All rights reserved.

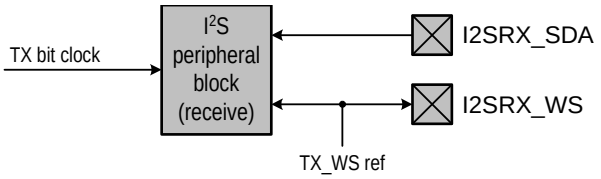


Fig 111. 4-wire receiver slave mode sharing the transmitter bit clock and WS

8. FIFO controller

Handling of data for transmission and reception is performed via the FIFO controller which can generate two DMA requests and an interrupt request. The controller consists of a set of comparators which compare FIFO levels with depth settings contained in registers. The current status of the level comparators can be seen in the APB status register.

Table 422. Conditions for FIFO level comparison

Level Comparison	Condition
dmareq_tx_1	tx_depth_dma1 >= tx_level
dmareq_rx_1	rx_depth_dma1 <= rx_level
dmareq_tx_2	tx_depth_dma2 >= tx_level
dmareq_rx_2	rx_depth_dma2 <= rx_level
irq_tx	tx_depth_irq >= tx_level
irq_rx	rx_depth_irq <= rx_level

System signaling occurs when a level detection is true and enabled.

Table 423. DMA and interrupt request generation

System Signaling	Condition
irq	(irq_rx & rx_irq_enable) | (irq_tx & tx_irq_enable)
dmareq[0]	(dmareq_tx_1 & tx_dma1_enable) | (dmareq_rx_1 & rx_dma1_enable)
dmareq[1]	(dmareq_tx_2 & tx_dma2_enable) | (dmareq_rx_2 & rx_dma2_enable)

Table 424. Status feedback in the I2SSTATE register

Status Feedback	Status
irq	irq_rx | irq_tx
dmareq1	(dmareq_tx_1 | dmareq_rx_1)
dmareq2	(dmareq_rx_2 | dmareq_tx_2)

© NXP B.V. 2010. All rights reserved.

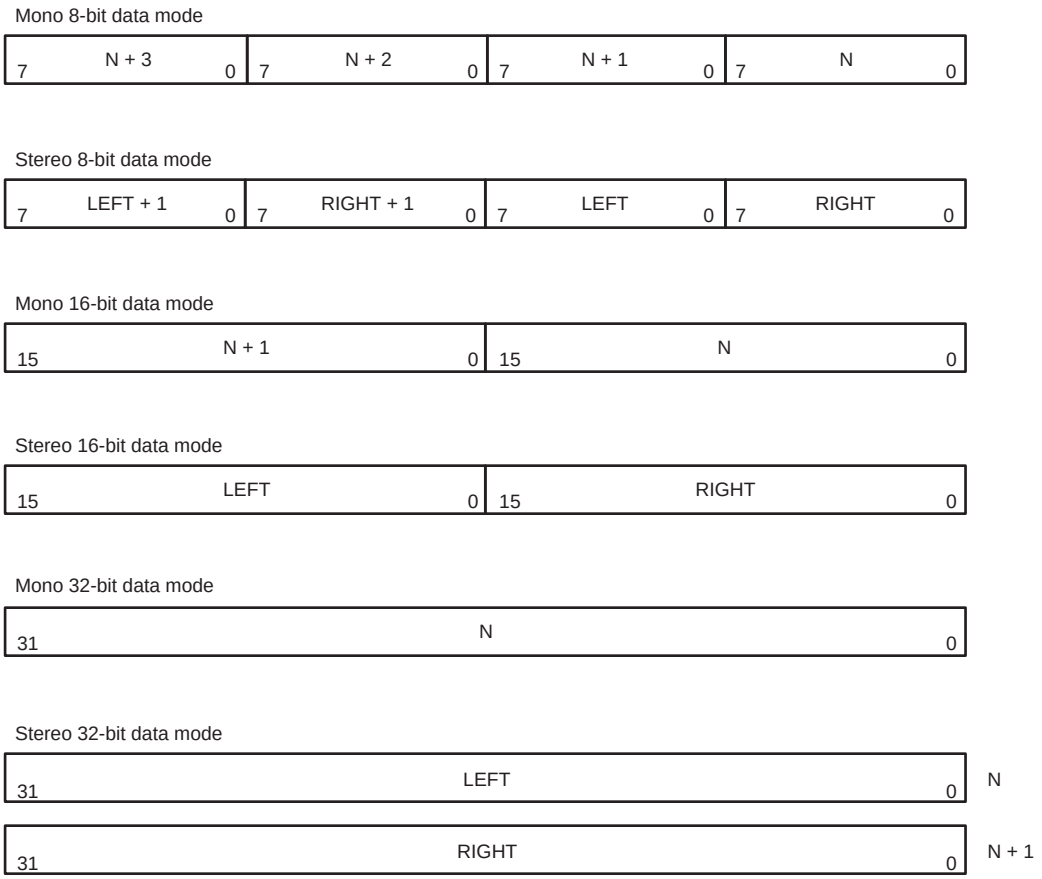


Fig 112. FIFO contents for various I²S modes

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The Timer 0, 1, 2, and 3 peripherals are configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bits PCTIM0/1/2/3.

   Remark: On reset, Timer0/1 are enabled (PCTIM0/1 = 1), and Timer2/3 are disabled (PCTIM2/3 = 0).
2. Peripheral clock: In the PCLKSEL0 register (Table 4–40), select PCLK_TIMER0/1; in the PCLKSEL1 register (Table 4–41), select PCLK_TIMER2/3.
3. Pins: Select timer pins through the PINSEL registers. Select the pin modes for the port pins with timer functions through the PINMODE registers (Section 8–5).
4. Interrupts: See register T0/1/2/3MCR (Table 21–430) and T0/1/2/3CCR (Table 21–431) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
5. DMA: Up to two match conditions can be used to generate timed DMA requests, see Table 31–544.

2. Features

Remark: The four Timer/Counters are identical except for the peripheral base address. A minimum of two Capture inputs and two Match outputs are pinned out for all four timers, with a choice of multiple pins for each. Timer 2 brings out all four Match outputs.

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Counter or Timer operation
- Up to two 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to four external outputs corresponding to match registers, with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.

© NXP B.V. 2010. All rights reserved.

3. Applications

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

4. Description

The Timer/Counter is designed to count cycles of the peripheral clock (PCLK) or an externally-supplied clock, and can optionally generate interrupts or perform other actions at specified timer values, based on four match registers. It also includes four capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

5. Pin description

Table 21–425 gives a brief summary of each of the Timer/Counter related pins.

Table 425. Timer/Counter pin description

Pin	Type	Description
CAP0[1:0] CAP1[1:0] CAP2[1:0] CAP3[1:0]	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. When more than one pin is selected for a Capture input on a single TIMER0/1 channel, the pin with the lowest Port number is used Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 21–6.3.
MAT0[1:0] MAT1[1:0] MAT2[3:0] MAT3[1:0]	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

5.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the Pin Select registers, which are described in Section 8–5. When more than one pin is selected for a MAT output, all such pins are driven identically. When more than one pin is selected for a CAP input, the pin with the lowest Port number is used. Note that match conditions may be used internally without the use of a device pin.

© NXP B.V. 2010. All rights reserved.

6. Register description

Each Timer/Counter contains the registers shown in Table 21–426 ("Reset Value" refers to the data stored in used bits only; it does not include reserved bits content). More detailed descriptions follow.

Table 426. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	T0IR - 0x4000 4000 T1IR - 0x4000 8000 T2IR - 0x4009 0000 T3IR - 0x4009 4000
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	T0TCR - 0x4000 4004 T1TCR - 0x4000 8004 T2TCR - 0x4009 0004 T3TCR - 0x4009 4004
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	T0TC - 0x4000 4008 T1TC - 0x4000 8008 T2TC - 0x4009 0008 T3TC - 0x4009 4008
PR	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0x4000 400C T1PR - 0x4000 800C T2PR - 0x4009 000C T3PR - 0x4009 400C
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	T0PC - 0x4000 4010 T1PC - 0x4000 8010 T2PC - 0x4009 0010 T3PC - 0x4009 4010
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	T0MCR - 0x4000 4014 T1MCR - 0x4000 8014 T2MCR - 0x4009 0014 T3MCR - 0x4009 4014
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	T0MR0 - 0x4000 4018 T1MR0 - 0x4000 8018 T2MR0 - 0x4009 0018 T3MR0 - 0x4009 4018
MR1	Match Register 1. See MR0 description.	R/W	0	T0MR1 - 0x4000 401C T1MR1 - 0x4000 801C T2MR1 - 0x4009 001C T3MR1 - 0x4009 401C
MR2	Match Register 2. See MR0 description.	R/W	0	T0MR2 - 0x4000 4020 T1MR2 - 0x4000 8020 T2MR2 - 0x4009 0020 T3MR2 - 0x4009 4020
MR3	Match Register 3. See MR0 description.	R/W	0	T0MR3 - 0x4000 4024 T1MR3 - 0x4000 8024 T2MR3 - 0x4009 0024 T3MR3 - 0x4009 4024
CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0	T0CCR - 0x4000 4028 T1CCR - 0x4000 8028 T2CCR - 0x4009 0028 T3CCR - 0x4009 4028

© NXP B.V. 2010. All rights reserved.

Table 426. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0(CAP0.0 or CAP1.0 respectively) input.	RO	0	T0CR0 - 0x4000 402C T1CR0 - 0x4000 802C T2CR0 - 0x4009 002C T3CR0 - 0x4009 402C
CR1	Capture Register 1. See CR0 description.	RO	0	T0CR1 - 0x4000 4030 T1CR1 - 0x4000 8030 T2CR1 - 0x4009 0030 T3CR1 - 0x4009 4030
EMR	External Match Register. The EMR controls the external match pins MATn.0-3 (MAT0.0-3 and MAT1.0-3 respectively).	R/W	0	T0EMR - 0x4000 403C T1EMR - 0x4000 803C T2EMR - 0x4009 003C T3EMR - 0x4009 403C
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	T0CTCR - 0x4000 4070 T1CTCR - 0x4000 8070 T2CTCR - 0x4009 0070 T3CTCR - 0x4009 4070

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

6.1 Interrupt Register (T[0/1/2/3]IR - 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000)

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request.

Table 427. Interrupt Register (T[0/1/2/3]IR - addresses 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000) bit description

Bit	Symbol	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
31:6	-	Reserved	-

6.2 Timer Control Register (T[0/1/2/3]CR - 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004)

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

© NXP B.V. 2010. All rights reserved.

Table 428. Timer Control Register (TCR, TIMERn: TnTCR - addresses 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When 1, 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
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.3 Count Control Register (T[0/1/2/3]CTCR - 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070)

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input can not exceed one quarter of the PCLK clock. Consequently, duration of the high/low levels on the same CAP input in this case can not be shorter than 1/(2 PCLK).

Table 429. Count Control Register (T[0/1/2/3]CTCR - addresses 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment the Timer's Prescale Counter (PC), or clear the PC and increment the Timer Counter (TC).	00
		00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. The Prescale Counter is incremented on 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.	

© NXP B.V. 2010. All rights reserved.

Table 429. Count Control Register (T[0/1/2/3]CTCR - addresses 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) bit description

Bit	Symbol	Value	Description	Reset Value
3:2	Count Input Select		When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking.	00
		00	CAPn.0 for TIMERn	
		01	CAPn.1 for TIMERn	
		10	Reserved	
		11	Reserved	
			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.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.4 Timer Counter registers (T0TC - T3TC, 0x4000 4008, 0x4000 8008, 0x4009 0008, 0x4009 4008)

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed.

6.5 Prescale register (T0PR - T3PR, 0x4000 400C, 0x4000 800C, 0x4009 000C, 0x4009 400C)

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

6.6 Prescale Counter register (T0PC - T3PC, 0x4000 4010, 0x4000 8010, 0x4009 0010, 0x4009 4010)

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK. This causes the Timer Counter to increment on every PCLK when PR = 0, every 2 pclks when PR = 1, etc.

© NXP B.V. 2010. All rights reserved.

6.7 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.

6.8 Match Control Register (T[0/1/2/3]MCR - 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014)

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 21–430.

Table 430. Match Control Register (T[0/1/2/3]MCR - addresses 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) bit description

Bit	Symbol	Value	Description	Reset Value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	
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.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6.9 Capture Registers (CR0 - CR1)

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.

6.10 Capture Control Register (T[0/1/2/3]CCR - 0x4000 4028, 0x4000 8028, 0x4009 0028, 0x4009 4028)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the Timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Table 431. Capture Control Register (T[0/1/2/3]CCR - addresses 0x4000 4028, 0x4000 8020, 0x4009 0028, 0x4009 4028) bit description

Bit	Symbol	Value	Description	Reset Value
0	CAP0RE	1	Capture on CAPn.0 rising edge: a sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
1	CAP0FE	1	Capture on CAPn.0 falling edge: a sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
2	CAP0I	1	Interrupt on CAPn.0 event: a CR0 load due to a CAPn.0 event will generate an interrupt.	0
		0	This feature is disabled.	
3	CAP1RE	1	Capture on CAPn.1 rising edge: a sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
4	CAP1FE	1	Capture on CAPn.1 falling edge: a sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
5	CAP1I	1	Interrupt on CAPn.1 event: a CR1 load due to a CAPn.1 event will generate an interrupt.	0
		0	This feature is disabled.	
31:6	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.11 External Match Register (T[0/1/2/3]EMR - 0x4000 403C, 0x4000 803C, 0x4009 003C, 0x4009 403C)

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the Timer number, 0 or 1, and “m” represent a Match number, 0 through 3.

© NXP B.V. 2010. All rights reserved.

Match events for Match 0 and Match 1 in each timer can cause a DMA request, see Section 21–6.12.

Table 432. External Match Register (T[0/1/2/3]EMR - addresses 0x4000 403C, 0x4000 803C, 0x4009 003C, 0x4009 403C) bit description

Bit	Symbol	Description	Reset Value
0	EM0	External Match 0. When a match occurs between the TC and MR0, this bit can either toggle, go low, go high, or do nothing, depending on bits 5:4 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
1	EM1	External Match 1. When a match occurs between the TC and MR1, this bit can either toggle, go low, go high, or do nothing, depending on bits 7:6 of this register. This bit can be driven onto a MATn.1 pin, in a positive-logic manner (0 = low, 1 = high).	0
2	EM2	External Match 2. When a match occurs between the TC and MR2, this bit can either toggle, go low, go high, or do nothing, depending on bits 9:8 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
3	EM3	External Match 3. When a match occurs between the TC and MR3, this bit can either toggle, go low, go high, or do nothing, depending on bits 11:10 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
5:4	EMC0	External Match Control 0. Determines the functionality of External Match 0. Table 21–433 shows the encoding of these bits.	00
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 21–433 shows the encoding of these bits.	00
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 21–433 shows the encoding of these bits.	00
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 21–433 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 433. External Match Control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

6.12 DMA operation

DMA requests are generated by 0 to 1 transitions of the External Match 0 and 1 bits of each timer. In order to have an effect, the GPDMA must be configured and the relevant timer DMA request selected as a DMA source via the DMAREQSEL register, see Section 31–5.15.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See Section 21–6.1. A DMA request will be cleared automatically when it is acted upon by the GPDMA controller.

© NXP B.V. 2010. All rights reserved.

7. Example timer operation

Figure 21–113 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 21–114 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.

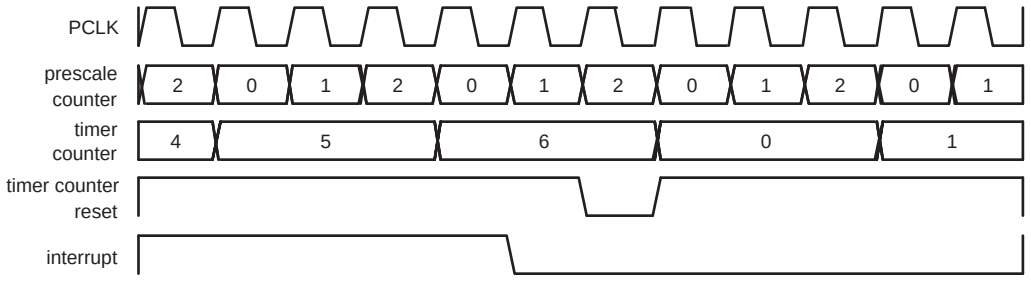


Fig 113. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

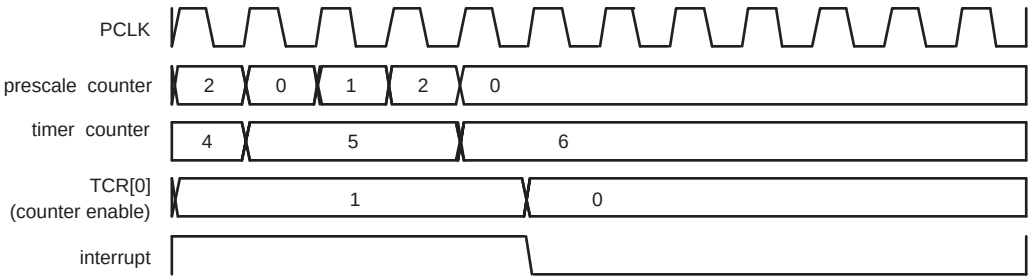


Fig 114. A timer Cycle in Which PR=2, MRx=6, and both interrupt and stop on match are enabled

© NXP B.V. 2010. All rights reserved.

8. Architecture

The block diagram for TIMER/COUNTER0 and TIMER/COUNTER1 is shown in Figure 21–115.

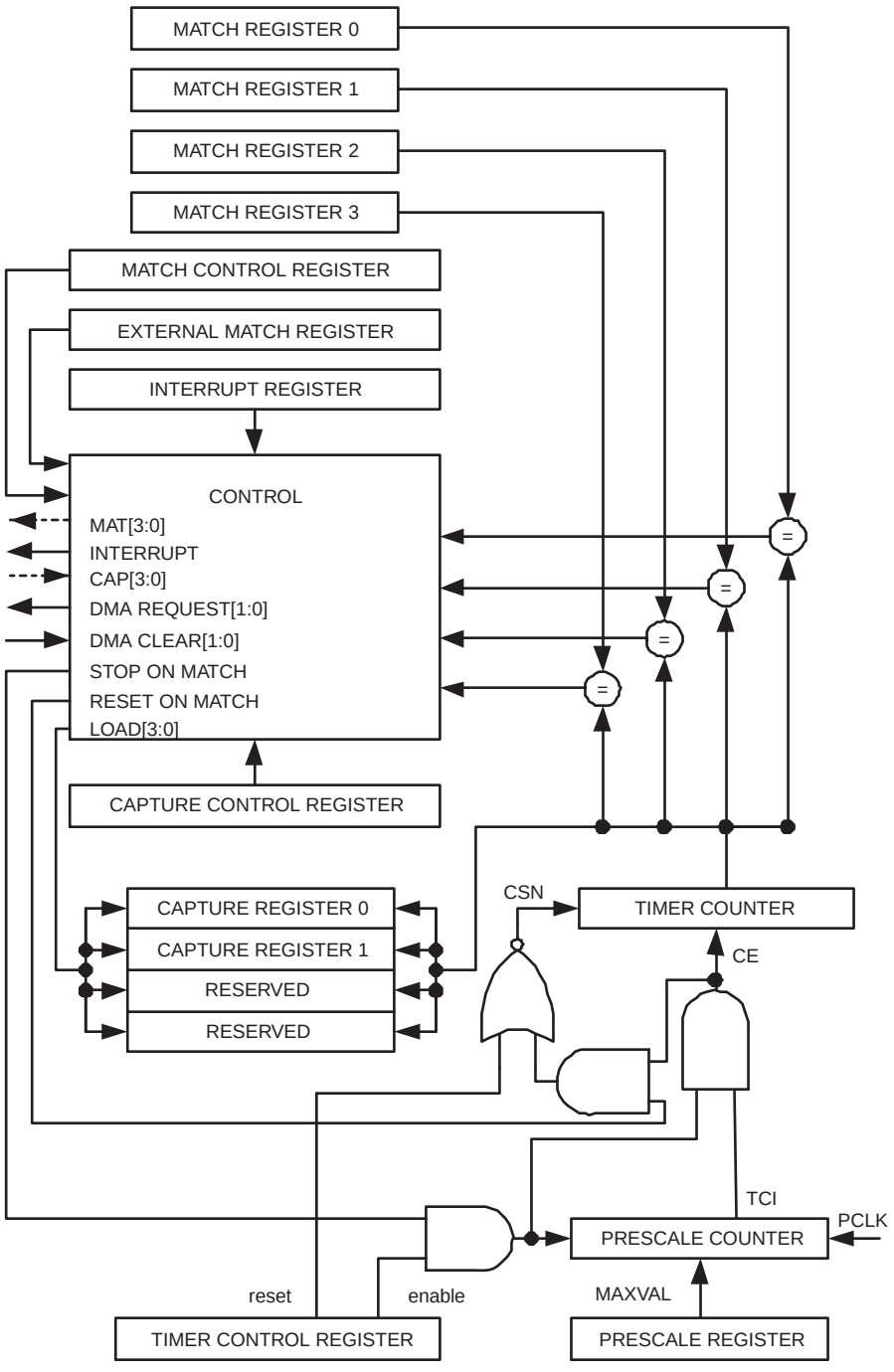


Fig 115. Timer block diagram

© NXP B.V. 2010. All rights reserved.

1. Features

- 32-bit counter running from PCLK. Counter can be free-running, or be reset by a generated interrupt.
- 32-bit compare value.
- 32-bit compare mask. An interrupt is generated when the counter value equals the compare value, after masking. This allows for combinations not possible with a simple compare.

2. Description

The Repetitive Interrupt Timer provides a versatile means of generating interrupts at specified time intervals, without using a standard timer. It is intended for repeating interrupts that aren't related to Operating System interrupts. However, it could be used as an alternative to the Cortex-M3 System Tick Timer (Section 23–1) if there are different system requirements.

3. Register description

Table 434. Repetitive Interrupt Timer register map

Name	Description	Access	Reset value[1]	Address
RICOMPVAL	Compare register	R/W	0xFFFF FFFF	0x400B 0000
RIMASK	Mask register. This register holds the 32-bit mask value. A '1' written to any bit will force a compare on the corresponding bit of the counter and compare register.	R/W	0	0x400B 0004
RICTRL	Control register.	R/W	0xC	0x400B 0008
RICOUNTER	32-bit counter	R/W	0	0x400B 000C

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

3.1 RI Compare Value register (RICOMPVAL - 0x400B 0000)

Table 435. RI Compare Value register (RICOMPVAL - address 0x400B 0000) bit description

Bit	Symbol	Description	Reset value
31:0	RICOMP	Compare register. Holds the compare value which is compared to the counter.	0xFFFF FFFF

3.2 RI Mask register (RIMASK - 0x400B 0004)

Table 436. RI Compare Value register (RICOMPVAL - address 0x400B 0004) bit description

Bit	Symbol	Description	Reset value
31:0	RIMASK	Mask register. This register holds the 32-bit mask value. A '1' written to any bit will force a compare on the corresponding bit of the counter and compare register.	0

© NXP B.V. 2010. All rights reserved.

3.3 RI Control register (RICTRL - 0x400B 0008)

Table 437. RI Control register (RICTRL - address 0x400B 0008) bit description

Bit	Symbol	Value	Description	Reset value
0	RITINT		Interrupt flag	0
		1	This bit is set to 1 by hardware whenever the counter value equals the masked compare value specified by the contents of RICOMPVAL and RIMASK registers. Writing a 1 to this bit will clear it to 0. Writing a 0 has no effect.	
		0	The counter value does not equal the masked compare value.	
1	RITENCLR		Timer enable clear	
		1	The timer will be cleared to 0 whenever the counter value equals the masked compare value specified by the contents of RICOMPVAL and RIMASK registers. This will occur on the same clock that sets the interrupt flag.	0
		0	The timer will not be cleared to 0.	
2	RITENBR		Timer enable for debug	1
		1	The timer is halted when the processor is halted for debugging.	
		0	Debug has no effect on the timer operation.	
3	RITEN		Timer enable.	1
		1	Timer enabled. **Remark:** This can be overruled by a debug halt if enabled in bit 2.	
		0	Timer disabled.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

3.4 RI Counter register (RICOUNTER - 0x400B 000C)

Table 438. RI Counter register (RICOUNTER - address 0x400B 000C) bit description

Bit	Symbol	Description	Reset value
31:0	RICOUNTER	32-bit up counter. Counts continuously unless RITEN bit in RICTRL register is cleared or debug mode is entered (if enabled by the RITNEBR bit in RICTRL). Can be loaded to any value in software.	0

4. RI timer operation

Following reset, the counter begins counting up from 00000000h. Whenever the counter value equals the value programmed into the RICOMPVAL register the interrupt flag will be set. Any bit or combination of bits can be removed from this comparison (i.e. forced to compare) by writing a ‘1’ to the corresponding bit(s) in the RIMASK register. If the enable_clr bit is low (default state), a valid comparison ONLY causes the interrupt flag to be set. It has no effect on the count sequence. Counting continues as usual. When the counter reaches FFFFFFFFh it rolls-over to 00000000h on the next clock and continues counting. If the enable_clr bit is set to ‘1’ a valid comparison will also cause the counter to be reset to zero. Counting will resume from there on the next clock edge.

© NXP B.V. 2010. All rights reserved.

Counting can be halted in software by writing a ‘0’ to the Enable_Timer bit - RICTRL(2). Counting will also be halted when the processor is halted for debugging provided the Enable_Break bit – RICTRL(1) is set. Both the Enable_Timer and Enable_Break bits are set on reset.

The interrupt flag can be cleared in software by writing a ‘1’ to the Interrupt bit – RICTRL(0).

Software can load the counter to any value at any time by writing to RICOUNTER.

The counter (RICOUNTER), RICOMPVAL register, RIMASK register and RICTRL register can all be read by software at any time.

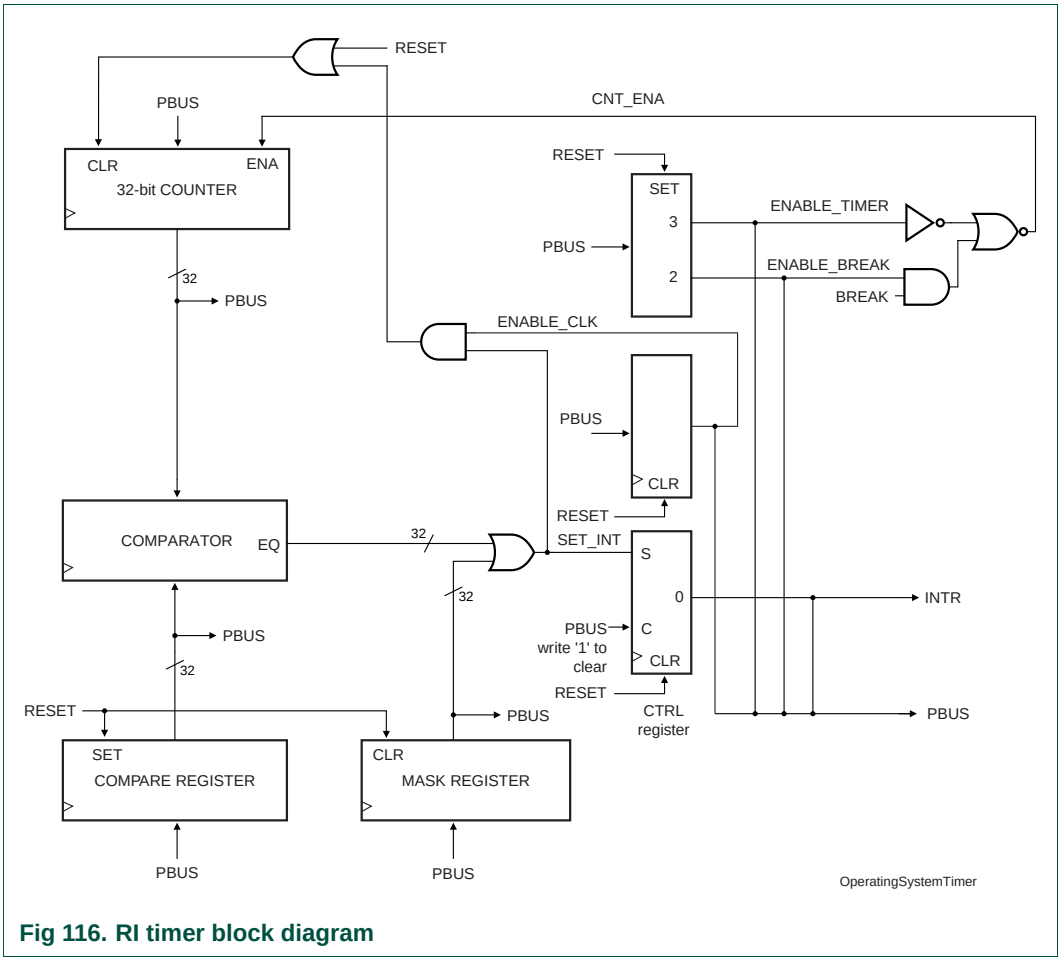


Fig 116. RI timer block diagram

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 23: LPC17xx System Tick Timer

Rev. 01 — 4 January 2010 User manual

1. Basic configuration

The System Tick Timer is configured using the following registers:

1. Clock Source: Select either the internal CCLK or external STCLK (P3.26) clock as the source in the STCTRL register.
2. Pins: If STCLK (P3.26) was selected as clock source enable the STCLK pin function in the PINMODE register (Section 8–5).
3. Interrupt: The System Tick Timer Interrupt is enabled in the NVIC using the appropriate Interrupt Set Enable register.

2. Features

- Times intervals of 10 milliseconds
- Dedicated exception vector
- Can be clocked internally by the CPU clock or by a clock input from a pin (STCLK)

3. Description

The System Tick Timer is an integral part of the Cortex-M3. The System Tick Timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the System Tick Timer is a part of the Cortex-M3, it facilitates porting of software by providing a standard timer that is available on Cortex-M3 based devices.

Refer to the Cortex-M3 User Guide appended to this manual (Section 34–4.4) for details of System Tick Timer operation.

4. Operation

The System Tick Timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The System Tick Timer may be clocked either from the CPU clock or from the external pin STCLK. The STCLK function shares pin P3.26 with other functions, and must be selected for use as the System Tick Timer clock. In order to generate recurring interrupts at a specific interval, the STRELOAD register must be initialized with the correct value for the desired interval. A default value is provided in the STCALIB register and may be changed by software. The default value gives a 10 millisecond interrupt rate if the CPU clock is set to 100 MHz.

The block diagram of the System Tick Timer is shown below in the Figure 23–117.

© NXP B.V. 2010. All rights reserved.

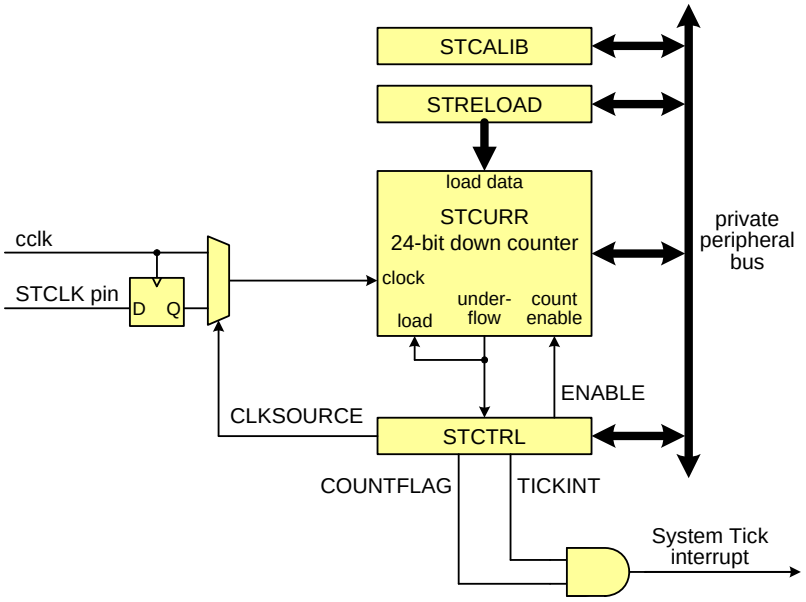


Fig 117. System Tick Timer block diagram

5. Register description

Table 439. System Tick Timer register map

Name	Description	Access	Reset value[1]	Address
STCTRL	System Timer Control and status register	R/W	0x4	0xE000 E010
STRELOAD	System Timer Reload value register	R/W	0	0xE000 E014
STCURR	System Timer Current value register	R/W	0	0xE000 E018
STCALIB	System Timer Calibration value register	R/W	0x000F 423F	0xE000 E01C

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

5.1 System Timer Control and status register (STCTRL - 0xE000 E010)

The STCTRL register contains control information for the System Tick Timer, and provides a status flag.

Table 440. System Timer Control and status register (STCTRL - 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0
2	CLKSOURCE	System Tick clock source selection. When 1, the CPU clock is selected. When 0, the external clock pin (STCLK) is selected.	1

© NXP B.V. 2010. All rights reserved.

Table 440. System Timer Control and status register (STCTRL - 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
15:3	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	COUNTFLAG	System Tick counter flag. This flag is set when the System Tick counter counts down to 0, and is cleared by reading this register.	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.2 System Timer Reload value register (STRELOAD - 0xE000 E014)

The STRELOAD register is set to the value that will be loaded into the System Tick Timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The STCALIB register may be read and used as the value for STRELOAD if the CPU or external clock is running at the frequency intended for use with the STCALIB value.

Table 441. System Timer Reload value register (STRELOAD - 0xE000 E014) bit description

Bit	Symbol	Description	Reset value
23:0	RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.3 System Timer Current value register (STCURR - 0xE000 E018)

The STCURR register returns the current count from the System Tick counter when it is read by software.

Table 442. System Timer Current value register (STCURR - 0xE000 E018) bit description

Bit	Symbol	Description	Reset value
23:0	CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in STCTRL.	0
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.4 System Timer Calibration value register (STCALIB - 0xE000 E01C)

The STCALIB register contains a value that is initialized by the Boot Code to a factory programmed value that is appropriate for generating an interrupt every 10 milliseconds if the System Tick Timer is clocked at a frequency of 100 MHz. This is the intended use of the System Tick Timer by ARM. It can be used to generate interrupts at other frequencies by selecting the correct reload value.

© NXP B.V. 2010. All rights reserved.

Table 443. System Timer Calibration value register (STCALIB - 0xE000 E01C) bit description

Bit	Symbol	Value	Description	Reset value
23:0	TENMS		Reload value to get a 10 millisecond System Tick underflow rate when running at 100 MHz. This value initialized at reset with a factory supplied value selected for the LPC17xx. The provided values of TENMS, SKEW, and NOREF are applicable only when using a CPU clock or external STCLK source of 100 MHz.	0x0F 423F
29:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	SKEW		Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is initialized at reset with a factory supplied value selected for the LPC17xx. See the description of TENMS above. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.	0
31	NOREF		Indicates whether an external reference clock is available. This bit is initialized at reset with a factory supplied value selected for the LPC17xx. See the description of TENMS above. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0

© NXP B.V. 2010. All rights reserved.

6. Example timer calculations

The following examples illustrate selecting System Tick Timer values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the System Tick Timer is intended to be used.

Example 1)

This example is for the System Tick Timer running from the CPU clock (cclk), which is 100 MHz.

STCTRL = 7. This enables the timer and its interrupt, and selects cclk as the clock source.

RELOAD = (cclk / 100) - 1 = 1,000,000 - 1 = 999,999 = 0xF423F

In this case, there is no rounding error, so the result is as accurate as cclk.

Example 2)

This example is for the System Tick Timer running from the CPU clock (cclk), which is 80 MHz.

STCTRL = 7. This enables the timer and its interrupt, and selects cclk as the clock source.

RELOAD = (cclk / 100) - 1 = 800,000 - 1 = 799,999 = 0xC34FF

In this case, there is no rounding error, so the result is as accurate as cclk.

Example 3)

This example is for the CPU clock (cclk) is taken from the Internal RC Oscillator (IRC), factory trimmed to 4 MHz.

STCTRL = 7. This enables the timer and its interrupt, and selects cclk as the clock source.

RELOAD = (F_{IRC} / 100) - 1 = 40,000 - 1 = 39,999 = 0x9C3F

In this case, there is no rounding error, so the result is as accurate as the IRC.

Example 4)

This example is for the System Tick Timer running from an external clock source (the STCLK pin), which in this case happens to be 32.768 kHz.

STCTRL = 3. This enables the timer and its interrupt, and selects the STCLK pin as the clock source. STCLK must be selected as the function of the relevant pin. See Section 8–5.6.

RELOAD = (cclk / 100) - 1 = 327.6 - 1 = 327 (rounded up) = 0x0147

In this case, there is rounding error, so the interrupt rate will drift slightly relative to the input frequency.

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The PWM is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCPWM1.

   Remark: On reset, the PWM is enabled (PCPWM1 = 1).
2. Peripheral clock: In the PCLKSEL0 register (Table 4–40), select PCLK_PWM1.
3. Pins: Select PWM pins through the PINSEL registers. Select pin modes for port pins with PWM1 functions through the PINMODE registers (Section 8–5).
4. Interrupts: See registers PWM1MCR (Table 24–450) and PWM1CCR (Table 24–451) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

2. Features

- Counter or Timer operation (may use the peripheral clock or one of the capture inputs as the clock source).
- Seven match registers allow up to 6 single edge controlled or 3 double edge controlled PWM outputs, or a mix of both types. The match registers also allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Supports single edge controlled and/or double edge controlled PWM outputs. Single edge controlled PWM outputs all go high at the beginning of each cycle unless the output is a constant low. Double edge controlled PWM outputs can have either edge occur at any position within a cycle. This allows for both positive going and negative going pulses.
- Pulse period and width can be any number of timer counts. This allows complete flexibility in the trade-off between resolution and repetition rate. All PWM outputs will occur at the same repetition rate.
- Double edge controlled PWM outputs can be programmed to be either positive going or negative going pulses.
- Match register updates are synchronized with pulse outputs to prevent generation of erroneous pulses. Software must "release" new match values before they can become effective.
- May be used as a standard timer if the PWM mode is not enabled.
- A 32-bit Timer/Counter with a programmable 32-bit prescaler.
- Two 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.

© NXP B.V. 2010. All rights reserved.

3. 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 LPC17xx. 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. 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 24–118 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.

© NXP B.V. 2010. All rights reserved.

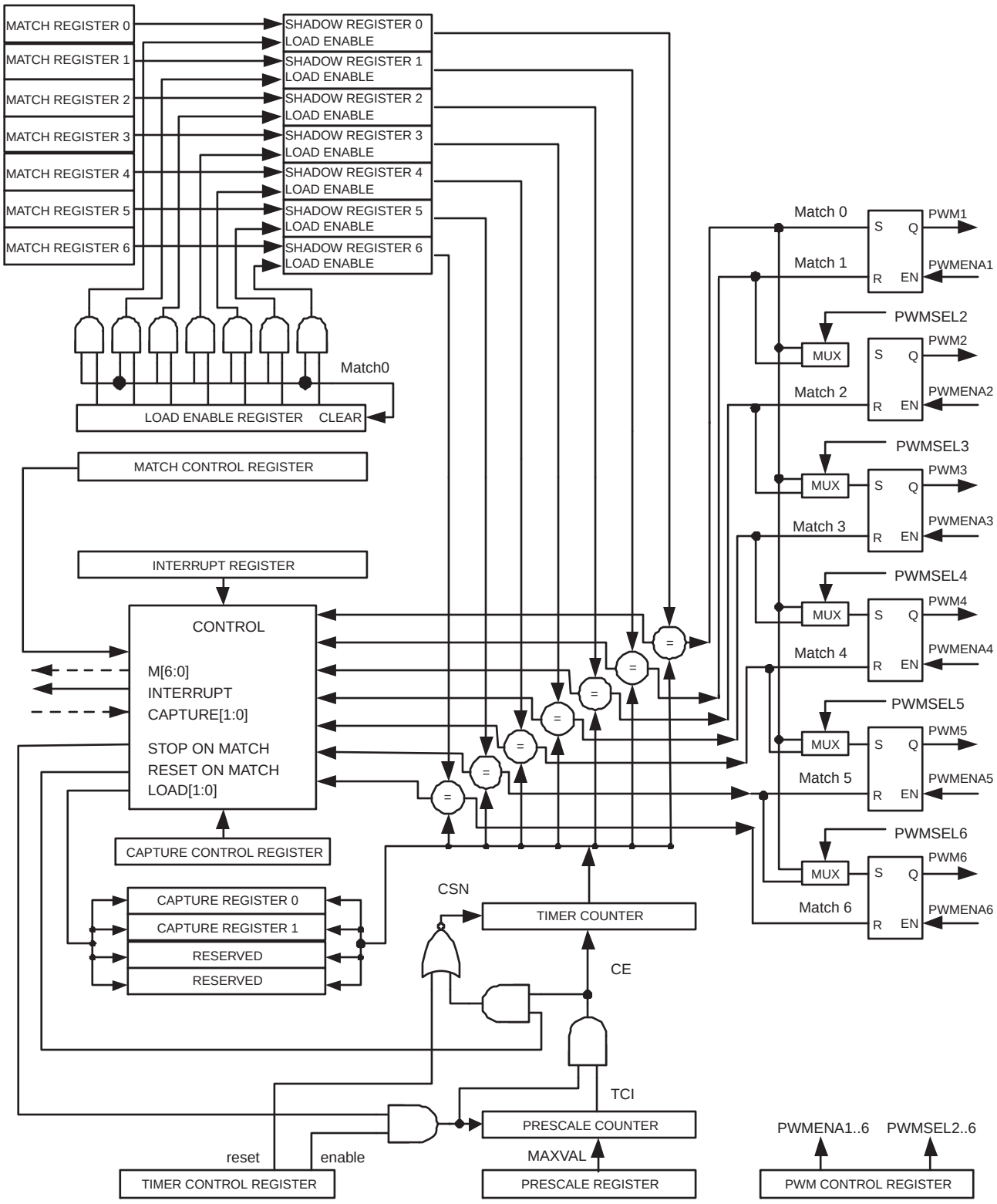
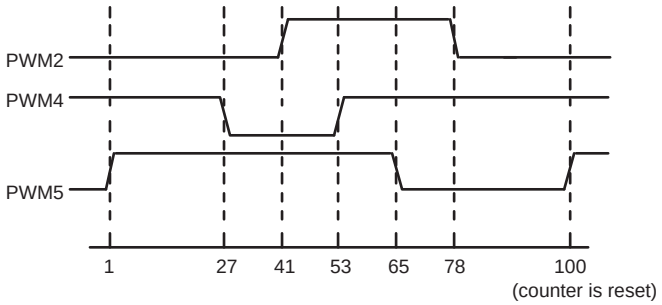


Fig 118. PWM block diagram

© NXP B.V. 2010. All rights reserved.

4. Sample waveform with rules for single and double edge control

A sample of how PWM values relate to waveform outputs is shown in Figure 24–119. PWM output logic is shown in Figure 24–118 that allows selection of either single or double edge controlled PWM outputs via the muxes controlled by the PWMSELn bits. The match register selections for various PWM outputs is shown in Table 24–444. 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 (counter resets to 1).

Match 0 is configured to reset the timer/counter when a match event occurs.

All PWM related Match registers are configured for toggle on match.

Control bits PWMSEL2 and PWMSEL4 are set.

The Match register values are as follows:

MR0 = 100 (PWM rate)

MR1 = 41, MR2 = 78 (PWM2 output)

MR3 = 53, MR4 = 27 (PWM4 output)

MR5 = 65 (PWM5 output)

Fig 119. Sample PWM waveforms

Table 444. Set and reset inputs for PWM Flip-Flops

PWM Channel	Single Edge PWM (PWMSELn = 0)		Double Edge PWM (PWMSELn = 1)	
	Set by	Reset by	Set by	Reset by
1	Match 0	Match 1	Match 0[1]	Match 1[1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2[2]	Match 3[2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4[2]	Match 5[2]
6	Match 0	Match 6	Match 5	Match 6

[1] Identical to single edge mode in this case since Match 0 is the neighboring match register. Essentially, PWM1 cannot be a double edged output.

[2] It is generally not advantageous to use PWM channels 3 and 5 for double edge PWM outputs because it would reduce the number of double edge PWM outputs that are possible. Using PWM 2, PWM4, and PWM6 for double edge PWM outputs provides the most pairings.

© NXP B.V. 2010. All rights reserved.

4.1 Rules for Single Edge Controlled PWM Outputs

1. All single edge controlled PWM outputs go high at the beginning of a PWM cycle unless their match value is equal to 0.
2. Each PWM output will go low when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM rate), the PWM output remains continuously high.

4.2 Rules for Double Edge Controlled PWM Outputs

Five rules are used to determine the next value of a PWM output when a new cycle is about to begin:

1. The match values for the **next** PWM cycle are used at the end of a PWM cycle (a time point which is coincident with the beginning of the next PWM cycle), except as noted in rule 3.
2. A match value equal to 0 or the current PWM rate (the same as the Match channel 0 value) have the same effect, except as noted in rule 3. For example, a request for a falling edge at the beginning of the PWM cycle has the same effect as a request for a falling edge at the end of a PWM cycle.
3. When match values are changing, if one of the "old" match values is equal to the PWM rate, it is used again once if the neither of the new match values are equal to 0 or the PWM rate, and there was no old match value equal to 0.
4. If both a set and a clear of a PWM output are requested at the same time, clear takes precedence. This can occur when the set and clear match values are the same as in, or when the set or clear value equals 0 and the other value equals the PWM rate.
5. If a match value is out of range (i.e. greater than the PWM rate value), no match event occurs and that match channel has no effect on the output. This means that the PWM output will remain always in one state, allowing always low, always high, or "no change" outputs.

5. Pin description

Table 24–445 gives a brief summary of each of PWM related pins.

Table 445. Pin summary

Pin	Type	Description
PWM1[1]	Output	Output from PWM channel 1.
PWM1[2]	Output	Output from PWM channel 2.
PWM1[3]	Output	Output from PWM channel 3.
PWM1[4]	Output	Output from PWM channel 4.
PWM1[5]	Output	Output from PWM channel 5.
PWM1[6]	Output	Output from PWM channel 6.
PCAP1[1:0]	Input	Capture Inputs. A transition on a capture pin can be configured to load the corresponding Capture Register with the value of the Timer Counter and optionally generate an interrupt. The PWM brings out 2 capture inputs.

© NXP B.V. 2010. All rights reserved.

6. Register description

The PWM1 function includes registers as shown in Table 24–446 below.

Table 446. PWM1 register map

Generic Name	Description	Access	Reset Value[1]	PWMn Register Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	PWM1IR - 0x4001 8000
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	PWM1TCR - 0x4001 8004
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	PWM1TC - 0x4001 8008
PR	Prescale Register. The TC is incremented every PR+1 cycles of PCLK.	R/W	0	PWM1PR - 0x4001 800C
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented. The PC is observable and controllable through the bus interface.	R/W	0	PWM1PC - 0x4001 8010
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	PWM1MCR - 0x4001 8014
MR0	Match Register 0. MR0 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC sets any PWM output that is in single-edge mode, and sets PWM1 if it's in double-edge mode.	R/W	0	PWM1MR0 - 0x4001 8018
MR1	Match Register 1. MR1 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM1 in either edge mode, and sets PWM2 if it's in double-edge mode.	R/W	0	PWM1MR1 - 0x4001 801C
MR2	Match Register 2. MR2 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM2 in either edge mode, and sets PWM3 if it's in double-edge mode.	R/W	0	PWM1MR2 - 0x4001 8020
MR3	Match Register 3. MR3 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM3 in either edge mode, and sets PWM4 if it's in double-edge mode.	R/W	0	PWM1MR3 - 0x4001 8024
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	PWM1CCR - 0x4001 8028
CR0	Capture Register 0. CR0 is loaded with the value of the TC when there is an event on the CAPn.0 input.	RO	0	PWM1CR0 - 0x4001 802C
CR1	Capture Register 1. See CR0 description.	RO	0	PWM1CR1 - 0x4001 8030
CR2	Capture Register 2. See CR0 description.	RO	0	PWM1CR2 - 0x4001 8034
CR3	Capture Register 3. See CR0 description.	RO	0	PWM1CR3 - 0x4001 8038

© NXP B.V. 2010. All rights reserved.

Table 446. PWM1 register map

Generic Name	Description	Access	Reset Value[1]	PWMn Register Name & Address
MR4	Match Register 4. MR4 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM4 in either edge mode, and sets PWM5 if it's in double-edge mode.	R/W	0	PWM1MR4 - 0x4001 8040
MR5	Match Register 5. MR5 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM5 in either edge mode, and sets PWM6 if it's in double-edge mode.	R/W	0	PWM1MR5 - 0x4001 8044
MR6	Match Register 6. MR6 can be enabled in the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between this value and the TC clears PWM6 in either edge mode.	R/W	0	PWM1MR6 - 0x4001 8048
PCR	PWM Control Register. Enables PWM outputs and selects PWM channel types as either single edge or double edge controlled.	R/W	0	PWM1PCR - 0x4001 804C
LER	Load Enable Register. Enables use of new PWM match values.	R/W	0	PWM1LER - 0x4001 8050
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	PWM1CTCR - 0x4001 8070

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

6.1 PWM Interrupt Register (PWM1IR - 0x4001 8000)

The PWM Interrupt Register consists of 11 bits (Table 24–447), 7 for the match interrupts and 4 reserved for the future use. If an interrupt is generated then the corresponding bit in the PWMIR will be high. Otherwise, the bit will be low. Writing a logic 1 to the corresponding IR bit will reset the interrupt. Writing a 0 has no effect.

Table 447: PWM Interrupt Register (PWM1IR - address 0x4001 8000) bit description

Bit	Symbol	Description	Reset Value
0	PWMMR0 Interrupt	Interrupt flag for PWM match channel 0.	0
1	PWMMR1 Interrupt	Interrupt flag for PWM match channel 1.	0
2	PWMMR2 Interrupt	Interrupt flag for PWM match channel 2.	0
3	PWMMR3 Interrupt	Interrupt flag for PWM match channel 3.	0
4	PWMCAP0 Interrupt	Interrupt flag for capture input 0	0
5	PWMCAP1 Interrupt	Interrupt flag for capture input 1.	0
7:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
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
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6.2 PWM Timer Control Register (PWM1TCR 0x4001 8004)

The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter. The function of each of the bits is shown in Table 24–448.

Table 448: PWM Timer Control Register (PWM1TCR address 0x4001 8004) bit description

Bit	Symbol	Value	Description	Reset Value
0	Counter Enable	1	The PWM Timer Counter and PWM Prescale Counter are enabled for counting.	0
		0	The counters are disabled.	
1	Counter Reset	1	The PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until this bit is returned to zero.	0
		0	Clear reset.	
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	1	PWM mode is enabled (counter resets to 1). PWM mode causes the shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match Register 0 - MR0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	0
		0	Timer mode is enabled (counter resets to 0).	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.3 PWM Count Control Register (PWM1CTCR - 0x4001 8070)

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting. The function of each of the bits is shown in Table 24–449.

Table 449: PWM Count control Register (PWM1CTCR - address 0x4001 8004) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Counter/ Timer Mode	00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register.	00
		01	Counter Mode: the TC is incremented on rising edges of the PCAP input selected by bits 3:2.	
		10	Counter Mode: the TC is incremented on falling edges of the PCAP input selected by bits 3:2.	
		11	Counter Mode: the TC is incremented on both edges of the PCAP input selected by bits 3:2.	
3:2	Count Input Select		When bits 1:0 of this register are not 00, these bits select which PCAP pin which carries the signal used to increment the TC.	00
		00	PCAP1.0	
		01	PCAP1.1 (Other combinations are reserved)	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6.4 PWM Match Control Register (PWM1MCR - 0x4001 8014)

The PWM Match Control Registers are used to control what operations are performed when one of the PWM Match Registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 24–450.

Table 450: Match Control Register (PWM1MCR - address 0x4000 4014) 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.	
4	PWMMR1R	1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.	0
		0	This feature is disabled.	
5	PWMMR1S	1	Stop on PWMMR1: the PWMTC and PWMPC will be stopped and PWMTCR[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	
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.	

© NXP B.V. 2010. All rights reserved.

Table 450: Match Control Register (PWM1MCR - address 0x4000 4014) bit description

Bit	Symbol	Value	Description	Reset Value
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
31:21	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.5 PWM Capture Control Register (PWM1CCR - 0x4001 8028)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when a capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the descriptions below, “n” represents the Timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Table 451: PWM Capture Control Register (PWM1CCR - address 0x4001 8028) bit description

Bit	Symbol	Value	Description	Reset Value
0	Capture on CAPn.0 rising edge	0	This feature is disabled.	0
		1	A synchronously sampled rising edge on the CAPn.0 input will cause CR0 to be loaded with the contents of the TC.	
1	Capture on CAPn.0 falling edge	0	This feature is disabled.	0
		1	A synchronously sampled falling edge on CAPn.0 will cause CR0 to be loaded with the contents of TC.	
2	Interrupt on CAPn.0 event	0	This feature is disabled.	0
		1	A CR0 load due to a CAPn.0 event will generate an interrupt.	

© NXP B.V. 2010. All rights reserved.

Table 451: PWM Capture Control Register (PWM1CCR - address 0x4001 8028) bit description

Bit	Symbol	Value	Description	Reset Value
3	Capture on CAPn.1rising edge	0	This feature is disabled.	0
		1	A synchronously sampled rising edge on the CAPn.1 input will cause CR1 to be loaded with the contents of the TC.	
4	Capture on CAPn.1falling edge	0	This feature is disabled.	0
		1	A synchronously sampled falling edge on CAPn.1 will cause CR1 to be loaded with the contents of TC.	
5	Interrupt on CAPn.1 event	0	This feature is disabled.	0
		1	A CR1 load due to a CAPn.1 event will generate an interrupt.	
31:6	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.6 PWM Control Register (PWM1PCR - 0x4001 804C)

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 24–452.

Table 452: PWM Control Register (PWM1PCR - address 0x4001 804C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Unused		Unused, always zero.	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.	
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.	

© NXP B.V. 2010. All rights reserved.

Table 452: PWM Control Register (PWM1PCR - address 0x4001 804C) bit description

Bit	Symbol	Value	Description	Reset Value
14	PWMENA6	1	The PWM6 output enabled.	0
		0	The PWM6 output disabled.	
31:15	Unused		Unused, always zero.	NA

6.7 PWM Latch Enable Register (PWM1LER - 0x4001 8050)

The PWM Latch Enable Registers are used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to the location of a PWM Match register while the Timer is in PWM mode, the value is captured, but not used immediately.

When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the shadow 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 LER. 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 LER is shown in Table 24–453.

Table 453: PWM Latch Enable Register (PWM1LER - address 0x4001 8050) 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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	0

© NXP B.V. 2010. All rights reserved.

Table 453: PWM Latch Enable Register (PWM1LER - address 0x4001 8050) bit description

Bit	Symbol	Description	Reset Value
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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	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 24–6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)".	0
31:7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

1. Introduction

The Motor Control PWM (MCPWM) is optimized for three-phase AC and DC motor control applications, but can be used in many other applications that need timing, counting, capture, and comparison.

2. Description

The MCPWM contains three independent channels, each including:

- a 32-bit Timer/Counter (TC)
- a 32-bit Limit register (LIM)
- a 32-bit Match register (MAT)
- a 10-bit dead-time register (DT) and an associated 10-bit dead-time counter
- a 32-bit capture register (CAP)
- two modulated outputs (MCOA and MCOB) with opposite polarities
- a period interrupt, a pulse-width interrupt, and a capture interrupt

Input pins MCI0-2 can trigger TC capture or increment a channel's TC. A global Abort input can force all of the channels into "A passive" state and cause an interrupt.

3. Pin description

Table 25–454 lists the MCPWM pins.

Table 454. Pin summary

Pin	Type	Description
MCOA0, MCOA1, MCOA2	O	Output A for channels 0, 1, 2
MCOB0, MCOB1, MCOB2	O	Output B for channels 0, 1, 2
$\overline{\text{MCABORT}}$	I	Low-active Fast Abort
MCI0, MCI1, MCI2	I	Input for channels 0, 1, 2

© NXP B.V. 2010. All rights reserved.

4. Block Diagram

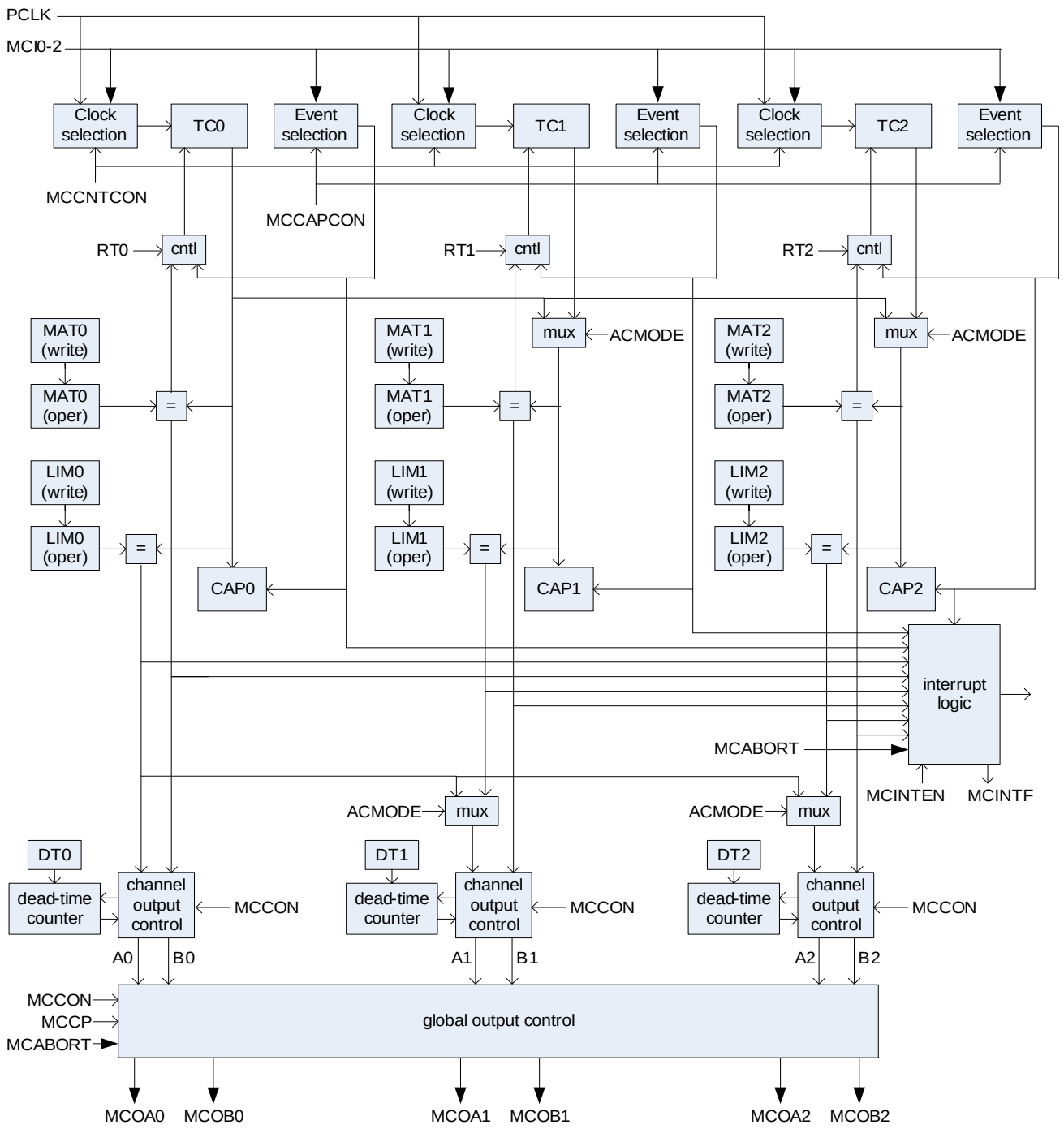


Fig 120. MCPWM Block Diagram

© NXP B.V. 2010. All rights reserved.

5. Configuring other modules for MCPWM use

Configure the following registers in other modules before using the Motor Control PWM:

1. Power: in the PCONP register (Table 4–46), set bit PCMCPWM.
   Remark: On reset the MCPWM is disabled (PCMCPWM = 0).
2. Peripheral clock: in the PCLKSEL1 register (Table 4–40) select PCLK_MCPWM.
3. Pins: select MCPWM functions through the PINSEL registers. Select modes for these pins through the PINMODE registers (Section 8–5).
4. Interrupts: See Section 25–7.3 for motor control PWM related interrupts. Interrupts can be enabled in the NVIC using the appropriate Interrupt Set Enable register.

6. General Operation

Section 25–8 includes detailed descriptions of the various modes of MCPWM operation, but a quick preview here will provide background for the register descriptions below.

The MCPWM includes 3 channels, each of which controls a pair of outputs that in turn can control something off-chip, like one set of coils in a motor. Each channel includes a Timer/Counter (TC) register that is incremented by a processor clock (timer mode) or by an input pin (counter mode).

Each channel has a Limit register that is compared to the TC value, and when a match occurs the TC is “recycled” in one of two ways. In “edge-aligned mode” the TC is reset to 0, while in “centered mode” a match switches the TC into a state in which it decrements on each processor clock or input pin transition until it reaches 0, at which time it starts counting up again.

Each channel also includes a Match register that holds a smaller value than the Limit register. In edge-aligned mode the channel’s outputs are switched whenever the TC matches either the Match or Limit register, while in center-aligned mode they are switched only when it matches the Match register.

So the Limit register controls the period of the outputs, while the Match register controls how much of each period the outputs spend in each state. Having a small value in the Limit register minimizes “ripple” if the output is integrated into a voltage, and allows the MCPWM to control devices that operate at high speed.

The “downside” of small values in the Limit register is that they reduce the resolution of the duty cycle controlled by the Match register. If you have 8 in the Limit register, the Match register can only select the duty cycle among 0%, 12.5%, 25%, …, 87.5%, or 100%. In general, the resolution of each step in the Match value is 1 divided by the Limit value.

This trade-off between resolution and period/frequency is inherent in the design of pulse width modulators.

© NXP B.V. 2010. All rights reserved.

7. Register description

"Control" registers and "interrupt" registers have separate read, set, and clear addresses. Reading such a register's read address(e.g. MCCON) yields the state of the register bits. Writing ones to the set address (e.g. MCCON_SET) sets register bit(s), and writing ones to the clear address (e.g. MCCON_CLR) clears register bit(s).

The Capture registers (MCCAP) are read-only, and the write-only MCCAP_CLR address can be used to clear one or more of them. All the other MCPWM registers (MCTIM, MCPER, MCPW, MCDEADTIME, and MCCP) are normal read-write registers.

Table 455. Motor Control Pulse Width Modulator (MCPWM) register map

Name	Description	Access	Reset value	Address
MCCON	PWM Control read address	RO	0	0x400B 8000
MCCON_SET	PWM Control set address	WO	-	0x400B 8004
MCCON_CLR	PWM Control clear address	WO	-	0x400B 8008
MCCAPCON	Capture Control read address	RO	0	0x400B 800C
MCCAPCON_SET	Capture Control set address	WO	-	0x400B 8010
MCCAPCON_CLR	Event Control clear address	WO	-	0x400B 8014
MCTC0	Timer Counter register, channel 0	R/W	0	0x400B 8018
MCTC1	Timer Counter register, channel 1	R/W	0	0x400B 801C
MCTC2	Timer Counter register, channel 2	R/W	0	0x400B 8020
MCLIM0	Limit register, channel 0	R/W	0xFFFF FFFF	0x400B 8024
MCLIM1	Limit register, channel 1	R/W	0xFFFF FFFF	0x400B 8028
MCLIM2	Limit register, channel 2	R/W	0xFFFF FFFF	0x400B 802C
MCMAT0	Match register, channel 0	R/W	0xFFFF FFFF	0x400B 8030
MCMAT1	Match register, channel 1	R/W	0xFFFF FFFF	0x400B 8034
MCMAT2	Match register, channel 2	R/W	0xFFFF FFFF	0x400B 8038
MCDT	Dead time register	R/W	0x3FFF FFFF	0x400B 803C
MCCP	Commutation Pattern register	R/W	0	0x400B 8040
MCCAP0	Capture register, channel 0	RO	0	0x400B 8044
MCCAP1	Capture register, channel 1	RO	0	0x400B 8048
MCCAP2	Capture register, channel 2	RO	0	0x400B 804C
MCINTEN	Interrupt Enable read address	RO	0	0x400B 8050
MCINTEN_SET	Interrupt Enable set address	WO	-	0x400B 8054
MCINTEN_CLR	Interrupt Enable clear address	WO	-	0x400B 8058
MCCNTCON	Count Control read address	RO	0	0x400B 805C
MCCNTCON_SET	Count Control set address	WO	-	0x400B 8060
MCCNTCON_CLR	Count Control clear address	WO	-	0x400B 8064
MCINTF	Interrupt flags read address	RO	0	0x400B 8068
MCINTF_SET	Interrupt flags set address	WO	-	0x400B 806C
MCINTF_CLR	Interrupt flags clear address	WO	-	0x400B 8070
MCCAP_CLR	Capture clear address	WO	-	0x400B 8074

© NXP B.V. 2010. All rights reserved.

7.1 MCPWM Control register

7.1.1 MCPWM Control read address (MCCON - 0x400B 8000)

The MCCON register controls the operation of all channels of the PWM. This address is read-only, but the underlying register can be modified by writing to addresses MCCON_SET and MCCON_CLR.

Table 456. MCPWM Control read address (MCCON - 0x400B 8000) bit description

Bit	Symbol	Value	Description	Reset value
0	RUN0		Stops/starts timer channel 0.	0
		0	Stop.	
		1	Run.	
1	CENTER0		Edge/center aligned operation for channel 0.	0
		0	Edge-aligned.	
		1	Center-aligned.	
2	POLA0		Selects polarity of the MCOA0 and MCOB0 pins.	0
		0	Passive state is LOW, active state is HIGH.	
		1	Passive state is HIGH, active state is LOW.	
3	DTE0		Controls the dead-time feature for channel 0.	0
		0	Dead-time disabled.	
		1	Dead-time enabled.	
4	DISUP0		Enable/disable updates of functional registers for channel 0 (see Section 25–8.2).	0
		0	Functional registers are updated from the write registers at the end of each PWM cycle.	
		1	Functional registers remain the same as long as the timer is running.	
7:5	-	-	Reserved.	
8	RUN1		Stops/starts timer channel 1.	0
		0	Stop.	
		1	Run.	
9	CENTER1		Edge/center aligned operation for channel 1.	0
		0	Edge-aligned.	
		1	Center-aligned.	
10	POLA1		Selects polarity of the MCOA1 and MCOB1 pins.	0
		0	Passive state is LOW, active state is HIGH.	
		1	Passive state is HIGH, active state is LOW.	
11	DTE1		Controls the dead-time feature for channel 1.	0
		0	Dead-time disabled.	
		1	Dead-time enabled.	
12	DISUP1		Enable/disable updates of functional registers for channel 1 (see Section 25–8.2).	0
		0	Functional registers are updated from the write registers at the end of each PWM cycle.	
		1	Functional registers remain the same as long as the timer is running.	
15:13	-	-	Reserved.	0

© NXP B.V. 2010. All rights reserved.

Table 456. MCPWM Control read address (MCCON - 0x400B 8000) bit description

Bit	Symbol	Value	Description	Reset value
16	RUN2		Stops/starts timer channel 2.	0
		0	Stop.	
		1	Run.	
17	CENTER2		Edge/center aligned operation for channel 2.	0
		0	Edge-aligned.	
		1	Center-aligned.	
18	POLA2		Selects polarity of the MCOA2 and MCOB2 pins.	0
		0	Passive state is LOW, active state is HIGH.	
		1	Passive state is HIGH, active state is LOW.	
19	DTE2		Controls the dead-time feature for channel 1.	0
		0	Dead-time disabled.	
		1	Dead-time enabled.	
20	DISUP2		Enable/disable updates of functional registers for channel 2 (see Section 25–8.2).	0
		0	Functional registers are updated from the write registers at the end of each PWM cycle.	
		1	Functional registers remain the same as long as the timer is running.	
28:21	-	-	Reserved.	
29	INVBDC		Controls the polarity of the MCOB outputs for all 3 channels. This bit is typically set to 1 only in 3-phase DC mode.	
		0	The MCOB outputs have opposite polarity from the MCOA outputs (aside from dead time).	
		1	The MCOB outputs have the same basic polarity as the MCOA outputs. (see Section 25–8.6)	
30	ACMODE		3-phase AC mode select (see Section 25–8.7).	0
		0	3-phase AC-mode off: Each PWM channel uses its own timer-counter and period register.	
		1	3-phase AC-mode on: All PWM channels use the timer-counter and period register of channel 0.	
31	DCMODE		3-phase DC mode select (see Section 25–8.6).	0
		0	3-phase DC mode off: PWM channels are independent (unless bit ACMODE = 1)	
		1	3-phase DC mode on: The internal MCOA0 output is routed through the MCCP (i.e. a mask) register to all six PWM outputs.	

7.1.2 MCPWM Control set address (MCCON_SET - 0x400B 8004)

Writing ones to this write-only address sets the corresponding bits in MCCON.

Table 457. MCPWM Control set address (MCCON_SET - 0x400B 8004) bit description

Bit	Description
31:0	Writing ones to this address sets the corresponding bits in the MCCON register. See Table 25–456.

7.1.3 MCPWM Control clear address (MCCON_CLR - 0x400B 8008)

Writing ones to this write-only address clears the corresponding bits in MCCON.

© NXP B.V. 2010. All rights reserved.

Table 458. MCPWM Control clear address (MCCON_CLR - 0x400B 8008) bit description

Bit	Description
31:0	Writing ones to this address clears the corresponding bits in the MCCON register. See Table 25–456.

7.2 MCPWM Capture Control register

7.2.1 MCPWM Capture Control read address (MCCAPCON - 0x400B 800C)

The MCCAPCON register controls detection of events on the MCI0-2 inputs for all MCPWM channels. Any of the three MCI inputs can be used to trigger a capture event on any or all of the three channels. This address is read-only, but the underlying register can be modified by writing to addresses MCCAPCON_SET and MCCAPCON_CLR.

Table 459. MCPWM Capture Control read address (MCCAPCON - 0x400B 800C) bit description

Bit	Symbol	Description	Reset Value
0	CAP0MCI0_RE	A 1 in this bit enables a channel 0 capture event on a rising edge on MCI0.	0
1	CAP0MCI0_FE	A 1 in this bit enables a channel 0 capture event on a falling edge on MCI0.	0
2	CAP0MCI1_RE	A 1 in this bit enables a channel 0 capture event on a rising edge on MCI1.	0
3	CAP0MCI1_FE	A 1 in this bit enables a channel 0 capture event on a falling edge on MCI1.	0
4	CAP0MCI2_RE	A 1 in this bit enables a channel 0 capture event on a rising edge on MCI2.	0
5	CAP0MCI2_FE	A 1 in this bit enables a channel 0 capture event on a falling edge on MCI2.	0
6	CAP1MCI0_RE	A 1 in this bit enables a channel 1 capture event on a rising edge on MCI0.	0
7	CAP1MCI0_FE	A 1 in this bit enables a channel 1 capture event on a falling edge on MCI0.	0
8	CAP1MCI1_RE	A 1 in this bit enables a channel 1 capture event on a rising edge on MCI1.	0
9	CAP1MCI1_FE	A 1 in this bit enables a channel 1 capture event on a falling edge on MCI1.	0
10	CAP1MCI2_RE	A 1 in this bit enables a channel 1 capture event on a rising edge on MCI2.	0
11	CAP1MCI2_FE	A 1 in this bit enables a channel 1 capture event on a falling edge on MCI2.	0
12	CAP2MCI0_RE	A 1 in this bit enables a channel 2 capture event on a rising edge on MCI0.	0
13	CAP2MCI0_FE	A 1 in this bit enables a channel 2 capture event on a falling edge on MCI0.	0
14	CAP2MCI1_RE	A 1 in this bit enables a channel 2 capture event on a rising edge on MCI1.	0
15	CAP2MCI1_FE	A 1 in this bit enables a channel 2 capture event on a falling edge on MCI1.	0
16	CAP2MCI2_RE	A 1 in this bit enables a channel 2 capture event on a rising edge on MCI2.	0
17	CAP2MCI2_FE	A 1 in this bit enables a channel 2 capture event on a falling edge on MCI2.	0
18	RT0	If this bit is 1, TC0 is reset by a channel 0 capture event.	0
19	RT1	If this bit is 1, TC1 is reset by a channel 1 capture event.	0
20	RT2	If this bit is 1, TC2 is reset by a channel 2 capture event.	0
21	HNFCAP0	Hardware noise filter: if this bit is 1, channel 0 capture events are delayed as described in Section 25–8.4.	0
22	HNFCAP1	Hardware noise filter: if this bit is 1, channel 1 capture events are delayed as described in Section 25–8.4.	0
23	HNFCAP2	Hardware noise filter: if this bit is 1, channel 2 capture events are delayed as described in Section 25–8.4.	0
31:24	-	Reserved.	-

7.2.2 MCPWM Capture Control set address (MCCAPCON_SET - 0x400B 8010)

Writing ones to this write-only address sets the corresponding bits in MCCAPCON.

© NXP B.V. 2010. All rights reserved.

Table 460. MCPWM Capture Control set address (MCCAPCON_SET - 0x400B 8010) bit description

Bit	Description
31:0	Writing ones to this address sets the corresponding bits in the MCCAPCON register. See Table 25–459.

7.2.3 MCPWM Capture control clear address (MCCAPCON_CLR - 0x400B 8014)

Writing ones to this write-only address clears the corresponding bits in MCCAPCON.

Table 461. MCPWM Capture control clear register (MCCAPCON_CLR - address 0x400B 8014) bit description

Bit	Description
31:0	Writing ones to this address clears the corresponding bits in the MCCAPCON register. See Table 25–459.

7.3 MCPWM Interrupt registers

The Motor Control PWM module includes the following interrupt sources:

Table 462. Motor Control PWM interrupts

Symbol	Description
ILIM0/1/2	Limit interrupts for channels 0, 1, 2.
IMAT0/1/2	Match interrupts for channels 0, 1, 2.
ICAP0/1/2	Capture interrupts for channels 0, 1, 2.
ABORT	Fast abort interrupt

All MCPWM interrupt registers contain one bit for each source as shown in Table 25–463.

Table 463. Interrupt sources bit allocation table

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	ABORT	-	-	-	-	ICAP2	IMAT2	ILIM2
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	-	ICAP1	IMAT1	ILIM1	-	ICAP0	IMAT0	ILIM0

7.3.1 MCPWM Interrupt Enable read address (MCINTEN - 0x400B 8050)

The MCINTEN register controls which of the MCPWM interrupts are enabled. This address is read-only, but the underlying register can be modified by writing to addresses MCINTEN_SET and MCINTEN_CLR.

Table 464. MCPWM Interrupt Enable read address (MCINTEN - 0x400B 8050) bit description

Bit	Value	Description	Reset value
31:0		See Table 25–463 for the bit allocation.	0
	1	Interrupt enabled.	
	0	Interrupt disabled.	

© NXP B.V. 2010. All rights reserved.

7.3.2 MCPWM Interrupt Enable set address (MCINTEN_SET - 0x400B 8054)

Writing ones to this write-only address sets the corresponding bits in MCINTEN, thus enabling interrupts.

Table 465. PWM interrupt enable set register (MCINTEN_SET - address 0x400B 8054) bit description

Bit	Description
31:0	Writing ones to this address sets the corresponding bits in MCINTEN, thus enabling interrupts. See Table 25–463.

7.3.3 MCPWM Interrupt Enable clear address (MCINTEN_CLR - 0x400B 8058)

Writing ones to this write-only address clears the corresponding bits in MCINTEN, thus disabling interrupts.

Table 466. PWM interrupt enable clear register (MCINTEN_CLR - address 0x400B 8058) bit description

Bit	Description
31:0	Writing ones to this address clears the corresponding bits in MCINTEN, thus disabling interrupts. See Table 25–463.

7.3.4 MCPWM Interrupt Flags read address (MCINTF - 0x400B 8068)

The MCINTF register includes all MCPWM interrupt flags, which are set when the corresponding hardware event occurs, or when ones are written to the MCINTF_SET address. When corresponding bits in this register and MCINTEN are both 1, the MCPWM asserts its interrupt request to the Interrupt Controller module. This address is read-only, but the bits in the underlying register can be modified by writing ones to addresses MCINTF_SET and MCINTF_CLR.

Table 467. MCPWM Interrupt Flags read address (MCINTF - 0x400B 8068) bit description

Bit	Value	Description	Reset value
31:0		See Table 25–463 for the bit allocation.	0
	1	If the corresponding bit in MCINTEN is 1, the MCPWM module is asserting its interrupt request to the Interrupt Controller.	
	0	This interrupt source is not contributing to the MCPWM interrupt request.	

7.3.5 MCPWM Interrupt Flags set address (MCINTF_SET - 0x400B 806C)

Writing one(s) to this write-only address sets the corresponding bit(s) in MCINTF, thus possibly simulating hardware interrupt(s).

Table 468. MCPWM Interrupt Flags set address (PWMINTF_SET - 0x400B 806C) bit description

Bit	Description
31:0	Writing one(s) to this write-only address sets the corresponding bit(s) in the MCINTF register, thus possibly simulating hardware interrupt(s). See Table 25–463.

7.3.6 MCPWM Interrupt Flags clear address (MCINTF_CLR - 0x400B 8070)

Writing one(s) to this write-only address sets the corresponding bit(s) in MCINTF, thus clearing the corresponding interrupt request(s). This is typically done in interrupt service routines.

© NXP B.V. 2010. All rights reserved.

Table 469. MCPWM Interrupt Flags clear address (PWMINTF_CLR - 0x400B 8070) bit description

Bit	Description
31:0	Writing one(s) to this write-only address sets the corresponding bit(s) in the MCINTF register, thus clearing the corresponding interrupt request(s). See Table 25–463.

7.4 MCPWM Count Control register

7.4.1 MCPWM Count Control read address (MCCNTCON - 0x400B 805C)

The MCCNTCON register controls whether the MCPWM channels are in timer or counter mode, and in counter mode whether the counter advances on rising and/or falling edges on any or all of the three MCI inputs. If timer mode is selected, the counter advances based on the PCLK clock.

This address is read-only. To set or clear the register bits, write ones to the MCCNTCON_SET or MCCNTCON_CLR address.

Table 470. MCPWM Count Control read address (MCCNTCON - 0x400B 805C) bit description

Bit	Symbol	Value	Description	Reset Value
0	TC0MCI0_RE	1	If MODE0 is 1, counter 0 advances on a rising edge on MCI0.	0
		0	A rising edge on MCI0 does not affect counter 0.	
1	TC0MCI0_FE	1	If MODE0 is 1, counter 0 advances on a falling edge on MCI0.	0
		0	A falling edge on MCI0 does not affect counter 0.	
2	TC0MCI1_RE	1	If MODE0 is 1, counter 0 advances on a rising edge on MCI1.	0
		0	A rising edge on MCI1 does not affect counter 0.	
3	TC0MCI1_FE	1	If MODE0 is 1, counter 0 advances on a falling edge on MCI1.	0
		0	A falling edge on MCI1 does not affect counter 0.	
4	TC0MCI2_RE	1	If MODE0 is 1, counter 0 advances on a rising edge on MCI2.	0
		0	A rising edge on MCI0 does not affect counter 0.	
5	TC0MCI2_FE	1	If MODE0 is 1, counter 0 advances on a falling edge on MCI2.	0
		0	A falling edge on MCI0 does not affect counter 0.	
6	TC1MCI0_RE	1	If MODE1 is 1, counter 1 advances on a rising edge on MCI0.	0
		0	A rising edge on MCI0 does not affect counter 1.	
7	TC1MCI0_FE	1	If MODE1 is 1, counter 1 advances on a falling edge on MCI0.	0
		0	A falling edge on MCI0 does not affect counter 1.	
8	TC1MCI1_RE	1	If MODE1 is 1, counter 1 advances on a rising edge on MCI1.	0
		0	A rising edge on MCI1 does not affect counter 1.	
9	TC1MCI1_FE	1	If MODE1 is 1, counter 1 advances on a falling edge on MCI1.	0
		0	A falling edge on MCI0 does not affect counter 1.	
10	TC1MCI2_RE	1	If MODE1 is 1, counter 1 advances on a rising edge on MCI2.	0
		0	A rising edge on MCI2 does not affect counter 1.	
11	TC1MCI2_FE	1	If MODE1 is 1, counter 1 advances on a falling edge on MCI2.	0
		0	A falling edge on MCI2 does not affect counter 1.	
12	TC2MCI0_RE	1	If MODE2 is 1, counter 2 advances on a rising edge on MCI0.	0
		0	A rising edge on MCI0 does not affect counter 2.	

© NXP B.V. 2010. All rights reserved.

Table 470. MCPWM Count Control read address (MCCNTCON - 0x400B 805C) bit description

Bit	Symbol	Value	Description	Reset Value
13	TC2MCI0_FE	1	If MODE2 is 1, counter 2 advances on a falling edge on MCI0.	0
		0	A falling edge on MCI0 does not affect counter 2.	
14	TC2MCI1_RE	1	If MODE2 is 1, counter 2 advances on a rising edge on MCI1.	0
		0	A rising edge on MCI1 does not affect counter 2.	
15	TC2MCI1_FE	1	If MODE2 is 1, counter 2 advances on a falling edge on MCI1.	0
		0	A falling edge on MCI1 does not affect counter 2.	
16	TC2MCI2_RE	1	If MODE2 is 1, counter 2 advances on a rising edge on MCI2.	0
		0	A rising edge on MCI2 does not affect counter 2.	
17	TC2MCI2_FE	1	If MODE2 is 1, counter 2 advances on a falling edge on MCI2.	0
		0	A falling edge on MCI2 does not affect counter 2.	
28:18	-	-	Reserved.	-
29	CNTR0	1	Channel 0 is in counter mode.	0
		0	Channel 0 is in timer mode.	
30	CNTR1	1	Channel 1 is in counter mode.	0
		0	Channel 1 is in timer mode.	
31	CNTR2	1	Channel 2 is in counter mode.	0
		0	Channel 2 is in timer mode.	

7.4.2 MCPWM Count Control set address (MCCNTCON_SET - 0x400B 8060)

Writing one(s) to this write-only address sets the corresponding bit(s) in MCCNTCON.

Table 471. MCPWM Count Control set address (MCCNTCON_SET - 0x400B 8060) bit description

Bit	Description
31:0	Writing one(s) to this write-only address sets the corresponding bit(s) in the MCCNTCON register. See Table 25–470.

7.4.3 MCPWM Count Control clear address (MCCNTCON_CLR - 0x400B 8064)

Writing one(s) to this write-only address clears the corresponding bit(s) in MCCNTCON.

Table 472. MCPWM Count Control clear address (MCCAPCON_CLR - 0x400B 8064) bit description

Bit	Description
31:0	Writing one(s) to this write-only address clears the corresponding bit(s) in the MCCNTCON register. See Table 25–470.

7.5 MCPWM Timer/Counter 0-2 registers (MCTC0-2 - 0x400B 8018, 0x400B 801C, 0x400B 8020)

These registers hold the current values of the 32-bit counter/timers for channels 0-2. Each value is incremented on every PCLK, or by edges on the MCI0-2 pins, as selected by MCCNTCON. The timer/counter counts up from 0 until it reaches the value in its corresponding MCPER register (or is stopped by writing to MCCON_CLR).

A TC register can be read at any time. In order to write to the TC register, its channel must be stopped. If not, the write will not take place, no exception is generated.

© NXP B.V. 2010. All rights reserved.

Table 473. MCPWM Timer/Counter 0-2 registers (MCTC0-2 - 0x400B 8018, 0x400B 801C, 0x400B 8020) bit description

Bit	Symbol	Description	Reset value
31:0	MCTC0/1/2	Timer/Counter values for channels 0, 1, 2.	0

7.6 MCPWM Limit 0-2 registers (MCLIM0-2 - 0x400B 8024, 0x400B 8028, 0x400B 802C)

These registers hold the limiting values for timer/counters 0-2. When a timer/counter reaches its corresponding limiting value: 1) in edge-aligned mode, it is reset and starts over at 0; 2) in center-aligned mode, it begins counting down until it reaches 0, at which time it begins counting up again.

If the channel's CENTER bit in MCCON is 0 selecting edge-aligned mode, the match between TC and LIM switches the channel's A output from "active" to "passive" state. If the channel's CENTER and DTE bits in MCCON are both 0, the match simultaneously switches the channel's B output from "passive" to "active" state.

If the channel's CENTER bit is 0 but the DTE bit is 1, the match triggers the channel's deadtime counter to begin counting -- when the deadtime counter expires, the channel's B output switches from "passive" to "active" state.

In center-aligned mode, matches between a channel's TC and LIM registers have no effect on its A and B outputs.

Writing to either a Limit or a Match (7.7) register loads a "write" register, and if the channel is stopped it also loads an "operating" register that is compared to the TC. If the channel is running and its "disable update" bit in MCCON is 0, the operating registers are loaded from the write registers: 1) in edge-aligned mode, when the TC matches the operating Limit register; 2) in center-aligned mode, when the TC counts back down to 0. If the channel is running and the "disable update" bit is 1, the operating registers are not loaded from the write registers until software stops the channel.

Reading an MCLIM address always returns the operating value.

Table 474. MCPWM Limit 0-2 registers (MCLIM0-2 - 0x400B 8024, 0x400B 8028, 0x400B 802C) bit description

Bit	Symbol	Description	Reset value
31:0	MCLIM0/1/2	Limit values for TC0, 1, 2.	0xFFFF FFFF

Note: *In timer mode, the period of a channel's modulated MCO outputs is determined by its Limit register, and the pulse width at the start of the period is determined by its Match register. If it suits your way of thinking, consider the Limit register to be the "Period register" and the Match register to be the "Pulse Width register".*

7.7 MCPWM Match 0-2 registers (MCMAT0-2 - 0x400B 8030, 0x400B 8034, 0x400B 8038)

These registers also have "write" and "operating" versions as described above for the Limit registers, and the operating registers are also compared to the channels' TCs. See 7.6 above for details of reading and writing both Limit and Match registers.

The Match and Limit registers control the MCO0-2 outputs. If a Match register is to have any effect on its channel's operation, it must contain a smaller value than the corresponding Limit register.

© NXP B.V. 2010. All rights reserved.

Table 475. MCPWM Match 0-2 registers (MCMAT0-2 - addresses 0x400B 8030, 0x400B 8034, 0x400B 8038) bit description

Bit	Symbol	Description	Reset value
31:0	MCMAT0/1/2	Match values for TC0, 1, 2.	0xFFFF FFFF

7.7.1 Match register in Edge-Aligned mode

If the channel's CENTER bit in MCCON is 0 selecting edge-aligned mode, a match between TC and MAT switches the channel's B output from "active" to "passive" state. If the channel's CENTER and DTE bits in MCCON are both 0, the match simultaneously switches the channel's A output from "passive" to "active" state.

If the channel's CENTER bit is 0 but the DTE bit is 1, the match triggers the channel's deadtime counter to begin counting -- when the deadtime counter expires, the channel's A output switches from "passive" to "active" state.

7.7.2 Match register in Center-Aligned mode

If the channel's CENTER bit in MCCON is 1 selecting center-aligned mode, a match between TC and MAT while the TC is incrementing switches the channel's B output from "active" to "passive" state, and a match while the TC is decrementing switches the A output from "active" to "passive". If the channel's CENTER bit in MCCON is 1 but the DTE bit is 0, a match simultaneously switches the channel's other output in the opposite direction.

If the channel's CENTER and DTE bits are both 1, a match between TC and MAT triggers the channel's deadtime counter to begin counting -- when the deadtime counter expires, the channel's B output switches from "passive" to "active" if the TC was counting up at the time of the match, and the channel's A output switches from "passive" to "active" if the TC was counting down at the time of the match.

7.7.3 0 and 100% duty cycle

To lock a channel's MCO outputs at the state "B active, A passive", write its Match register with a higher value than you write to its Limit register. The match never occurs.

To lock a channel's MCO outputs at the opposite state, "A active, B passive", simply write 0 to its Match register.

7.8 MCPWM Dead-time register (MCDT - 0x400B 803C)

This register holds the dead-time values for the three channels. If a channel's DTE bit in MCCON is 1 to enable its dead-time counter, the counter counts down from this value whenever one its channel's outputs changes from "active" to "passive" state. When the dead-time counter reaches 0, the channel changes its other output from "passive" to "active" state.

The motivation for the dead-time feature is that power transistors, like those driven by the A and B outputs in a motor-control application, take longer to fully turn off than they take to start to turn on. If the A and B transistors are ever turned on at the same time, a wasteful and damaging current will flow between the power rails through the transistors. In such applications, the dead-time register should be programmed with the number of PCLK periods that is greater than or equal to the transistors' maximum turn-off time minus their minimum turn-on time.

© NXP B.V. 2010. All rights reserved.

Table 476. MCPWM Dead-time register (MCDT - address 0x400B 803C) bit description

Bit	Symbol	Description	Reset value
9:0	DT0	Dead time for channel 0.[1]	0x3FF
19:10	DT1	Dead time for channel 1.[2]	0x3FF
29:20	DT2	Dead time for channel 2.[2]	0x3FF
31:30	-	reserved	

[1] If ACMODE is 1 selecting AC-mode, this field controls the dead time for all three channels.

[2] If ACMODE is 0.

7.9 MCPWM Commutation Pattern register (MCCP - 0x400B 8040)

This register is used in DC mode only. The internal MCOA0 signal is routed to any or all of the six output pins under the control of the bits in this register. Like the Match and Limit registers, this register has “write” and “operational” versions. See 7.6 and 8.2 for more about this subject.

Table 477. MCPWM Commutation Pattern register (MCCP - address 0x400B 8040) bit description

Bit	Symbol	Description	Reset value
0	CCPA0	0 = MCOA0 passive, 1 = internal MCOA0.	0
1	CCPB0	0 = MCOB0 passive, 1 = MCOB0 tracks internal MCOA0.	0
2	CCPA1	0 = MCOA1 passive, 1 = MCOA1 tracks internal MCOA0.	0
3	CCPB1	0 = MCOB1 passive, 1 = MCOB1 tracks internal MCOA0.	0
4	CCPA2	0 = MCOA2 passive, 1 = MCOA2 tracks internal MCOA0.	0
5	CCPB2	0 = MCOB2 passive, 1 = MCOB2 tracks internal MCOA0.	0
31:6	-	Reserved.	

7.10 MCPWM Capture Registers

7.10.1 MCPWM Capture read addresses (MCCAP0-2 - 0x400B 8044, 0x400B 8048, 0x400B 804C)

The MCCAPCON register (Table 25–459) allows software to select any edge(s) on any of the MCI0-2 inputs as a capture event for each channel. When a channel’s capture event occurs, the current TC value for that channel is stored in its read-only Capture register. These addresses are read-only, but the underlying registers can be cleared by writing to the CAP_CLR address

Table 478. MCPWM Capture read addresses (MCCAP0/1/2 - 0x400B 8044, 0x400B 8048, 0x400B 804C) bit description

Bit	Symbol	Description	Reset value
31:0	CAP0/1/2	TC value at a capture event for channels 0, 1, 2.	0x0000 0000

7.10.2 MCPWM Capture clear address (MCCAP_CLR - 0x400B 8074)

Writing ones to this write-only address clears the selected CAP register(s).

© NXP B.V. 2010. All rights reserved.

Table 479. MCPWM Capture clear address (CAP_CLR - 0x400B 8074) bit description

Bit	Symbol	Description
0	CAP_CLR0	Writing a 1 to this bit clears the MCCAP0 register.
1	CAP_CLR1	Writing a 1 to this bit clears the MCCAP1 register.
2	CAP_CLR2	Writing a 1 to this bit clears the MCCAP2 register.
31:3	-	Reserved

© NXP B.V. 2010. All rights reserved.

8. PWM operation

8.1 Pulse-width modulation

Each channel of the MCPWM has two outputs, A and B, that can drive a pair of transistors to switch a controlled point between two power rails. Most of the time the two outputs have opposite polarity, but a dead-time feature can be enabled (on a per-channel basis) to delay both signals' transitions from "passive" to "active" state so that the transistors are never both turned on simultaneously. In a more general view, the states of each output pair can be thought of "high", "low", and "floating" or "up", "down", and "center-off".

Each channel's mapping from "active" and "passive" to "high" and "low" is programmable. After Reset, the three A outputs are passive/low, and the B outputs are active/high.

The MCPWM can perform edge-aligned and center-aligned pulse-width modulation.

Note: *In timer mode, the period of a channel's modulated MCO outputs is determined by its Limit register, and the pulse width at the start of the period is determined by its Match register. If it suits your way of thinking, consider the Limit register to be the "Period register" and the Match register to be the "Pulse Width register".*

Edge-aligned PWM without dead-time

In this mode the timer TC counts up from 0 to the value in the LIM register. As shown in Figure 25–121, the MCO state is "A passive" until the TC matches the Match register, at which point it changes to "A active". When the TC matches the Limit register, the MCO state changes back to "A passive", and the TC is reset and starts counting up again.

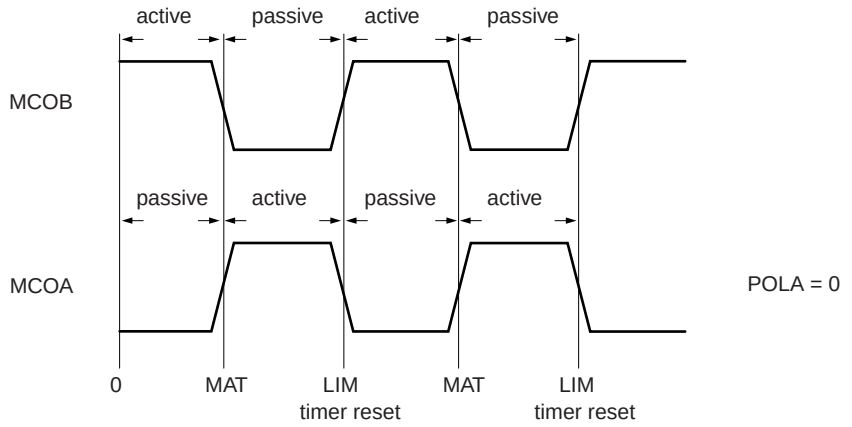


Fig 121. Edge-aligned PWM waveform without dead time, POLA = 0

Center-aligned PWM without dead-time

In this mode the timer TC counts up from 0 to the value in the LIM register, then counts back down to 0 and repeats. As shown in Figure 25–122, while the timer counts up, the MCO state is "A passive" until the TC matches the Match register, at which point it changes to "A active". When the TC matches the Limit register it starts counting down. When the TC matches the Match register on the way down, the MCO state changes back to "A passive".

© NXP B.V. 2010. All rights reserved.

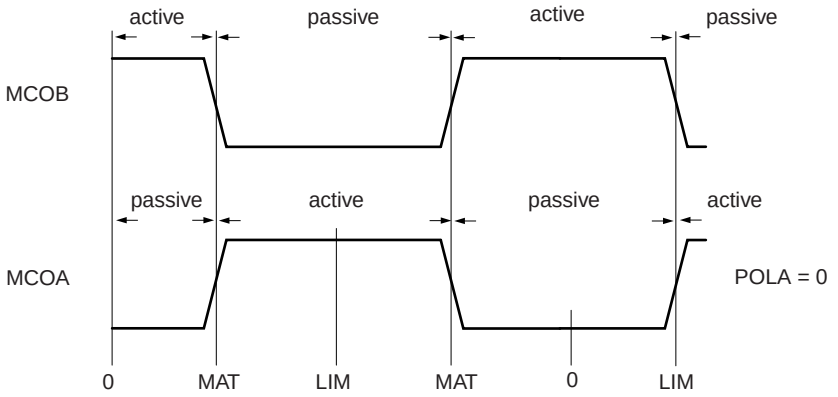


Fig 122. Center-aligned PWM waveform without dead time, POLA = 0

Dead-time counter

When the a channel's DTE bit is set in MCCON, the dead-time counter delays the passive-to-active transitions of both MCO outputs. The dead-time counter starts counting down, from the channel's DT value (in the MCDT register) to 0, whenever the channel's A or B output changes from active to passive. The transition of the other output from passive to active is delayed until the dead-time counter reaches 0. During the dead time, the MCOA and MCOB output levels are both passive. Figure 25–123 shows operation in edge aligned mode with dead time, and Figure 25–124 shows center-aligned operation with dead time.

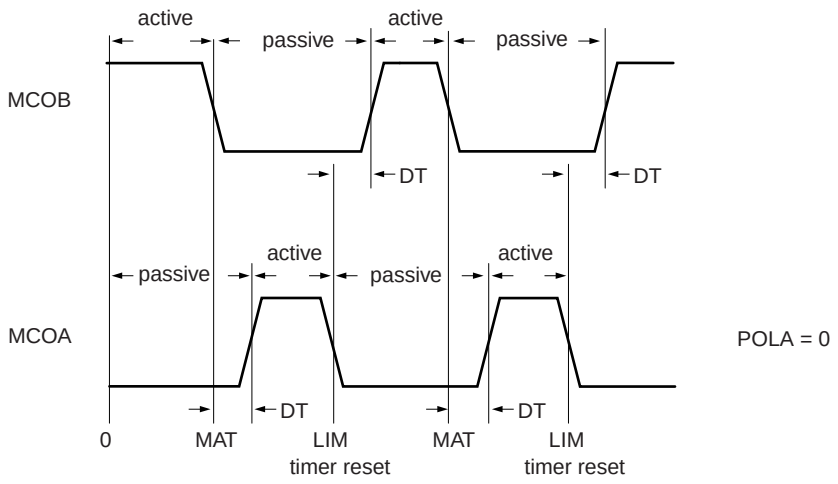


Fig 123. Edge-aligned PWM waveform with dead time, POLA = 0

© NXP B.V. 2010. All rights reserved.

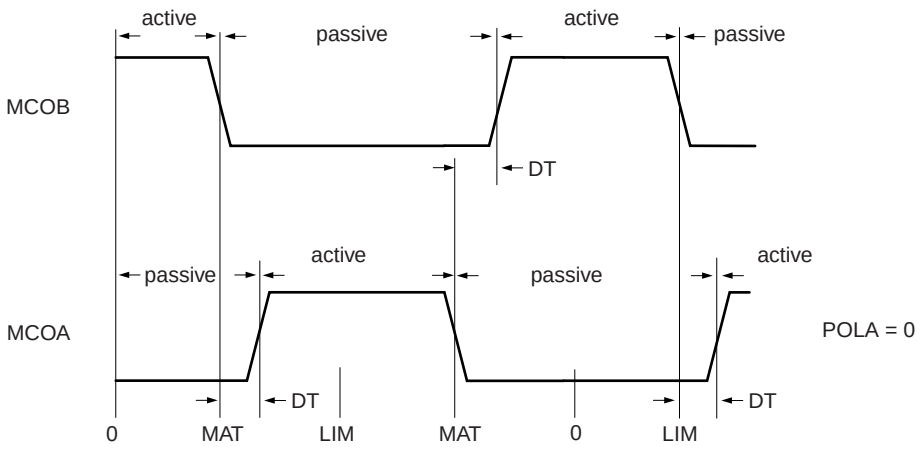


Fig 124. Center-aligned waveform with dead time, POLA = 0

8.2 Shadow registers and simultaneous updates

The Limit, Match, and Commutation Pattern registers (MCLIM, MCMAT, and MCCP) are implemented as register pairs, each consisting of a write register and an operational register. Software writes into the write registers. The operational registers control the actual operation of each channel and are loaded with the current value in the write registers when the TC starts counting up from 0.

Updating of the functional registers can be disabled by setting a channel's DISUP bit in the MCCON register. If the DISUP bits are set, the functional registers are not updated until software stops the channel.

If a channel is not running when software writes to its LIM or MAT register, the functional register is updated immediately.

Software can write to a TC register only when its channel is stopped.

8.3 Fast Abort (ABORT)

The MCPWM has an external input $\overline{\text{MCABORT}}$. When this input goes low, all six MCO outputs assume their "A passive" states, and the Abort interrupt is generated if enabled. The outputs remain locked in "A passive" state until the ABORT interrupt flag is cleared or the Abort interrupt is disabled. The ABORT flag may not be cleared before the $\overline{\text{MCABORT}}$ input goes high.

In order to clear an ABORT flag, a 1 must be written to bit 15 of the MCINTF_CLR register. This will remove the interrupt request. The interrupt can also be disabled by writing a 1 to bit 15 of the MCINTEN_CLR register.

8.4 Capture events

Each PWM channel can take a snapshot of its TC when an input signal transitions. Any channel may use any combination of rising and/or falling edges on any or all of the MCI0-2 inputs as a capture event, under control of the MCCAPCON register. Rising or falling edges on the inputs are detected synchronously with respect to PCLK.

© NXP B.V. 2010. All rights reserved.

If a channel's HNF bit in the MCCAPCON register is set to enable "noise filtering", a selected edge on an MCI pin starts the dead-time counter for that channel, and the capture event actions described below are delayed until the dead-time counter reaches 0. This function is targeted specifically for performing three-phase brushless DC motor control with Hall sensors.

A capture event on a channel (possibly delayed by HNF) causes the following:

- The current value of the TC is stored in the Capture register (CAP).
- If the channel's capture event interrupt is enabled (see Table 25–464), the capture event interrupt flag is set.
- If the channel's RT bit is set in the MCCAPCON register, enabling reset on a capture event, the input event has the same effect as matching the channel's TC to its LIM register. This includes resetting the TC and switching the MCO pin(s) in edge-aligned mode as described in 7.6 and 8.1.

8.5 External event counting (Counter mode)

If a channel's MODE bit is 1 in MCCNTCON, its TC is incremented by rising and/or falling edge(s) (synchronously detected) on the MCI0-2 input(s), rather than by PCLK. The PWM functions and capture functions are unaffected.

8.6 Three-phase DC mode

The three-phase DC mode is selected by setting the DCMODE bit in the MCCON register.

In this mode, the internal MCOA0 signal can be routed to any or all of the MCO outputs. Each MCO output is masked by a bit in the current Commutation Pattern register MCCP. If a bit in the MCCP register is 0, its output pin has the logic level for the passive state of output MCOA0. The polarity of the off state is determined by the POLA0 bit.

All MCO outputs that have 1 bits in the MCCP register are controlled by the internal MCOA0 signal.

The three MCOB output pins are inverted when the INVBDC bit is 1 in the MCCON register. This feature accommodates bridge-drivers that have active-low inputs for the low-side switches.

The MCCP register is implemented as a shadow register pair, so that changes to the active commutation pattern occur at the beginning of a new PWM cycle. See 7.6 and 8.2 for more about writing and reading such registers.

Figure 25–125 shows sample waveforms of the MCO outputs in three-phase DC mode. Bits 1 and 3 in the MCCP register (corresponding to outputs MCOB1 and MCOB0) are set to 0 so that these outputs are masked and in the off state. Their logic level is determined by the POLA0 bit (here, POLA0 = 0 so the passive state is logic LOW). The INVBDC bit is set to 0 (logic level not inverted) so that the B output have the same polarity as the A outputs. Note that this mode differs from other modes in that the MCOB outputs are **not** the opposite of the MCOA outputs.

In the situation shown in Figure 25–125, bits 0, 2, 4, and 5 in the MCCP register are set to 1. That means that MCOA1 and both MCO outputs for channel 2 follow the MCOA0 signal.

© NXP B.V. 2010. All rights reserved.

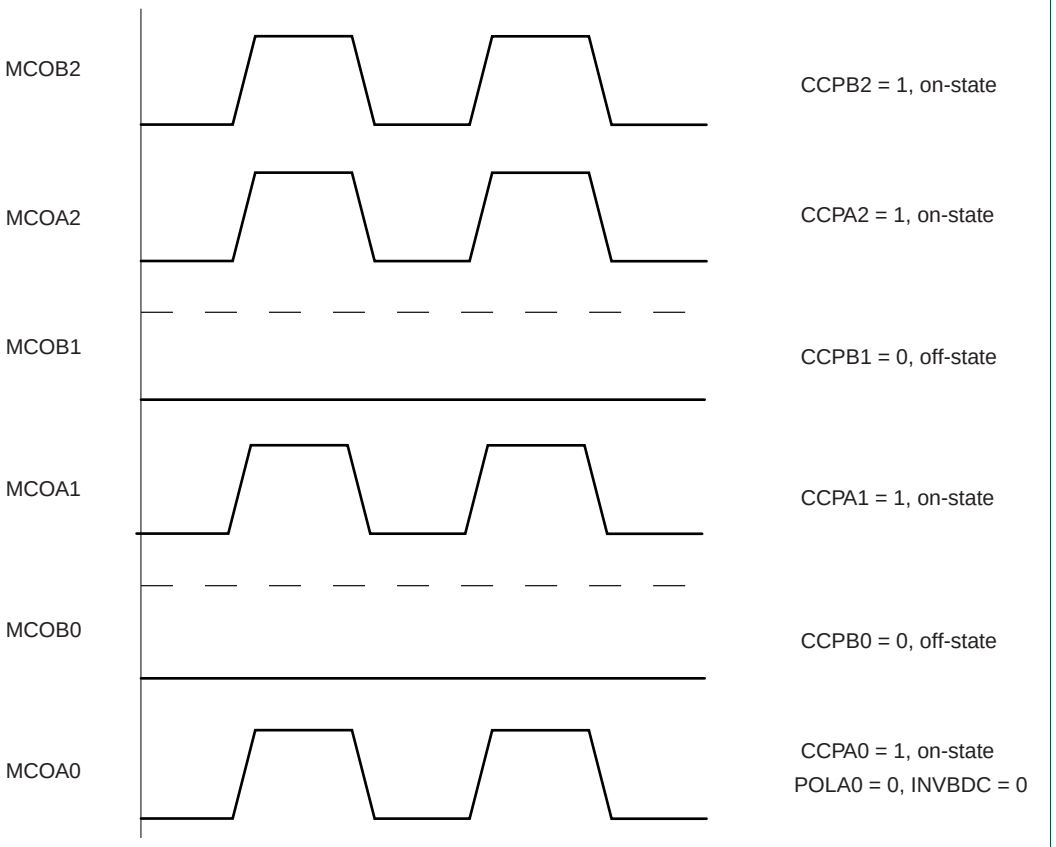


Fig 125. Three-phase DC mode sample waveforms

8.7 Three phase AC mode

The three-phase AC-mode is selected by setting the ACMODE bit in the MCCON register.

In this mode, the value of channel 0's TC is routed to all channels for comparison with their MAT registers. (The LIM1-2 registers are not used.)

Each channel controls its MCO output by comparing its MAT value to TC0.

Figure 25–126 shows sample waveforms for the six MCO outputs in three-phase AC mode. The POLA bits are set to 0 for all three channels, so that for all MCO outputs the active levels are high and the passive levels are low. Each channel has a different MAT value which is compared to the MCTC0 value. In this mode the period value is identical for all three channels and is determined by MCLIM0. The dead-time mode is disabled.

© NXP B.V. 2010. All rights reserved.

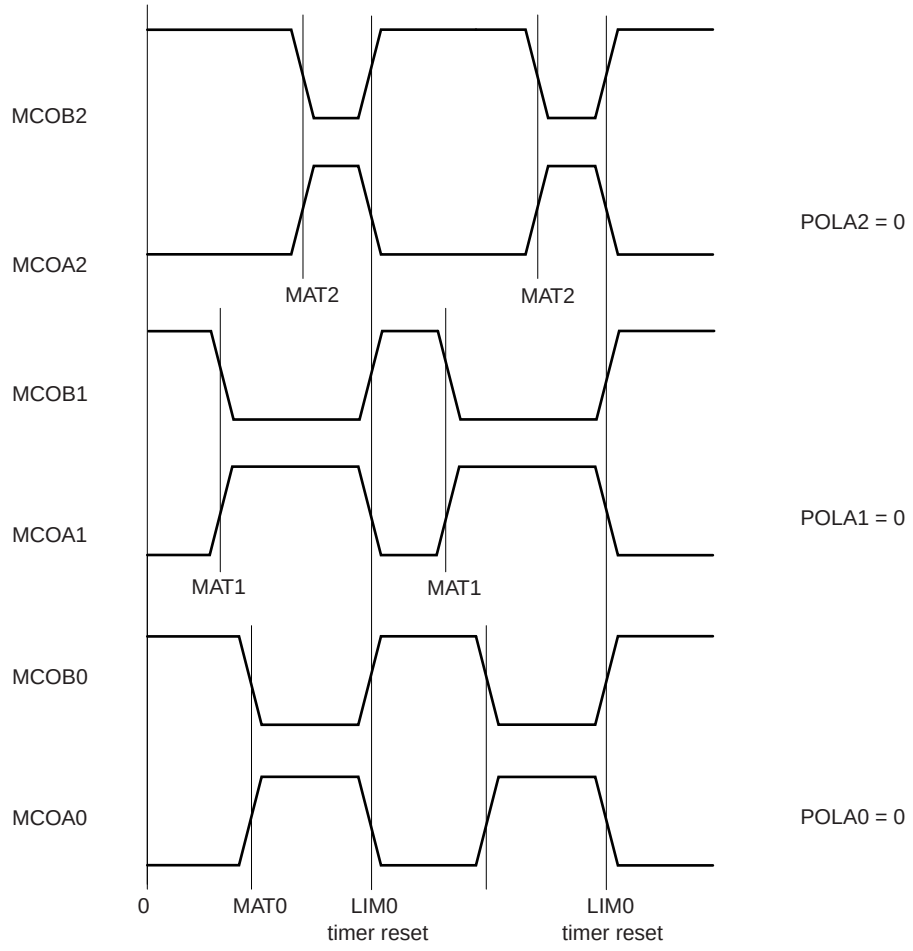


Fig 126. Three-phase AC mode sample waveforms, edge aligned PWM mode

8.8 Interrupts

The MCPWM includes 10 possible interrupt sources:

- When any channel's TC matches its Match register.
- When any channel's TC matches its Limit register.
- When any channel captures the value of its TC into its Capture register, because a selected edge occurs on any of MCI0-2.
- When all three channels' outputs are forced to "A passive" state because the $\overline{\text{MCABORT}}$ pin goes low.

Section 25–7.3 "MCPWM Interrupt registers" explains how to enable these interrupts, and Section 25–7.2 "MCPWM Capture Control register" describes how to map edges on the MCI0-2 inputs to "capture events" on the three channels.

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The QEI is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCQEI.

   Remark: On reset, the QEI is disabled (PCQEI = 0).
2. Peripheral clock: In the PCLKSEL0 register (Table 4–40), select PCLK_QEI.
3. Pins: Select QEI pins through the PINSEL registers. Select pin modes for port pins with QEI functions through the PINMODE registers (Section 8–5).
4. Interrupts: See Section 26–6.4. The QEI interrupt is enabled in the NVIC using the appropriate Interrupt Set Enable register.

2. Features

This Quadrature Encoder Interface (QEI) has the following features:

- tracks encoder position.
- increments/ decrements depending on direction.
- programmable for 2X or 4X position counting.
- velocity capture using built-in timer.
- velocity compare function with less than interrupt.
- uses 32-bit registers for position and velocity.
- three position compare registers with interrupts.
- index counter for revolution counting.
- index compare register with interrupts.
- can combine index and position interrupts to produce an interrupt for whole and partial revolution displacement.
- digital filter with programmable delays for encoder input signals.
- can accept decoded signal inputs (clock and direction).

3. Introduction

A quadrature encoder, also known as a 2-channel incremental encoder, converts angular displacement into two pulse signals. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and velocity. In addition, a third channel, or index signal, can be used to reset the position counter. This quadrature encoder interface module decodes the digital pulses from a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture the velocity of the encoder wheel.

© NXP B.V. 2010. All rights reserved.

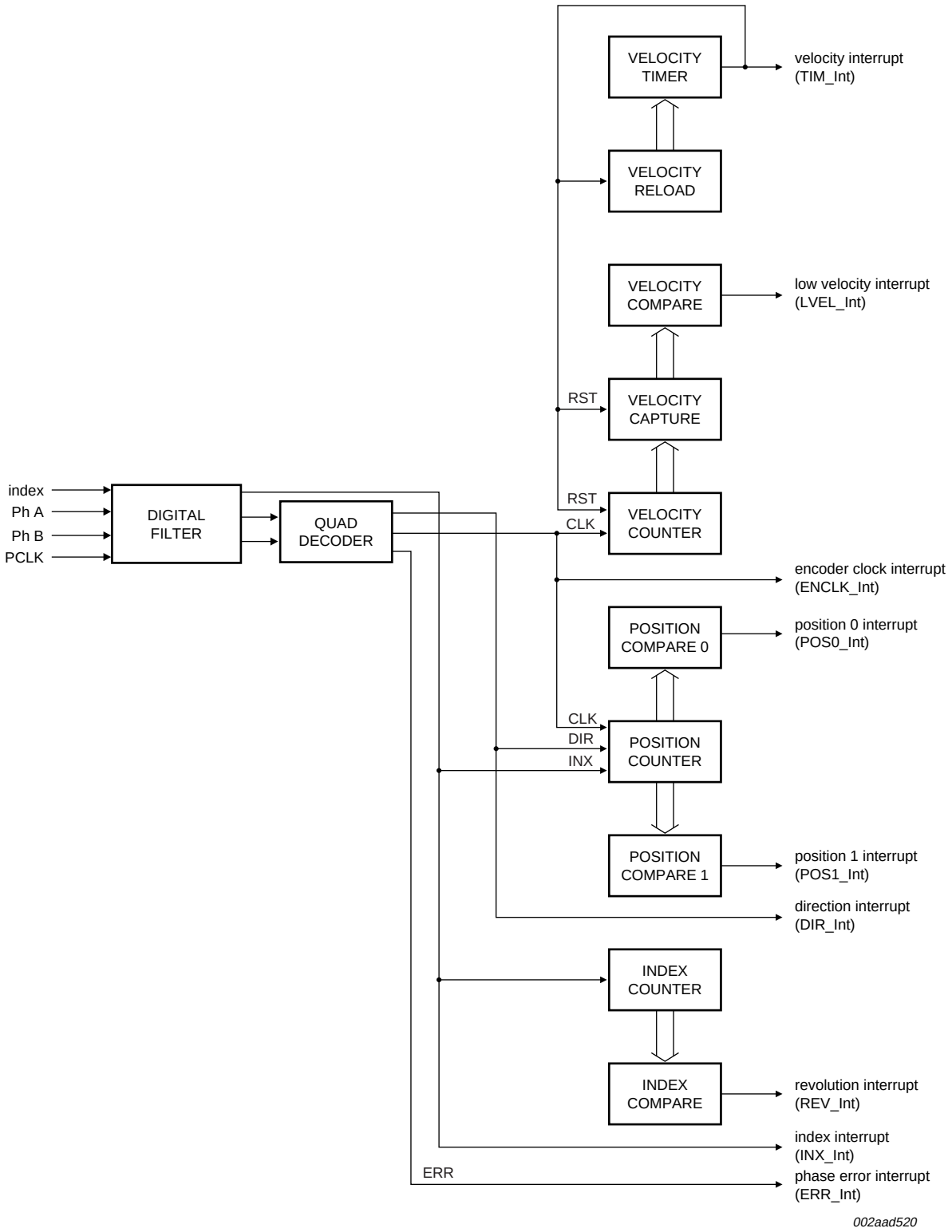


Fig 127.Encoder interface block diagram

© NXP B.V. 2010. All rights reserved.

4. Functional description

The QEI module interprets the two-bit gray code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture the velocity of the encoder wheel.

4.1 Input signals

The QEI module supports two modes of signal operation: quadrature phase mode and clock/direction mode. In quadrature phase mode, the encoder produces two clocks that are 90 degrees out of phase; the edge relationship is used to determine the direction of rotation. In clock/direction mode, the encoder produces a clock signal to indicate steps and a direction signal to indicate the direction of rotation.).

This mode is determined by the SigMode bit of the QEI Control (QEICON) register (See Table 26–485). When the SigMode bit = 1, the quadrature decoder is bypassed and the PhA pin functions as the direction signal and PhB pin functions as the clock signal for the counters, etc. When the SigMode bit = 0, the PhA pin and PhB pins are decoded by the quadrature decoder. In this mode the quadrature decoder produces the direction and clock signals for the counters, etc. In both modes the direction signal is subject to the effects of the direction invert (DIRINV) bit.

4.1.1 Quadrature input signals

When edges on PhA lead edges on PhB, the position counter is incremented. When edges on PhB lead edges on PhA, the position counter is decremented. When a rising and falling edge pair is seen on one of the phases without any edges on the other, the direction of rotation has changed.

Table 480. Encoder states

Phase A	Phase B	state
1	0	1
1	1	2
0	1	3
0	0	4

Table 481. Encoder state transitions[1]

from state	to state	Direction
1	2	positive
2	3	
3	4	
4	1	
4	3	negative
3	2	
2	1	
1	4	

[1] All other state transitions are illegal and should set the ERR bit.

Interchanging of the PhA and PhB input signals are compensated by complementing the DIR bit. When set = 1, the direction inversion bit (DIRINV) complements the DIR bit.

© NXP B.V. 2010. All rights reserved.

Table 482. Encoder direction

DIR bit	DIRINV bit	direction
0	0	forward
1	0	reverse
0	1	reverse
1	1	forward

Figure 26–128 shows how quadrature encoder signals equate to direction and count.

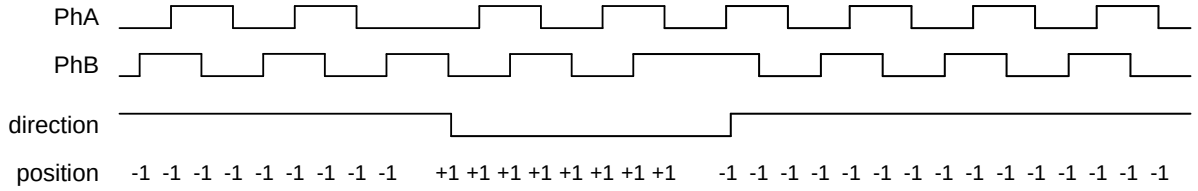


Fig 128.Quadrature Encoder Basic Operation

4.1.2 Digital input filtering

All three encoder inputs (PhA, PhB, and index) require digital filtering. The number of sample clocks is user programmable from 1 to 4,294,967,295 (0xFFFF FFFF). In order for a transition to be accepted, the input signal must remain in new state for the programmed number of sample clocks.

4.2 Position capture

The capture mode for the position integrator can be set to update the position counter on every edge of the PhA signal or to update on every edge of both PhA and PhB. Updating the position counter on every PhA and PhB provides more positional resolution at the cost of less range in the positional counter.

The position integrator and velocity capture can be independently enabled. Alternatively, the phase signals can be interpreted as a clock and direction signal as output by some encoders.

The position counter is automatically reset on one of two conditions. Incrementing past the maximum position value (QEIMAXPOS) will reset the position counter to zero. If the reset on index bit (RESPI) is set, sensing the index pulse will reset the position counter to zero.

4.3 Velocity capture

The velocity capture has a programmable timer and a capture register. It counts the number of phase edges (using the same configuration as for the position integrator) in a given time period. When the velocity timer (QEITIME) overflows the contents of the velocity counter (QEIVEL) are transferred to the capture (QEICAP) register. The velocity counter is then cleared. The velocity timer is loaded with the contents of the velocity reload register (QEILOAD). Finally, the velocity interrupt (TIM_Int) is asserted. The

© NXP B.V. 2010. All rights reserved.

number of edges counted in a given time period is directly proportional to the velocity of the encoder. Setting the reset velocity bit (RESV) has the same effect as an overflow of the velocity timer, except that the setting the RESV bit will not generate a velocity interrupt.

The following equation converts the velocity counter value into an RPM value:

```
RPM = (PCLK * QEICAP * 60) ÷ (QEILOAD * PPR * Edges)
```

where:

- **PCLK** is the peripheral clock rate for the QEI block. See Section 4–7.3 for more on the possibilities for PCLK).
- **QEICAP** is the captured velocity counter value for the last velocity timer period.
- **QEILOAD** is the velocity timer reload value.
- **PPR** is the number of pulses per revolution of the physical encoder used in the application
- **Edges** is 2 or 4, based on the capture mode set in the QEICON register (2 for CapMode set to 0 and 4 for CapMode set to 1)

For example, consider a motor running at 600 RPM. A 2048 pulse per revolution quadrature encoder is attached to the motor, producing 8192 phase edges per revolution (PPR * Edges). This results in 81,920 pulses per second (the motor turns 10 times per second at 600 RPM and there are 8092 edges per revolution). If the timer were clocked at 10,000 Hz, and the QEILOAD was 2,500 (corresponding to ¼ of a second), it would count 20,480 pulses per update. Using the above equation:

RPM = (10000 * 1 * 20480 * 60) ÷ (2500 * 2048 * 4) = 600 RPM

Now, consider that the motor is sped up to 3000 RPM. This results in 409,600 pulses per second, or 102,400 every ¼ of a second. Again, the above equation gives:

RPM = (10000 * 1 * 102400 * 60) ÷ (2500 * 2048 * 4) = 3000 RPM

These are simple examples, real-world values will have a higher rate for PCLK, and probably a larger value for QEILOAD as well.

4.4 Velocity compare

In addition to velocity capture, the velocity measurement system includes a programmable velocity compare register. After every velocity capture event the contents of the velocity capture register (QEICAP) is compared with the contents of the velocity compare register (VELCOMP). If the captured velocity is less than the compare value an interrupt is asserted provided that the velocity compare interrupt enable bit is set. This can be used to determine if a motor shaft is either stalled or moving too slow.

© NXP B.V. 2010. All rights reserved.

5. Pin description

Table 483. QEI pin description

Pin name	I/O	Description
MCI0 [1]	I	Used as the Phase A (PhA) input to the Quadrature Encoder Interface.
MCI1 [1]	I	Used as the Phase B (PhB) input to the Quadrature Encoder Interface.
MCI2 [1]	I	Used as the Index (IDX) input to the Quadrature Encoder Interface.

[1] The Quadrature Encoder Interface uses the same pin functions as the Motor Control PWM feedback inputs and are connected when the Motor Control PWM function is selected on these pins. If used as part of motor control, the QEI is an alternative to feedback directly to the MCPWM.

© NXP B.V. 2010. All rights reserved.

6. Register description

6.1 Register summary

Table 484. QEI Register summary

Name	Description	Access	Reset value	Address
Control registers				
QEICON	Control register	WO	0	0x400B C000
QEICONF	Configuration register	R/W	0	0x400B C008
QEISTAT	Encoder status register	RO	0	0x400B C004
Position, index, and timer registers				
QEIPOS	Position register	RO	0	0x400B C00C
QEIMAXPOS	Maximum position register	R/W	0	0x400B C010
CMPOS0	position compare register 0	R/W	0	0x400B C014
CMPOS1	position compare register 1	R/W	0	0x400B C018
CMPOS2	position compare register 2	R/W	0	0x400B C01C
INXCNT	Index count register	RO	0	0x400B C020
INXCMP	Index compare register	R/W	0	0x400B C024
QEILOAD	Velocity timer reload register	R/W	0	0x400B C028
QEITIME	Velocity timer register	RO	0	0x400B C02C
QEIVEL	Velocity counter register	RO	0	0x400B C030
QEICAP	Velocity capture register	RO	0	0x400B C034
VELCOMP	Velocity compare register	R/W	0	0x400B C038
FILTER	Digital filter register	R/W	0	0x400B C03C
Interrupt registers				
QEIINTSTAT	Interrupt status register	RO	0	0x400B CFE0
QEISET	Interrupt status set register	WO	0	0x400B CFEC
QEICLR	Interrupt status clear register	WO	0	0x400B CFE8
QEIIE	Interrupt enable register	RO	0	0x400B CFE4
QEIIES	Interrupt enable set register	WO	0	0x400B CFDC
QEIIEC	Interrupt enable clear register	WO	0	0x400B CFD8

© NXP B.V. 2010. All rights reserved.

6.2 Control registers

6.2.1 QEI Control register (QEICON - 0x400B C000)

This register contains bits which control the operation of the position and velocity counters of the QEI module.

Table 485: QEI Control register (QEICON - address 0x400B C000) bit description

Bit	Symbol	Description	Reset value
0	RESP	Reset position counter. When set = 1, resets the position counter to all zeros. Autoclears when the position counter is cleared.	0
1	RESPI	Reset position counter on index. When set = 1, resets the position counter to all zeros when an index pulse occurs. Autoclears when the position counter is cleared.	0
2	RESV	Reset velocity. When set = 1, resets the velocity counter to all zeros and reloads the velocity timer. Autoclears when the velocity counter is cleared.	0
3	RESI	Reset index counter. When set = 1, resets the index counter to all zeros. Autoclears when the index counter is cleared.	0
31:4	-	reserved	0

6.2.2 QEI Configuration register (QEICONF - 0x400B C008)

This register contains the configuration of the QEI module.

Table 486: QEI Configuration register (QEICONF - address 0x400B C008) bit description

Bit	Symbol	Description	Reset value
0	DIRINV	Direction invert. When = 1, complements the DIR bit.	0
1	SIGMODE	Signal Mode. When = 0, PhA and PhB function as quadrature encoder inputs. When = 1, PhA functions as the direction signal and PhB functions as the clock signal.	0
2	CAPMODE	Capture Mode. When = 0, only PhA edges are counted (2X). When = 1, BOTH PhA and PhB edges are counted (4X), increasing resolution but decreasing range.	0
3	INVINX	Invert Index. When set, inverts the sense of the index input.	0
31:4	-	reserved	0

6.2.3 QEI Status register (QEISTAT - 0x400B C004)

This register provides the status of the encoder interface.

Table 487: QEI Interrupt Status register (QEISTAT - address 0x400B C004) bit description

Bit	Symbol	Description	Reset value
0	DIR	Direction bit. In combination with DIRINV bit indicates forward or reverse direction. See Table 26–482.	
31:1	-	reserved	0

© NXP B.V. 2010. All rights reserved.

6.3 Position, index and timer registers

6.3.1 QEI Position register (QEIPOS - 0x400B C00C)

This register contains the current value of the encoder position. Increments or decrements when encoder counts occur, depending on the direction of rotation.

Table 488: QEI Position register (QEIPOS - address 0x400B C00C) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current position value.	0

6.3.2 QEI Maximum Position register (QEIMAXPOS - 0x400B C010)

This register contains the maximum value of the encoder position. In forward rotation the position register resets to zero when the position register exceeds this value. In reverse rotation the position register resets to this value when the position register decrements from zero.

Table 489: QEI Maximum Position register (QEIMAXPOS - address 0x400B C010) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current maximum position value.	0

6.3.3 QEI Position Compare register 0 (CMPOS0 - 0x400B C014)

This register contains a position compare value. This value is compared against the current value of the position register. Interrupts can be enabled to interrupt when the compare value is equal to the current value of the position register.

Table 490: QEI Position Compare register 0 (CMPOS0 - address 0x400B C014) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current position value.	0

6.3.4 QEI Position Compare register 1 (CMPOS1 - 0x400B C018)

This register contains a position compare value. This value is compared against the current value of the position register. Interrupts can be enabled to interrupt when the compare value is equal to the current value of the position register.

Table 491: QEI Position Compare register 1 (CMPOS1 - address 0x400B C018) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current position value.	0

© NXP B.V. 2010. All rights reserved.

6.3.5 QEI Position Compare register 2 (CMPOS2 - 0x400B C01C)

This register contains a position compare value. This value is compared against the current value of the position register. Interrupts can be enabled to interrupt when the compare value is equal to the current value of the position register.

Table 492: QEI Position Compare register 2 (CMPOS2 - address 0x400B C01C) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current position value.	0

6.3.6 QEI Index Count register (INXCNT - 0x400B C020)

This register contains the current value of the encoder position. Increments or decrements when encoder counts occur, depending on the direction of rotation.

Table 493: QEI Index Count register (CMPOS - address 0x400B C020) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current position value.	0

6.3.7 QEI Index Compare register (INXCMP - 0x400B C024)

This register contains an index compare value. This value is compared against the current value of the index count register. Interrupts can be enabled to interrupt when the compare value is equal to the current value of the index count register.

Table 494: QEI Index Compare register (CMPOS - address 0x400B C024) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current position value.	0

6.3.8 QEI Timer Reload register (QEILOAD - 0x400B C028)

This register contains the reload value of the velocity timer. When the timer (QEITIME) overflows or the RESV bit is asserted, this value is loaded into the timer (QEITIME).

Table 495: QEI Timer Load register (QEILOAD - address 0x400B C028) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current velocity timer load value.	0

6.3.9 QEI Timer register (QEITIME - 0x400B C02C)

This register contains the current value of the velocity timer. When this timer overflows the value of velocity counter (QEIVEL) is stored in the velocity capture register (QEICAP), the velocity counter is reset to zero, the timer is reloaded with the value stored in the velocity reload register (QEILOAD), and the velocity interrupt (TIM_Int) is asserted.

Table 496: QEI Timer register (QEITIME - address 0x400B C02C) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current velocity timer value.	0

© NXP B.V. 2010. All rights reserved.

6.3.10 QEI Velocity register (QEIVEL - 0x400B C030)

This register contains the running count of velocity pulses for the current time period. When the velocity timer (QEITIME) overflows the contents of this register is captured in the velocity capture register (QEICAP). After capture, this register is set to zero. This register is also reset when the velocity reset bit (RESV) is asserted.

Table 497: QEI Velocity register (QEIVEL - address 0x400B C030) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current velocity pulse count.	0

6.3.11 QEI Velocity Capture register (QEICAP - 0x400B C034)

This register contains the most recently measured velocity of the encoder. This corresponds to the number of velocity pulses counted in the previous velocity timer period.The current velocity count is latched into this register when the velocity timer overflows.

Table 498: QEI Velocity Capture register (QEICAP - address 0x400B C034) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current velocity pulse count.	0

6.3.12 QEI Velocity Compare register (VELCOMP - 0x400B C038)

This register contains a velocity compare value. This value is compared against the captured velocity in the velocity capture register. If the capture velocity is less than the value in this compare register, a velocity compare interrupt (VELC_Int) will be asserted, if enabled.

Table 499: QEI Velocity Compare register (VELCOMP - address 0x400B C038) bit description

Bit	Symbol	Description	Reset value
31:0	-	Current velocity pulse count.	0

6.3.13 QEI Digital Filter register (FILTER - 0x400B C03C)

This register contains the sampling count for the digital filter. A sampling count of zero bypasses the filter.

Table 500: QEI Digital Filter register (FILTER - address 0x400B C03C) bit description

Bit	Symbol	Description	Reset value
31:0	-	Digital filter sampling delay	0x0

© NXP B.V. 2010. All rights reserved.

6.4 Interrupt registers

6.4.1 QEI Interrupt Status register (QEIINTSTAT)

This register provides the status of the encoder interface and the current set of interrupt sources that are asserted to the controller. Bits set to 1 indicate the latched events that have occurred; a zero bit indicates that the event in question has not occurred. Writing a 0 to a bit position clears the corresponding interrupt.

Table 501: QEI Interrupt Status register (QEIINTSTAT - address 0x400B CFE0) bit description

Bit	Symbol	Description	Reset value
0	INX_Int	Indicates that an index pulse was detected.	0
1	TIM_Int	Indicates that a velocity timer overflow occurred	0
2	VELC_Int	Indicates that captured velocity is less than compare velocity.	0
3	DIR_Int	Indicates that a change of direction was detected.	0
4	ERR_Int	Indicates that an encoder phase error was detected.	0
5	ENCLK_Int	Indicates that and encoder clock pulse was detected.	
6	POS0_Int	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_Int	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_Int	Indicates that the position 2 compare value is equal to the current position.	0
9	REV_Int	Indicates that the index compare value is equal to the current index count.	0
10	POS0REV_Int	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV_Int is set.	0
11	POS1REV_Int	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV_Int is set.	0
12	POS2REV_Int	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV_Int is set.	0
31:13	-	reserved	0

6.4.2 QEI Interrupt Set register (QEISET - 0x400B CFEC)

Writing a one to a bit in this register sets the corresponding bit in the QEI Interrupt Status register (QEISTAT).

Table 502: QEI Interrupt Set register (QEISET - address 0x400B CFEC) bit description

Bit	Symbol	Description	Reset value
0	INX_Int	Indicates that an index pulse was detected.	0
1	TIM_Int	Indicates that a velocity timer overflow occurred	0
2	VELC_Int	Indicates that captured velocity is less than compare velocity.	0
3	DIR_Int	Indicates that a change of direction was detected.	0
4	ERR_Int	Indicates that an encoder phase error was detected.	0
5	ENCLK_Int	Indicates that and encoder clock pulse was detected.	
6	POS0_Int	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_Int	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_Int	Indicates that the position 2 compare value is equal to the current position.	0
9	REV_Int	Indicates that the index compare value is equal to the current index count.	0

© NXP B.V. 2010. All rights reserved.

Table 502: QEI Interrupt Set register (QEISET - address 0x400B CFEC) bit description

Bit	Symbol	Description	Reset value
10	POS0REV_Int	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV_Int is set.	0
11	POS1REV_Int	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV_Int is set.	0
12	POS2REV_Int	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV_Int is set.	0
31:13	-	reserved	0

6.4.3 QEI Interrupt Clear register (QEICLR - 0x400B CFE8)

Writing a 1 to a bit in this register clears the corresponding bit in the QEI Interrupt Status register (QEISTAT).

Table 503: QEI Interrupt Clear register (QEICLR - 0x400B CFE8) bit description

Bit	Symbol	Description	Reset value
0	INX_Int	Indicates that an index pulse was detected.	0
1	TIM_Int	Indicates that a velocity timer overflow occurred	0
2	VELC_Int	Indicates that captured velocity is less than compare velocity.	0
3	DIR_Int	Indicates that a change of direction was detected.	0
4	ERR_Int	Indicates that an encoder phase error was detected.	0
5	ENCLK_Int	Indicates that and encoder clock pulse was detected.	
6	POS0_Int	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_Int	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_Int	Indicates that the position 2 compare value is equal to the current position.	0
9	REV_Int	Indicates that the index compare value is equal to the current index count.	0
10	POS0REV_Int	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV_Int is set.	0
11	POS1REV_Int	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV_Int is set.	0
12	POS2REV_Int	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV_Int is set.	0
31:13	-	reserved	0

6.4.4 QEI Interrupt Enable register (QEIIE - 0x400B CFE4)

This register enables interrupt sources. Bits set to 1 enable the corresponding interrupt; a 0 bit disables the corresponding interrupt.

Table 504: QEI Interrupt Enable register (QEIIE - address 0x400B CFE4) bit description

Bit	Symbol	Description	Reset value
0	INX_Int	Indicates that an index pulse was detected.	0
1	TIM_Int	Indicates that a velocity timer overflow occurred	0
2	VELC_Int	Indicates that captured velocity is less than compare velocity.	0
3	DIR_Int	Indicates that a change of direction was detected.	0

© NXP B.V. 2010. All rights reserved.

Table 504: QEI Interrupt Enable register (QEIIE - address 0x400B CFE4) bit description

Bit	Symbol	Description	Reset value
4	ERR_Int	Indicates that an encoder phase error was detected.	0
5	ENCLK_Int	Indicates that and encoder clock pulse was detected.	0
6	POS0_Int	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_Int	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_Int	Indicates that the position 2 compare value is equal to the current position.	0
9	REV_Int	Indicates that the index compare value is equal to the current index count.	0
10	POS0REV_Int	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV_Int is set.	0
11	POS1REV_Int	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV_Int is set.	0
12	POS2REV_Int	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV_Int is set.	0
31:13	-	reserved	0

6.4.5 QEI Interrupt Enable Set register (QEIIES - 0x400B CFDC)

Writing a 1 to a bit in this register sets the corresponding bit in the QEI Interrupt Enable register (QEIIE).

Table 505: QEI Interrupt Enable Set register (QEIIES - address 0x400B CFDC) bit description

Bit	Symbol	Description	Reset value
0	INX_EN	Indicates that an index pulse was detected.	0
1	TIM_EN	Indicates that a velocity timer overflow occurred	0
2	VELC_EN	Indicates that captured velocity is less than compare velocity.	0
3	DIR_EN	Indicates that a change of direction was detected.	0
4	ERR_EN	Indicates that an encoder phase error was detected.	0
5	ENCLK_EN	Indicates that and encoder clock pulse was detected.	0
6	POS0_Int	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_Int	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_Int	Indicates that the position 2 compare value is equal to the current position.	0
9	REV_Int	Indicates that the index compare value is equal to the current index count.	0
10	POS0REV_Int	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV_Int is set.	0
11	POS1REV_Int	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV_Int is set.	0
12	POS2REV_Int	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV_Int is set.	0
31:13	-	reserved	0

© NXP B.V. 2010. All rights reserved.

6.4.6 QEI Interrupt Enable Clear register (QEIIEC - 0x400B CFD8)

Writing a 1 to a bit in this register clears the corresponding bit in the QEI Interrupt Enable register (QEIIE).

Table 506: QEI Interrupt Enable Clear register (QEIIEC - address 0x400B CFD8) bit description

Bit	Symbol	Description	Reset value
0	INX_EN	Indicates that an index pulse was detected.	0
1	TIM_EN	Indicates that a velocity timer overflow occurred	0
2	VELC_EN	Indicates that captured velocity is less than compare velocity.	0
3	DIR_EN	Indicates that a change of direction was detected.	0
4	ERR_EN	Indicates that an encoder phase error was detected.	0
5	ENCLK_EN	Indicates that and encoder clock pulse was detected.	0
6	POS0_Int	Indicates that the position 0 compare value is equal to the current position.	0
7	POS1_Int	Indicates that the position 1compare value is equal to the current position.	0
8	POS2_Int	Indicates that the position 2 compare value is equal to the current position.	0
9	REV_Int	Indicates that the index compare value is equal to the current index count.	0
10	POS0REV_Int	Combined position 0 and revolution count interrupt. Set when both the POS0_Int bit is set and the REV_Int is set.	0
11	POS1REV_Int	Combined position 1 and revolution count interrupt. Set when both the POS1_Int bit is set and the REV_Int is set.	0
12	POS2REV_Int	Combined position 2 and revolution count interrupt. Set when both the POS2_Int bit is set and the REV_Int is set.	0
31:13	-	reserved	0

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The RTC is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bits PCRTC.

   Remark: On reset, the RTC is enabled. See Section 27–7 for power saving options.
2. Clock: The RTC uses the 1 Hz clock output from the RTC oscillator as the only clock source. The peripheral clock rate for accessing registers is CCLK/8.
3. Interrupts: See Section 27–6.1 for RTC interrupt handling. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

2. Features

- Measures the passage of time to maintain a calendar and clock. Provides seconds, minutes, hours, day of month, month, year, day of week, and day of year.
- Ultra-low power design to support battery powered systems. Less than 1 microamp required for battery operation. Uses power from the CPU power supply when it is present.
- 20 bytes of Battery-backed storage and RTC operation when power is removed from the CPU.
- Dedicated 32 kHz ultra low power oscillator.
- Dedicated battery power supply pin.
- RTC power supply is isolated from the rest of the chip.
- Calibration counter allows adjustment to better than ±1 sec/day with 1 sec resolution.
- Periodic interrupts can be generated from increments of any field of the time registers and selected fractional second values.
- Alarm interrupt can be generated for a specific date/time.

3. Description

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off. It uses very little power when its registers are not being accessed by the CPU, especially reduced power modes. On the LPC17xx, the RTC is clocked by a separate 32 kHz oscillator that produces a 1 Hz internal time reference. The RTC is powered by its own power supply pin, VBAT, which can be connected to a battery, externally tied to a 3V supply, or left floating.

The RTC power domain is shown in conceptual form in Figure 27–129. A detailed view of the time keeping portion of the RTC is shown in Figure 27–130.

© NXP B.V. 2010. All rights reserved.

4. Architecture

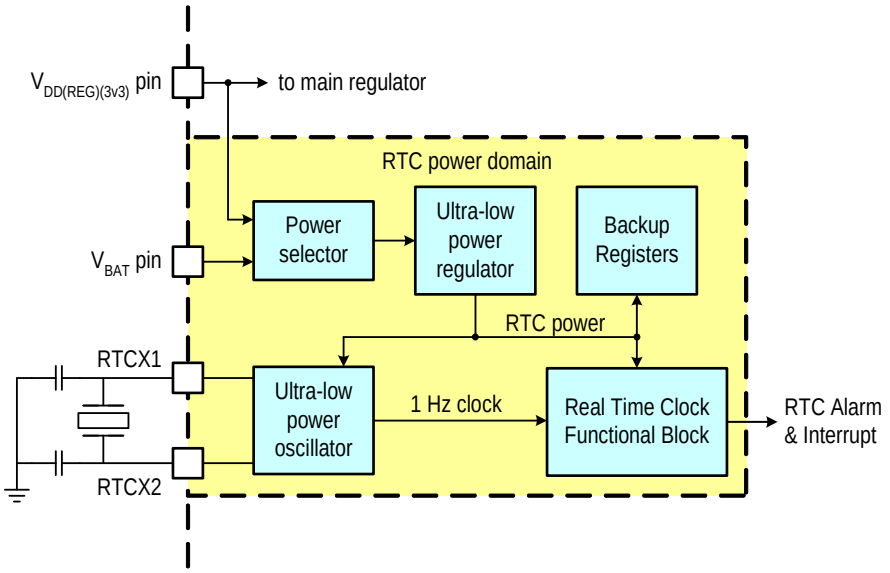


Fig 129. RTC domain conceptual diagram

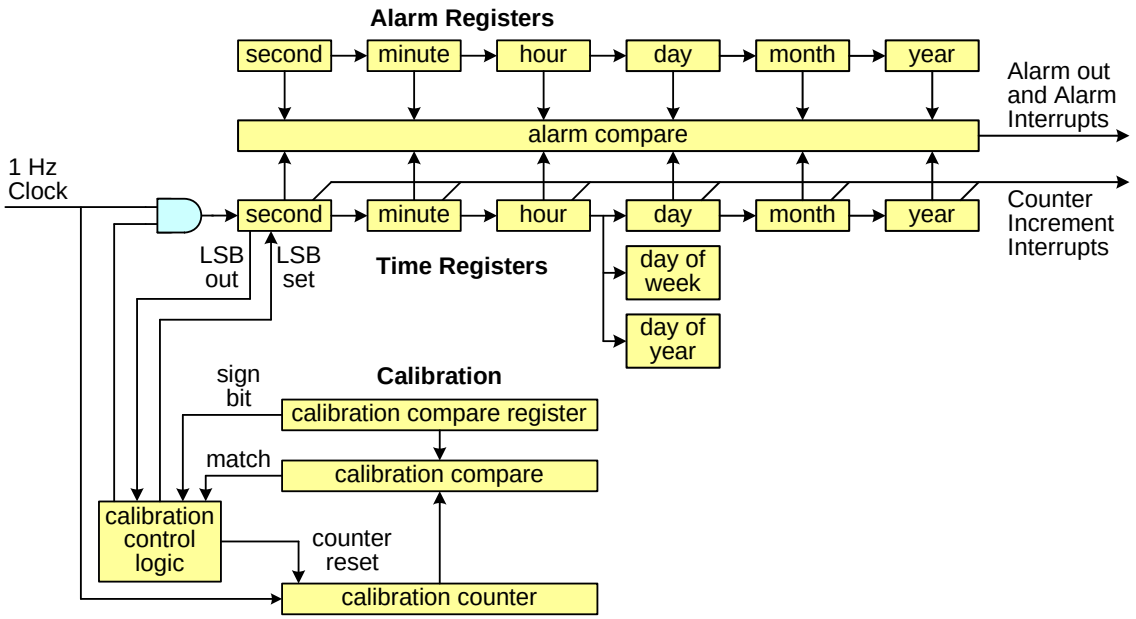


Fig 130. RTC functional block diagram

© NXP B.V. 2010. All rights reserved.

5. Pin description

Table 507. RTC pin description

Name	Type	Description
RTCX1	I	Input to the RTC oscillator circuit.
RTCX2	O	Output from the RTC oscillator circuit. **Remark:** If the RTC is not used, the RTCX1/2 pins can be left floating.
V_{BAT}	I	**RTC power supply:** Typically connected to an external 3V battery. If this pin is not powered, the RTC is still powered internally if $V_{DD(REG)(3V3)}$ is present.

6. Register description

The RTC includes a number of registers, shown in Table 27–508. Detailed descriptions of the registers follow. In these descriptions, for most of the registers the Reset Value column shows "NC", meaning that these registers are Not Changed by a Reset. Software must initialize these registers between power-on and setting the RTC into operation. The registers are split into five sections by functionality.

© NXP B.V. 2010. All rights reserved.

Table 508. Real-Time Clock register map

Name	Description	Access	Reset Value[1]	Address
Miscellaneous registers (see Section 27–6.2)				
ILR	Interrupt Location Register	R/W	0	0x4002 4000
CCR	Clock Control Register	R/W	NC	0x4002 4008
CIIR	Counter Increment Interrupt Register	R/W	0	0x4002 400C
AMR	Alarm Mask Register	R/W	0	0x4002 4010
RTC_AUX	RTC Auxiliary control register	R/W	0x8	0x4002 405C
RTC_AUXEN	RTC Auxiliary Enable register	R/W	0	0x4002 4058
Consolidated time registers (see Section 27–6.3)				
CTIME0	Consolidated Time Register 0	RO	NC	0x4002 4014
CTIME1	Consolidated Time Register 1	RO	NC	0x4002 4018
CTIME2	Consolidated Time Register 2	RO	NC	0x4002 401C
Time counter registers (see Section 27–6.4)				
SEC	Seconds Counter	R/W	NC	0x4002 4020
MIN	Minutes Register	R/W	NC	0x4002 4024
HOUR	Hours Register	R/W	NC	0x4002 4028
DOM	Day of Month Register	R/W	NC	0x4002 402C
DOW	Day of Week Register	R/W	NC	0x4002 4030
DOY	Day of Year Register	R/W	NC	0x4002 4034
MONTH	Months Register	R/W	NC	0x4002 4038
YEAR	Years Register	R/W	NC	0x4002 403C
CALIBRATION	Calibration Value Register	R/W	NC	0x4002 4040
General purpose registers (see Section 27–6.6)				
GPREG0	General Purpose Register 0	R/W	NC	0x4002 4044
GPREG1	General Purpose Register 1	R/W	NC	0x4002 4048
GPREG2	General Purpose Register 2	R/W	NC	0x4002 404C
GPREG3	General Purpose Register 3	R/W	NC	0x4002 4050
GPREG4	General Purpose Register 4	R/W	NC	0x4002 4054
Alarm register group (see Section 27–6.7)				
ALSEC	Alarm value for Seconds	R/W	NC	0x4002 4060
ALMIN	Alarm value for Minutes	R/W	NC	0x4002 4064
ALHOUR	Alarm value for Hours	R/W	NC	0x4002 4068
ALDOM	Alarm value for Day of Month	R/W	NC	0x4002 406C
ALDOW	Alarm value for Day of Week	R/W	NC	0x4002 4070
ALDOY	Alarm value for Day of Year	R/W	NC	0x4002 4074
ALMON	Alarm value for Months	R/W	NC	0x4002 4078
ALYEAR	Alarm value for Year	R/W	NC	0x4002 407C

[1] Reset values apply only to a power-up of the RTC block, other types of reset have no effect on this block. Since the RTC is powered whenever either of the $V_{DD(REG)(3V3)}$, or V_{BAT} supplies are present, power-up reset occurs only when both supplies were absent and then one is turned on. Most registers are not affected by power-up of the RTC and must be initialized by software if the RTC is enabled. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

© NXP B.V. 2010. All rights reserved.

6.1 RTC interrupts

Interrupt generation is controlled through the Interrupt Location Register (ILR), Counter Increment Interrupt Register (CIIR), the alarm registers, and the Alarm Mask Register (AMR). Interrupts are generated only by the transition into the interrupt state. The ILR separately enables CIIR and AMR interrupts. Each bit in CIIR corresponds to one of the time counters. If CIIR is enabled for a particular counter, then every time the counter is incremented an interrupt is generated. The alarm registers allow the user to specify a date and time for an interrupt to be generated. The AMR provides a mechanism to mask alarm compares. If all non-masked alarm registers match the value in their corresponding time counter, then an interrupt is generated.

The RTC interrupt can bring the microcontroller out of Power-down mode when the RTC is operating from its own oscillator on the RTCX1-2 pins. When the RTC interrupt is enabled for wake-up and its selected event occurs, the oscillator wake-up cycle associated with the XTAL1/2 pins is started. For details on the RTC based wake-up process see Section 4–8.8 “Wake-up from Reduced Power Modes” on page 63 and Section 4–9 “Wake-up timer” on page 66.

6.2 Miscellaneous register group

6.2.1 Interrupt Location Register (ILR - 0x4002 4000)

The Interrupt Location Register is a 2-bit register that specifies which blocks are generating an interrupt (see Table 27–509). 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 509. Interrupt Location Register (ILR - address 0x4002 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.	0
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.	0
31:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2.2 Clock Control Register (CCR - 0x4002 4008)

The clock register is a 4-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 27–510. All NC bits in this register should be initialized when the RTC is first turned on.

Table 510. Clock Control Register (CCR - address 0x4002 4008) bit description

Bit	Symbol	Value	Description	Reset value
0	CLKEN		Clock Enable.	NC
		1	The time counters are enabled.	
		0	The time counters are disabled so that they may be initialized.	

© NXP B.V. 2010. All rights reserved.

Table 510. Clock Control Register (CCR - address 0x4002 4008) bit description

Bit	Symbol	Value	Description	Reset value
1	CTCRST		CTC Reset.	0
		1	When one, the elements in the internal oscillator divider are reset, and remain reset until CCR[1] is changed to zero. This is the divider that generates the 1 Hz clock from the 32.768 kHz crystal. The state of the divider is not visible to software.	
		0	No effect.	
3:2	-		Internal test mode controls. These bits must be 0 for normal RTC operation.	NC
4	CCALEN		Calibration counter enable.	NC
		1	The calibration counter is disabled and reset to zero.	
		0	The calibration counter is enabled and counting, using the 1 Hz clock. When the calibration counter is equal to the value of the CALIBRATION register, the counter resets and repeats counting up to the value of the CALIBRATION register. See Section 27–6.4.2 and Section 27–6.5.	
31:5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2.3 Counter Increment Interrupt Register (CIIR - 0x4002 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 1 to bit 0 of the Interrupt Location Register (ILR[0]).

Table 511. Counter Increment Interrupt Register (CIIR - address 0x4002 400C) bit description

Bit	Symbol	Description	Reset value
0	IMSEC	When 1, an increment of the Second value generates an interrupt.	0
1	IMMIN	When 1, an increment of the Minute value generates an interrupt.	0
2	IMHOUR	When 1, an increment of the Hour value generates an interrupt.	0
3	IMDOM	When 1, an increment of the Day of Month value generates an interrupt.	0
4	IMDOW	When 1, an increment of the Day of Week value generates an interrupt.	0
5	IMDOY	When 1, an increment of the Day of Year value generates an interrupt.	0
6	IMMON	When 1, an increment of the Month value generates an interrupt.	0
7	IMYEAR	When 1, an increment of the Year value generates an interrupt.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2.4 Alarm Mask Register (AMR - 0x4002 4010)

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 27–512 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.

© NXP B.V. 2010. All rights reserved.

Table 512. Alarm Mask Register (AMR - address 0x4002 4010) bit description

Bit	Symbol	Description	Reset value
0	AMRSEC	When 1, the Second value is not compared for the alarm.	0
1	AMRMIN	When 1, the Minutes value is not compared for the alarm.	0
2	AMRHOUR	When 1, the Hour value is not compared for the alarm.	0
3	AMRDOM	When 1, the Day of Month value is not compared for the alarm.	0
4	AMRDOW	When 1, the Day of Week value is not compared for the alarm.	0
5	AMRDOY	When 1, the Day of Year value is not compared for the alarm.	0
6	AMRMON	When 1, the Month value is not compared for the alarm.	0
7	AMRYEAR	When 1, the Year value is not compared for the alarm.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2.5 RTC Auxiliary control register (RTC_AUX - 0x4002 405C)

The RTC Auxiliary Control register contains added interrupt flags for functions that are not part of the Real Time Clock itself (the part recording the passage of time and generating other time related functions). On the LPC17xx, the only added interrupt flag is for failure of the RTC oscillator.

Table 513. RTC Auxiliary control register (RTC_AUX - address 0x4002 405C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	RTC_OSCF	RTC Oscillator Fail detect flag. Read: this bit is set if the RTC oscillator stops, and when RTC power is first turned on. An interrupt will occur when this bit is set, the RTC_OSCFEN bit in RTC_AUXEN is a 1, and the RTC interrupt is enabled in the NVIC. Write: writing a 1 to this bit clears the flag.	1
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.2.6 RTC Auxiliary Enable register (RTC_AUXEN - 0x4002 4058)

The RTC Auxiliary Enable Register controls whether additional interrupt sources represented in the RTC Auxiliary control register are enabled.

Table 514. RTC Auxiliary Enable register (RTC_AUXEN - address 0x4002 4058) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	RTC_OSCFEN	Oscillator Fail Detect interrupt enable. When 0: the RTC Oscillator Fail detect interrupt is disabled. When 1: the RTC Oscillator Fail detect interrupt is enabled. See Section 27–6.2.5.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6.3 Consolidated time registers

The values of the Time Counters can optionally be read in a consolidated format which allows the programmer to read all time counters with only three read operations. The various registers are packed into 32-bit values as shown in Table 27–515, Table 27–516, and Table 27–517. 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.

6.3.1 Consolidated Time Register 0 (CTIME0 - 0x4002 4014)

The Consolidated Time Register 0 contains the low order time values: Seconds, Minutes, Hours, and Day of Week.

Table 515. Consolidated Time register 0 (CTIME0 - address 0x4002 4014) bit description

Bit	Symbol	Description	Reset value
5:0	Seconds	Seconds value in the range of 0 to 59	NC
7:6	-	Reserved, 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	NC
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	NC
23:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NC
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.	NC

6.3.2 Consolidated Time Register 1 (CTIME1 - 0x4002 4018)

The Consolidate Time Register 1 contains the Day of Month, Month, and Year values.

Table 516. Consolidated Time register 1 (CTIME1 - address 0x4002 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).	NC
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.	NC
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.	NC
31:28	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.3.3 Consolidated Time Register 2 (CTIME2 - 0x4002 401C)

The Consolidate Time Register 2 contains just the Day of Year value.

© NXP B.V. 2010. All rights reserved.

Table 517. Consolidated Time register 2 (CTIME2 - address 0x4002 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).	NC
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.4 Time Counter Group

The time value consists of the eight counters shown in Table 27–518 and Table 27–519. These counters can be read or written at the locations shown in Table 27–519.

Table 518. Time Counter relationships and values

Counter	Size	Enabled by	Minimum value	Maximum value
Second	6	1 Hz Clock	0	59
Minute	6	Second	0	59
Hour	5	Minute	0	23
Day of Month	5	Hour	1	28, 29, 30 or 31
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 519. Time Counter registers

Name	Size	Description	Access	Address
SEC	6	Seconds value in the range of 0 to 59	R/W	0x4002 4020
MIN	6	Minutes value in the range of 0 to 59	R/W	0x4002 4024
HOUR	5	Hours value in the range of 0 to 23	R/W	0x4002 4028
DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).[1]	R/W	0x4002 402C
DOW	3	Day of week value in the range of 0 to 6[1]	R/W	0x4002 4030
DOY	9	Day of year value in the range of 1 to 365 (366 for leap years)[1]	R/W	0x4002 4034
MONTH	4	Month value in the range of 1 to 12	R/W	0x4002 4038
YEAR	12	Year value in the range of 0 to 4095	R/W	0x4002 403C

[1] These values are simply incremented at the appropriate intervals and reset at the defined overflow point. They are not calculated and must be correctly initialized in order to be meaningful.

6.4.1 Leap year calculation

The RTC does a simple bit comparison to see if the two lowest order bits of the year counter are zero. If true, then the RTC considers that year a leap year. The RTC considers all years evenly divisible by 4 as leap years. This algorithm is accurate from the year 1901 through the year 2099, but fails for the year 2100, which is not a leap year. The only effect of leap year on the RTC is to alter the length of the month of February for the month, day of month, and year counters.

6.4.2 Calibration register (CALIBRATION - address 0x4002 4040)

The following register is used to calibrate the time counter.

© NXP B.V. 2010. All rights reserved.

Table 520. Calibration register (CALIBRATION - address 0x4002 4040) bit description

Bit	Symbol	Value	Description	Reset value
16:0	CALVAL	-	If enabled, the calibration counter counts up to this value. The maximum value is 131, 072 corresponding to about 36.4 hours. Calibration is disabled if CALVAL = 0.	NC
17	CALDIR		Calibration direction	NC
		1	Backward calibration. When CALVAL is equal to the calibration counter, the RTC timers will stop incrementing for 1 second.	
		0	Forward calibration. When CALVAL is equal to the calibration counter, the RTC timers will jump by 2 seconds.	
31:12			Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.5 Calibration procedure

The calibration logic can periodically adjust the time counter either by not incrementing the counter, or by incrementing the counter by 2 instead of 1. This allows calibrating the RTC oscillator under some typical voltage and temperature conditions without the need to externally trim the RTC oscillator.

A recommended method for determining the calibration value is to use the CLKOUT feature to unintrusively observe the RTC oscillator frequency under the conditions it is to be trimmed for, and calculating the number of clocks that will be seen before the time is off by one second. That value is used to determine CALVAL.

If the RTC oscillator is trimmed externally, the same method of unintrusively observing the RTC oscillator frequency may be helpful in that process.

Backward calibration

Enable the RTC timer and calibration in the CCR register (set bits CLKEN = 1 and CCALEN = 0). In the CALIBRATION register, set the calibration value CALVAL $\geq$ 1 and select CALDIR = 1.

- The SEC timer and the calibration counter count up for every 1 Hz clock cycle.
- When the calibration counter reaches CALVAL, a calibration match occurs and all RTC timers will be stopped for one clock cycle so that the timers will not increment in the next cycle.
- If an alarm match event occurs in the same cycle as the calibration match, the alarm interrupt will be delayed by one cycle to avoid a double alarm interrupt.

Forward calibration

Enable the RTC timer and calibration in the CCR register (set bits CLKEN = 1 and CCALEN = 0). In the CALIBRATION register, set the calibration value CALVAL $\geq$ 1 and select CALDIR = 0.

- The SEC timer and the calibration counter count up for every 1 Hz clock cycle.
- When the calibration counter reaches CALVAL, a calibration match occurs and the RTC timers are incremented by 2.
- When the calibration event occurs, the LSB of the ALSEC register is forced to be one so that the alarm interrupt will not be missed when skipping a second.

© NXP B.V. 2010. All rights reserved.

6.6 General purpose registers

6.6.1 General purpose registers 0 to 4 (GPREG0 to GPREG4 - addresses 0x4002 4044 to 0x4002 4054)

These registers can be used to store important information when the main power supply is off. The value in these registers is not affected by chip reset.

Table 521. General purpose registers 0 to 4 (GPREG0 to GPREG4 - addresses 0x4002 4044 to 0x4002 4054) bit description

Bit	Symbol	Description	Reset value
31:0	GP0 to GP4	General purpose storage.	N/A

6.7 Alarm register group

The alarm registers are shown in Table 27–522. The values in these registers are compared with the time counters. If all the unmasked (See Section 27–6.2.4 “Alarm Mask Register (AMR - 0x4002 4010)” on page 560) alarm registers match their corresponding time counters then an interrupt is generated. The interrupt is cleared when a 1 is written to bit 1 of the Interrupt Location Register (ILR[1]).

Table 522. Alarm registers

Name	Size	Description	Access	Address
ALSEC	6	Alarm value for Seconds	R/W	0x4002 4060
ALMIN	6	Alarm value for Minutes	R/W	0x4002 4064
ALHOUR	5	Alarm value for Hours	R/W	0x4002 4068
ALDOM	5	Alarm value for Day of Month	R/W	0x4002 406C
ALDOW	3	Alarm value for Day of Week	R/W	0x4002 4070
ALDOY	9	Alarm value for Day of Year	R/W	0x4002 4074
ALMON	4	Alarm value for Months	R/W	0x4002 4078
ALYEAR	12	Alarm value for Years	R/W	0x4002 407C

7. RTC usage notes

If the RTC is used, V_{BAT} may be connected to an independent power supply (typically an external battery), or left floating. The RTC domain will always be internally powered if $V_{DD(REG)(3V3)}$ is present, even if there is no power applied to V_{BAT}. As long as power is available on either $V_{DD(REG)(3V3)}$ or V_{BAT}, the RTC will lose its time value and backup register contents. If both $V_{DD(REG)(3V3)}$ and V_{BAT} are not present, all RTC information will be lost. RTC incrementation will stop or be unpredictable if the clock source is lost, interrupted, or altered.

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 28: LPC17xx Watchdog Timer (WDT)

Rev. 01 — 4 January 2010 User manual

1. Features

- Internally resets chip if not periodically reloaded.
- Debug mode.
- Enabled by software but requires a hardware reset or a Watchdog reset/interrupt to be disabled.
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled.
- Flag to indicate Watchdog reset.
- Programmable 32-bit timer with internal pre-scaler.
- Selectable time period from ($T_{WDCLK} \times 256 \times 4$) to ($T_{WDCLK} \times 2^{32} \times 4$) in multiples of $T_{WDCLK} \times 4$.
- The Watchdog clock (WDCLK) source can be selected from the Internal RC oscillator (IRC), the APB peripheral clock (PCLK, see Table 4–40), or the RTC oscillator. This gives a wide range of potential timing choices for Watchdog operation under different power reduction conditions. For increased reliability, it also provides the ability to run the Watchdog timer from an entirely internal source that is not dependent on an external crystal and its associated components and wiring.
- The Watchdog timer can be configured to run in Deep Sleep mode when using the IRC as the clock source.

2. Applications

The purpose of the Watchdog is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state. When enabled, the Watchdog will generate a system reset if the user program fails to "feed" (or reload) the Watchdog within a predetermined amount of time.

For interaction of the on-chip watchdog and other peripherals, especially the reset and boot-up procedures, please read Section 3–4 “Reset” on page 18 of this document.

© NXP B.V. 2010. All rights reserved.

3. Description

The Watchdog consists of a divide by 4 fixed pre-scaler and a 32-bit counter. The clock is fed to the timer via a pre-scaler. The timer decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{32} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup the Watchdog timer operating mode in WDMOD register.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register.
- The Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.

When the Watchdog is in the reset mode and the counter underflows, the CPU will be reset, loading the stack pointer and program counter from the vector table as in the case of external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

The watchdog timer block uses two clocks: PCLK and WDCLK. PCLK is used for the APB accesses to the watchdog registers. The WDCLK is used for the watchdog timer counting.

There is some synchronization logic between these two clock domains. When the WDMOD and WDTC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain. When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the PCLK for reading as the WDTV register by the CPU.

4. Register description

The Watchdog contains 4 registers as shown in Table 28–523 below.

Table 523. 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	0x4000 0000
WDTC	Watchdog timer constant register. This register determines the time-out value.	R/W	0xFF	0x4000 0004
WDFEED	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	WO	NA	0x4000 0008
WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	RO	0xFF	0x4000 000C
WDCLKSEL	Watchdog clock source selection register.	R/W	0	0x4000 0010

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

© NXP B.V. 2010. All rights reserved.

4.1 Watchdog Mode register (WDMOD - 0x4000 0000)

The WDMOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 524: Watchdog Mode register (WDMOD - address 0x4000 0000) bit description

Bit	Symbol	Description	Reset Value
0	WDEN	WDEN Watchdog enable bit (set-only). When 1, the watchdog timer is running.	0
1	WDRESET	WDRESET Watchdog reset enable bit (set -only). When 1, a watchdog timeout will cause a chip reset.	0
2	WDTOF	WDTOF Watchdog time-out flag. Set when the watchdog timer times out, cleared by software.	0 (Only after external reset)
3	WDINT	WDINT Watchdog interrupt flag (read-only, not clearable by software).	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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 NVIC or the watchdog interrupt request will be generated indefinitely. the intent of the watchdog interrupt is to allow debugging watchdog activity without resetting the device when the watchdog overflows.

Watchdog reset or interrupt will occur any time the watchdog is running and has an operating clock source. Any clock source works in Sleep mode, and the IRC works in Deep Sleep mode. If a watchdog interrupt occurs in Sleep or Deep Sleep mode, it will wake up the device.

Table 525. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: debug with the Watchdog interrupt but no WDRESET enabled. When this mode is selected, a watchdog counter underflow will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: operate with the Watchdog interrupt and WDRESET enabled. When this mode is selected, a watchdog counter underflow will reset the microcontroller. Although the Watchdog interrupt is also enabled in this case (WDEN = 1) it will not be recognized since the watchdog reset will clear the WDINT flag.

© NXP B.V. 2010. All rights reserved.

4.2 Watchdog Timer Constant register (WDTC - 0x4000 0004)

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. It's a 32-bit register with 8 LSB set to 1 on reset. Writing values below 0xFF will cause 0x0000 00FF to be loaded to the WDTC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

Table 526: Watchdog Constant register (WDTC - address 0x4000 0004) bit description

Bit	Symbol	Description	Reset Value
31:0	Count	Watchdog time-out interval.	0x0000 00FF

4.3 Watchdog Feed register (WDFEED - 0x4000 0008)

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the WDTC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors. After writing 0xAA to WDFEED, access to any Watchdog register other than writing 0x55 to WDFEED causes an immediate reset/interrupt when the Watchdog is enabled. The reset will be generated during the second PCLK following an incorrect access to a Watchdog register during a feed sequence.

Interrupts should be disabled during the feed sequence. An abort condition will occur if an interrupt happens during the feed sequence.

Table 527: Watchdog Feed register (WDFEED - address 0x4000 0008) bit description

Bit	Symbol	Description	Reset Value
7:0	Feed	Feed value should be 0xAA followed by 0x55.	NA
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.4 Watchdog Timer Value register (WDTV - 0x4000 000C)

The WDTV register is used to read the current value of Watchdog timer.

When reading the value of the 32-bit timer, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 PCLK cycles, so the value of WDTV is older than the actual value of the timer when it's being read by the CPU.

Table 528: Watchdog Timer Value register (WDTV - address 0x4000 000C) bit description

Bit	Symbol	Description	Reset Value
31:0	Count	Counter timer value.	0x0000 00FF

4.5 Watchdog Timer Clock Source Selection register (WDCLKSEL - 0x4000 0010)

This register allows selecting the clock source for the Watchdog timer. The possibilities are the Internal RC oscillator (IRC) or the APB peripheral clock (pclk). The function of bits in WDCLKSEL are shown in Table 28–529. The clock source selection can be locked by software, so that it cannot be modified. On reset, the clock source selection bits are always unlocked.

© NXP B.V. 2010. All rights reserved.

When the IRC is chosen as the watchdog clock source, the watchdog timer can remain running in deep sleep mode, and can reset or wake up the device from that mode.

Table 529: Watchdog Timer Clock Source Selection register (WDCLKSEL - address 0x4000 0010) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WDSEL		These bits select the clock source for the Watchdog timer as described below. **Warning:** Improper setting of this value may result in incorrect operation of the Watchdog timer, which could adversely affect system operation. If the WDLOCK bit in this register is set, the WDSEL bit cannot be modified.	0
		00	Selects the Internal RC oscillator (irc_clk) as the Watchdog clock source (default).	
		01	Selects the APB peripheral clock (watchdog pclk) as the Watchdog clock source.	
		10	Selects the RTC oscillator (rtc_clk) as the Watchdog clock source.	
		11	Reserved, this setting should not be used.	
30:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31	WDLOCK	0	This bit is set to 0 on any reset. It cannot be cleared by software.	0
		1	Software can set this bit to 1 at any time. Once WDLOCK is set, the bits of this register cannot be modified.	

5. Block diagram

The block diagram of the Watchdog is shown below in the Figure 28–131. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

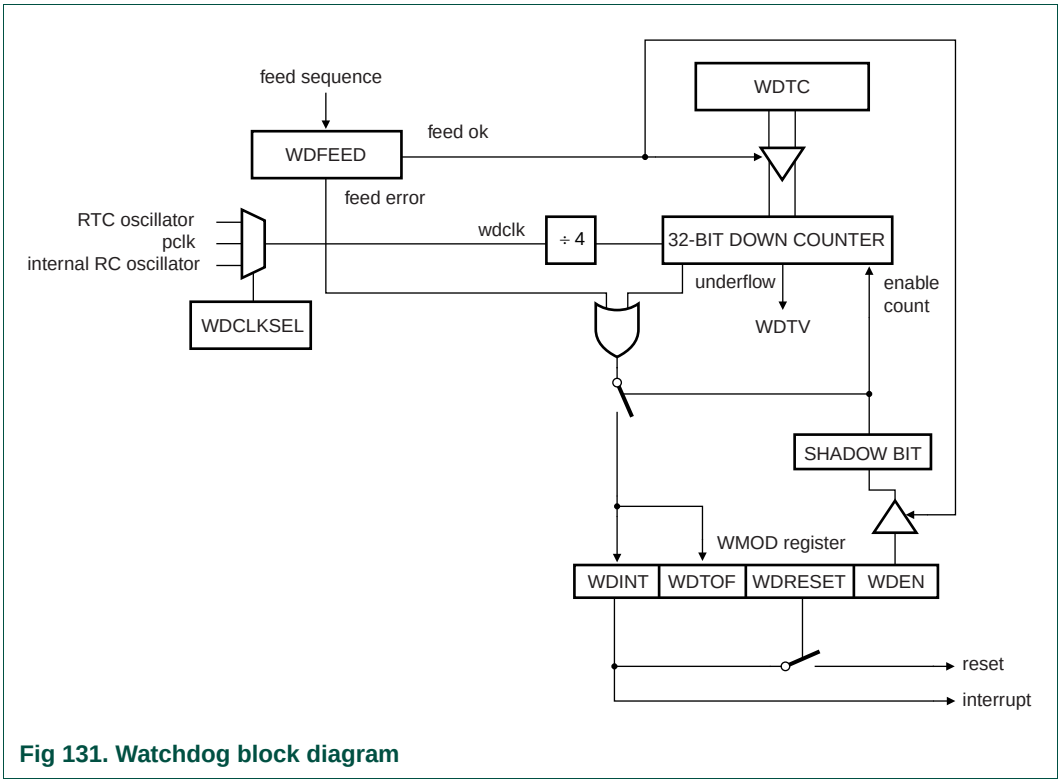


Fig 131. Watchdog block diagram

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The ADC is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set the PCADC bit.

   Remark: On reset, the ADC is disabled. To enable the ADC, first set the PCADC bit, and then enable the ADC in the AD0CR register (bit PDN Table 29–532). To disable the ADC, first clear the PDN bit, and then clear the PCADC bit.
2. Clock: In the PCLKSEL0 register (Table 4–40), select PCLK_ADC. To scale the clock for the ADC, see bits CLKDIV in Table 29–532.
3. Pins: Enable ADC0 pins through PINSEL registers. Select the pin modes for the port pins with ADC0 functions through the PINMODE registers (Section 8–5).
4. Interrupts: To enable interrupts in the ADC, see Table 29–536. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. Disable the ADC interrupt in the NVIC using the appropriate Interrupt Set Enable register.
5. DMA: See Section 29–6.4. For GPDMA system connections, see Table 31–544.

2. Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among 8 pins.
- Power-down mode.
- Measurement range V_{REFN} to V_{REFP} (typically 3 V; not to exceed V_{DDA} voltage level).
- 12-bit conversion rate of 200 kHz.
- Burst conversion mode for single or multiple inputs.
- Optional conversion on transition on input pin or Timer Match signal.

3. Description

Basic clocking for the A/D converters is provided by the APB clock. A programmable divider is included in each converter to scale this clock to the clock (maximum 13 MHz) needed by the successive approximation process. A fully accurate conversion requires 65 of these clocks.

© NXP B.V. 2010. All rights reserved.

4. Pin description

Table 29–530 gives a brief summary of each of ADC related pins.

Table 530. ADC pin description

Pin	Type	Description
AD0.7 to AD0.0	Input	**Analog Inputs.** The ADC cell can measure the voltage on any of these input signals. Digital signals are disconnected from the ADC input pins when the ADC function is selected on that pin in the Pin Select register. **Warning:** if the ADC is used, signal levels on analog input pins must not be above the level of V_{DDA} at any time. Otherwise, A/D converter readings will be invalid. If the A/D converter is not used in an application then the pins associated with A/D inputs can be used as 5 V tolerant digital IO pins.
V_{REFP}, V_{REFN}	Reference	**Voltage References.** These pins provide a voltage reference level for the ADC and DAC. **Note: V_{REFP} should be tied to VDD(3V3) and V_{REFN} should be tied to V_{SS} if the ADC and DAC are not used.**
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. **Note: V_{DDA} should be tied to VDD(3V3) and V_{SSA} should be tied to V_{SS} if the ADC and DAC are not used.**

© NXP B.V. 2010. All rights reserved.

5. Register description

The A/D Converter registers are shown in Table 29–531.

Table 531. ADC registers

Generic Name	Description	Access	Reset value[1]	AD0 Name & Address
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	1	AD0CR - 0x4003 4000
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	AD0GDR - 0x4003 4004
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	0x100	AD0INTEN - 0x4003 400C
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	AD0DR0 - 0x4003 4010
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	AD0DR1 - 0x4003 4014
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	AD0DR2 - 0x4003 4018
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	AD0DR3 - 0x4003 401C
ADDR4	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	AD0DR4 - 0x4003 4020
ADDR5	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	AD0DR5 - 0x4003 4024
ADDR6	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	AD0DR6 - 0x4003 4028
ADDR7	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	AD0DR7 - 0x4003 402C
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/DMA flag.	RO	0	AD0STAT - 0x4003 4030
ADTRM	ADC trim register.	R/W	0	AD0TRM - 0x4003 4034

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

© NXP B.V. 2010. All rights reserved.

5.1 A/D Control Register (AD0CR - 0x4003 4000)

Table 532: A/D Control Register (AD0CR - address 0x4003 4000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	SEL		Selects which of the AD0.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones is allowed. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The APB clock (PCLK_ADC0) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 13 MHz. Typically, software should program the smallest value in this field that yields a clock of 13 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 up to 200 kHz, scanning (if necessary) through the pins selected by bits set to ones in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. **Remark:** START bits must be 000 when BURST = 1 or conversions will not start.	0
		0	Conversions are software controlled and require 65 clocks.	
20:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PDN	1	The A/D converter is operational.	0
		0	The A/D converter is in power-down mode.	
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	
		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on the P2.10 / EINT0 / NMI pin.	
		011	Start conversion when the edge selected by bit 27 occurs on the P1.27 / CLKOUT / USB_OVRCRn / CAP0.1 pin.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1. Note that this does not require that the MAT0.1 function appear on a device pin.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3. Note that it is not possible to cause the MAT0.3 function to appear on a device pin.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0. Note that this does not require that the MAT1.0 function appear on a device pin.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1. Note that this does not require that the MAT1.1 function appear on a device pin.	
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

5.2 A/D Global Data Register (AD0GDR - 0x4003 4004)

The A/D Global Data Register holds the result of the most recent A/D conversion that has completed, and also includes copies of the status flags that go with that conversion.

Results of ADC conversion can be read in one of two ways. One is to use the A/D Global Data Register to read all data from the ADC. Another is to use the A/D Channel Data Registers. It is important to use one method consistently because the DONE and OVERRUN flags can otherwise get out of synch between the AD0GDR and the A/D Channel Data Registers, potentially causing erroneous interrupts or DMA activity.

Table 533: A/D Global Data Register (AD0GDR - address 0x4003 4004) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin selected by the SEL field, as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0x3FF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	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	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

5.3 A/D Interrupt Enable register (AD0INTEN - 0x4003 400C)

This register allows control over which A/D channels generate an interrupt when a conversion is complete. For example, it may be desirable to use some A/D channels to monitor sensors by continuously performing conversions on them. The most recent results are read by the application program whenever they are needed. In this case, an interrupt is not desirable at the end of each conversion for some A/D channels.

Table 534: A/D Status register (AD0INTEN - address 0x4003 400C) bit description

Bit	Symbol	Value	Description	Reset value
0	ADINTEN0	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.	

© NXP B.V. 2010. All rights reserved.

Table 534: A/D Status register (AD0INTEN - address 0x4003 400C) bit description

Bit	Symbol	Value	Description	Reset value
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.	
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	ADINTEN7	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

5.4 A/D Data Registers (AD0DR0 to AD0DR7 - 0x4003 4010 to 0x4003 402C)

The A/D Data Registers hold the result of the last conversion for each A/D channel, when an A/D conversion is complete. They also include the flags that indicate when a conversion has been completed and when a conversion overrun has occurred.

Results of ADC conversion can be read in one of two ways. One is to use the A/D Global Data Register to read all data from the ADC. Another is to use the A/D Channel Data Registers. It is important to use one method consistently because the DONE and OVERRUN flags can otherwise get out of synch between the AD0GDR and the A/D Channel Data Registers, potentially causing erroneous interrupts or DMA activity.

Table 535: A/D Data Registers (AD0DR0 to AD0DR7 - 0x4003 4010 to 0x4003 402C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0x3FF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	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.	
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.	NA

© NXP B.V. 2010. All rights reserved.

5.5 A/D Status register (ADSTAT - 0x4003 4030)

The A/D Status register allows checking the status of all A/D channels simultaneously. The DONE and OVERRUN flags appearing in the ADDRn register for each A/D channel are mirrored in ADSTAT. The interrupt flag (the logical OR of all DONE flags) is also found in ADSTAT.

Table 536: A/D Status register (AD0STAT - address 0x4003 4030) bit description

Bit	Symbol	Description	Reset value
0	DONE0	This bit mirrors the DONE status flag from the result register for A/D channel 0.	0
1	DONE1	This bit mirrors the DONE status flag from the result register for A/D channel 1.	0
2	DONE2	This bit mirrors the DONE status flag from the result register for A/D channel 2.	0
3	DONE3	This bit mirrors the DONE status flag from the result register for A/D channel 3.	0
4	DONE4	This bit mirrors the DONE status flag from the result register for A/D channel 4.	0
5	DONE5	This bit mirrors the DONE status flag from the result register for A/D channel 5.	0
6	DONE6	This bit mirrors the DONE status flag from the result register for A/D channel 6.	0
7	DONE7	This bit mirrors the DONE status flag from the result register for A/D channel 7.	0
8	OVERRUN0	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 0.	0
9	OVERRUN1	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 1.	0
10	OVERRUN2	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 2.	0
11	OVERRUN3	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 3.	0
12	OVERRUN4	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 4.	0
13	OVERRUN5	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 5.	0
14	OVERRUN6	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 6.	0
15	OVERRUN7	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 7.	0
16	ADINT	This bit is the A/D interrupt flag. It is one when any of the individual A/D channel Done flags is asserted and enabled to contribute to the A/D interrupt via the ADINTEN register.	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

5.6 A/D Trim register (ADTRIM - 0x4003 4034)

This register will be set by the bootcode on start-up. It contains the trim values for the DAC and the ADC. The offset trim values for the ADC can be overwritten by the user. All 12 bits are visible when this register is read.

Table 537: A/D Trim register (ADTRM - address 0x4003 4034) bit description

Bit	Symbol	Description	Reset value
3:0	-	reserved.	NA
7:4	ADCOFFS	Offset trim bits for ADC operation. Initialized by the boot code. Can be overwritten by the user.	0
11:8	TRIM	written-to by boot code. Can **not** be overwritten by the user. These bits are locked after boot code write.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

6. Operation

Once an ADC conversion is started, it cannot be interrupted. A new software write to launch a new conversion or a new edge-trigger event will be ignored while the previous conversion is in progress.

6.1 Hardware-triggered conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the ADC will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

6.2 Interrupts

An interrupt request is asserted to the NVIC when the DONE bit is 1. Software can use the Interrupt Enable bit for the A/D Converter in the NVIC to control whether this assertion results in an interrupt. DONE is negated when the ADDR is read.

6.3 Accuracy vs. digital receiver

The ADC function must be selected via the PINSEL registers in order to get accurate voltage readings on the monitored pin. The PINMODE should also be set to the mode for which neither pull-up nor pull-down resistor is enabled. For a pin hosting an ADC input, it is not possible to have a have a digital function selected and yet get valid ADC readings. An inside circuit disconnects ADC hardware from the associated pin whenever a digital function is selected on that pin.

6.4 DMA control

A DMA transfer request is generated from the ADC interrupt request line. To generate a DMA transfer the same conditions must be met as the conditions for generating an interrupt (see Section 29–6.2 and Section 29–5.3).

Remark: If the DMA is used, the ADC interrupt must be disabled in the NVIC.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register (see Section 31–5.20). If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8 - see Section 31–5.20), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted (see Section 31–5.20). Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists (see Section 31–5.19).

© NXP B.V. 2010. All rights reserved.

1. Basic configuration

The DAC is configured using the following registers:

1. Power: The DAC is always connected to V_{DDA}. Register access is determined by PINSEL and PINMODE settings (see below).
2. Clock: In the PCLKSEL0 register (Table 4–40), select PCLK_DAC.
3. Pins: Enable the DAC pin through the PINSEL registers. Select pin mode for port pin with DAC through the PINMODE registers (Section 8–5). This must be done before accessing any DAC registers.
4. DMA: The DAC can be connected to the GPDMA controller (see Section 30–4.2). For GPDMA connections, see Table 31–544.

2. Features

- 10-bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power-down mode
- Selectable speed vs. power
- Maximum update rate of 1 MHz.

3. Pin description

Table 30–538 gives a brief summary of each of DAC related pins.

Table 538. D/A Pin Description

Pin	Type	Description
AOUT	Output	**Analog Output.** After the selected settling time after the DACR is written with a new value, the voltage on this pin (with respect to V_{SSA}) is VALUE × ((V_{REFP} - V_{REFN})/1024) + V_{REFN}.
V_{REFP}, V_{REFN}	Reference	**Voltage References.** These pins provide a voltage reference level for the ADC and DAC. **Note: V_{REFP} should be tied to VDD(3V3) and V_{REFN} should be tied to V_{SS} if the ADC and DAC are not used.**
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. **Note: V_{DDA} should be tied to VDD(3V3) and V_{SSA} should be tied to V_{SS} if the ADC and DAC are not used.**

© NXP B.V. 2010. All rights reserved.

4. Register description

The DAC registers are shown in Table 30–539. Note that the DAC does not have a control bit in the PCONP register. To enable the DAC, its output must be selected to appear on the related pin, P0.26, by configuring the PINSEL1 register. See Section 8–5.2 “Pin Function Select Register 1 (PINSEL1 - 0x4002 C004)”. the DAC must be enabled in this manner prior to accessing any DAC registers.

Table 539. DAC registers

Name	Description	Access	Reset value[1]	Address
DACR	D/A Converter Register. This register contains the digital value to be converted to analog and a power control bit.	R/W	0	0x4008 C000
DACCTRL	DAC Control register. This register controls DMA and timer operation.	R/W	0	0x4008 C004
DACCNTVAL	DAC Counter Value register. This register contains the reload value for the DAC DMA/Interrupt timer.	R/W	0	0x4008 C008

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

4.1 D/A Converter Register (DACR - 0x4008 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 540: D/A Converter Register (DACR - address 0x4008 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 × ((V_{REFP} - V_{REFN})/1024) + V_{REFN}.	0
16	BIAS[1]	0	The settling time of the DAC is 1 μs max, and the maximum current is 700 μA. This allows a maximum update rate of 1 MHz.	0
		1	The settling time of the DAC is 2.5 μs and the maximum current is 350 μA. This allows a maximum update rate of 400 kHz.	
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] The settling times noted in the description of the BIAS bit are valid for a capacitance load on the AOUT pin not exceeding 100 pF. A load impedance value greater than that value will cause settling time longer than the specified time. One or more graph(s) of load impedance vs. settling time will be included in the final data sheet.

4.2 D/A Converter Control register (DACCTRL - 0x4008 C004)

This read/write register enables the DMA operation and controls the DMA timer.

© NXP B.V. 2010. All rights reserved.

Table 541. D/A Control register (DACCTRL - address 0x4008 C004) bit description

Bit	Symbol	Value	Description	Reset Value
0	INT_DMA_REQ	0	This bit is cleared on any write to the DACR register.	0
		1	This bit is set by hardware when the timer times out.	
1	DBLBUF_ENA	0	DACR double-buffering is disabled.	0
		1	When this bit and the CNT_ENA bit are both set, the double-buffering feature in the DACR register will be enabled. Writes to the DACR register are written to a pre-buffer and then transferred to the DACR on the next time-out of the counter.	
2	CNT_ENA	0	Time-out counter operation is disabled.	0
		1	Time-out counter operation is enabled.	
3	DMA_ENA	0	DMA access is disabled.	0
		1	DMA Burst Request Input 7 is enabled for the DAC (see Table 31–544).	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

4.3 D/A Converter Counter Value register (DACCNTVAL - 0x4008 C008)

This read/write register contains the reload value for the Interrupt/DMA counter.

Table 542: D/A Converter register (DACR - address 0x4008 C000) bit description

Bit	Symbol	Description	Reset Value
15:0	VALUE	16-bit reload value for the DAC interrupt/DMA timer.	0

5. Operation

5.1 DMA counter

When the counter enable bit CNT_ENA in DACCTRL is set, a 16-bit counter will begin counting down, at the rate selected by PCLK_DAC (see Table 4–40), from the value programmed into the DACCNTVAL register. The counter is decremented Each time the counter reaches zero, the counter will be reloaded by the value of DACCNTVAL and the DMA request bit INT_DMA_REQ will be set in hardware.

Note that the contents of the DACCTRL and DACCNTVAL registers are read and write accessible, but the timer itself is not accessible for either read or write.

If the DMA_ENA bit is set in the DACCTRL register, the DAC DMA request will be routed to the GPDMA. When the DMA_ENA bit is cleared, the default state after a reset, DAC DMA requests are blocked.

5.2 Double buffering

Double-buffering is enabled only if both, the CNT_ENA and the DBLBUF_ENA bits are set in DACCTRL. In this case, any write to the DACR register will only load the pre-buffer, which shares its register address with the DACR register. The DACR itself will be loaded from the pre-buffer whenever the counter reaches zero and the DMA request is set. At the same time the counter is reloaded with the COUNTVAL register value.

Reading the DACR register will only return the contents of the DACR register itself, not the contents of the pre-buffer register.

© NXP B.V. 2010. All rights reserved.

If either the CNT_ENA or the DBLBUF_ENA bits are 0, any writes to the DACR address will go directly to the DACR register.

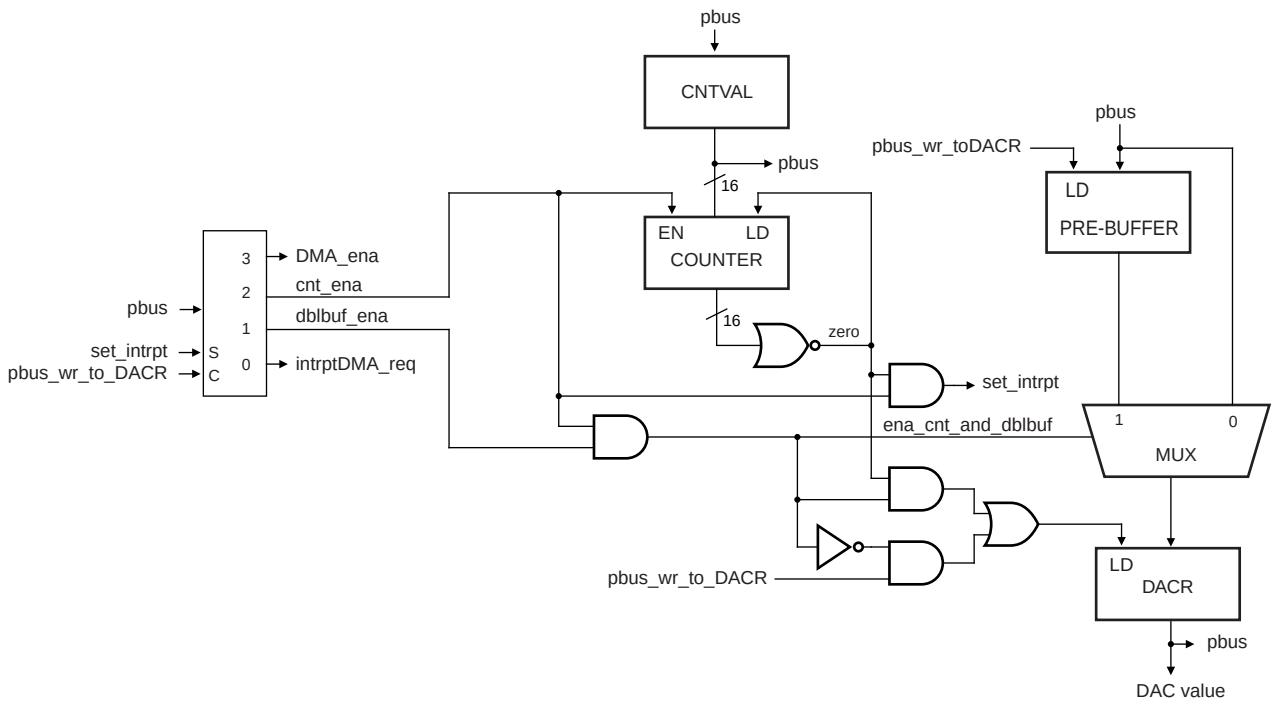


Fig 132. DAC control with DMA interrupt and timer

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 31: LPC17xx General Purpose DMA (GPDMA) controller

Rev. 01 — 4 January 2010 User manual

1. Basic configuration

The GPDMA is configured using the following registers:

1. Power: In the PCONP register (Table 4–46), set bit PCGPDMA.

   Remark: On reset, the GPDMA is disabled (PCGPDMA = 0).
2. Clock: see Table 4–38.
3. Interrupts: Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
4. Programming: see Section 31–6.

2. Introduction

The DMA controller allows peripheral-to memory, memory-to-peripheral, and memory-to-memory transactions. Each DMA stream provides unidirectional serial DMA transfers for a single source and destination. For example, a bi-directional port requires one stream for transmit and one for receives. The source and destination areas can each be either a memory region or a peripheral.

3. Features

- Eight DMA channels. Each channel can support an unidirectional transfer.
- 16 DMA request lines.
- Memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers are supported.
- GPDMA supports the SSP, I2S, UART, A/D Converter, and D/A Converter peripherals. DMA can also be triggered by a timer match condition. Memory-to-memory transfers and transfers to or from GPIO are also supported.
- Scatter or gather DMA is supported through the use of linked lists. This means that the source and destination areas do not have to occupy contiguous areas of memory.
- Hardware DMA channel priority.
- AHB slave DMA programming interface. The DMA Controller is programmed by writing to the DMA control registers over the AHB slave interface.
- One AHB bus master for transferring data. The interface transfers data when a DMA request goes active.
- 32-bit AHB master bus width.
- Incrementing or non-incrementing addressing for source and destination.
- Programmable DMA burst size. The DMA burst size can be programmed to more efficiently transfer data.
- Internal four-word FIFO per channel.

© NXP B.V. 2010. All rights reserved.

- Supports 8-bit, 16-bit, and 32-bit wide transactions.
- Big-endian and little-endian support. The DMA Controller defaults to little-endian mode on reset.
- An interrupt to the processor can be generated on a DMA completion or when a DMA error has occurred.
- Raw interrupt status. The DMA error and DMA count raw interrupt status can be read prior to masking.
- DMA can operate in Sleep mode. (Note that in Sleep mode the GPDMA cannot access the flash memory).

4. Functional description

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

4.1 DMA controller functional description

The DMA Controller enables peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral, and memory-to-memory transactions. Each DMA stream provides unidirectional serial DMA transfers for a single source and destination. For example, a bidirectional port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral, and can be accessed through the AHB master. Figure 31–133 shows a block diagram of the DMA Controller.

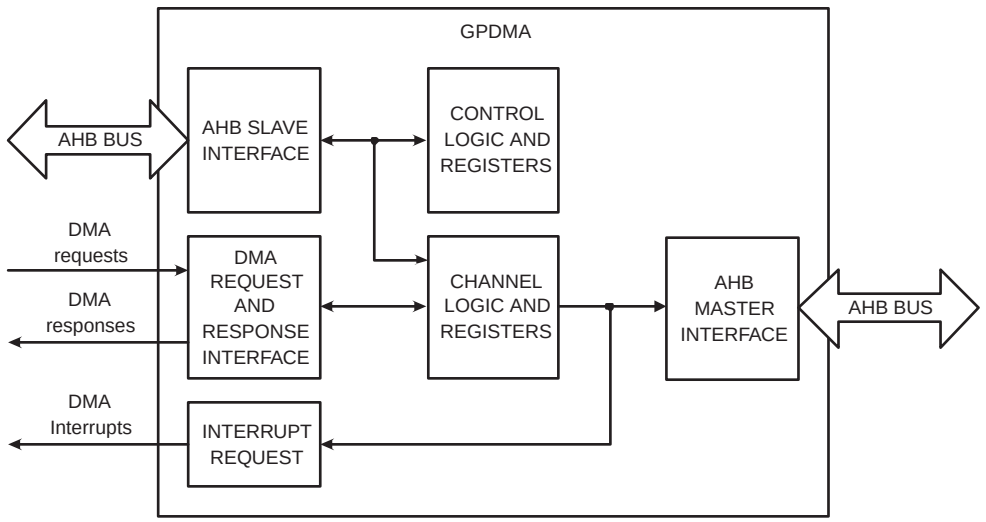


Fig 133. DMA controller block diagram

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

4.1.1 AHB slave interface

All transactions to DMA Controller registers on the AHB slave interface are 32 bits wide. 8-bit and 16-bit accesses are not supported and will result in an exception.

© NXP B.V. 2010. All rights reserved.

4.1.2 Control logic and register bank

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

4.1.3 DMA request and response interface

See Section 31–4.2 for information on the DMA request and response interface.

4.1.4 Channel logic and channel register bank

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

4.1.5 Interrupt request

The interrupt request generates the interrupt to the ARM processor.

4.1.6 AHB master interface

The DMA Controller contains one AHB master interface. The AHB master is capable of dealing with all types of AHB transactions, including:

- Split, retry, and error responses from slaves. If a peripheral performs a split or retry, the DMA Controller stalls and waits until the transaction can complete.
- Locked transfers for source and destination of each stream.
- Setting of protection bits for transfers on each stream.

4.1.6.1 Bus and transfer widths

The physical width of the AHB bus is 32 bits. Source and destination transfers can be of differing widths and can be the same width or narrower than the physical bus width. The DMA Controller packs or unpacks data as appropriate.

4.1.6.2 Endian behavior

The DMA Controller can cope with both little-endian and big-endian addressing.

Internally the DMA Controller treats all data as a stream of bytes instead of 16-bit or 32-bit quantities. This means that when performing mixed-endian activity, where the endianness of the source and destination are different, byte swapping of the data within the 32-bit data bus is observed.

Note: If byte swapping is not required, then use of different endianness between the source and destination addresses must be avoided. Table 31–543 shows endian behavior for different source and destination combinations.

© NXP B.V. 2010. All rights reserved.

Table 543. Endian behavior

Source endian	Destination endian	Source width	Destination width	Source transfer no/byte lane	Source data	Destination transfer no/byte lane	Destination data
Little	Little	8	8	1/[7:0]	21	1/[7:0]	21212121
				2/[15:8]	43	2/[15:8]	43434343
				3/[23:16]	65	3/[23:16]	65656565
				4/[31:24]	87	4/[31:24]	87878787
Little	Little	8	16	1/[7:0]	21	1/[15:0]	43214321
				2/[15:8]	43	2/[31:16]	87658765
				3/[23:16]	65		
				4/[31:24]	87		
Little	Little	8	32	1/[7:0]	21	1/[31:0]	87654321
				2/[15:8]	43		
				3/[23:16]	65		
				4/[31:24]	87		
Little	Little	16	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				2/[23:16]	65	3/[23:16]	65656565
				2/[31:24]	87	4/[31:24]	87878787
Little	Little	16	16	1/[7:0]	21	1/[15:0]	43214321
				1/[15:8]	43	2/[31:16]	87658765
				2/[23:16]	65		
				2/[31:24]	87		
Little	Little	16	32	1/[7:0]	21	1/[31:0]	87654321
				1/[15:8]	43		
				2/[23:16]	65		
				2/[31:24]	87		
Little	Little	32	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				1/[23:16]	65	3/[23:16]	65656565
				1/[31:24]	87	4/[31:24]	87878787
Little	Little	32	16	1/[7:0]	21	1/[15:0]	43214321
				1/[15:8]	43	2/[31:16]	87658765
				1/[23:16]	65		
				1/[31:24]	87		
Little	Little	32	32	1/[7:0]	21	1/[31:0]	87654321
				1/[15:8]	43		
				1/[23:16]	65		
				1/[31:24]	87		
Big	Big	8	8	1/[31:24]	12	1/[31:24]	12121212
				2/[23:16]	34	2/[23:16]	34343434
				3/[15:8]	56	3/[15:8]	56565656
				4/[7:0]	78	4/[7:0]	78787878

© NXP B.V. 2010. All rights reserved.

Table 543. Endian behavior *…continued*

Source endian	Destination endian	Source width	Destination width	Source transfer no/byte lane	Source data	Destination transfer no/byte lane	Destination data
Big	Big	8	16	1/[31:24]	12	1/[15:0]	12341234
				2/[23:16]	34	2/[31:16]	56785678
				3/[15:8]	56		
				4/[7:0]	78		
Big	Big	8	32	1/[31:24]	12	1/[31:0]	12345678
				2/[23:16]	34		
				3/[15:8]	56		
				4/[7:0]	78		
Big	Big	16	8	1/[31:24]	12	1/[31:24]	12121212
				1/[23:16]	34	2/[23:16]	34343434
				2/[15:8]	56	3/[15:8]	56565656
				2/[7:0]	78	4/[7:0]	78787878
Big	Big	16	16	1/[31:24]	12	1/[15:0]	12341234
				1/[23:16]	34	2/[31:16]	56785678
				2/[15:8]	56		
				2/[7:0]	78		
Big	Big	16	32	1/[31:24]	12	1/[31:0]	12345678
				1/[23:16]	34		
				2/[15:8]	56		
				2/[7:0]	78		
Big	Big	32	8	1/[31:24]	12	1/[31:24]	12121212
				1/[23:16]	34	2/[23:16]	34343434
				1/[15:8]	56	3/[15:8]	56565656
				1/[7:0]	78	4/[7:0]	78787878
Big	Big	32	16	1/[31:24]	12	1/[15:0]	12341234
				1/[23:16]	34	2/[31:16]	56785678
				1/[15:8]	56		
				1/[7:0]	78		
Big	Big	32	32	1/[31:24]	12	1/[31:0]	12345678
				1/[23:16]	34		
				1/[15:8]	56		
				1/[7:0]	78		

4.1.6.3 Error conditions

An error during a DMA transfer is flagged directly by the peripheral by asserting an Error response on the AHB bus during the transfer. The DMA Controller automatically disables the DMA stream after the current transfer has completed, and can optionally generate an error interrupt to the CPU. This error interrupt can be masked.

© NXP B.V. 2010. All rights reserved.

4.1.7 Channel hardware

Each stream is supported by a dedicated hardware channel, including source and destination controllers, as well as a FIFO. This enables better latency than a DMA controller with only a single hardware channel shared between several DMA streams and simplifies the control logic.

4.1.8 DMA request priority

DMA channel priority is fixed. DMA channel 0 has the highest priority and DMA channel 7 has the lowest priority.

If the DMA Controller is transferring data for the lower priority channel and then the higher priority channel goes active, it completes the number of transfers delegated to the master interface by the lower priority channel before switching over to transfer data for the higher priority channel. Transfers delegated to the master interface are staged in the DMA channel FIFO, so the amount of data that needs to transfer could be as large as a 4 words.

It is recommended that memory-to-memory transactions use the lowest priority channel.

4.1.9 Interrupt generation

A combined interrupt output is generated as an OR function of the individual interrupt requests of the DMA Controller and is connected to the interrupt controller.

4.2 DMA system connections

4.2.1 DMA request signals

The DMA request signals are used by peripherals to request a data transfer. The DMA request signals indicate whether a single or burst transfer of data is required. The DMA available request signals are:

DMACBREQ[15:0] — Burst request signals. These cause a programmed burst number of data to be transferred.

DMACSREQ[15:0] — Single transfer request signals. These cause a single data to be transferred. The DMA controller transfers a single transfer to or from the peripheral.

DMACLBREQ[15:0] — Last burst request signals.

DMACLSREQ[15:0] — Last single transfer request signals.

Note that peripherals on this device do not support “last” request types, and many do not support both single and burst request types. See Section 31–4.2.3.

4.2.2 DMA response signals

The DMA response signals indicate whether the transfer initiated by the DMA request signal has completed. The response signals can also be used to indicate whether a complete packet has been transferred. The DMA response signals from the DMA controller are:

DMACCLR[15:0] — DMA clear or acknowledge signals. The DMACCLR signal is used by the DMA controller to acknowledge a DMA request from the peripheral.

© NXP B.V. 2010. All rights reserved.

DMACTC[15:0] — DMA terminal count signals. The DMACTC signal can be used by the DMA controller to indicate to the peripheral that the DMA transfer is complete.

4.2.3 DMA request connections

The connection of the GPDMA to the supported peripheral devices depends on the DMA functions implemented in those peripherals. Table 31–544 shows the DMA Request numbers used by the supported peripherals. UART and timer DMA requests on channels 8 through 15 are chosen via the DMAREQSEL register, see Section 31–5.15.

Table 544. DMA Connections

Peripheral Function	DMA Single Request Input (DMACSREQ)	DMA Burst Request Input (DMACBREQ)	DMA Request Signal
SSP0 Tx	0	0	Dedicated DMA requests
SSP0 Rx	1	1	Dedicated DMA requests
SSP1 Tx	2	2	Dedicated DMA requests
SSP1 Rx	3	3	Dedicated DMA requests
ADC	4	4	ADC interrupt request [1]
I^2S channel 0	-	5	Dedicated DMA request
I^2S channel 1	-	6	Dedicated DMA request
DAC	-	7	Dedicated DMA request
UART0 Tx / MAT0.0	-	8	Dedicated DMA requests
UART0 Rx / MAT0.1	-	9	Dedicated DMA requests
UART1 Tx / MAT1.0	-	10	Dedicated DMA requests
UART1 Rx / MAT1.1	-	11	Dedicated DMA requests
UART2 Tx / MAT2.0	-	12	Dedicated DMA requests
UART2 Rx / MAT2.1	-	13	Dedicated DMA requests
UART3 Tx / MAT3.0	-	14	Dedicated DMA requests
UART3 Rx / MAT3.1	-	15	Dedicated DMA requests

[1] Generates an interrupt and/or DMA request depending on software setup.

© NXP B.V. 2010. All rights reserved.

5. Register description

The DMA Controller supports 8 channels. Each channel has registers specific to the operation of that channel. Other registers controls aspects of how source peripherals relate to the DMA Controller. There are also global DMA control and status registers.

The DMA Controller registers are shown in Table 31–545.

Table 545. GPDMA register map

Name	Description	Access	Reset state	Address
General registers				
DMACIntStat	DMA Interrupt Status Register	RO	0	0x5000 4000
DMACIntTCStat	DMA Interrupt Terminal Count Request Status Register	RO	0	0x5000 4004
DMACIntTCClear	DMA Interrupt Terminal Count Request Clear Register	WO	-	0x5000 4008
DMACIntErrStat	DMA Interrupt Error Status Register	RO	0	0x5000 400C
DMACIntErrClr	DMA Interrupt Error Clear Register	WO	-	0x5000 4010
DMACRawIntTCStat	DMA Raw Interrupt Terminal Count Status Register	RO	0	0x5000 4014
DMACRawIntErrStat	DMA Raw Error Interrupt Status Register	RO	0	0x5000 4018
DMACEnbldChns	DMA Enabled Channel Register	RO	0	0x5000 401C
DMACSoftBReq	DMA Software Burst Request Register	R/W	0	0x5000 4020
DMACSoftSReq	DMA Software Single Request Register	R/W	0	0x5000 4024
DMACSoftLBReq	DMA Software Last Burst Request Register	R/W	0	0x5000 4028
DMACSoftLSReq	DMA Software Last Single Request Register	R/W	0	0x5000 402C
DMACConfig	DMA Configuration Register	R/W	0	0x5000 4030
DMACSync	DMA Synchronization Register	R/W	0	0x5000 4034
DMAREQSEL	Selects between UART and timer DMA requests on channels 8 through 15	R/W	0	0x400F C1C4
Channel 0 registers				
DMACC0SrcAddr	DMA Channel 0 Source Address Register	R/W	0	0x5000 4100
DMACC0DestAddr	DMA Channel 0 Destination Address Register	R/W	0	0x5000 4104
DMACC0LLI	DMA Channel 0 Linked List Item Register	R/W	0	0x5000 4108
DMACC0Control	DMA Channel 0 Control Register	R/W	0	0x5000 410C
DMACC0Config	DMA Channel 0 Configuration Register	R/W	0 [1]	0x5000 4110
Channel 1 registers				
DMACC1SrcAddr	DMA Channel 1 Source Address Register	R/W	0	0x5000 4120
DMACC1DestAddr	DMA Channel 1 Destination Address Register	R/W	0	0x5000 4124
DMACC1LLI	DMA Channel 1 Linked List Item Register	R/W	0	0x5000 4128
DMACC1Control	DMA Channel 1 Control Register	R/W	0	0x5000 412C
DMACC1Config	DMA Channel 1 Configuration Register	R/W	0 [1]	0x5000 4130
Channel 2 registers				
DMACC2SrcAddr	DMA Channel 2 Source Address Register	R/W	0	0x5000 4140
DMACC2DestAddr	DMA Channel 2 Destination Address Register	R/W	0	0x5000 4144
DMACC2LLI	DMA Channel 2 Linked List Item Register	R/W	0	0x5000 4148
DMACC2Control	DMA Channel 2 Control Register	R/W	0	0x5000 414C

© NXP B.V. 2010. All rights reserved.

Table 545. GPDMA register map

Name	Description	Access	Reset state	Address
DMACC2Config	DMA Channel 2 Configuration Register	R/W	0 [1]	0x5000 4150
Channel 3 registers				
DMACC3SrcAddr	DMA Channel 3 Source Address Register	R/W	0	0x5000 4160
DMACC3DestAddr	DMA Channel 3 Destination Address Register	R/W	0	0x5000 4164
DMACC3LLI	DMA Channel 3 Linked List Item Register	R/W	0	0x5000 4168
DMACC3Control	DMA Channel 3 Control Register	R/W	0	0x5000 416C
DMACC3Config	DMA Channel 3 Configuration Register	R/W	0 [1]	0x5000 4170
Channel 4 registers				
DMACC4SrcAddr	DMA Channel 4 Source Address Register	R/W	0	0x5000 4180
DMACC4DestAddr	DMA Channel 4 Destination Address Register	R/W	0	0x5000 4184
DMACC4LLI	DMA Channel 4 Linked List Item Register	R/W	0	0x5000 4188
DMACC4Control	DMA Channel 4 Control Register	R/W	0	0x5000 418C
DMACC4Config	DMA Channel 4 Configuration Register	R/W	0 [1]	0x5000 4190
Channel 5 registers				
DMACC5SrcAddr	DMA Channel 5 Source Address Register	R/W	0	0x5000 41A0
DMACC5DestAddr	DMA Channel 5 Destination Address Register	R/W	0	0x5000 41A4
DMACC5LLI	DMA Channel 5 Linked List Item Register	R/W	0	0x5000 41A8
DMACC5Control	DMA Channel 5 Control Register	R/W	0	0x5000 41AC
DMACC5Config	DMA Channel 5 Configuration Register	R/W	0 [1]	0x5000 41B0
Channel 6 registers				
DMACC6SrcAddr	DMA Channel 6 Source Address Register	R/W	0	0x5000 41C0
DMACC6DestAddr	DMA Channel 6 Destination Address Register	R/W	0	0x5000 41C4
DMACC6LLI	DMA Channel 6 Linked List Item Register	R/W	0	0x5000 41C8
DMACC6Control	DMA Channel 6 Control Register	R/W	0	0x5000 41CC
DMACC6Config	DMA Channel 6 Configuration Register	R/W	0 [1]	0x5000 41D0
Channel 7 registers				
DMACC7SrcAddr	DMA Channel 7 Source Address Register	R/W	0	0x5000 41E0
DMACC7DestAddr	DMA Channel 7 Destination Address Register	R/W	0	0x5000 41E4
DMACC7LLI	DMA Channel 7 Linked List Item Register	R/W	0	0x5000 41E8
DMACC7Control	DMA Channel 7 Control Register	R/W	0	0x5000 41EC
DMACC7Config	DMA Channel 7 Configuration Register	R/W	0 [1]	0x5000 41F0

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

© NXP B.V. 2010. All rights reserved.

5.1 DMA Interrupt Status register (DMACIntStat - 0x5000 4000)

The DMACIntStat Register is read-only and shows the status of the interrupts after masking. A 1 bit indicates that a specific DMA channel interrupt request is active. The request can be generated from either the error or terminal count interrupt requests. Table 31–546 shows the bit assignments of the DMACIntStat Register.

Table 546. DMA Interrupt Status register (DMACIntStat - 0x5000 4000)

Bit	Name	Function
7:0	IntStat	Status of DMA channel interrupts after masking. Each bit represents one channel: 0 - the corresponding channel has no active interrupt request. 1 - the corresponding channel does have an active interrupt request.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.2 DMA Interrupt Terminal Count Request Status register (DMACIntTCStat - 0x5000 4004)

The DMACIntTCStat Register is read-only and indicates the status of the terminal count after masking. Table 31–547 shows the bit assignments of the DMACIntTCStat Register.

Table 547. DMA Interrupt Terminal Count Request Status register (DMACIntTCStat - 0x5000 4004)

Bit	Name	Function
7:0	IntTCStat	Terminal count interrupt request status for DMA channels. Each bit represents one channel: 0 - the corresponding channel has no active terminal count interrupt request. 1 - the corresponding channel does have an active terminal count interrupt request.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.3 DMA Interrupt Terminal Count Request Clear register (DMACIntTCClear - 0x5000 4008)

The DMACIntTCClear Register is write-only and clears one or more terminal count interrupt requests. When writing to this register, each data bit that contains a 1 causes the corresponding bit in the status register (DMACIntTCStat) to be cleared. Data bits that are 0 have no effect. Table 31–548 shows the bit assignments of the DMACIntTCClear Register.

Table 548. DMA Interrupt Terminal Count Request Clear register (DMACIntTCClear - 0x5000 4008)

Bit	Name	Function
7:0	IntTCClear	Allows clearing the Terminal count interrupt request (IntTCStat) for DMA channels. Each bit represents one channel: 0 - writing 0 has no effect. 1 - clears the corresponding channel terminal count interrupt.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.4 DMA Interrupt Error Status register (DMACIntErrStat - 0x5000 400C)

The DMACIntErrStat Register is read-only and indicates the status of the error request after masking. Table 31–549 shows the bit assignments of the DMACIntErrStat Register.

© NXP B.V. 2010. All rights reserved.

Table 549. DMA Interrupt Error Status register (DMACIntErrStat - 0x5000 400C)

Bit	Name	Function
7:0	IntErrStat	Interrupt error status for DMA channels. Each bit represents one channel: 0 - the corresponding channel has no active error interrupt request. 1 - the corresponding channel does have an active error interrupt request.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.5 DMA Interrupt Error Clear register (DMACIntErrClr - 0x5000 4010)

The DMACIntErrClr Register is write-only and clears the error interrupt requests. When writing to this register, each data bit that is 1 causes the corresponding bit in the status register to be cleared. Data bits that are 0 have no effect on the corresponding bit in the register. Table 31–550 shows the bit assignments of the DMACIntErrClr Register.

Table 550. DMA Interrupt Error Clear register (DMACIntErrClr - 0x5000 4010)

Bit	Name	Function
7:0	IntErrClr	Writing a 1 clears the error interrupt request (IntErrStat) for DMA channels. Each bit represents one channel: 0 - writing 0 has no effect. 1 - clears the corresponding channel error interrupt.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.6 DMA Raw Interrupt Terminal Count Status register (DMACRawIntTCStat - 0x5000 4014)

The DMACRawIntTCStat Register is read-only and indicates which DMA channel is requesting a transfer complete (terminal count interrupt) prior to masking. (Note: the DMACIntTCStat Register contains the same information after masking.) A 1 bit indicates that the terminal count interrupt request is active prior to masking. Table 31–551 shows the bit assignments of the DMACRawIntTCStat Register.

Table 551. DMA Raw Interrupt Terminal Count Status register (DMACRawIntTCStat - 0x5000 4014)

Bit	Name	Function
7:0	RawIntTCStat	Status of the terminal count interrupt for DMA channels prior to masking. Each bit represents one channel: 0 - the corresponding channel has no active terminal count interrupt request. 1 - the corresponding channel does have an active terminal count interrupt request.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.7 DMA Raw Error Interrupt Status register (DMACRawIntErrStat - 0x5000 4018)

The DMACRawIntErrStat Register is read-only and indicates which DMA channel is requesting an error interrupt prior to masking. (Note: the DMACIntErrStat Register contains the same information after masking.) A 1 bit indicates that the error interrupt request is active prior to masking. Table 31–552 shows the bit assignments of register of the DMACRawIntErrStat Register.

© NXP B.V. 2010. All rights reserved.

Table 552. DMA Raw Error Interrupt Status register (DMACRawIntErrStat - 0x5000 4018)

Bit	Name	Function
7:0	RawIntErrStat	Status of the error interrupt for DMA channels prior to masking. Each bit represents one channel: 0 - the corresponding channel has no active error interrupt request. 1 - the corresponding channel does have an active error interrupt request.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.8 DMA Enabled Channel register (DMACEnbldChns - 0x5000 401C)

The DMACEnbldChns Register is read-only and indicates which DMA channels are enabled, as indicated by the Enable bit in the DMACCxConfig Register. A 1 bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer. Table 31–553 shows the bit assignments of the DMACEnbldChns Register.

Table 553. DMA Enabled Channel register (DMACEnbldChns - 0x5000 401C)

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

5.9 DMA Software Burst Request register (DMACSoftBReq - 0x5000 4020)

The DMACSoftBReq Register is read/write and enables DMA burst requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting DMA burst transfers. A request can be generated from either a peripheral or the software request register. Each bit is cleared when the related transaction has completed. Table 31–554 shows the bit assignments of the DMACSoftBReq Register.

Table 554. DMA Software Burst Request register (DMACSoftBReq - 0x5000 4020)

Bit	Name	Function
15:0	SoftBReq	Software burst request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function (refer to Table 31–544 for peripheral hardware connections to the DMA controller): 0 - writing 0 has no effect. 1 - writing 1 generates a DMA burst request for the corresponding request line.
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

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

© NXP B.V. 2010. All rights reserved.

5.10 DMA Software Single Request register (DMACSoftSReq - 0x5000 4024)

The DMACSoftSReq Register is read/write and enables DMA single transfer requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting single DMA transfers. A request can be generated from either a peripheral or the software request register. Table 31–555 shows the bit assignments of the DMACSoftSReq Register.

Table 555. DMA Software Single Request register (DMACSoftSReq - 0x5000 4024)

Bit	Name	Function
15:0	SoftSReq	Software single transfer request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function: 0 - writing 0 has no effect. 1 - writing 1 generates a DMA single transfer request for the corresponding request line.
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.11 DMA Software Last Burst Request register (DMACSoftLBReq - 0x5000 4028)

The DMACSoftLBReq Register is read/write and enables DMA last burst requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting last burst DMA transfers. A request can be generated from either a peripheral or the software request register. Table 31–556 shows the bit assignments of the DMACSoftLBReq Register.

Table 556. DMA Software Last Burst Request register (DMACSoftLBReq - 0x5000 4028)

Bit	Name	Function
15:0	SoftLBReq	Software last burst request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function: 0 - writing 0 has no effect. 1 - writing 1 generates a DMA last burst request for the corresponding request line.
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.12 DMA Software Last Single Request register (DMACSoftLSReq - 0x5000 402C)

The DMACSoftLSReq Register is read/write and enables DMA last single requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Reading the register indicates which sources are requesting last single DMA transfers. A request can be generated from either a peripheral or the software request register. Table 31–557 shows the bit assignments of the DMACSoftLSReq Register.

© NXP B.V. 2010. All rights reserved.

Table 557. DMA Software Last Single Request register (DMACSoftLSReq - 0x5000 402C)

Bit	Name	Function
15:0	SoftLSReq	Software last single transfer request flags for each of 16 possible sources. Each bit represents one DMA request line or peripheral function: 0 - writing 0 has no effect. 1 - writing 1 generates a DMA last single transfer request for the corresponding request line.
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.13 DMA Configuration register (DMACConfig - 0x5000 4030)

The DMACConfig Register is read/write and configures the operation of the DMA Controller. The endianness of the AHB master interface can be altered by writing to the M bit of this register. The AHB master interface is set to little-endian mode on reset. Table 31–558 shows the bit assignments of the DMACConfig Register.

Table 558. DMA Configuration register (DMACConfig - 0x5000 4030)

Bit	Name	Function
0	E	DMA Controller enable: 0 = disabled (default). Disabling the DMA Controller reduces power consumption. 1 = enabled.
1	M	AHB Master endianness configuration: 0 = little-endian mode (default). 1 = big-endian mode.
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.14 DMA Synchronization register (DMACSync - 0x5000 4034)

The DMACSync Register is read/write and enables or disables synchronization logic for the DMA request signals. The DMA request signals consist of the DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0], and DMACLSREQ[15:0]. A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests. This register is reset to 0, synchronization logic enabled. Table 31–559 shows the bit assignments of the DMACSync Register.

Table 559. DMA Synchronization register (DMACSync - 0x5000 4034)

Bit	Name	Function
15:0	DMACSync	Controls the synchronization logic for DMA request signals. Each bit represents one set of DMA request lines as described in the preceding text: 0 - synchronization logic for the corresponding DMA request signals are disabled. 1 - synchronization logic for the corresponding request line signals are enabled.
31:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.15 DMA Request Select register (DMAReqSel - 0x400F C1C4)

DMAReqSel is a read/write register that allows selecting between UART or Timer DMA requests for DMA inputs 8 through 15. Table 31–560 shows the bit assignments of the DMAReqSel Register.

Table 560. DMA Request Select register (DMAReqSel - 0x400F C1C4)

Bit	Name	Function
0	DMASEL08	Selects the DMA request for GPDMA input 8: 0 - UART0 TX is selected. 1 - Timer 0 match 0 is selected.
1	DMASEL09	Selects the DMA request for GPDMA input 9: 0 - UART0 RX is selected. 1 - Timer 0 match 1 is selected.
2	DMASEL10	Selects the DMA request for GPDMA input 10: 0 - UART1 TX is selected. 1 - Timer 1match 0 is selected.
3	DMASEL11	Selects the DMA request for GPDMA input 11: 0 - UART1 RX is selected. 1 - Timer 1match 1 is selected.
4	DMASEL12	Selects the DMA request for GPDMA input 12: 0 - UART2 TX is selected. 1 - Timer 2 match 0 is selected.
5	DMASEL13	Selects the DMA request for GPDMA input 13: 0 - UART2 RX is selected. 1 - Timer 2 match 1 is selected.
6	DMASEL14	Selects the DMA request for GPDMA input 14: 0 - UART3 TX is selected. 1 - Timer 3 match 0 is selected.
7	DMASEL15	Selects the DMA request for GPDMA input 15: 0 - UART3 RX is selected. 1 - Timer 3 match 1 is selected.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

5.16 DMA Channel registers

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

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

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

© NXP B.V. 2010. All rights reserved.

5.17 DMA Channel Source Address registers (DMACCxSrcAddr - 0x5000 41x0)

The eight read/write DMACCxSrcAddr Registers (DMACC0SrcAddr to DMACC7SrcAddr) contain the current source address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the appropriate channel is enabled. When the DMA channel is enabled this register is updated:

- As the source address is incremented.
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time software has processed the value read, the address may have progressed. It is intended to be read-only when the channel has stopped, in which case it shows the source address of the last item read.

Note: The source and destination addresses must be aligned to the source and destination widths.

Table 31–561 shows the bit assignments of the DMACCxSrcAddr Registers.

Table 561. DMA Channel Source Address registers (DMACCxSrcAddr - 0x5000 41x0)

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

5.18 DMA Channel Destination Address registers (DMACCxDestAddr - 0x5000 41x4)

The eight read/write DMACCxDestAddr Registers (DMACC0DestAddr to DMACC7DestAddr) contain the current destination address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the channel is enabled. When the DMA channel is enabled the register is updated as the destination address is incremented and by following the linked list when a complete packet of data has been transferred. Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the address may have progressed. It is intended to be read-only when a channel has stopped, in which case it shows the destination address of the last item read. Table 31–562 shows the bit assignments of the DMACCxDestAddr Register.

Table 562. DMA Channel Destination Address registers (DMACCxDestAddr - 0x5000 41x4)

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

5.19 DMA Channel Linked List Item registers (DMACCxLLI - 0x5000 41x8)

The eight read/write DMACCxLLI Registers (DMACC0LLI to DMACC7LLI) contain a word-aligned address of the next Linked List Item (LLI). If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled when all DMA transfers associated with it are completed. Programming this register when the DMA channel is enabled may have unpredictable side effects. Table 31–563 shows the bit assignments of the DMACCxLLI Register.

© NXP B.V. 2010. All rights reserved.

Table 563. DMA Channel Linked List Item registers (DMACCxLLI - 0x5000 41x8)

Bit	Name	Function
1:0	-	Reserved, and must be written as 0.
31:2	LLI	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.

5.20 DMA channel control registers (DMACCxControl - 0x5000 41xC)

The eight read/write DMACCxControl Registers (DMACC0Control to DMACC7Control) contain DMA channel control information such as the transfer size, burst size, and transfer width. Each register is programmed directly by software before the DMA channel is enabled. When the channel is enabled the register is updated by following the linked list when a complete packet of data has been transferred. Reading the register while the channel is active does not give useful information. This is because by the time software has processed the value read, the channel may have advanced. It is intended to be read-only when a channel has stopped. Table 31–564 shows the bit assignments of the DMACCxControl Register.

5.20.1 Protection and access information

AHB access information is provided to the source and/or destination peripherals when a transfer occurs, although on the LPC17xx this has no effect. The transfer information is provided by programming the DMA channel (the Prot bits of the DMACCxControl Register, and the Lock bit of the DMACCxConfig Register). These bits are programmed by software, and can be used by peripherals. Three bits of information are provided, and are used as shown in Table 31–564.

© NXP B.V. 2010. All rights reserved.

Table 564. DMA channel control registers (DMACCxControl - 0x5000 41xC)

Bit	Name	Function
11:0	TransferSize	Transfer size. This field sets the size of the transfer. The transfer size value must be set before the channel is enabled. Transfer size is updated as data transfers are completed. A read from this field indicates the number of transfers completed on the destination bus. Reading the register when the channel is active does not give useful information because by the time that the software has processed the value read, the channel might have progressed. It is intended to be used only when a channel is enabled and then disabled.
14:12	SBSize	Source burst size. Indicates the number of transfers that make up a source burst. This value must be set to the burst size of the source peripheral, or if the source is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACBREQ signal goes active in the source peripheral. 000 - 1 001 - 4 010 - 8 011 - 16 100 - 32 101 - 64 110 - 128 111 - 256
17:15	DBSize	Destination burst size. Indicates the number of transfers that make up a destination burst transfer request. This value must be set to the burst size of the destination peripheral or, if the destination is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACBREQ signal goes active in the destination peripheral. 000 - 1 001 - 4 010 - 8 011 - 16 100 - 32 101 - 64 110 - 128 111 - 256
20:18	SWidth	Source transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required. 000 - Byte (8-bit) 001 - Halfword (16-bit) 010 - Word (32-bit) 011 to 111 - Reserved
23:21	DWidth	Destination transfer width. Transfers wider than the AHB master bus width are not supported. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required. 000 - Byte (8-bit) 001 - Halfword (16-bit) 010 - Word (32-bit) 011 to 111 - Reserved
25:24	-	Reserved, and must be written as 0.

© NXP B.V. 2010. All rights reserved.

Table 564. DMA channel control registers (DMACCxControl - 0x5000 41xC) *…continued*

Bit	Name	Function
26	SI	Source increment: 0 - the source address is not incremented after each transfer. 1 - the source address is incremented after each transfer.
27	DI	Destination increment: 0 - the destination address is not incremented after each transfer. 1 - the destination address is incremented after each transfer.
28	Prot1	This is provided to the peripheral during a DMA bus access and indicates that the access is in user mode or privileged mode. This information is not used in the LPC17xx. 0 - access is in user mode. 1 - access is in privileged mode.
29	Prot2	This is provided to the peripheral during a DMA bus access and indicates to the peripheral that the access is bufferable or not bufferable. This information is not used in the LPC17xx. 0 - access is not bufferable. 1 - access is bufferable.
30	Prot3	This is provided to the peripheral during a DMA bus access and indicates to the peripheral that the access is cacheable or not cacheable. This information is not used in the LPC17xx. 0 - access is not cacheable. 1 - access is cacheable.
31	I	Terminal count interrupt enable bit. 0 - the terminal count interrupt is disabled. 1 - the terminal count interrupt is enabled.

5.21 DMA Channel Configuration registers (DMACCxConfig - 0x5000 41x0)

The eight DMACCxConfig Registers (DMACC0Config to DMACC7Config) are read/write with the exception of bit[17] which is read-only. Used these to configure the DMA channel. The registers are not updated when a new LLI is requested. Table 31–565 shows the bit assignments of the DMACCxConfig Register.

© NXP B.V. 2010. All rights reserved.

Table 565. DMA Channel Configuration registers (DMACCxConfig - 0x5000 41x0)

Bit	Name	Function
0	E	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0 = channel disabled. 1 = channel enabled. The Channel Enable bit status can also be found by reading the DMACEnbldChns Register. A channel is enabled by setting this bit. A channel can be disabled by clearing the Enable bit. This causes the current AHB transfer (if one is in progress) to complete and the channel is then disabled. Any data in the FIFO of the relevant channel is lost. Restarting the channel by setting the Channel Enable bit has unpredictable effects, the channel must be fully re-initialized. The channel is also disabled, and Channel Enable bit cleared, when the last LLI is reached, the DMA transfer is completed, or if a channel error is encountered. If a channel must be disabled without losing data in the FIFO, the Halt bit must be set so that further DMA requests are ignored. The Active bit must then be polled until it reaches 0, indicating that there is no data left in the FIFO. Finally, the Channel Enable bit can be cleared.
5:1	SrcPeripheral	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory. See Table 31–544 for peripheral identification.
10:6	DestPeripheral	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory. See Table 31–544 for peripheral identification.
13:11	TransferType	This value indicates the type of transfer. The transfer type can be memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral. Refer to Table 31–566 for the encoding of this field.
14	IE	Interrupt error mask. When cleared, this bit masks out the error interrupt of the relevant channel.
15	ITC	Terminal count interrupt mask. When cleared, this bit masks out the terminal count interrupt of the relevant channel.
16	L	Lock. When set, this bit enables locked transfers. This information is not used in the LPC17xx.
17	A	Active: 0 = there is no data in the FIFO of the channel. 1 = the channel FIFO has data. This value can be used with the Halt and Channel Enable bits to cleanly disable a DMA channel. This is a read-only bit.
18	H	Halt: 0 = enable DMA requests. 1 = ignore further source DMA requests. The contents of the channel FIFO are drained. This value can be used with the Active and Channel Enable bits to cleanly disable a DMA channel.
31:19	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

© NXP B.V. 2010. All rights reserved.

5.21.1 Lock control

The lock control may set the lock bit by writing a 1 to bit 16 of the DMACCxConfig Register. When a burst occurs, the AHB arbiter will not de-grant the master during the burst until the lock is de-asserted. The DMA Controller can be locked for a a single burst such as a long source fetch burst or a long destination drain burst. The DMA Controller does not usually assert the lock continuously for a source fetch burst followed by a destination drain burst.

There are situations when the DMA Controller asserts the lock for source transfers followed by destination transfers. This is possible when internal conditions in the DMA Controller permit it to perform a source fetch followed by a destination drain back-to-back.

5.21.2 Transfer type

Table 31–566 lists the bit values of the transfer type bits identified in Table 31–565.

Table 566. Transfer type bits

Bit value	Transfer type	Controller
000	Memory to memory	DMA
001	Memory to peripheral	DMA
010	Peripheral to memory	DMA
011	Source peripheral to destination peripheral	DMA
100 to 111	Reserved, do not use these combinations	-

© NXP B.V. 2010. All rights reserved.

6. Using the DMA controller

6.1 Programming the DMA controller

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

6.1.1 Enabling the DMA controller

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

6.1.2 Disabling the DMA controller

To disable the DMA controller:

- Read the DMACEnbldChns register and ensure that all the DMA channels have been disabled. If any channels are active, see Disabling a DMA channel.
- Disable the DMA controller by writing 0 to the DMA Enable bit in the DMACConfig register.

6.1.3 Enabling a DMA channel

To enable the DMA channel set the channel enable bit in the relevant DMA channel configuration register. Note that the channel must be fully initialized before it is enabled.

6.1.4 Disabling a DMA channel

A DMA channel can be disabled in three ways:

- By writing directly to the channel enable bit. Any outstanding data in the FIFO's is lost if this method is used.
- By using the active and halt bits in conjunction with the channel enable bit.
- By waiting until the transfer completes. This automatically clears the channel.

Disabling a DMA channel and losing data in the FIFO

Clear the relevant channel enable bit in the relevant channel configuration register. The current AHB transfer (if one is in progress) completes and the channel is disabled. Any data in the FIFO is lost.

Disabling the DMA channel without losing data in the FIFO

- Set the halt bit in the relevant channel configuration register. This causes any future DMA request to be ignored.
- Poll the active bit in the relevant channel configuration register until it reaches 0. This bit indicates whether there is any data in the channel that has to be transferred.
- Clear the channel enable bit in the relevant channel configuration register

6.1.5 Setting up a new DMA transfer

To set up a new DMA transfer:

If the channel is not set aside for the DMA transaction:

© NXP B.V. 2010. All rights reserved.

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

6.1.6 Halting a DMA channel

Set the halt bit in the relevant DMA channel configuration register. The current source request is serviced. Any further source DMA request is ignored until the halt bit is cleared.

6.1.7 Programming a DMA channel

1. Choose a free DMA channel with the priority needed. DMA channel 0 has the highest priority and DMA channel 7 the lowest priority.
2. Clear any pending interrupts on the channel to be used by writing to the DMACIntTCClear and DMACIntErrClear register. The previous channel operation might have left interrupt active.
3. Write the source address into the DMACCxSrcAddr register.
4. Write the destination address into the DMACCxDestAddr register.
5. Write the address of the next LLI into the DMACCxLLI register. If the transfer comprises of a single packet of data then 0 must be written into this register.
6. Write the control information into the DMACCxControl register.
7. Write the channel configuration information into the DMACCxConfig register. If the enable bit is set then the DMA channel is automatically enabled.

6.2 Flow control

The device that controls the length of the packet is known as the flow controller. On the LPC17xx, the flow controller is always the DMA Controller, and the packet length is programmed by software before the DMA channel is enabled.

When the DMA transfer is completed:

1. The DMA Controller issues an acknowledge to the peripheral in order to indicate that the transfer has finished.
2. A TC interrupt is generated, if enabled.
3. The DMA Controller moves on to the next LLI.

The following sections describe the DMA Controller data flow sequences for the four allowed transfer types:

- Memory-to-peripheral.
- Peripheral-to-memory.
- Memory-to-memory.
- Peripheral-to-peripheral.

Table 31–567 indicates the request signals used for each type of transfer.

© NXP B.V. 2010. All rights reserved.

Table 567. DMA request signal usage

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

6.2.1 Peripheral-to-memory or memory-to-peripheral DMA flow

For a peripheral-to-memory or memory-to-peripheral DMA flow, the following sequence occurs:

1. Program and enable the DMA channel.
2. Wait for a DMA request.
3. The DMA Controller starts transferring data when:
   - The DMA request goes active.
   - The DMA stream has the highest pending priority.
   - The DMA Controller is the bus master of the AHB bus.
4. If an error occurs while transferring the data, an error interrupt is generated and disables the DMA stream, and the flow sequence ends.
5. Decrement the transfer count.
6. If the transfer has completed (indicated by the transfer count reaching 0):
   - The DMA Controller responds with a DMA acknowledge.
   - The terminal count interrupt is generated (this interrupt can be masked).
   - If the DMACCxLLI Register is not 0, then reload the DMACCxSrcAddr, DMACCxDestAddr, DMACCxLLI, and DMACCxControl registers and go to back to step 2. However, if DMACCxLLI is 0, the DMA stream is disabled and the flow sequence ends.

6.2.2 Peripheral-to-peripheral DMA flow

For a peripheral-to-peripheral DMA flow, the following sequence occurs:

1. Program and enable the DMA channel.
2. Wait for a source DMA request.
3. The DMA Controller starts transferring data when:
   - The DMA request goes active.
   - The DMA stream has the highest pending priority.
   - The DMA Controller is the bus master of the AHB bus.
4. If an error occurs while transferring the data an error interrupt is generated, the DMA stream is disabled, and the flow sequence ends.
5. Decrement the transfer count.
6. If the transfer has completed (indicated by the transfer count reaching 0):
   - The DMA Controller responds with a DMA acknowledge to the source peripheral.
   - Further source DMA requests are ignored.

© NXP B.V. 2010. All rights reserved.

7. When the destination DMA request goes active and there is data in the DMA Controller FIFO, transfer data into the destination peripheral.
8. If an error occurs while transferring the data, an error interrupt is generated, the DMA stream is disabled, and the flow sequence ends.
9. If the transfer has completed it is indicated by the transfer count reaching 0. The following happens:
   - The DMA Controller responds with a DMA acknowledge to the destination peripheral.
   - The terminal count interrupt is generated (this interrupt can be masked).
   - If the DMACCxLLI Register is not 0, then reload the DMACCxSrcAddr, DMACCxDestAddr, DMACCxLLI, and DMACCxControl Registers and go to back to step 2. However, if DMACCxLLI is 0, the DMA stream is disabled and the flow sequence ends.

6.2.3 Memory-to-memory DMA flow

For a memory-to-memory DMA flow the following sequence occurs:

1. Program and enable the DMA channel.
2. Transfer data whenever the DMA channel has the highest pending priority and the DMA Controller gains mastership of the AHB bus.
3. If an error occurs while transferring the data, generate an error interrupt and disable the DMA stream.
4. Decrement the transfer count.
5. If the count has reached zero:
   - Generate a terminal count interrupt (the interrupt can be masked).
   - If the DMACCxLLI Register is not 0, then reload the DMACCxSrcAddr, DMACCxDestAddr, DMACCxLLI, and DMACCxControl Registers and go to back to step 2. However, if DMACCxLLI is 0, the DMA stream is disabled and the flow sequence ends.

Note: Memory-to-memory transfers should be programmed with a low channel priority, otherwise other DMA channels cannot access the bus until the memory-to-memory transfer has finished, or other AHB masters cannot perform any transaction.

6.3 Interrupt requests

Interrupt requests can be generated when an AHB error is encountered or at the end of a transfer (terminal count), after all the data corresponding to the current LLI has been transferred to the destination. The interrupts can be masked by programming bits in the relevant DMACCxControl and DMACCxConfig Channel Registers. The interrupt requests from all DMA channels can be found in the DMACRawIntTCStat and DMACRawIntErrStat registers. The masked versions of the DMA interrupt data is contained in the DMACIntTCStat and DMACIntErrStat registers. The DMACIntStat register then combines the DMACIntTCStat and DMACIntErrStat requests into a single register to enable the source of an interrupt to be found quickly. Writing to the DMACIntTCClear or the DMACIntErrClr Registers with a bit set to 1 enables selective clearing of interrupts.

© NXP B.V. 2010. All rights reserved.

6.3.1 Hardware interrupt sequence flow

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

1. Read the DMACIntTCStat Register to determine whether the interrupt was generated due to the end of the transfer (terminal count). A 1 bit indicates that the transfer completed. If more than one request is active, it is recommended that the highest priority channels be checked first.
2. Read the DMACIntErrStat Register to determine whether the interrupt was generated due to an error occurring. A 1 bit indicates that an error occurred.
3. Service the interrupt request.
4. For a terminal count interrupt, write a 1 to the relevant bit of the DMACIntTCClr Register. For an error interrupt write a 1 to the relevant bit of the DMACIntErrClr Register to clear the interrupt request.

6.4 Address generation

Address generation can be either incrementing or non-incrementing (address wrapping is not supported).

Some devices, especially memories, disallow burst accesses across certain address boundaries. The DMA controller assumes that this is the case with any source or destination area, which is configured for incrementing addressing. This boundary is assumed to be aligned with the specified burst size. For example, if the channel is set for 16-transfer burst to a 32-bit wide device then the boundary is 64-bytes aligned (that is address bits [5:0] equal 0). If a DMA burst is to cross one of these boundaries, then, instead of a burst, that transfer is split into separate AHB transactions.

6.4.1 Word-aligned transfers across a boundary

The channel is configured for 16-transfer bursts, each transfer 32-bits wide, to a destination for which address incrementing is enabled. The start address for the current burst is 0x0C000024, the next boundary (calculated from the burst size and transfer width) is 0x0C000040.

The transfer will be split into two AHB transactions:

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

6.5 Scatter/gather

Scatter/gather is supported through the use of linked lists. This means that the source and destination areas do not have to occupy contiguous areas in memory. Where scatter/gather is not required, the DMACCxLLI Register must be set to 0.

The source and destination data areas are defined by a series of linked lists. Each Linked List Item (LLI) controls the transfer of one block of data, and then optionally loads another LLI to continue the DMA operation, or stops the DMA stream. The first LLI is programmed into the DMA Controller.

The data to be transferred described by an LLI (referred to as the packet of data) usually requires one or more DMA bursts (to each of the source and destination).

© NXP B.V. 2010. All rights reserved.

6.5.1 Linked list items

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

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

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

6.5.1.1 Programming the DMA controller for scatter/gather DMA

To program the DMA Controller for scatter/gather DMA:

1. Write the LLIs for the complete DMA transfer to memory. Each linked list item contains four words:
   - Source address.
   - Destination address.
   - Pointer to next LLI.
   - Control word.

   The last LLI has its linked list word pointer set to 0.
2. Choose a free DMA channel with the priority required. DMA channel 0 has the highest priority and DMA channel 7 the lowest priority.
3. Write the first linked list item, previously written to memory, to the relevant channel in the DMA Controller.
4. Write the channel configuration information to the channel Configuration Register and set the Channel Enable bit. The DMA Controller then transfers the first and then subsequent packets of data as each linked list item is loaded.
5. An interrupt can be generated at the end of each LLI depending on the Terminal Count bit in the DMACCxControl Register. If this bit is set an interrupt is generated at the end of the relevant LLI. The interrupt request must then be serviced and the relevant bit in the DMACIntTCClear Register must be set to clear the interrupt.

6.5.1.2 Example of scatter/gather DMA

See Figure 31–134 for an example of an LLI. A section of memory is to be transferred to a peripheral. The addresses of each LLI entry are given, in hexadecimal, at the left-hand side of the figure. In this example, the LLIs describing the transfer are to be stored contiguously from address 0x2002 0000, but they could be located anywhere. The right side of the figure shows the memory containing the data to be transferred.

© NXP B.V. 2010. All rights reserved.

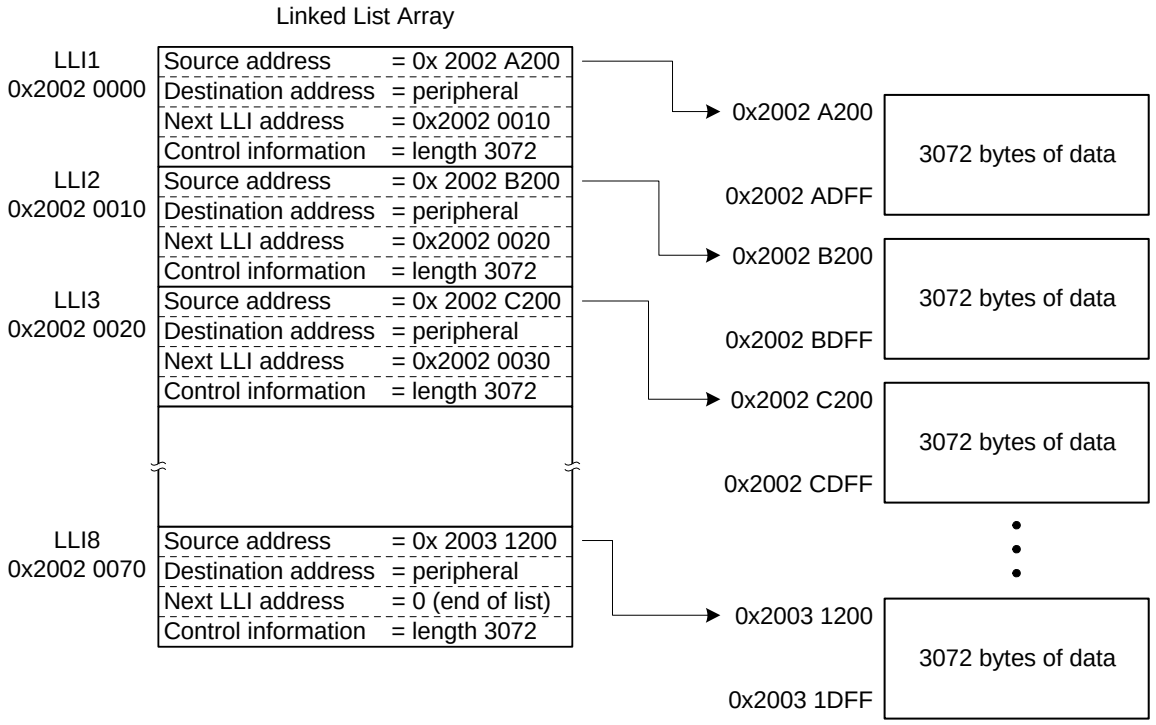


Fig 134. LLI example

The first LLI, stored at 0x2002 0000, defines the first block of data to be transferred, which is the data stored from address 0x2002 A200 to 0x2002 ADFF:

- Source start address 0x2002 A200.
- Destination address set to the destination peripheral address.
- Transfer width, word (32-bit).
- Transfer size, 3072 bytes (0XC00).
- Source and destination burst sizes, 16 transfers.
- Next LLI address, 0x2002 0010.

The second LLI, stored at 0x2002 0010, describes the next block of data to be transferred:

- Source start address 0x2002 B200.
- Destination address set to the destination peripheral address.
- Transfer width, word (32-bit).
- Transfer size, 3072 bytes (0xC00).
- Source and destination burst sizes, 16 transfers.
- Next LLI address, 0x2002 0020.

A chain of descriptors is built up, each one pointing to the next in the series. To initialize the DMA stream, the first LLI, 0x2002 0000, is programmed into the DMA Controller. When the first packet of data has been transferred the next LLI is automatically loaded.

The final LLI is stored at 0x2002 0070 and contains:

© NXP B.V. 2010. All rights reserved.

- Source start address 0x2003 1200.
- Destination address set to the destination peripheral address.
- Transfer width, word (32-bit).
- Transfer size, 3072 bytes (0xC00).
- Source and destination burst sizes, 16 transfers.
- Next LLI address, 0x0.

Because the next LLI address is set to zero, this is the last descriptor, and the DMA channel is disabled after transferring the last item of data. The channel is probably set to generate an interrupt at this point to indicate to the ARM processor that the channel can be reprogrammed.

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 32: LPC17xx Flash memory interface and programming

Rev. 01 — 4 January 2010 User manual

1. Introduction

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

2. Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and UART0 serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.
- Flash signature generation: built-in hardware can generate a signature for a range of flash addresses, or for the entire flash memory.

3. Description

The flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A LOW level after reset at pin P2.10 is considered an external hardware request to start the ISP command handler. Assuming that power supply pins are on their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before P2.10 is sampled and the decision on whether to continue with user code or ISP handler is made. If P2.10 is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P2.10 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

Pin P2.10 is used as a hardware request signal for ISP and therefore requires special attention. Since P2.10 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

When ISP mode is entered after a power on reset, the IRC and PLL are used to generate the CCLK of 14.748 MHz. The baud rates that can easily be obtained in this case are: 9600 baud, 19200 baud, 38400 baud, 57600 baud, 115200 baud, and 230400 baud. This may not be the case when ISP is invoked by the user application (see Section 32–8.9 “Re-invoke ISP” on page 633).

© NXP B.V. 2010. All rights reserved.

A hardware flash signature generation capability is built into the flash memory. this feature can be used to create a signature that can then be used to verify flash contents. Details of flash signature generation are in Section 32–10.

3.1 Memory map after any reset

When a user program begins execution after reset, the interrupt vectors are set to point to the beginning of flash memory.

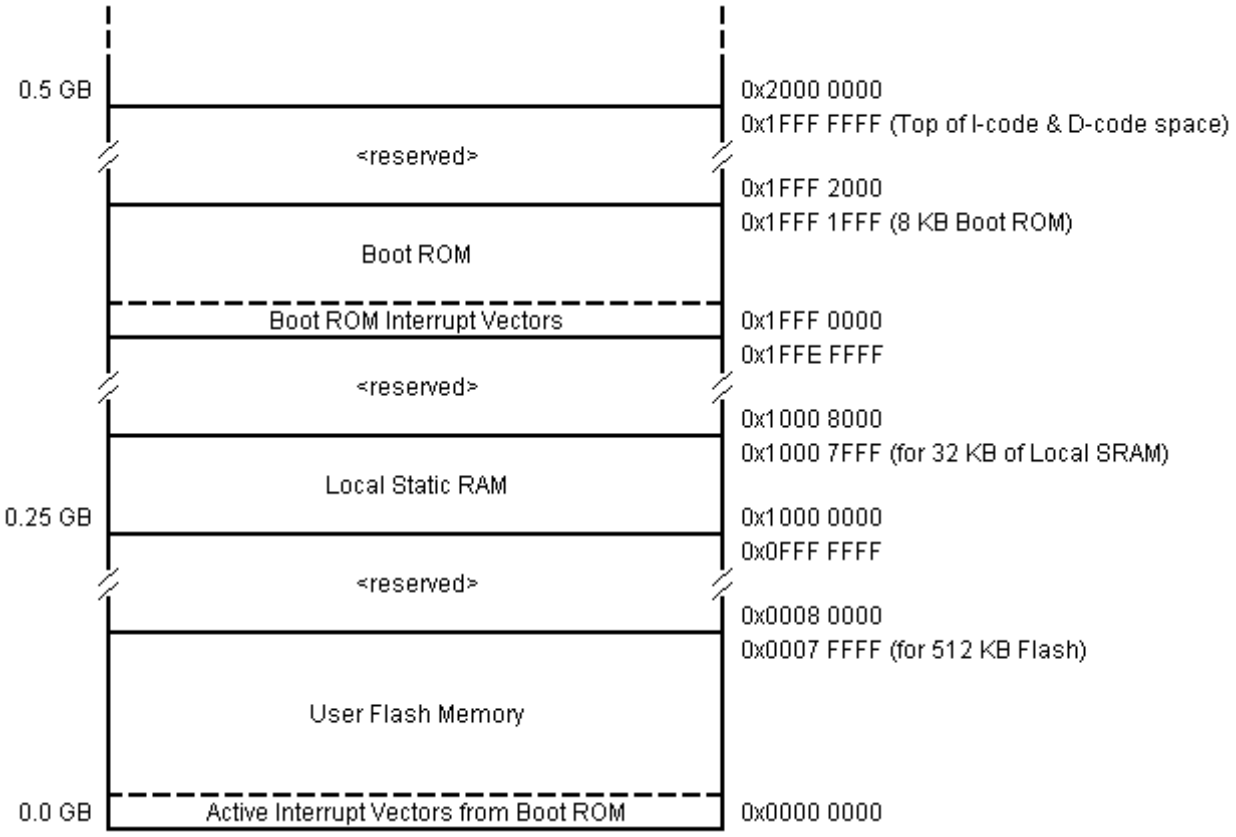


Fig 135. Map of lower memory

3.1.1 Criterion for Valid User Code

The reserved Cortex-M3 exception vector location 7 (offset 0x 001C in the vector table) should contain the 2's complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The boot loader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. The host should send a "?" (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also

© NXP B.V. 2010. All rights reserved.

sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this the host should send the same string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. For example, if the part is running at 10 MHz, the response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the CCLK frequency should be greater than or equal to 10 MHz.

For more details on Reset, PLL and startup/boot code interaction see Section 4–5.1.1 “PLL0 and startup/boot code interaction”.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 32–7 “ISP commands” on page 620.

3.2 Communication protocol

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

3.2.1 ISP command format

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

3.2.2 ISP response format

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

3.2.3 ISP data format

The data stream is in UU-encoded format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

© NXP B.V. 2010. All rights reserved.

3.2.4 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (0x13) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (0x11). The host should also support the same flow control scheme.

3.2.5 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC" (0x1B). This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

3.2.6 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.

3.2.7 RAM used by ISP command handler

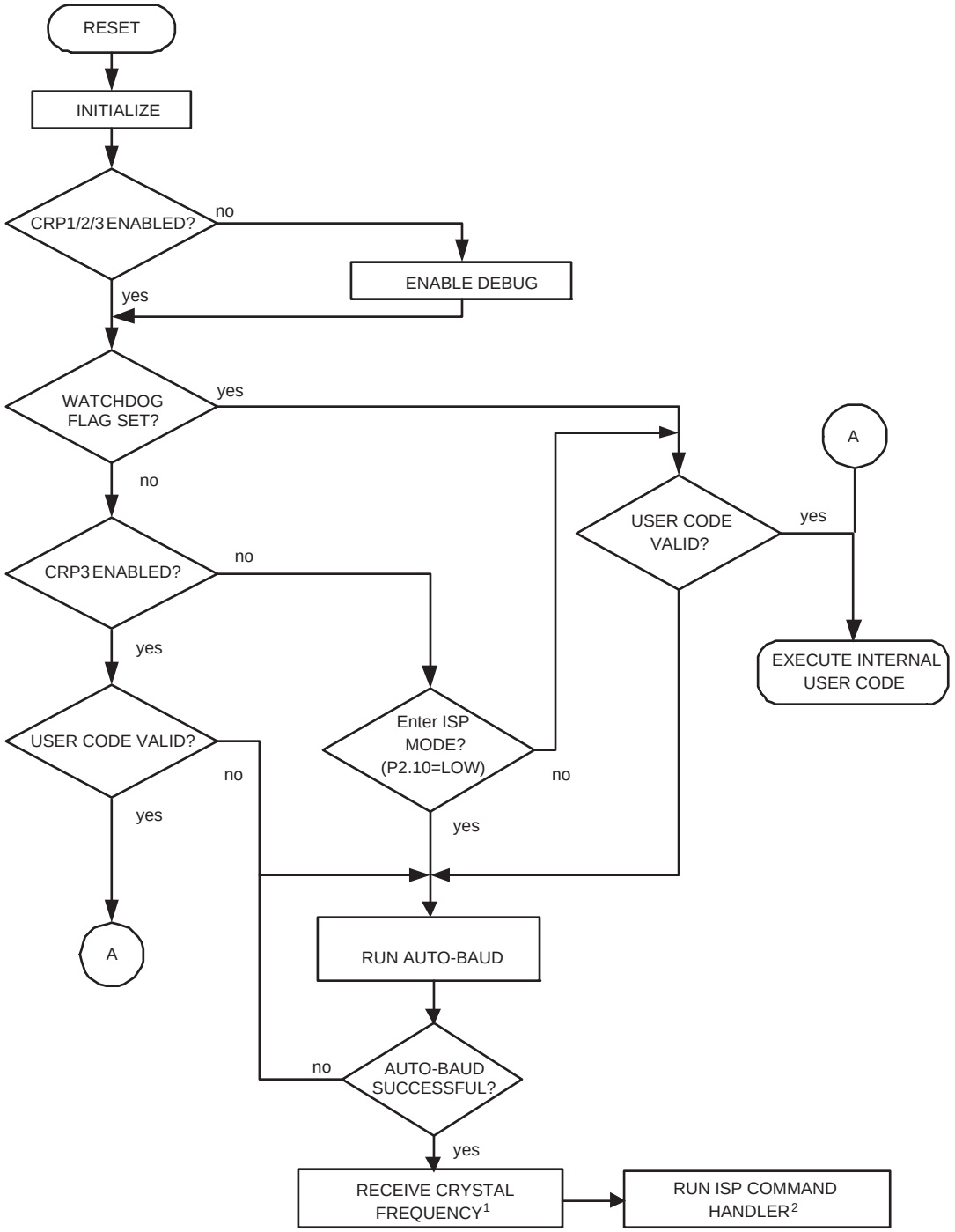
ISP commands use on-chip RAM from 0x1000 0118 to 0x1000 01FF. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 256 bytes and it grows downwards.

3.2.8 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.

© NXP B.V. 2010. All rights reserved.

4. Boot process flowchart



(1) For details on handling the crystal frequency, see Section 32–8.9 “Re-invoke ISP” on page 633

(2) For details on available ISP commands based on the CRP settings see Section 32–6 “Code Read Protection (CRP)”

Fig 136. Boot process flowchart

© NXP B.V. 2010. All rights reserved.

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 LPC17xx devices containing 32, 64, 128, 256 and 512 kB of flash respectively. IAP and ISP routines are located in the Boot ROM.

Table 568. Sectors in a LPC17xx device

Sector Number	Sector Size [kB]	Start Address	End Address	32 kB Part	64 kB Part	128 kB Part	256 kB Part	512 kB Part
0	4	0X0000 0000	0X0000 0FFF	x	x	x	x	x
1	4	0X0000 1000	0X0000 1FFF	x	x	x	x	x
2	4	0X0000 2000	0X0000 2FFF	x	x	x	x	x
3	4	0X0000 3000	0X0000 3FFF	x	x	x	x	x
4	4	0X0000 4000	0X0000 4FFF	x	x	x	x	x
5	4	0X0000 5000	0X0000 5FFF	x	x	x	x	x
6	4	0X0000 6000	0X0000 6FFF	x	x	x	x	x
7	4	0X0000 7000	0X0000 7FFF	x	x	x	x	x
8	4	0x0000 8000	0X0000 8FFF		x	x	x	x
9	4	0x0000 9000	0X0000 9FFF		x	x	x	x
10 (0x0A)	4	0x0000 A000	0X0000 AFFF		x	x	x	x
11 (0x0B)	4	0x0000 B000	0X0000 BFFF		x	x	x	x
12 (0x0C)	4	0x0000 C000	0X0000 CFFF		x	x	x	x
13 (0x0D)	4	0x0000 D000	0X0000 DFFF		x	x	x	x
14 (0X0E)	4	0x0000 E000	0X0000 EFFF		x	x	x	x
15 (0x0F)	4	0x0000 F000	0X0000 FFFF		x	x	x	x
16 (0x10)	32	0x0001 0000	0x0001 7FFF			x	x	x
17 (0x11)	32	0x0001 8000	0x0001 FFFF			x	x	x
18 (0x12)	32	0x0002 0000	0x0002 7FFF				x	x
19 (0x13)	32	0x0002 8000	0x0002 FFFF				x	x
20 (0x14)	32	0x0003 0000	0x0003 7FFF				x	x
21 (0x15)	32	0x0003 8000	0x0003 FFFF				x	x
22 (0x16)	32	0x0004 0000	0x0004 7FFF					x
23 (0x17)	32	0x0004 8000	0x0004 FFFF					x
24 (0x18)	32	0x0005 0000	0x0005 7FFF					x
25 (0x19)	32	0x0005 8000	0x0005 FFFF					x
26 (0x1A)	32	0x0006 0000	0x0006 7FFF					x
27 (0x1B)	32	0x0006 8000	0x0006 FFFF					x
28 (0x1C)	32	0x0007 0000	0x0007 7FFF					x
29 (0x1D)	32	0x0007 8000	0x0007 FFFF					x

© NXP B.V. 2010. All rights reserved.

6. Code Read Protection (CRP)

Code Read Protection is a mechanism that allows user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in flash location at 0x000002FC. IAP commands are not affected by the code read protection.

Important: Any CRP change becomes effective only after the device has gone through a power cycle.

Table 569. Code Read Protection options

Name	Pattern programmed in 0x000002FC	Description
CRP1	0x12345678	Access to chip via the JTAG pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command can not access RAM below 0x10000200. This is due to use of the RAM by the ISP code, see Section 32–3.2.7. • Read Memory command: disabled. • Copy RAM to Flash command: cannot write to Sector 0. • Go command: disabled. • Erase sector(s) command: can erase any individual sector except sector 0 only, or can erase all sectors at once. • Compare command: disabled This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. The compare command is disabled, so in the case of partial flash updates the secondary loader should implement a checksum mechanism to verify the integrity of the flash.
CRP2	0x87654321	This is similar to CRP1 with the following additions: • Write to RAM command: disabled. • Copy RAM to Flash: disabled. • Erase command: only allows erase of all sectors.
CRP3	0x43218765	This is similar to CRP2, but ISP entry by pulling P2.10 LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using the P2.10 pin. It is up to the user's application to provide for flash updates by using IAP calls or by invoking ISP with UART0. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

© NXP B.V. 2010. All rights reserved.

Table 570. Code Read Protection hardware/software interaction

CRP option	User Code Valid	P2.10 pin at reset	JTAG enabled	LPC17xx enters ISP mode	partial flash update in ISP mode
None	No	X	Yes	Yes	Yes
	Yes	High	Yes	No	NA
	Yes	Low	Yes	Yes	Yes
CRP1	No	x	No	Yes	Yes
	Yes	High	No	No	NA
	Yes	Low	No	Yes	Yes
CRP2	No	x	No	Yes	No
	Yes	High	No	No	NA
	Yes	Low	No	Yes	No
CRP3	No	x	No	Yes	No
	Yes	x	No	No	NA

If any CRP mode is enabled and access to the chip is allowed via the ISP, an unsupported or restricted ISP command will be terminated with return code CODE_READ_PROTECTION_ENABLED.

© NXP B.V. 2010. All rights reserved.

7. ISP commands

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

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

Table 571. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 32–572
Set Baud Rate	B <Baud Rate> <stop bit>	Table 32–573
Echo	A <setting>	Table 32–575
Write to RAM	W <start address> <number of bytes>	Table 32–576
Read Memory	R <address> <number of bytes>	Table 32–577
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 32–578
Copy RAM to Flash	C <flash address> <RAM address> <number of bytes>	Table 32–579
Go	G <address> <Mode>	Table 32–580
Erase sector(s)	E <start sector number> <end sector number>	Table 32–581
Blank check sector(s)	I <start sector number> <end sector number>	Table 32–582
Read Part ID	J	Table 32–583
Read Boot Code version	K	Table 32–585
Read serial number	N	Table 32–586
Compare	M <address1> <address2> <number of bytes>	Table 32–587

7.1 Unlock <Unlock code>

Table 572. 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.

© NXP B.V. 2010. All rights reserved.

7.2 Set Baud Rate <Baud Rate> <stop bit>

Table 573. 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 574. Correlation between possible ISP baudrates and CCLK frequency (in MHz)

ISP Baudrate .vs. CCLK Frequency	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456[1]	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

[1] ISP entry after reset uses the on chip IRC and PLL to run the device at CCLK = 14.748 MHz

7.3 Echo <setting>

Table 575. 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.

7.4 Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent.

© NXP B.V. 2010. All rights reserved.

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 576. 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 levels CRP2 or CRP3 are enabled.
Example	"W 268435968 4<CR><LF>" writes 4 bytes of data to address 0x1000 0200.

7.5 Read Memory <address> <no. of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 577. 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 any level of code read protection is enabled.
Example	"R 268435968 4<CR><LF>" reads 4 bytes of data from address 0x1000 0200.

© NXP B.V. 2010. All rights reserved.

7.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 578. ISP Prepare sector(s) for write operation command

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

7.7 Copy RAM to Flash <flash address> <RAM address> <no of bytes>

Table 579. ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. This command is blocked when code read protection levels CRP2 or CRP3 are enabled. When code read protection level CRP1 is enabled, individual sectors other than sector 0 can be written.
Example	"C 0 268468224 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 8000 to the flash address 0.

© NXP B.V. 2010. All rights reserved.

7.8 Go <address> <mode>

Table 580. 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 (retained for backward compatibility):** T (Execute program in Thumb Mode) | A (not allowed).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when any level of code read protection is enabled.
Example	"G 0 T<CR><LF>" branches to address 0x0000 0000.

When the GO command is used, execution begins at the specified address (assuming it is an executable address) with the device left as it was configured for the ISP code. This means that some things are different than they would be for entering user code directly following a chip reset. Most importantly, the main PLL will be running and connected, configured to generate a CPU clock with a frequency of approximately 14.7456 MHz.

7.9 Erase sector(s) <start sector number> <end sector number>

Table 581. ISP Erase sector command

Command	E
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip flash memory. This command is blocked when code read protection level CRP3 is enabled. When code read protection level CRP1 is enabled, individual sectors other than sector 0 can be erased. All sectors can be erased at once in CRP1 and CRP2.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

© NXP B.V. 2010. All rights reserved.

7.10 Blank check sector(s) <sector number> <end sector number>

Table 582. ISP Blank check sector command

Command	I
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR |
Description	This command is used to blank check one or more sectors of on-chip flash memory.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

7.11 Read Part Identification number

Table 583. ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 32–584 "LPC17xx part identification numbers").
Description	This command is used to read the part identification number. The part identification number maps to a feature subset within a device family. This number will not normally change as a result of technical revisions.

Table 584. LPC17xx part identification numbers

Device	ASCII/dec coding	Hex coding
LPC1769	638664503	0x2611 3F37
LPC1768	637615927	0x2601 3F37
LPC1767	637610039	0x2601 2837
LPC1766	637615923	0x2601 3F33
LPC1765	637613875	0x2601 3733
LPC1764	637606178	0x2601 1922
LPC1759	621885239	0x2511 3737
LPC1758	620838711	0x2501 3F37
LPC1756	620828451	0x2501 1723
LPC1754	620828450	0x2501 1722
LPC1752	620761377	0x2500 1121
LPC1751	620761368	0x2500 1118

© NXP B.V. 2010. All rights reserved.

7.12 Read Boot Code version number

Table 585. 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.

7.13 Read device serial number

Table 586. ISP Read device serial number command

Command	N
Input	None.
Return Code	CMD_SUCCESS followed by the device serial number in 4 decimal ASCII groups, each representing a 32-bit value.
Description	This command is used to read the device serial number. The serial number may be used to uniquely identify a single unit among all LPC17xx devices.

7.14 Compare <address1> <address2> <no of bytes>

Table 587. 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. This command is blocked when any level of code read protection is enabled.
Example	"M 8192 268435968 4<CR><LF>" compares 4 bytes from the RAM address 0x1000 0200 to the 4 bytes from the flash address 0x2000.

© NXP B.V. 2010. All rights reserved.

7.15 ISP Return Codes

Table 588. ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken into consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken into consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.
19	CODE_READ_PROTECTION_ENABLED	Code read protection enabled.

© NXP B.V. 2010. All rights reserved.

8. IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. 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 32–137. The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to Flash" command. The maximum number of results is 4, returned by the "Read device serial number" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at location 0x1FFF 1FF0.

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

Define the IAP location entry point. Bit 0 of the IAP location is set since the Cortex-M3 uses only Thumb mode.

```
#define IAP_LOCATION 0x1FFF1FF1
```

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

```
unsigned long command[5];
unsigned long result[5];
```

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.

© NXP B.V. 2010. All rights reserved.

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 589. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50_{10}	Table 32–590
Copy RAM to Flash	51_{10}	Table 32–591
Erase sector(s)	52_{10}	Table 32–592
Blank check sector(s)	53_{10}	Table 32–593
Read part ID	54_{10}	Table 32–594
Read Boot Code version	55_{10}	Table 32–595
Read device serial number	58_{10}	Table 32–596
Compare	56_{10}	Table 32–597
Reinvoke ISP	57_{10}	Table 32–598

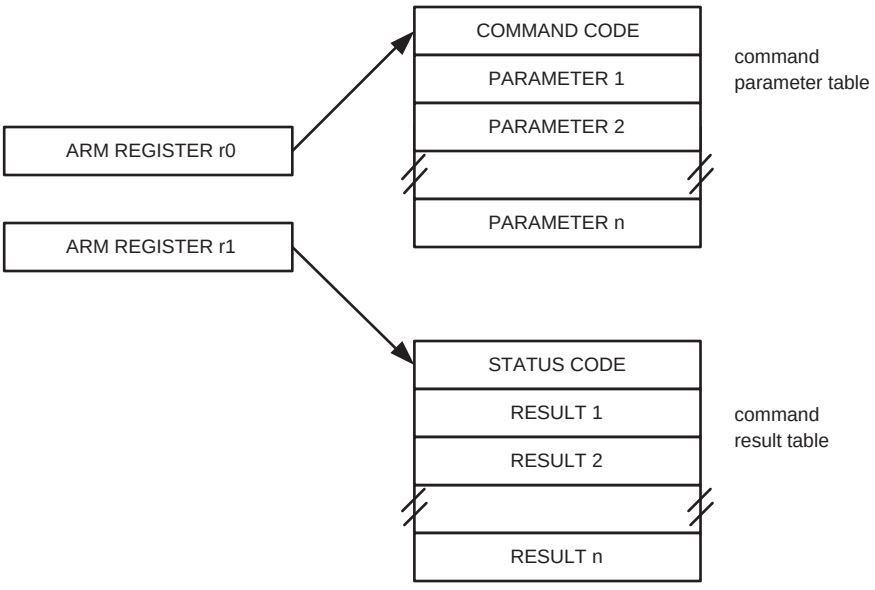


Fig 137. IAP parameter passing

© NXP B.V. 2010. All rights reserved.

8.1 Prepare sector(s) for write operation

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

Table 590. 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. To prepare a single sector use the same "Start" and "End" sector numbers.

8.2 Copy RAM to Flash

Table 591. 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:** CPU Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY |
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed.

© NXP B.V. 2010. All rights reserved.

8.3 Erase Sector(s)

Table 592. 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:** CPU Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip flash memory. To erase a single sector use the same "Start" and "End" sector numbers.

8.4 Blank check sector(s)

Table 593. 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.

8.5 Read part identification number

Table 594. IAP Read part identification number command

Command	Read part identification number
Input	**Command code: 54_{10}** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** Part Identification Number.
Description	This command is used to read the part identification number. The value returned is the hexadecimal version of the part ID. See Table 32–584 “LPC17xx part identification numbers”.

© NXP B.V. 2010. All rights reserved.

8.6 Read Boot Code version number

Table 595. 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.

8.7 Read device serial number

Table 596. IAP Read device serial number command

Command	Read device serial number
Input	**Command code: 58_{10}** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** First 32-bit word of Device Identification Number (at the lowest address) **Result1:** Second 32-bit word of Device Identification Number **Result2:** Third 32-bit word of Device Identification Number **Result3:** Fourth 32-bit word of Device Identification Number
Description	This command is used to read the device identification number. The serial number may be used to uniquely identify a single unit among all LPC17xx devices.

8.8 Compare <address1> <address2> <no of bytes>

Table 597. 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.**

© NXP B.V. 2010. All rights reserved.

8.9 Re-invoke ISP

Table 598. Re-invoke ISP

Command	Compare
Input	**Command code: 57_{10}**
Return Code	None
Result	**None.**
Description	This command is used to invoke the boot loader in ISP mode. It maps boot vectors, sets PCLK = CCLK / 4, configures UART0 pins Rx and Tx, resets TIMER1 and resets the U0FDR (see Section 14–4.12). This command may be used when a valid user program is present in the internal flash memory and the P2.10 pin is not accessible to force the ISP mode. The command does not disable the PLL hence it is possible to invoke the boot loader when the part is running off the PLL. In such case the ISP utility should pass the CCLK (crystal or PLL output depending on the clock source selection Section 4–4.1) frequency after auto-baud handshake. Another option is to disable the PLL and select the IRC as the clock source before making this IAP call. In this case frequency sent by ISP is ignored and IRC and PLL are used to generate CCLK = 14.748 MHz.

8.10 IAP Status Codes

Table 599. IAP Status Codes Summary

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

9. JTAG flash programming interface

Debug tools can write parts of the flash image to the RAM and then execute the IAP call "Copy RAM to Flash" repeatedly with proper offset.

© NXP B.V. 2010. All rights reserved.

10. Flash signature generation

The flash module contains a built-in signature generator. This generator can produce a 128-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (e.g. during programming).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 128-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature generation is complete. Code outside of the flash (e.g. internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

10.1 Register description for signature generation

Table 600. Register overview: FMC (base address 0x2020 0000)

Name	Description	Access	Reset Value	Address	Reference
FMSSTART	Signature start address register	R/W	0	0x4008 4020	Table 32–601
FMSSTOP	Signature stop-address register	R/W	0	0x4008 4024	Table 32–602
FMSW0	128-bit signature Word 0	R	-	0x4008 402C	Table 32–603
FMSW1	128-bit signature Word 1	R	-	0x4008 4030	Table 32–604
FMSW2	128-bit signature Word 2	R	-	0x4008 4034	Table 32–605
FMSW3	128-bit signature Word 3	R	-	0x4008 4038	Table 32–606
FMSTAT	Signature generation status register	R	0	0x4008 4FE0	Section 32–10.1.3
FMSTATCLR	Signature generation status clear register	W	-	0x4008 4FE8	Section 32–10.1.4

© NXP B.V. 2010. All rights reserved.

10.1.1 Signature generation address and control registers

These registers control automatic signature generation. A signature can be generated for any part of the flash memory contents. The address range to be used for generation is defined by writing the start address to the signature start address register (FMSSTART) and the stop address to the signature stop address register (FMSSTOP. The start and stop addresses must be aligned to 128-bit boundaries and can be derived by dividing the byte address by 16.

Signature generation is started by setting the SIG_START bit in the FMSSTOP register. Setting the SIG_START bit is typically combined with the signature stop address in a single write.

Table 32–601 and Table 32–602 show the bit assignments in the FMSSTART and FMSSTOP registers respectively.

Table 601. Flash Module Signature Start register (FMSSTART - 0x4008 4020) bit description

Bit	Symbol	Description	Reset Value
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0

Table 602. Flash Module Signature Stop register (FMSSTOP - 0x4008 4024) bit description

Bit	Symbol	Value	Description	Reset Value
31:18	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
17	SIG_START		Start control bit for signature generation.	0
		0	Signature generation is stopped	
		1	Initiate signature generation	
16:0	STOP		BIST stop address divided by 16 (corresponds to AHB byte address [20:4]).	0

10.1.2 Signature generation result registers

The signature generation result registers return the flash signature produced by the embedded signature generator. The 128-bit signature is reflected by the four registers FMSW0, FMSW1, FMSW2 and FMSW3.

The generated flash signature can be used to verify the flash memory contents. The generated signature can be compared with an expected signature and thus makes saves time and code space. The method for generating the signature is described in Section 32–10.2.

Table 32–606 show bit assignment of the FMSW0 and FMSW1, FMSW2, FMSW3 registers respectively.

Table 603. FMSW0 register bit description (FMSW0, address: 0x2020 002C)

Bit	Symbol	Description	Reset Value
31:0	SW0[31:0]	Word 0 of 128-bit signature (bits 31 to 0).	-

Table 604. FMSW1 register bit description (FMSW1, address: 0x2020 0030)

Bit	Symbol	Description	Reset Value
31:0	SW1[63:32]	Word 1 of 128-bit signature (bits 63 to 32).	-

© NXP B.V. 2010. All rights reserved.

Table 605. FMSW2 register bit description (FMSW2, address: 0x2020 0034)

Bit	Symbol	Description	Reset Value
31:0	SW2[95:64]	Word 2 of 128-bit signature (bits 95 to 64).	-

Table 606. FMSW3 register bit description (FMSW3, address: 0x2020 0038)

Bit	Symbol	Description	Reset Value
31:0	SW3[127:96]	Word 3 of 128-bit signature (bits 127 to 96).	-

10.1.3 Flash Module Status register (FMSTAT - 0x0x4008 4FE0)

The read-only FMSTAT register provides a means of determining when signature generation has completed. Completion of signature generation can be checked by polling the SIG_DONE bit in FMSTAT. SIG_DONE should be cleared via the FMSTATCLR register before starting a signature generation operation, otherwise the status might indicate completion of a previous operation.

Table 607. Flash module Status register (FMSTAT - 0x4008 4FE0) bit description

Bit	Symbol	Description	Reset Value
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	SIG_DONE	When 1, a previously started signature generation has completed. See FMSTATCLR register description for clearing this flag.	0
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.1.4 Flash Module Status Clear register (FMSTATCLR - 0x0x4008 4FE8)

The FMSTATCLR register is used to clear the signature generation completion flag.

Table 608. Flash Module Status Clear register (FMSTATCLR - 0x0x4008 4FE8) bit description

Bit	Symbol	Description	Reset Value
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	SIG_DONE_CLR	Writing a 1 to this bits clears the signature generation completion flag (SIG_DONE) in the FMSTAT register.	0
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

© NXP B.V. 2010. All rights reserved.

10.2 Algorithm and procedure for signature generation

Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

The signature generation is started by writing a '1' to FMSSTOP.MISR_START. Starting the signature generation is typically combined with defining the stop address, which is done in another field FMSSTOP.FMSSTOP of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. Reading of the flash memory for signature generation uses a self-timed read mechanism and does not depend on any configurable timing settings for the flash. A safe estimation for the duration of the signature generation is:

Duration = int((60 / tcy) + 3) x (FMSSTOP - FMSSTART + 1)

When signature generation is triggered via software, the duration is in AHB clock cycles, and tcy is the time in ns for one AHB clock. The SIG_DONE bit in FMSTAT can be polled by software to determine when signature generation is complete.

If signature generation is triggered via JTAG, the duration is in JTAG tck cycles, and tcy is the time in ns for one JTAG clock. Polling the SIG_DONE bit in FMSTAT is not possible in this case.

After signature generation, a 128-bit signature can be read from the FMSW0 to FMSW3 registers. The 128-bit signature reflects the corrected data read from the flash. The 128-bit signature reflects flash parity bits and check bit values.

Content verification

The signature as it is read from the FMSW0 to FMSW3 registers must be equal to the reference signature. The algorithms to derive the reference signature is given in Figure 32–138.

```
sign = 0
FOR address = FMSTART.FMSTART TO FMSTOP.FMSTOP
{
        FOR i = 0 TO 126
                nextSign[i] = f_Q[address][i] XOR sign[i+1]
                nextSign[127] = f_Q[address][127] XOR sign[0] XOR sign[2] XOR
                        sign[27] XOR sign[29]
        sign = nextSign
}
signature128 = sign
```

Fig 138. Algorithm for generating a 128 bit signature

© NXP B.V. 2010. All rights reserved.

UM10360

Chapter 33: LPC17xx JTAG, Serial Wire Debug, and Trace

Rev. 01 — 4 January 2010 User manual

1. Features

- Supports both standard JTAG and ARM Serial Wire Debug modes.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Trace port provides CPU instruction trace capability. Output can be via a 4-bit trace data port, or Serial Wire Viewer.
- Eight Breakpoints. Six instruction breakpoints that can also be used to remap instruction addresses for code patches. Two data comparators that can be used to remap addresses for patches to literal values.
- Four data Watchpoints that can also be used as trace triggers.
- Instrumentation Trace Macrocell allows additional software controlled trace.

2. Introduction

Debug and trace functions are integrated into the ARM Cortex-M3. Serial wire debug and trace functions are supported in addition to a standard JTAG debug and parallel trace functions. The ARM Cortex-M3 is configured to support up to eight breakpoints and four watchpoints.

3. Description

Debugging with the LPC17xx defaults to JTAG. Once in the JTAG debug mode, the debug tool can switch to Serial Wire Debug mode.

Trace can be done using either a 4-bit parallel interface or the Serial Wire Output. When the Serial Wire Output is used, less data can be traced, but it uses no application related pins. Parallel trace has a greater bandwidth, but uses 5 functional pins that may be needed in the application. Note that the trace function available for the Cortex-M3 is functionally very different than the trace that was available for previous ARM7 based devices, using only 5 pins instead of 10.

4. Pin Description

The tables below indicate the various pin functions related to debug and trace. Some of these functions share pins with other functions which therefore may not be used at the same time. Use of the JTAG port excludes use of Serial Wire Debug and Serial Wire Output. Use of the parallel trace requires 5 pins that may be part of the user application, limiting debug possibilities for those features. Trace using the Serial Wire Output does not have this limitation, but has a limited bandwidth.

© NXP B.V. 2010. All rights reserved.

Table 609. JTAG pin description

Pin Name	Type	Description
TCK	Input	**JTAG Test Clock.** This pin is the clock for debug logic when in the JTAG debug mode.
TMS	Input	**JTAG Test Mode Select.** The TMS pin selects the next state in the TAP state machine.
TDI	Input	**JTAG Test Data In.** This is the serial data input for the shift register.
TDO	Output	**JTAG Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal.
$\overline{\text{TRST}}$	Input	**JTAG Test Reset.** The $\overline{\text{TRST}}$ pin can be used to reset the test logic within the debug logic.
RTCK	Output	**JTAG Returned Test Clock.** This is an extra signal added to the JTAG port, and is included for backward pin compatibility with LPC23xx series devices that share the same pinout as this device. RTCK is not normally used with the Cortex-M3. For designs based on ARM7TDMI-S processor core, this signal could be used by external JTAG host interface logic to maintain synchronization with targets having a slow or varying clock frequency. For details refer to "Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)".

Table 610. Serial Wire Debug pin description

Pin Name	Type	Description
SWDCLK	Input	**Serial Wire Clock.** This pin is the clock for debug logic when in the Serial Wire Debug mode.
SWDIO	Input / Output	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the Cortex-M3 CPU.
SWO	Output	**Serial Wire Output.** The SWO pin optionally provides data from the ITM and/or the ETM for an external debug tool to evaluate.

Table 611. Parallel Trace pin description

Pin Name	Type	Description
TRACECLK	Input	**Trace Clock.** This pin provides the sample clock for trace data on the TRACEDATA pins when tracing is enabled by an external debug tool.
TRACEDATA[3:0]	Output	**Trace Data bits 3 to 0.** These pins provide ETM trace data when tracing is enabled by an external debug tool. The debug tool can then interpret the compressed information and make it available to the user.

5. Debug Notes

Important: The user should be aware of certain limitations during debugging. The most important is that, due to limitations of the Cortex-M3 integration, the LPC17xx cannot wake up in the usual manner from Deep Sleep and Power-down modes. It is recommended not to use these modes during debug.

© NXP B.V. 2010. All rights reserved.

Another issue is that debug mode changes the way in which reduced power modes are handled by the Cortex-M3 CPU. This causes power modes at the device level to be different from normal modes operation. These differences mean that power measurements should not be made while debugging, the results will be higher than during normal operation in an application.

During a debugging session, the System Tick Timer and the Repetitive Interrupt Timers are automatically stopped whenever the CPU is stopped. Other peripherals are not affected. If the Repetitive Interrupt Timer is configured such that its PCLK rate is lower than the CPU clock rate, the RIT may not increment predictably during some debug operations, such as single stepping.

Debugging is disabled if code read protection is enabled.

6. Debug memory re-mapping

Following chip reset, a portion of the Boot ROM is mapped to address 0 so that it will be automatically executed. The Boot ROM switches the map to point to Flash memory prior to user code being executed. In this way a user normally does not need to know that this re-mapping occurs.

However, when a debugger halts CPU execution immediately following reset, the Boot ROM is still mapped to address 0 and can cause confusion. Ideally, the debugger should correct the mapping automatically in this case, so that a user does not need to deal with it.

6.1 Memory Mapping Control register (MEMMAP - 0x400F C040)

The MEMMAP register allows switch the mapping of the bottom of memory, including default reset and interrupt vectors, between the Boot ROM and the bottom of on-chip Flash memory.

Table 612. Memory Mapping Control register (MEMMAP - 0x400F C040) bit description

Bit	Symbol	Value	Description	Reset value
0	MAP		Memory map control.	0
		0	Boot mode. A portion of the Boot ROM is mapped to address 0.	
		1	User mode. The on-chip Flash memory is mapped to address 0.	
31:1	-		Reserved. The value read from a reserved bit is not defined.	NA

7. JTAG TAP Identification

The JTAG TAP controller contains device ID that can be used by debugging software to identify the general type of device. More detailed device information is available through ISP/IAP commands (see Section 32–7 and Section 32–8). For the LPC17xx family, this ID value is 0x4BA0 0477.

© NXP B.V. 2010. All rights reserved.

1. ARM Cortex-M3 User Guide: Introduction

The material in this appendix is provided by ARM Limited for inclusion in the User Manuals of devices containing the Cortex-M3 CPU. Minimal changes have been made to reflect implementation options and other distinctions that apply specifically to LPC17xx devices.

1.1 About the processor and core peripherals

The Cortex-M3 processor is a high performance 32-bit processor designed for the microcontroller market. It offers significant benefits to developers, including:

- outstanding processing performance combined with fast interrupt handling
- enhanced system debug with extensive breakpoint and trace capabilities
- efficient processor core, system and memories
- ultra-low power consumption with integrated sleep modes
- platform security, with optional integrated **memory protection unit** (MPU).

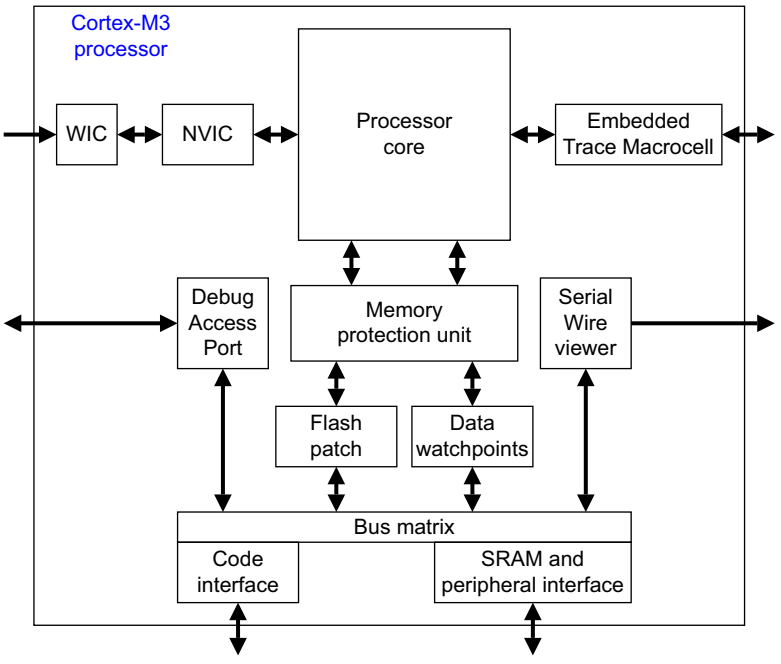


Fig 139. Typical Cortex-M3 implementation

The Cortex-M3 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including single-cycle 32x32 multiplication and dedicated hardware division.

© NXP B.V. 2010. All rights reserved.

To facilitate the design of cost-sensitive devices, the Cortex-M3 processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M3 processor implements a version of the Thumb instruction set, ensuring high code density and reduced program memory requirements. The Cortex-M3 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M3 processor closely integrates a configurable **nested interrupt controlle**r (NVIC), to deliver industry-leading interrupt performance. The NVIC includes a **non-maskable interrupt** (NMI), and provides up to 256 interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of **interrupt service routines** (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and store-multiple operations. Interrupt handlers do not require any assembler stubs, removing any code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the entire device to be rapidly powered down.

LPC17xx devices support additional reduced power modes, see Section 4–8 “Power control” for details.

1.1.1 System level interface

The Cortex-M3 processor provides multiple interfaces using AMBA technology to provide high speed, low latency memory accesses. It supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks and thread-safe Boolean data handling.

The Cortex-M3 processor has an optional **memory protection unit** (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are becoming critical in many embedded applications such as automotive. The MPU is included in LPC17xx devices.

1.1.2 Integrated configurable debug

The Cortex-M3 processor implements a complete hardware debug solution. This provides high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin **Serial Wire Debug** (SWD) port that is ideal for microcontrollers and other small package devices. The MCU vendor determines the debug feature configuration and therefore this can differ across different devices and families.

For system trace the processor integrates an **Instrumentation Trace Macrocell** (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system events these generate, a **Serial Wire Viewer** (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

The optional **Embedded Trace Macrocell** (ETM) delivers unrivalled instruction trace capture in an area far smaller than traditional trace units, enabling many low cost MCUs to implement full instruction trace for the first time.

© NXP B.V. 2010. All rights reserved.

LPC17xx devices support JTAG and Serial Wire Debug, Serial Wire Viewer, and include the Embedded Trace Macrocell. See Section 33–1 for additional information.

1.1.3 Cortex-M3 processor features and benefits summary

- tight integration of system peripherals reduces area and development costs
- Thumb instruction set combines high code density with 32-bit performance
- code-patch ability for ROM system updates
- power control optimization of system components
- integrated sleep modes for low power consumption
- fast code execution permits slower processor clock or increases sleep mode time
- hardware division and fast multiplier
- deterministic, high-performance interrupt handling for time-critical applications
- optional **memory protection unit** (MPU) for safety-critical applications
- extensive debug and trace capabilities:
  - Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing.

1.1.4 Cortex-M3 core peripherals

These are:

- **Nested Vectored Interrupt Controller**

  The **Nested Vectored Interrupt Controller** (NVIC) is an embedded interrupt controller that supports low latency interrupt processing.

- **System control block**

  The **System control block** (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

- **System timer**

  The system timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.

- **Memory protection unit**

  The **Memory protection unit** (MPU) improves system reliability by defining the memory attributes for different memory regions. It provides up to eight different regions, and an optional predefined background region.

© NXP B.V. 2010. All rights reserved.

2. ARM Cortex-M3 User Guide: Instruction Set

2.1 Instruction set summary

The processor implements a version of the Thumb instruction set. Table 34–613 lists the supported instructions.

Note

In Table 34–613:

- angle brackets, <>, enclose alternative forms of the operand
- braces, {}, enclose optional operands
- the Operands column is not exhaustive
- `Op2` is a flexible second operand that can be either a register or a constant
- most instructions can use an optional condition code suffix.

For more information on the instructions and operands, see the instruction descriptions.

Table 613. Cortex-M3 instructions

Mnemonic	Operands	Brief description	Flags	Page
ADC, ADCS	{Rd,} Rn, Op2	Add with Carry	N,Z,C,V	Section 34–2.5.1
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V	Section 34–2.5.1
ADD, ADDW	{Rd,} Rn, #imm12	Add	N,Z,C,V	Section 34–2.5.1
ADR	Rd, label	Load PC-relative address	-	Section 34–2.4.1
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C	Section 34–2.5.2
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic Shift Right	N,Z,C	Section 34–2.5.3
B	label	Branch	-	Section 34–2.9.1
BFC	Rd, #lsb, #width	Bit Field Clear	-	Section 34–2.8.1
BFI	Rd, Rn, #lsb, #width	Bit Field Insert	-	Section 34–2.8.1
BIC, BICS	{Rd,} Rn, Op2	Bit Clear	N,Z,C	Section 34–2.5.2
BKPT	#imm	Breakpoint	-	Section 34–2.10.1
BL	label	Branch with Link	-	Section 34–2.9.1
BLX	Rm	Branch indirect with Link	-	Section 34–2.9.1
BX	Rm	Branch indirect	-	Section 34–2.9.1
CBNZ	Rn, label	Compare and Branch if Non Zero	-	Section 34–2.9.2
CBZ	Rn, label	Compare and Branch if Zero	-	Section 34–2.9.2
CLREX	-	Clear Exclusive	-	Section 34–2.4.9
CLZ	Rd, Rm	Count leading zeros	-	Section 34–2.5.4
CMN, CMNS	Rn, Op2	Compare Negative	N,Z,C,V	Section 34–2.5.5
CMP, CMPS	Rn, Op2	Compare	N,Z,C,V	Section 34–2.5.5
CPSID	iflags	Change Processor State, Disable Interrupts	-	Section 34–2.10.2
CPSIE	iflags	Change Processor State, Enable Interrupts	-	Section 34–2.10.2
DMB	-	Data Memory Barrier	-	Section 34–2.10.3
DSB	-	Data Synchronization Barrier	-	Section 34–2.10.4
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N,Z,C	Section 34–2.5.2

© NXP B.V. 2010. All rights reserved.

Table 613. Cortex-M3 instructions *…continued*

Mnemonic	Operands	Brief description	Flags	Page
ISB	-	Instruction Synchronization Barrier	-	Section 34–2.10.5
IT	-	If-Then condition block	-	Section 34–2.9.3
LDM	Rn{!}, reglist	Load Multiple registers, increment after	-	Section 34–2.4.6
LDMDB, LDMEA	Rn{!}, reglist	Load Multiple registers, decrement before	-	Section 34–2.4.6
LDMFD, LDMIA	Rn{!}, reglist	Load Multiple registers, increment after	-	Section 34–2.4.6
LDR	Rt, [Rn, #offset]	Load Register with word	-	Section 34–2.4
LDRB, LDRBT	Rt, [Rn, #offset]	Load Register with byte	-	Section 34–2.4
LDRD	Rt, Rt2, [Rn, #offset]	Load Register with two bytes	-	Section 34–2.4.2
LDREX	Rt, [Rn, #offset]	Load Register Exclusive	-	Section 34–2.4.8
LDREXB	Rt, [Rn]	Load Register Exclusive with byte	-	Section 34–2.4.8
LDREXH	Rt, [Rn]	Load Register Exclusive with halfword	-	Section 34–2.4.8
LDRH, LDRHT	Rt, [Rn, #offset]	Load Register with halfword	-	Section 34–2.4
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load Register with signed byte	-	Section 34–2.4
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load Register with signed halfword	-	Section 34–2.4
LDRT	Rt, [Rn, #offset]	Load Register with word	-	Section 34–2.4
LSL, LSLS	Rd, Rm, <Rs|#n>	Logical Shift Left	N,Z,C	Section 34–2.5.3
LSR, LSRS	Rd, Rm, <Rs|#n>	Logical Shift Right	N,Z,C	Section 34–2.5.3
MLA	Rd, Rn, Rm, Ra	Multiply with Accumulate, 32-bit result	-	Section 34–2.6.1
MLS	Rd, Rn, Rm, Ra	Multiply and Subtract, 32-bit result	-	Section 34–2.6.1
MOV, MOVS	Rd, Op2	Move	N,Z,C	Section 34–2.5.6
MOVT	Rd, #imm16	Move Top	-	Section 34–2.5.7
MOVW, MOV	Rd, #imm16	Move 16-bit constant	N,Z,C	Section 34–2.5.6
MRS	Rd, spec_reg	Move from special register to general register	-	Section 34–2.10.6
MSR	spec_reg, Rm	Move from general register to special register	N,Z,C,V	Section 34–2.10.7
MUL, MULS	{Rd,} Rn, Rm	Multiply, 32-bit result	N,Z	Section 34–2.6.1
MVN, MVNS	Rd, Op2	Move NOT	N,Z,C	Section 34–2.5.6
NOP	-	No Operation	-	Section 34–2.10.8
ORN, ORNS	{Rd,} Rn, Op2	Logical OR NOT	N,Z,C	Section 34–2.5.2
ORR, ORRS	{Rd,} Rn, Op2	Logical OR	N,Z,C	Section 34–2.5.2
POP	reglist	Pop registers from stack	-	Section 34–2.4.7
PUSH	reglist	Push registers onto stack	-	Section 34–2.4.7
RBIT	Rd, Rn	Reverse Bits	-	Section 34–2.5.8
REV	Rd, Rn	Reverse byte order in a word	-	Section 34–2.5.8
REV16	Rd, Rn	Reverse byte order in each halfword	-	Section 34–2.5.8
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	-	Section 34–2.5.8
ROR, RORS	Rd, Rm, <Rs|#n>	Rotate Right	N,Z,C	Section 34–2.5.3
RRX, RRXS	Rd, Rm	Rotate Right with Extend	N,Z,C	Section 34–2.5.3
RSB, RSBS	{Rd,} Rn, Op2	Reverse Subtract	N,Z,C,V	Section 34–2.5.1
SBC, SBCS	{Rd,} Rn, Op2	Subtract with Carry	N,Z,C,V	Section 34–2.5.1
SBFX	Rd, Rn, #lsb, #width	Signed Bit Field Extract	-	Section 34–2.8.2

© NXP B.V. 2010. All rights reserved.

Table 613. Cortex-M3 instructions *…continued*

Mnemonic	Operands	Brief description	Flags	Page
SDIV	{Rd,} Rn, Rm	Signed Divide	-	Section 34–2.6.3
SEV	-	Send Event	-	Section 34–2.10.9
SMLAL	RdLo, RdHi, Rn, Rm	Signed Multiply with Accumulate (32 x 32 + 64), 64-bit result	-	Section 34–2.6.2
SMULL	RdLo, RdHi, Rn, Rm	Signed Multiply (32 x 32), 64-bit result	-	Section 34–2.6.2
SSAT	Rd, #n, Rm {,shift #s}	Signed Saturate	Q	Section 34–2.7.1
STM	Rn{!}, reglist	Store Multiple registers, increment after	-	Section 34–2.4.6
STMDB, STMEA	Rn{!}, reglist	Store Multiple registers, decrement before	-	Section 34–2.4.6
STMFD, STMIA	Rn{!}, reglist	Store Multiple registers, increment after	-	Section 34–2.4.6
STR	Rt, [Rn, #offset]	Store Register word	-	Section 34–2.4
STRB, STRBT	Rt, [Rn, #offset]	Store Register byte	-	Section 34–2.4
STRD	Rt, Rt2, [Rn, #offset]	Store Register two words	-	Section 34–2.4.2
STREX	Rd, Rt, [Rn, #offset]	Store Register Exclusive	-	Section 34–2.4.8
STREXB	Rd, Rt, [Rn]	Store Register Exclusive byte	-	Section 34–2.4.8
STREXH	Rd, Rt, [Rn]	Store Register Exclusive halfword	-	Section 34–2.4.8
STRH, STRHT	Rt, [Rn, #offset]	Store Register halfword	-	Section 34–2.4
STRT	Rt, [Rn, #offset]	Store Register word	-	Section 34–2.4
SUB, SUBS	{Rd,} Rn, Op2	Subtract	N,Z,C,V	Section 34–2.5.1
SUB, SUBW	{Rd,} Rn, #imm12	Subtract	N,Z,C,V	Section 34–2.5.1
SVC	#imm	Supervisor Call	-	Section 34–2.10.10
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	-	Section 34–2.8.3
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	-	Section 34–2.8.3
TBB	[Rn, Rm]	Table Branch Byte	-	Section 34–2.9.4
TBH	[Rn, Rm, LSL #1]	Table Branch Halfword	-	Section 34–2.9.4
TEQ	Rn, Op2	Test Equivalence	N,Z,C	Section 34–2.5.9
TST	Rn, Op2	Test	N,Z,C	Section 34–2.5.9
UBFX	Rd, Rn, #lsb, #width	Unsigned Bit Field Extract	-	Section 34–2.8.2
UDIV	{Rd,} Rn, Rm	Unsigned Divide	-	Section 34–2.6.3
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned Multiply with Accumulate (32 x 32 + 64), 64-bit result	-	Section 34–2.6.2
UMULL	RdLo, RdHi, Rn, Rm	Unsigned Multiply (32 x 32), 64-bit result	-	Section 34–2.6.2
USAT	Rd, #n, Rm {,shift #s}	Unsigned Saturate	Q	Section 34–2.7.1
UXTB	{Rd,} Rm {,ROR #n}	Zero extend a byte	-	Section 34–2.8.3
UXTH	{Rd,} Rm {,ROR #n}	Zero extend a halfword	-	Section 34–2.8.3
WFE	-	Wait For Event	-	Section 34–2.10.11
WFI	-	Wait For Interrupt	-	Section 34–2.10.12

© NXP B.V. 2010. All rights reserved.

2.2 Intrinsic functions

ANSI cannot directly access some Cortex-M3 instructions. This section describes intrinsic functions that can generate these instructions, provided by the CMIS and that might be provided by a C compiler. If a C compiler does not support an appropriate intrinsic function, you might have to use inline assembler to access some instructions.

The CMSIS provides the following intrinsic functions to generate instructions that ANSI cannot directly access:

Table 614. CMSIS intrinsic functions to generate some Cortex-M3 instructions

Instruction	CMSIS intrinsic function
CPSIE I	void __enable_irq(void)
CPSID I	void __disable_irq(void)
CPSIE F	void __enable_fault_irq(void)
CPSID F	void __disable_fault_irq(void)
ISB	void __ISB(void)
DSB	void __DSB(void)
DMB	void __DMB(void)
REV	uint32_t __REV(uint32_t int value)
REV16	uint32_t __REV16(uint32_t int value)
REVSH	uint32_t __REVSH(uint32_t int value)
RBIT	uint32_t __RBIT(uint32_t int value)
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

The CMSIS also provides a number of functions for accessing the special registers using MRS and MSR instructions:

Table 615. CMSIS intrinsic functions to access the special registers

Special register	Access	CMSIS function
PRIMASK	Read	uint32_t __get_PRIMASK (void)
	Write	void __set_PRIMASK (uint32_t value)
FAULTMASK	Read	uint32_t __get_FAULTMASK (void)
	Write	void __set_FAULTMASK (uint32_t value)
BASEPRI	Read	uint32_t __get_BASEPRI (void)
	Write	void __set_BASEPRI (uint32_t value)
CONTROL	Read	uint32_t __get_CONTROL (void)
	Write	void __set_CONTROL (uint32_t value)
MSP	Read	uint32_t __get_MSP (void)
	Write	void __set_MSP (uint32_t TopOfMainStack)
PSP	Read	uint32_t __get_PSP (void)
	Write	void __set_PSP (uint32_t TopOfProcStack)

2.3 About the instruction descriptions

The following sections give more information about using the instructions:

© NXP B.V. 2010. All rights reserved.

- Section 34–2.3.1 “Operands”
- Section 34–2.3.2 “Restrictions when using PC or SP”
- Section 34–2.3.3 “Flexible second operand”
- Section 34–2.3.4 “Shift Operations”
- Section 34–2.3.5 “Address alignment”
- Section 34–2.3.6 “PC-relative expressions”
- Section 34–2.3.7 “Conditional execution”
- Section 34–2.3.8 “Instruction width selection”.

2.3.1 Operands

An instruction operand can be an ARM register, a constant, or another instruction-specific parameter. Instructions act on the operands and often store the result in a destination register. When there is a destination register in the instruction, it is usually specified before the operands.

Operands in some instructions are flexible in that they can either be a register or a constant. See Section 34–2.3.3.

2.3.2 Restrictions when using PC or SP

Many instructions have restrictions on whether you can use the **Program Counter** (PC) or **Stack Pointer** (SP) for the operands or destination register. See instruction descriptions for more information.

Remark: Bit[0] of any address you write to the PC with a `BX`, `BLX`, `LDM`, `LDR`, or `POP` instruction must be 1 for correct execution, because this bit indicates the required instruction set, and the Cortex-M3 processor only supports Thumb instructions.

2.3.3 Flexible second operand

Many general data processing instructions have a flexible second operand. This is shown as *Operand2* in the descriptions of the syntax of each instruction.

Operand2 can be a:

- Section 34–2.3.3.1 “Constant”
- Section 34–2.3.3.2 “Register with optional shift”

2.3.3.1 Constant

You specify an Operand2 constant in the form:

```
#constant
```

where `constant` can be:

- any constant that can be produced by shifting an 8-bit value left by any number of bits within a 32-bit word
- any constant of the form `0x00XY00XY`
- any constant of the form `0xXY00XY00`
- any constant of the form `0xXYXYXYXY`.

© NXP B.V. 2010. All rights reserved.

Remark: In the constants shown above, `X` and `Y` are hexadecimal digits.

In addition, in a small number of instructions, `constant` can take a wider range of values. These are described in the individual instruction descriptions.

When an Operand2 constant is used with the instructions `MOVS`, `MVNS`, `ANDS`, `ORRS`, `ORNS`, `EORS`, `BICS`, `TEQ` or `TST`, the carry flag is updated to bit[31] of the constant, if the constant is greater than 255 and can be produced by shifting an 8-bit value. These instructions do not affect the carry flag if Operand2 is any other constant.

Instruction substitution: Your assembler might be able to produce an equivalent instruction in cases where you specify a constant that is not permitted. For example, an assembler might assemble the instruction `CMP Rd, #0xFFFFFFFE` as the equivalent instruction `CMN Rd, #0x2`.

2.3.3.2 Register with optional shift

You specify an Operand2 register in the form:

Rm {, *shift*}

where:

Rm is the register holding the data for the second operand.

shift is an optional shift to be applied to *Rm*. It can be one of:

ASR#n: arithmetic shift right *n* bits, $1 \le n \le 32$.

LSL#*n:* logical shift left *n* bits, $1 \le n \le 31$.

LSR#*n*: logical shift right *n* bits, $1 \le n \le 32$.

ROR#*n*: rotate right *n* bits, $1 \le n \le 31$.

RRX: rotate right one bit, with extend.

—: if omitted, no shift occurs, equivalent to LSL#*0*.

If you omit the shift, or specify LSL #0, the instruction uses the value in *Rm*.

If you specify a shift, the shift is applied to the value in *Rm*, and the resulting 32-bit value is used by the instruction. However, the contents in the register *Rm* remains unchanged. Specifying a register with shift also updates the carry flag when used with certain instructions. For information on the shift operations and how they affect the carry flag, see Section 34–2.3.4 “Shift Operations”

2.3.4 Shift Operations

Register shift operations move the bits in a register left or right by a specified number of bits, the **shift length**. Register shift can be performed:

- directly by the instructions `ASR`, `LSR`, `LSL`, `ROR`, and `RRX`, and the result is written to a destination register
- during the calculation of *Operand2* by the instructions that specify the second operand as a register with shift, see Section 34–2.3.3. The result is used by the instruction.

© NXP B.V. 2010. All rights reserved.

The permitted shift lengths depend on the shift type and the instruction, see the individual instruction description or Section 34–2.3.3. If the shift length is 0, no shift occurs. Register shift operations update the carry flag except when the specified shift length is 0. The following sub-sections describe the various shift operations and how they affect the carry flag. In these descriptions, *Rm* is the register containing the value to be shifted, and *n* is the shift length.

2.3.4.1 ASR

Arithmetic shift right by *n* bits moves the left-hand 32-*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32-*n* bits of the result. And it copies the original bit[31] of the register into the left-hand *n* bits of the result. See Figure 34–140.

You can use the ASR #*n* operation to divide the value in the register *Rm* by 2^n, with the result being rounded towards negative-infinity.

When the instruction is ASRS or when ASR #*n* is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit shifted out, bit[*n*-1], of the register *Rm*.

Note

- If *n* is 32 or more, then all the bits in the result are set to the value of bit[31] of *Rm*.
- If *n* is 32 or more and the carry flag is updated, it is updated to the value of bit[31] of *Rm*.

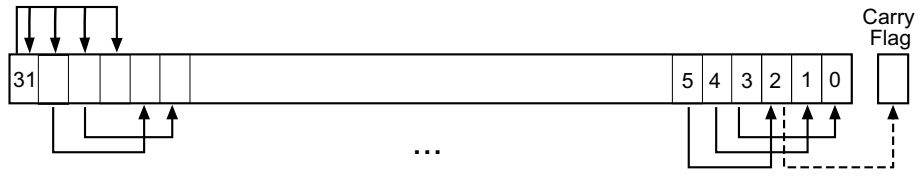


Fig 140. ASR #3

2.3.4.2 LSR

Logical shift right by *n* bits moves the left-hand 32-*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32-*n* bits of the result. And it sets the left-hand *n* bits of the result to 0. See Figure 34–141.

You can use the LSR #*n* operation to divide the value in the register *Rm* by 2^n, if the value is regarded as an unsigned integer.

When the instruction is LSRS or when LSR #*n* is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit shifted out, bit[*n*-1], of the register *Rm*.

Note

- If *n* is 32 or more, then all the bits in the result are cleared to 0.

© NXP B.V. 2010. All rights reserved.

- If *n* is 33 or more and the carry flag is updated, it is updated to 0.

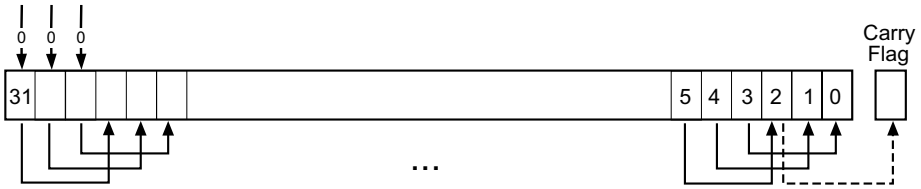


Fig 141. LSR#3

2.3.4.3 LSL

Logical shift left by *n* bits moves the right-hand 32-*n* bits of the register *Rm*, to the left by *n* places, into the left-hand 32-*n* bits of the result. And it sets the right-hand *n* bits of the result to 0. See Figure 34–142.

You can use he LSL #*n* operation to multiply the value in the register *Rm* by 2^n, if the value is regarded as an unsigned integer or a two's complement signed integer. Overflow can occur without warning.

When the instruction is LSLS or when LSL #*n*, with non-zero *n*, is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit shifted out, bit[32-*n*], of the register *Rm*. These instructions do not affect the carry flag when used with *LSL #0*.

Note

- If *n* is 32 or more, then all the bits in the result are cleared to 0.
- If *n* is 33 or more and the carry flag is updated, it is updated to 0.

Fig 142. LSL#3

2.3.4.4 ROR

Rotate right by *n* bits moves the left-hand 32-*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32-*n* bits of the result. And it moves the right-hand *n* bits of the register into the left-hand *n* bits of the result. See Figure 34–143.

When the instruction is RORS or when ROR #*n* is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit rotation, bit[*n*-1], of the register *Rm*.

Note

© NXP B.V. 2010. All rights reserved.

- If *n* is 32, then the value of the result is same as the value in *Rm*, and if the carry flag is updated, it is updated to bit[31] of *Rm*.
- `ROR` with shift length, *n*, more than 32 is the same as `ROR` with shift length *n*-32.

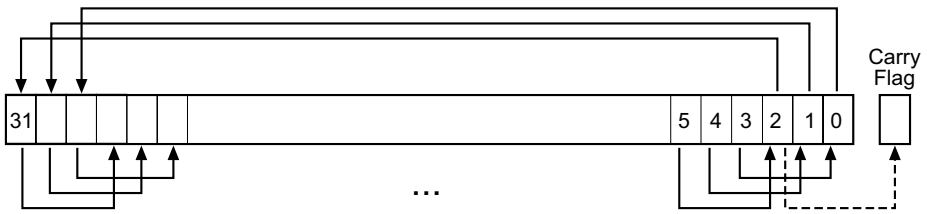


Fig 143. ROR#3

2.3.4.5 RRX

Rotate right with extend moves the bits of the register *Rm* to the right by one bit. And it copies the carry flag into bit[31] of the result. See Figure 34–144.

When the instruction is `RRXS` or when RRX is used in *Operand2* with the instructions `MOVS`, `MVNS`, `ANDS`, `ORRS`, `ORNS`, `EORS`, `BICS`, `TEQ` or `TST`, the carry flag is updated to bit[0] of the register *Rm*.

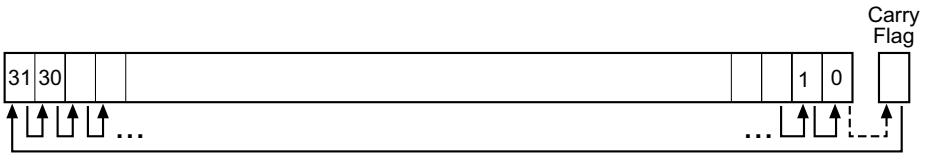


Fig 144. RRX

2.3.5 Address alignment

An aligned access is an operation where a word-aligned address is used for a word, dual word, or multiple word access, or where a halfword-aligned address is used for a halfword access. Byte accesses are always aligned.

The Cortex-M3 processor supports unaligned access only for the following instructions:

- `LDR`, `LDRT`
- `LDRH`, `LDRHT`
- `LDRSH`, `LDRSHT`
- `STR`, `STRT`
- `STRH`, `STRHT`

All other load and store instructions generate a usage fault exception if they perform an unaligned access, and therefore their accesses must be address aligned. For more information about usage faults see Section 34–3.4 “Fault handling”.

© NXP B.V. 2010. All rights reserved.

Unaligned accesses are usually slower than aligned accesses. In addition, some memory regions might not support unaligned accesses. Therefore, ARM recommends that programmers ensure that accesses are aligned. To avoid accidental generation of unaligned accesses, use the UNALIGN_TRP bit in the Configuration and Control Register to trap all unaligned accesses, see Section 34–4.3.8 “Configuration and Control Register”.

2.3.6 PC-relative expressions

A PC-relative expression or **label** is a symbol that represents the address of an instruction or literal data. It is represented in the instruction as the PC value plus or minus a numeric offset. The assembler calculates the required offset from the label and the address of the current instruction. If the offset is too big, the assembler produces an error.

Note

- For `B`, `BL`, `CBNZ`, and `CBZ` instructions, the value of the PC is the address of the current instruction plus 4 bytes.
- For all other instructions that use labels, the value of the PC is the address of the current instruction plus 4 bytes, with bit[1] of the result cleared to 0 to make it word-aligned.
- Your assembler might permit other syntaxes for PC-relative expressions, such as a label plus or minus a number, or an expression of the form `[PC, #number]`.

2.3.7 Conditional execution

Most data processing instructions can optionally update the condition flags in the **Application Program Status Register** (APSR) according to the result of the operation, see Section 34–3.1.3.5 “Program Status Register”. Some instructions update all flags, and some only update a subset. If a flag is not updated, the original value is preserved. See the instruction descriptions for the flags they affect.

You can execute an instruction conditionally, based on the condition flags set in another instruction, either:

- immediately after the instruction that updated the flags
- after any number of intervening instructions that have not updated the flags.

Conditional execution is available by using conditional branches or by adding condition code suffixes to instructions. See Table 34–616 for a list of the suffixes to add to instructions to make them conditional instructions. The condition code suffix enables the processor to test a condition based on the flags. If the condition test of a conditional instruction fails, the instruction:

- does not execute
- does not write any value to its destination register
- does not affect any of the flags
- does not generate any exception.

Conditional instructions, except for conditional branches, must be inside an If-Then instruction block. See Section 34–2.9.3 for more information and restrictions when using the `IT` instruction. Depending on the vendor, the assembler might automatically insert an `IT` instruction if you have conditional instructions outside the IT block.

© NXP B.V. 2010. All rights reserved.

Use the CBZ and CBNZ instructions to compare the value of a register against zero and branch on the result.

This section describes:

- Section 34–2.3.7.1 “The condition flags”
- Section 34–2.3.7.2 “Condition code suffixes”.

2.3.7.1 The condition flags

The APSR contains the following condition flags:

N — Set to 1 when the result of the operation was negative, cleared to 0 otherwise.

Z — Set to 1 when the result of the operation was zero, cleared to 0 otherwise.

C — Set to 1 when the operation resulted in a carry, cleared to 0 otherwise.

V — Set to 1 when the operation caused overflow, cleared to 0 otherwise.

For more information about the APSR see Section 34–3.1.3.5 “Program Status Register”.

A carry occurs:

- if the result of an addition is greater than or equal to 2^{32}
- if the result of a subtraction is positive or zero
- as the result of an inline barrel shifter operation in a move or logical instruction.

Overflow occurs if the result of an add, subtract, or compare is greater than or equal to 2^{31}, or less than -2^{31}.

Remark: Most instructions update the status flags only if the S suffix is specified. See the instruction descriptions for more information.

2.3.7.2 Condition code suffixes

The instructions that can be conditional have an optional condition code, shown in syntax descriptions as {*cond*}. Conditional execution requires a preceding IT instruction. An instruction with a condition code is only executed if the condition code flags in the APSR meet the specified condition. Table 34–616 shows the condition codes to use.

You can use conditional execution with the IT instruction to reduce the number of branch instructions in code.

Table 34–616 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

Table 616. Condition code suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned ≥
CC or LO	C = 0	Lower, unsigned <
MI	N = 1	Negative

© NXP B.V. 2010. All rights reserved.

Table 616. Condition code suffixes

Suffix	Flags	Meaning
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned >
LS	C = 0 or Z = 1	Lower or same, unsigned ≤
GE	N = V	Greater than or equal, signed ≥
LT	N != V	Less than, signed <
GT	Z = 0 and N = V	Greater than, signed >
LE	Z = 1 and N != V	Less than or equal, signed ≤
AL	Can have any value	Always. This is the default when no suffix is specified.

Section 34– shows the use of a conditional instruction to find the absolute value of a number. R0 = ABS(R1).

Section 34– shows the use of conditional instructions to update the value of R4 if the signed values R0 is greater than R1 and R2 is greater than R3.

Example: Absolute value:

```
MOVS    R0, R1          ; R0 = R1, setting flags
IT      MI              ; IT instruction for the negative condition
RSBMI   R0, R1, #0      ; If negative, R0 = -R1
```

Example: Compare and update value:

```
CMP     R0, R1       ; Compare R0 and R1, setting flags
ITT     GT           ; IT instruction for the two GT conditions
CMPGT   R2, R3       ; If 'greater than', compare R2 and R3, setting flags
MOVGT   R4, R5       ; If still 'greater than', do R4 = R5
```

2.3.8 Instruction width selection

There are many instructions that can generate either a 16-bit encoding or a 32-bit encoding depending on the operands and destination register specified. For some of these instructions, you can force a specific instruction size by using an instruction width suffix. The .W suffix forces a 32-bit instruction encoding. The .N suffix forces a 16-bit instruction encoding.

If you specify an instruction width suffix and the assembler cannot generate an instruction encoding of the requested width, it generates an error.

Remark: In some cases it might be necessary to specify the .W suffix, for example if the operand is the label of an instruction or literal data, as in the case of branch instructions. This is because the assembler might not automatically generate the right size encoding.

To use an instruction width suffix, place it immediately after the instruction mnemonic and condition code, if any. Section 34–2.3.8.1 shows instructions with the instruction width suffix.

© NXP B.V. 2010. All rights reserved.

2.3.8.1 Example: Instruction width selection

```
BCS.W  label      ; creates a 32-bit instruction even for a short branch

ADDS.W R0, R0, R1 ; creates a 32-bit instruction even though the same
                  ; operation can be done by a 16-bit instruction
```


© NXP B.V. 2010. All rights reserved.

2.4 Memory access instructions

Table 34–617 shows the memory access instructions:

Table 617. Memory access instructions

Mnemonic	Brief description	See
ADR	Load PC-relative address	Section 34–2.4.1
CLREX	Clear Exclusive	Section 34–2.4.9
LDM{mode}	Load Multiple registers	Section 34–2.4.6
LDR{type}	Load Register using immediate offset	Section 34–2.4.2
LDR{type}	Load Register using register offset	Section 34–2.4.3
LDR{type}T	Load Register with unprivileged access	Section 34–2.4.4
LDR	Load Register using PC-relative address	Section 34–2.4.5
LDREX{type}	Load Register Exclusive	Section 34–2.4.8
POP	Pop registers from stack	Section 34–2.4.7
PUSH	Push registers onto stack	Section 34–2.4.7
STM{mode}	Store Multiple registers	Section 34–2.4.6
STR{type}	Store Register using immediate offset	Section 34–2.4.2
STR{type}	Store Register using register offset	Section 34–2.4.3
STR{type}T	Store Register with unprivileged access	Section 34–2.4.4
STREX{type}	Store Register Exclusive	Section 34–2.4.8

© NXP B.V. 2010. All rights reserved.

2.4.1 ADR

Load PC-relative address.

2.4.1.1 Syntax

`ADR`{*cond*} *Rd*, *label*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

label is a PC-relative expression. See Section 34–2.3.6 “PC-relative expressions”.

2.4.1.2 Operation

`ADR` determines the address by adding an immediate value to the PC, and writes the result to the destination register.

`ADR` produces position-independent code, because the address is PC-relative.

If you use `ADR` to generate a target address for a `BX` or `BLX` instruction, you must ensure that bit[0] of the address you generate is set to1 for correct execution.

Values of *label* must be within the range of –4095 to +4095 from the address in the PC.

Remark: You might have to use the `.W` suffix to get the maximum offset range or to generate addresses that are not word-aligned. See Section 34–2.3.8 “Instruction width selection”.

2.4.1.3 Restrictions

Rd must not be SP and must not be PC.

2.4.1.4 Condition flags

This instruction does not change the flags.

2.4.1.5 Examples

```
ADR     R1, TextMessage   ; Write address value of a location labelled as
                          ; TextMessage to R1
```


© NXP B.V. 2010. All rights reserved.

2.4.2 LDR and STR, immediate offset

Load and Store with immediate offset, pre-indexed immediate offset, or post-indexed immediate offset.

2.4.2.1 Syntax

op{*type*}{*cond*} *Rt*, [*Rn* {, #*offset*}] ; immediate offset

op{*type*}{*cond*} *Rt*, [*Rn*, #*offset*]! ; pre-indexed

op{*type*}{*cond*} *Rt*, [*Rn*], #*offset* ; post-indexed

*op*D{*cond*} *Rt*, *Rt2*, [*Rn* {, #*offset*}] ; immediate offset, two words

*op*D{*cond*} *Rt*, *Rt2*, [*Rn*, #*offset*]! ; pre-indexed, two words

*op*D{*cond*} *Rt*, *Rt2*, [*Rn*], #*offset* ; post-indexed, two words

where:

op is one of:

`LDR`: Load register.

`STR`: Store register.

type is one of:

`B`: unsigned byte, zero extend to 32 bits on loads.

`SB`: signed byte, sign extend to 32 bits (`LDR` only).

`H`: unsigned halfword, zero extend to 32 bits on loads.

`SH`: signed halfword, sign extend to 32 bits (`LDR` only).

—: omit, for word.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rt is the register to load or store.

Rn is the register on which the memory address is based.

offset is an offset from *Rn*. If *offset* is omitted, the address is the contents of *Rn*.

Rt2 is the additional register to load or store for two-word operations.

2.4.2.2 Operation

`LDR` instructions load one or two registers with a value from memory.

`STR` instructions store one or two register values to memory.

Load and store instructions with immediate offset can use the following addressing modes:

- Offset addressing

  The offset value is added to or subtracted from the address obtained from the register *Rn*. The result is used as the address for the memory access. The register *Rn* is unaltered. The assembly language syntax for this mode is:

© NXP B.V. 2010. All rights reserved.

```
[Rn, #offset]
```

- Pre-indexed addressing

  The offset value is added to or subtracted from the address obtained from the register *Rn*. The result is used as the address for the memory access and written back into the register *Rn*. The assembly language syntax for this mode is:

  ```
  [Rn, #offset]!
  ```

- Post-indexed addressing

  The address obtained from the register *Rn* is used as the address for the memory access. The offset value is added to or subtracted from the address, and written back into the register *Rn*. The assembly language syntax for this mode is:

  ```
  [Rn], #offset
  ```

The value to load or store can be a byte, halfword, word, or two words. Bytes and halfwords can either be signed or unsigned. See Section 34–2.3.5 “Address alignment”.

Table 34–618 shows the ranges of offset for immediate, pre-indexed and post-indexed forms.

Table 618. Offset ranges

Instruction type	Immediate offset	Pre-indexed	Post-indexed
Word, halfword, signed halfword, byte, or signed byte	–255 to 4095	–255 to 255	–255 to 255
Two words	multiple of 4 in the range –1020 to 1020	multiple of 4 in the range -1020 to 1020	multiple of 4 in the range –1020 to 1020

2.4.2.3 Restrictions

For load instructions:

- *Rt* can be SP or PC for word loads only
- *Rt* must be different from *Rt2* for two-word loads
- *Rn* must be different from *Rt* and *Rt2* in the pre-indexed or post-indexed forms.

When *Rt* is PC in a word load instruction:

- bit[0] of the loaded value must be 1 for correct execution
- a branch occurs to the address created by changing bit[0] of the loaded value to 0
- if the instruction is conditional, it must be the last instruction in the IT block.

For store instructions:

- *Rt* can be SP for word stores only
- *Rt* must not be PC
- *Rn* must not be PC
- *Rn* must be different from *Rt* and *Rt2* in the pre-indexed or post-indexed forms.

2.4.2.4 Condition flags

These instructions do not change the flags.

© NXP B.V. 2010. All rights reserved.

2.4.2.5 Examples

```
LDR     R8, [R10]             ; Loads R8 from the address in R10.
LDRNE   R2, [R5, #960]!       ; Loads (conditionally) R2 from a word
                              ; 960 bytes above the address in R5, and
                              ; increments R5 by 960.
STR     R2, [R9,#const-struc] ; const-struc is an expression evaluating
                              ; to a constant in the range 0-4095.
STRH    R3, [R4], #4          ; Store R3 as halfword data into address in
                              ; R4, then increment R4 by 4
LDRD    R8, R9, [R3, #0x20]   ; Load R8 from a word 32 bytes above the
                              ; address in R3, and load R9 from a word 36
                              ; bytes above the address in R3
STRD    R0, R1, [R8], #-16    ; Store R0 to address in R8, and store R1 to
                              ; a word 4 bytes above the address in R8,
                              ; and then decrement R8 by 16.
```


© NXP B.V. 2010. All rights reserved.

2.4.3 LDR and STR, register offset

Load and Store with register offset.

2.4.3.1 Syntax

op{*type*}{*cond*} *Rt*, [*Rn*, *Rm* {, LSL #*n*}]

where:

op is one of:

- `LDR`: Load Register.
- `STR`: Store Register.

type is one of:

- `B`: unsigned byte, zero extend to 32 bits on loads.
- `SB`: signed byte, sign extend to 32 bits (`LDR` only).
- `H`: unsigned halfword, zero extend to 32 bits on loads.
- `SH`: signed halfword, sign extend to 32 bits (`LDR` only).
- —: omit, for word.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rt is the register to load or store.

Rn is the register on which the memory address is based.

Rm is a register containing a value to be used as the offset.

`LSL` #*n* is an optional shift, with *n* in the range 0 to 3.

2.4.3.2 Operation

`LDR` instructions load a register with a value from memory.

`STR` instructions store a register value into memory.

The memory address to load from or store to is at an offset from the register *Rn*. The offset is specified by the register *Rm* and can be shifted left by up to 3 bits using `LSL`.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See Section 34–2.3.5 “Address alignment”.

2.4.3.3 Restrictions

In these instructions:

- *Rn* must not be PC
- *Rm* must not be SP and must not be PC
- *Rt* can be SP only for word loads and word stores
- *Rt* can be PC only for word loads.

When *Rt* is PC in a word load instruction:

© NXP B.V. 2010. All rights reserved.

- bit[0] of the loaded value must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

2.4.3.4 Condition flags

These instructions do not change the flags.

2.4.3.5 Examples

```
STR    R0, [R5, R1]         ; Store value of R0 into an address equal to
                            ; sum of R5 and R1
LDRSB  R0, [R5, R1, LSL #1] ; Read byte value from an address equal to
                            ; sum of R5 and two times R1, sign extended it
                            ; to a word value and put it in R0
STR    R0, [R1, R2, LSL #2] ; Stores R0 to an address equal to sum of R1
                            ; and four times R2
```


© NXP B.V. 2010. All rights reserved.

2.4.4 LDR and STR, unprivileged

Load and Store with unprivileged access.

2.4.4.1 Syntax

op{*type*}T{*cond*} *Rt*, [*Rn* {, #*offset*}] ; immediate offset

where:

op is one of:

`LDR`: Load Register.

`STR`: Store Register.

type is one of:

`B`: unsigned byte, zero extend to 32 bits on loads.

`SB`: signed byte, sign extend to 32 bits (`LDR` only).

`H`: unsigned halfword, zero extend to 32 bits on loads.

`SH`: signed halfword, sign extend to 32 bits (`LDR` only).

—: omit, for word.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rt is the register to load or store.

Rn is the register on which the memory address is based.

offset is an offset from *Rn* and can be 0 to 255. If *offset* is omitted, the address is the value in *Rn*.

2.4.4.2 Operation

These load and store instructions perform the same function as the memory access instructions with immediate offset, see Section 34–2.4.2. The difference is that these instructions have only unprivileged access even when used in privileged software.

When used in unprivileged software, these instructions behave in exactly the same way as normal memory access instructions with immediate offset.

2.4.4.3 Restrictions

In these instructions:

- *Rn* must not be PC
- *Rt* must not be SP and must not be PC.

2.4.4.4 Condition flags

These instructions do not change the flags.

© NXP B.V. 2010. All rights reserved.

2.4.4.5 Examples

```
STRBTEQ  R4, [R7]       ; Conditionally store least significant byte in
                        ; R4 to an address in R7, with unprivileged access
LDRHT    R2, [R2, #8]   ; Load halfword value from an address equal to
                        ; sum of R2 and 8 into R2, with unprivileged access
```


© NXP B.V. 2010. All rights reserved.

2.4.5 LDR, PC-relative

Load register from memory.

2.4.5.1 Syntax

LDR{*type*}{*cond*} *Rt*, *label*

LDRD{*cond*} *Rt*, *Rt2*, *label* ; Load two words

type is one of:

- `B`: unsigned byte, zero extend to 32 bits on loads.
- `SB`: signed byte, sign extend to 32 bits (`LDR` only).
- `H`: unsigned halfword, zero extend to 32 bits on loads.
- `SH`: signed halfword, sign extend to 32 bits (`LDR` only).
- —: omit, for word.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rt is the register to load or store.

Rt2 is the second register to load or store.

label is a PC-relative expression. See Section 34–2.3.6 “PC-relative expressions”.

2.4.5.2 Operation

`LDR` loads a register with a value from a PC-relative memory address. The memory address is specified by a label or by an offset from the PC.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See Section 34–2.3.5 “Address alignment”.

label must be within a limited range of the current instruction. Table 34–619 shows the possible offsets between *label* and the PC.

Table 619. Offset ranges

Instruction type	Offset range
Word, halfword, signed halfword, byte, signed byte	–4095 to 4095
Two words	–1020 to 1020

Remark: You might have to use the `.W` suffix to get the maximum offset range. See Section 34–2.3.8 “Instruction width selection”.

2.4.5.3 Restrictions

In these instructions:

- *Rt* can be SP or PC only for word loads
- *Rt2* must not be SP and must not be PC
- *Rt* must be different from *Rt2*.

When *Rt* is PC in a word load instruction:

© NXP B.V. 2010. All rights reserved.

- bit[0] of the loaded value must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

2.4.5.4 Condition flags

These instructions do not change the flags.

2.4.5.5 Examples

```
LDR     R0, LookUpTable   ; Load R0 with a word of data from an address
                          ; labelled as LookUpTable
LDRSB   R7, localdata     ; Load a byte value from an address labelled
                          ; as localdata, sign extend it to a word
                          ; value, and put it in R7
```


© NXP B.V. 2010. All rights reserved.

2.4.6 LDM and STM

Load and Store Multiple registers.

2.4.6.1 Syntax

op{*addr_mode*}{*cond*} *Rn*{!}, *reglist*

where:

op is one of:

- `LDM`: Load Multiple registers.
- `STM`: Store Multiple registers.

addr_mode is any one of the following:

- `IA`: Increment address After each access. This is the default.
- `DB`: Decrement address Before each access.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rn is the register on which the memory addresses are based.

! is an optional writeback suffix. If ! is present the final address, that is loaded from or stored to, is written back into *Rn*.

reglist is a list of one or more registers to be loaded or stored, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range, see Section 34–2.4.6.5.

`LDM` and `LDMFD` are synonyms for `LDMIA`. `LDMFD` refers to its use for popping data from Full Descending stacks.

`LDMEA` is a synonym for `LDMDB`, and refers to its use for popping data from Empty Ascending stacks.

`STM` and `STMEA` are synonyms for `STMIA`. `STMEA` refers to its use for pushing data onto Empty Ascending stacks.

`STMFD` is s synonym for `STMDB`, and refers to its use for pushing data onto Full Descending stacks

2.4.6.2 Operation

`LDM` instructions load the registers in *reglist* with word values from memory addresses based on *Rn*.

`STM` instructions store the word values in the registers in *reglist* to memory addresses based on *Rn*.

For `LDM`, `LDMIA`, `LDMFD`, `STM`, `STMIA`, and `STMEA` the memory addresses used for the accesses are at 4-byte intervals ranging from *Rn* to *Rn* + 4 * (*n*-1), where *n* is the number of registers in *reglist*. The accesses happens in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest number register using the highest memory address. If the writeback suffix is specified, the value of *Rn* + 4 * (*n*-1) is written back to *Rn*.

© NXP B.V. 2010. All rights reserved.

For LDMDB, LDMEA, STMDB, and STMFD the memory addresses used for the accesses are at 4-byte intervals ranging from *Rn* to *Rn* - 4 * (*n*-1), where *n* is the number of registers in *reglist*. The accesses happen in order of decreasing register numbers, with the highest numbered register using the highest memory address and the lowest number register using the lowest memory address. If the writeback suffix is specified, the value of *Rn* - 4 * (*n*-1) is written back to *Rn*.

The PUSH and POP instructions can be expressed in this form. See Section 34–2.4.7 for details.

2.4.6.3 Restrictions

In these instructions:

- *Rn* must not be PC
- *reglist* must not contain SP
- in any STM instruction, *reglist* must not contain PC
- in any LDM instruction, *reglist* must not contain PC if it contains LR
- *reglist* must not contain *Rn* if you specify the writeback suffix.

When PC is in *reglist* in an LDM instruction:

- bit[0] of the value loaded to the PC must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

2.4.6.4 Condition flags

These instructions do not change the flags.

2.4.6.5 Examples

```
LDM     R8,{R0,R2,R9}      ; LDMIA is a synonym for LDM
STMDB   R1!,{R3-R6,R11,R12}
```

2.4.6.6 Incorrect examples

```
STM     R5!,{R5,R4,R9} ; Value stored for R5 is unpredictable
LDM     R2, {}         ; There must be at least one register in the list
```


© NXP B.V. 2010. All rights reserved.

2.4.7 PUSH and POP

Push registers onto, and pop registers off a full-descending stack.

2.4.7.1 Syntax

`PUSH`{*cond*} *reglist*

`POP`{*cond*} *reglist*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

reglist is a non-empty list of registers, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range.

`PUSH` and `POP` are synonyms for `STMDB` and `LDM` (or `LDMIA`) with the memory addresses for the access based on SP, and with the final address for the access written back to the SP. `PUSH` and `POP` are the preferred mnemonics in these cases.

2.4.7.2 Operation

`PUSH` stores registers on the stack in order of decreasing the register numbers, with the highest numbered register using the highest memory address and the lowest numbered register using the lowest memory address.

`POP` loads registers from the stack in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

See Section 34–2.4.6 for more information.

2.4.7.3 Restrictions

In these instructions:

- *reglist* must not contain SP
- for the `PUSH` instruction, *reglist* must not contain PC
- for the `POP` instruction, *reglist* must not contain PC if it contains LR.

When PC is in *reglist* in a `POP` instruction:

- bit[0] of the value loaded to the PC must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

2.4.7.4 Condition flags

These instructions do not change the flags.

2.4.7.5 Examples

```
PUSH    {R0,R4-R7}
PUSH    {R2,LR}
POP     {R0,R10,PC}
```


© NXP B.V. 2010. All rights reserved.

2.4.8 LDREX and STREX

Load and Store Register Exclusive.

2.4.8.1 Syntax

LDREX{*cond*} *Rt*, [*Rn* {, #*offset*}]

STREX{*cond*} *Rd*, *Rt*, [*Rn* {, #*offset*}]

LDREXB{*cond*} *Rt*, [*Rn*]

STREXB{*cond*} *Rd*, *Rt*, [*Rn*]

LDREXH{*cond*} *Rt*, [*Rn*]

STREXH{*cond*} *Rd*, *Rt*, [*Rn*]

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register for the returned status.

Rt is the register to load or store.

Rn is the register on which the memory address is based.

offset is an optional offset applied to the value in *Rn*. If *offset* is omitted, the address is the value in *Rn*.

2.4.8.2 Operation

`LDREX`, `LDREXB`, and `LDREXH` load a word, byte, and halfword respectively from a memory address.

`STREX`, `STREXB`, and `STREXH` attempt to store a word, byte, and halfword respectively to a memory address. The address used in any Store-Exclusive instruction must be the same as the address in the most recently executed Load-exclusive instruction. The value stored by the Store-Exclusive instruction must also have the same data size as the value loaded by the preceding Load-exclusive instruction. This means software must always use a Load-exclusive instruction and a matching Store-Exclusive instruction to perform a synchronization operation, see Section 34–3.2.7 “Synchronization primitives”

If an Store-Exclusive instruction performs the store, it writes 0 to its destination register. If it does not perform the store, it writes 1 to its destination register. If the Store-Exclusive instruction writes 0 to the destination register, it is guaranteed that no other process in the system has accessed the memory location between the Load-exclusive and Store-Exclusive instructions.

For reasons of performance, keep the number of instructions between corresponding Load-Exclusive and Store-Exclusive instruction to a minimum.

Remark: The result of executing a Store-Exclusive instruction to an address that is different from that used in the preceding Load-Exclusive instruction is unpredictable.

2.4.8.3 Restrictions

In these instructions:

© NXP B.V. 2010. All rights reserved.

- do not use PC
- do not use SP for *Rd* and *Rt*
- for `STREX`, *Rd* must be different from both *Rt* and *Rn*
- the value of *offset* must be a multiple of four in the range 0-1020.

2.4.8.4 Condition flags

These instructions do not change the flags.

2.4.8.5 Examples

```
    MOV     R1, #0x1            ; Initialize the 'lock taken' value
try
    LDREX   R0, [LockAddr]      ; Load the lock value
    CMP     R0, #0              ; Is the lock free?
    ITT     EQ                  ; IT instruction for STREXEQ and CMPEQ
    STREXEQ R0, R1, [LockAddr]  ; Try and claim the lock
    CMPEQ   R0, #0              ; Did this succeed?
    BNE     try                 ; No - try again
    ....                        ; Yes - we have the lock
```


© NXP B.V. 2010. All rights reserved.

2.4.9 CLREX

Clear Exclusive.

2.4.9.1 Syntax

CLREX{*cond*}

where:

*cond i*s an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.4.9.2 Operation

Use `CLREX` to make the next `STREX`, `STREXB`, or `STREXH` instruction write 1 to its destination register and fail to perform the store. It is useful in exception handler code to force the failure of the store exclusive if the exception occurs between a load exclusive instruction and the matching store exclusive instruction in a synchronization operation.

See Section 34–3.2.7 “Synchronization primitives” for more information.

2.4.9.3 Condition flags

These instructions do not change the flags.

2.4.9.4 Examples

```
CLREX
```


© NXP B.V. 2010. All rights reserved.

2.5 General data processing instructions

Table 34–620 shows the data processing instructions:

Table 620. Data processing instructions

Mnemonic	Brief description	See
ADC	Add with Carry	Section 34–2.5.1
ADD	Add	Section 34–2.5.1
ADDW	Add	Section 34–2.5.1
AND	Logical AND	Section 34–2.5.2
ASR	Arithmetic Shift Right	Section 34–2.5.3
BIC	Bit Clear	Section 34–2.5.2
CLZ	Count leading zeros	Section 34–2.5.4
CMN	Compare Negative	Section 34–2.5.5
CMP	Compare	Section 34–2.5.5
EOR	Exclusive OR	Section 34–2.5.2
LSL	Logical Shift Left	Section 34–2.5.3
LSR	Logical Shift Right	Section 34–2.5.3
MOV	Move	Section 34–2.5.6
MOVT	Move Top	Section 34–2.5.7
MOVW	Move 16-bit constant	Section 34–2.5.6
MVN	Move NOT	Section 34–2.5.6
ORN	Logical OR NOT	Section 34–2.5.2
ORR	Logical OR	Section 34–2.5.2
RBIT	Reverse Bits	Section 34–2.5.8
REV	Reverse byte order in a word	Section 34–2.5.8
REV16	Reverse byte order in each halfword	Section 34–2.5.8
REVSH	Reverse byte order in bottom halfword and sign extend	Section 34–2.5.8
ROR	Rotate Right	Section 34–2.5.3
RRX	Rotate Right with Extend	Section 34–2.5.3
RSB	Reverse Subtract	Section 34–2.5.1
SBC	Subtract with Carry	Section 34–2.5.1
SUB	Subtract	Section 34–2.5.1
SUBW	Subtract	Section 34–2.5.1
TEQ	Test Equivalence	Section 34–2.5.9
TST	Test	Section 34–2.5.9

© NXP B.V. 2010. All rights reserved.

2.5.1 ADD, ADC, SUB, SBC, and RSB

Add, Add with carry, Subtract, Subtract with carry, and Reverse Subtract.

2.5.1.1 Syntax

op{S}{*cond*} {*Rd*,} *Rn*, *Operand2*

op{*cond*} {*Rd*,} *Rn*, *#imm12* ; ADD and SUB only

where:

op is one of:

- `ADD`: Add.
- `ADC`: Add with Carry.
- `SUB`: Subtract.
- `RSB`: Reverse Subtract.

S: is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see Section 34–2.3.7 “Conditional execution”.

*cond: i*s an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See Section 34–2.3.3 for details of the options.

imm12 is any value in the range 0-4095.

2.5.1.2 Operation

The `ADD` instruction adds the value of *Operand2* or *imm12* to the value in *Rn*.

The `ADC` instruction adds the values in *Rn* and *Operand2*, together with the carry flag.

The `SUB` instruction subtracts the value of *Operand2* or *imm12* from the value in *Rn*.

The `SBC` instruction subtracts the value of *Operand2* from the value in *Rn*. If the carry flag is clear, the result is reduced by one.

The `RSB` instruction subtracts the value in *Rn* from the value of *Operand2*. This is useful because of the wide range of options for *Operand2*.

Use `ADC` and `SBC` to synthesize multiword arithmetic, see Section 34–2.5.1.6.

See also Section 34–2.4.1.

Remark: `ADDW` is equivalent to the `ADD` syntax that uses the *imm12* operand. `SUBW` is equivalent to the `SUB` syntax that uses the *imm12* operand.

2.5.1.3 Restrictions

- *Operand2* must not be SP and must not be PC
- *Rd* can be SP only in `ADD` and `SUB`, and only with the additional restrictions:

© NXP B.V. 2010. All rights reserved.

- *Rn* must also be SP
- any shift in *Operand2* must be limited to a maximum of 3 bits using `LSL`

- *Rn* can be SP only in `ADD` and `SUB`
- *Rd* can be PC only in the *cond* instruction where:
  - you must not specify the S suffix
  - *Rm* must not be PC and must not be SP
  - if the instruction is conditional, it must be the last instruction in the IT block
- with the exception of the *cond* instruction, *Rn* can be PC only in `ADD` and `SUB`, and only with the additional restrictions:
  - you must not specify the S suffix
  - the second operand must be a constant in the range 0 to 4095.

Note

- When using the PC for an addition or a subtraction, bits[1:0] of the PC are rounded to b00 before performing the calculation, making the base address for the calculation word-aligned.
- If you want to generate the address of an instruction, you have to adjust the constant based on the value of the PC. ARM recommends that you use the `ADR` instruction instead of `ADD` or `SUB` with *Rn* equal to the PC, because your assembler automatically calculates the correct constant for the `ADR` instruction.

When *Rd* is PC in the *cond* instruction:

- bit[0] of the value written to the PC is ignored
- a branch occurs to the address created by forcing bit[0] of that value to 0.

2.5.1.4 Condition flags

If S is specified, these instructions update the N, Z, C and V flags according to the result.

2.5.1.5 Examples

```
ADD     R2, R1, R3
SUBS    R8, R6, #240        ; Sets the flags on the result
RSB     R4, R4, #1280       ; Subtracts contents of R4 from 1280
ADCHI   R11, R0, R3         ; Only executed if C flag set and Z
                            ; flag clear
```

2.5.1.6 Multiword arithmetic examples

Section 34– shows two instructions that add a 64-bit integer contained in R2 and R3 to another 64-bit integer contained in R0 and R1, and place the result in R4 and R5.

Multiword values do not have to use consecutive registers. Section 34– shows instructions that subtract a 96-bit integer contained in R9, R1, and R11 from another contained in R6, R2, and R8. The example stores the result in R6, R9, and R2.

64-bit addition:

```
ADDS    R4, R0, R2    ; add the least significant words
ADC     R5, R1, R3    ; add the most significant words with carry
```


© NXP B.V. 2010. All rights reserved.

96-bit subtraction:

```
SUBS    R6, R6, R9     ; subtract the least significant words
SBCS    R9, R2, R1     ; subtract the middle words with carry
SBC     R2, R8, R11    ; subtract the most significant words with carry
```


© NXP B.V. 2010. All rights reserved.

2.5.2 AND, ORR, EOR, BIC, and ORN

Logical AND, OR, Exclusive OR, Bit Clear, and OR NOT.

2.5.2.1 Syntax

op{S}{*cond*} {*Rd*,} *Rn*, *Operand2*

where:

op is one of:

- `AND`: logical AND.
- `ORR`: logical OR, or bit set.
- `EOR`: logical Exclusive OR.
- `BIC`: logical AND NOT, or bit clear.
- `ORN`: logical OR NOT.

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see Section 34–2.3.7 “Conditional execution”.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See Section 34–2.3.3 for details of the options.

2.5.2.2 Operation

The `AND`, `EOR`, and `ORR` instructions perform bitwise AND, Exclusive OR, and OR operations on the values in *Rn* and *Operand2*.

The `BIC` instruction performs an AND operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

The `ORN` instruction performs an OR operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

2.5.2.3 Restrictions

Do not use SP and do not use PC.

2.5.2.4 Condition flags

If S is specified, these instructions:

- update the N and Z flags according to the result
- can update the C flag during the calculation of *Operand2*, see Section 34–2.3.3
- do not affect the V flag.

2.5.2.5 Examples

```
AND     R9, R2, #0xFF00
ORREQ   R2, R0, R5
ANDS    R9, R8, #0x19
```


© NXP B.V. 2010. All rights reserved.

```
EORS    R7, R11, #0x18181818
BIC     R0, R1, #0xab
ORN     R7, R11, R14, ROR #4
ORNS    R7, R11, R14, ASR #32
```


© NXP B.V. 2010. All rights reserved.

2.5.3 ASR, LSL, LSR, ROR, and RRX

Arithmetic Shift Right, Logical Shift Left, Logical Shift Right, Rotate Right, and Rotate Right with Extend.

2.5.3.1 Syntax

op{S}{*cond*} *Rd*, *Rm*, *Rs*

op{S}{*cond*} *Rd*, *Rm*, #*n*

RRX{S}{*cond*} *Rd*, *Rm*

where:

op is one of:

- `ASR`: Arithmetic Shift Right.
- `LSL`: Logical Shift Left.
- `LSR`: Logical Shift Right.
- `ROR`: Rotate Right.

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

Rm is the register holding the shift length to apply to the value in *Rm*. Only the least significant byte is used and can be in the range 0 to 255.

Rs s the register holding the shift length to apply to the value in *Rm*. Only the least significant byte is used and can be in the range 0 to 255.

n is the shift length. The range of shift length depends on the instruction:

- `ASR`: shift length from 1 to 32
- `LSL`: shift length from 0 to 31
- `LSR`: shift length from 1 to 32
- `ROR`: shift length from 1 to 31

Remark: `MOV{S}{cond} Rd, Rm` is the preferred syntax for `LSL{S}{cond} Rd, Rm, #0`.

2.5.3.2 Operation

`ASR`, `LSL`, `LSR`, and `ROR` move the bits in the register *Rm* to the left or right by the number of places specified by constant *n* or register *Rs*.

`RRX` moves the bits in register *Rm* to the right by 1.

In all these instructions, the result is written to *Rd*, but the value in register *Rm* remains unchanged. For details on what result is generated by the different instructions, see Section 34–2.3.4 “Shift Operations”.

2.5.3.3 Restrictions

Do not use SP and do not use PC.

© NXP B.V. 2010. All rights reserved.

2.5.3.4 Condition flags

If S is specified:

- these instructions update the N and Z flags according to the result
- the C flag is updated to the last bit shifted out, except when the shift length is 0, see Section 34–2.3.4 “Shift Operations”.

2.5.3.5 Examples

```
ASR    R7, R8, #9  ; Arithmetic shift right by 9 bits
LSLS   R1, R2, #3  ; Logical shift left by 3 bits with flag update
LSR    R4, R5, #6  ; Logical shift right by 6 bits
ROR    R4, R5, R6  ; Rotate right by the value in the bottom byte of R6
RRX    R4, R5      ; Rotate right with extend
```


© NXP B.V. 2010. All rights reserved.

2.5.4 CLZ

Count Leading Zeros.

2.5.4.1 Syntax

CLZ{cond} Rd, Rm

where:

cond is an optional condition code, see Section 34–2.3.7.

Rd is the destination register.

Rm is the operand register.

2.5.4.2 Operation

The `CLZ` instruction counts the number of leading zeros in the value in *Rm* and returns the result in *Rd*. The result value is 32 if no bits are set in the source register, and zero if bit[31] is set.

2.5.4.3 Restrictions

Do not use SP and do not use PC.

2.5.4.4 Condition flags

This instruction does not change the flags.

2.5.4.5 Examples

```
CLZ     R4,R9
CLZNE   R2,R3
```


© NXP B.V. 2010. All rights reserved.

2.5.5 CMP and CMN

Compare and Compare Negative.

2.5.5.1 Syntax

CMP{cond} Rn, Operand2

CMN{cond} Rn, Operand2

where:

cond is an optional condition code, see Section 34–2.3.7.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See Flexible second operand on page 3-10for details of the options.

2.5.5.2 Operation

These instructions compare the value in a register with *Operand2*. They update the condition flags on the result, but do not write the result to a register.

The `CMP` instruction subtracts the value of *Operand2* from the value in *Rn*. This is the same as a `SUBS` instruction, except that the result is discarded.

The `CMN` instruction adds the value of *Operand2* to the value in *Rn*. This is the same as an `ADDS` instruction, except that the result is discarded.

2.5.5.3 Restrictions

In these instructions:

- do not use PC
- *Operand2* must not be SP.

2.5.5.4 Condition flags

These instructions update the N, Z, C and V flags according to the result.

2.5.5.5 Examples

```
CMP     R2, R9
CMN     R0, #6400
CMPGT   SP, R7, LSL #2
```


© NXP B.V. 2010. All rights reserved.

2.5.6 MOV and MVN

Move and Move NOT.

2.5.6.1 Syntax

MOV{S}{cond} Rd, Operand2

MOV{cond} Rd, #imm16

MVN{S}{cond} Rd, Operand2

where:

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see Section 34–2.3.7.

cond is an optional condition code, see Section 34–2.3.7.

Rd is the destination register.

Operand2 is a flexible second operand. See Flexible second operand on page 3-10for details of the options.

imm16 is any value in the range 0-65535.

2.5.6.2 Operation

The `MOV` instruction copies the value of *Operand2* into *Rd*.

When *Operand2* in a `MOV` instruction is a register with a shift other than `LSL #0`, the preferred syntax is the corresponding shift instruction:

- `ASR{S}{cond} Rd, Rm, #n` is the preferred syntax for `MOV{S}{cond} Rd, Rm, ASR #n`
- `LSL{S}{cond} Rd, Rm, #n` is the preferred syntax for `MOV{S}{cond} Rd, Rm, LSL #n`*n* if != 0
- `LSR{S}{cond} Rd, Rm, #n` is the preferred syntax for `MOV{S}{cond} Rd, Rm, LSR #n`
- `ROR{S}{cond} Rd, Rm, #n` is the preferred syntax for `MOV{S}{cond} Rd, Rm, ROR #n`
- `RRX{S}{cond} Rd, Rm` is the preferred syntax for `MOV{S}{cond} Rd, Rm, RRX`.

Also, the `MOV` instruction permits additional forms of *Operand2* as synonyms for shift instructions:

- `MOV{S}{cond} Rd, Rm, ASR Rs` is a synonym for `ASR{S}{cond} Rd, Rm, Rs`
- `MOV{S}{cond} Rd, Rm, LSL Rs` is a synonym for `LSL{S}{cond} Rd, Rm, Rs`
- `MOV{S}{cond} Rd, Rm, LSR Rs` is a synonym for `LSR{S}{cond} Rd, Rm, Rs`
- `MOV{S}{cond} Rd, Rm, ROR Rs` is a synonym for `ROR{S}{cond} Rd, Rm, Rs`

See Section 34–2.5.3.

The `MVN` instruction takes the value of *Operand2*, performs a bitwise logical NOT operation on the value, and places the result into *Rd*.

Remark: The `MOVW` instruction provides the same function as `MOV`, but is restricted to using the *imm16* operand.

© NXP B.V. 2010. All rights reserved.

2.5.6.3 Restrictions

You can use SP and PC only in the `MOV` instruction, with the following restrictions:

- the second operand must be a register without shift
- you must not specify the S suffix.

When *Rd* is PC in a `MOV` instruction:

- bit[0] of the value written to the PC is ignored
- a branch occurs to the address created by forcing bit[0] of that value to 0.

Remark: Though it is possible to use `MOV` as a branch instruction, ARM strongly recommends the use of a `BX` or `BLX` instruction to branch for software portability to the ARM instruction set.

2.5.6.4 Condition flags

If S is specified, these instructions:

- update the N and Z flags according to the result
- can update the C flag during the calculation of *Operand2*, see Section 34–2.3.3
- do not affect the V flag.

2.5.6.5 Example

```
MOVS  R11, #0x000B    ; Write value of 0x000B to R11, flags get updated
      MOV   R1, #0xFA05     ; Write value of 0xFA05 to R1, flags are not updated
      MOVS  R10, R12        ; Write value in R12 to R10, flags get updated
      MOV   R3, #23         ; Write value of 23 to R3
      MOV   R8, SP          ; Write value of stack pointer to R8
      MVNS  R2, #0xF        ; Write value of 0xFFFFFFF0 (bitwise inverse of 0xF)
                            ; to the R2 and update flags
```


© NXP B.V. 2010. All rights reserved.

2.5.7 MOVT

Move Top.

2.5.7.1 Syntax

MOVT{cond} Rd, #imm16

where:

cond is an optional condition code, see Section 34–2.3.7.

Rd is the destination register.

imm16 is a 16-bit immediate constant.

2.5.7.2 Operation

`MOVT` writes a 16-bit immediate value, *imm16*, to the top halfword, *Rd*[31:16], of its destination register. The write does not affect *Rd*[15:0].

The `MOV`, `MOVT` instruction pair enables you to generate any 32-bit constant.

2.5.7.3 Restrictions

Rd must not be SP and must not be PC.

2.5.7.4 Condition flags

This instruction does not change the flags.

2.5.7.5 Examples

```
MOVT   R3, #0xF123 ; Write 0xF123 to upper halfword of R3, lower halfword
                   ; and APSR are unchanged
```


© NXP B.V. 2010. All rights reserved.

2.5.8 REV, REV16, REVSH, and RBIT

Reverse bytes and Reverse bits.

2.5.8.1 Syntax

op{cond} Rd, Rn

where:

op is any of:

`REV` Reverse byte order in a word.

`REV16` Reverse byte order in each halfword independently.

`REVSH` Reverse byte order in the bottom halfword, and sign extend to

`RBIT` Reverse the bit order in a 32-bit word.

cond is an optional condition code, see Section 34–2.3.7.

Rd is the destination register.

Rn is the register holding the operand.

2.5.8.2 Operation

Use these instructions to change endianness of data:

`REV`: converts 32-bit big-endian data into little-endian data or 32-bit little-endian data into big-endian data.

`REV16` converts 16-bit big-endian data into little-endian data or 16-bit little-endian data into big-endian data.

`REVSH` converts either:

16-bit signed big-endian data into 32-bit signed little-endian data

16-bit signed little-endian data into 32-bit signed big-endian data.

2.5.8.3 Restrictions

Do not use SP and do not use PC.

2.5.8.4 Condition flags

These instructions do not change the flags.

2.5.8.5 Examples

```
REV    R3, R7  ; Reverse byte order of value in R7 and write it to R3
REV16  R0, R0  ; Reverse byte order of each 16-bit halfword in R0
REVSH  R0, R5  ; Reverse Signed Halfword
REVHS  R3, R7  ; Reverse with Higher or Same condition
RBIT   R7, R8  ; Reverse bit order of value in R8 and write the result to R7
```


© NXP B.V. 2010. All rights reserved.

2.5.9 TST and TEQ

Test bits and Test Equivalence.

2.5.9.1 Syntax

`TST`{*cond*} *Rn*, *Operand2*

`TEQ`{*cond*} *Rn*, *Operand2*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See Section 34–2.3.3 for details of the options.

2.5.9.2 Operation

These instructions test the value in a register against *Operand2*. They update the condition flags based on the result, but do not write the result to a register.

The `TST` instruction performs a bitwise AND operation on the value in *Rn* and the value of *Operand2*. This is the same as the `ANDS` instruction, except that it discards the result.

To test whether a bit of *Rn* is 0 or 1, use the `TST` instruction with an *Operand2* constant that has that bit set to 1 and all other bits cleared to 0.

The `TEQ` instruction performs a bitwise Exclusive OR operation on the value in *Rn* and the value of *Operand2*. This is the same as the `EORS` instruction, except that it discards the result.

Use the `TEQ` instruction to test if two values are equal without affecting the V or C flags.

`TEQ` is also useful for testing the sign of a value. After the comparison, the N flag is the logical Exclusive OR of the sign bits of the two operands.

2.5.9.3 Restrictions

Do not use SP and do not use PC.

2.5.9.4 Condition flags

These instructions:

- update the N and Z flags according to the result
- can update the C flag during the calculation of *Operand2*, see Section 34–2.3.3
- do not affect the V flag.

2.5.9.5 Examples

```
TST     R0, #0x3F8  ; Perform bitwise AND of R0 value to 0x3F8,
                    ; APSR is updated but result is discarded
TEQEQ   R10, R9     ; Conditionally test if value in R10 is equal to
                    ; value in R9, APSR is updated but result is discarded
```


© NXP B.V. 2010. All rights reserved.

2.6 Multiply and divide instructions

Table 34–621 shows the multiply and divide instructions:

Table 621. Multiply and divide instructions

Mnemonic	Brief description	See
MLA	Multiply with Accumulate, 32-bit result	Section 34–2.6.1
MLS	Multiply and Subtract, 32-bit result	Section 34–2.6.1
MUL	Multiply, 32-bit result	Section 34–2.6.1
SDIV	Signed Divide	Section 34–2.6.3
SMLAL	Signed Multiply with Accumulate (32x32+64), 64-bit result	Section 34–2.6.2
SMULL	Signed Multiply (32x32), 64-bit result	Section 34–2.6.2
UDIV	Unsigned Divide	Section 34–2.6.3
UMLAL	Unsigned Multiply with Accumulate (32x32+64), 64-bit result	Section 34–2.6.2
UMULL	Unsigned Multiply (32x32), 64-bit result	Section 34–2.6.2

© NXP B.V. 2010. All rights reserved.

2.6.1 MUL, MLA, and MLS

Multiply, Multiply with Accumulate, and Multiply with Subtract, using 32-bit operands, and producing a 32-bit result.

2.6.1.1 Syntax

`MUL{S}`{*cond*} {*Rd*,} *Rn*, *Rm* ; Multiply

`MLA`{*cond*} *Rd*, *Rn*, *Rm*, *Ra* ; Multiply with accumulate

MLS{*cond*} *Rd*, *Rn*, *Rm*, *Ra* ; Multiply with subtract

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn, Rm are registers holding the values to be multiplied.

Ra is a register holding the value to be added or subtracted from.

2.6.1.2 Operation

The `MUL` instruction multiplies the values from *Rn* and *Rm*, and places the least significant 32 bits of the result in *Rd*.

The `MLA` instruction multiplies the values from *Rn* and *Rm*, adds the value from *Ra*, and places the least significant 32 bits of the result in *Rd*.

The `MLS` instruction multiplies the values from *Rn* and *Rm*, subtracts the product from the value from *Ra*, and places the least significant 32 bits of the result in *Rd*.

The results of these instructions do not depend on whether the operands are signed or unsigned.

2.6.1.3 Restrictions

In these instructions, do not use SP and do not use PC.

If you use the S suffix with the `MUL` instruction:

- *Rd*, *Rn*, and *Rm* must all be in the range `R0` to `R7`
- *Rd* must be the same as *Rm*
- you must not use the *cond* suffix.

2.6.1.4 Condition flags

If S is specified, the `MUL` instruction:

- updates the N and Z flags according to the result
- does not affect the C and V flags.

© NXP B.V. 2010. All rights reserved.

2.6.1.5 Examples

```
MUL     R10, R2, R5     ; Multiply, R10 = R2 x R5
MLA     R10, R2, R1, R5 ; Multiply with accumulate, R10 = (R2 x R1) + R5
MULS    R0, R2, R2      ; Multiply with flag update, R0 = R2 x R2
MULLT   R2, R3, R2      ; Conditionally multiply, R2 = R3 x R2
MLS     R4, R5, R6, R7  ; Multiply with subtract, R4 = R7 - (R5 x R6)
```


© NXP B.V. 2010. All rights reserved.

2.6.2 UMULL, UMLAL, SMULL, and SMLAL

Signed and Unsigned Long Multiply, with optional Accumulate, using 32-bit operands and producing a 64-bit result.

2.6.2.1 Syntax

op{*cond*} *RdLo*, *RdHi*, *Rn*, *Rm*

where:

op is one of:

- `UMULL`: Unsigned Long Multiply.
- `UMLAL`: Unsigned Long Multiply, with Accumulate.
- `SMULL`: Signed Long Multiply.
- `SMLAL`: Signed Long Multiply, with Accumulate.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

RdHi, *RdLo* are the destination registers. For `UMLAL` and `SMLAL` they also hold the accumulating value.

Rn, Rm are registers holding the operands.

2.6.2.2 Operation

The `UMULL` instruction interprets the values from *Rn* and *Rm* as unsigned integers. It multiplies these integers and places the least significant 32 bits of the result in *RdLo*, and the most significant 32 bits of the result in *RdHi*.

The `UMLAL` instruction interprets the values from *Rn* and *Rm* as unsigned integers. It multiplies these integers, adds the 64-bit result to the 64-bit unsigned integer contained in *RdHi* and *RdLo*, and writes the result back to *RdHi* and *RdLo*.

The `SMULL` instruction interprets the values from *Rn* and *Rm* as two’s complement signed integers. It multiplies these integers and places the least significant 32 bits of the result in *RdLo*, and the most significant 32 bits of the result in *RdHi*.

The `SMLAL` instruction interprets the values from *Rn* and *Rm* as two’s complement signed integers. It multiplies these integers, adds the 64-bit result to the 64-bit signed integer contained in *RdHi* and *RdLo*, and writes the result back to *RdHi* and *RdLo*.

2.6.2.3 Restrictions

In these instructions:

- do not use SP and do not use PC
- *RdHi* and *RdLo* must be different registers.

2.6.2.4 Condition flags

These instructions do not affect the condition code flags.

© NXP B.V. 2010. All rights reserved.

2.6.2.5 Examples

```
UMULL      R0, R4, R5, R6   ; Unsigned (R4,R0) = R5 x R6
SMLAL      R4, R5, R3, R8   ; Signed (R5,R4) = (R5,R4) + R3 x R8
```


© NXP B.V. 2010. All rights reserved.

2.6.3 SDIV and UDIV

Signed Divide and Unsigned Divide.

2.6.3.1 Syntax

SDIV{*cond*} {*Rd*,} *Rn*, *Rm*

UDIV{*cond*} {*Rd*,} *Rn, Rm*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn is the register holding the value to be divided.

Rm is a register holding the divisor.

2.6.3.2 Operation

`SDIV` performs a signed integer division of the value in *Rn* by the value in *Rm*.

`UDIV` performs an unsigned integer division of the value in *Rn* by the value in *Rm*.

For both instructions, if the value in *Rn* is not divisible by the value in *Rm*, the result is rounded towards zero.

2.6.3.3 Restrictions

Do not use SP and do not use PC.

2.6.3.4 Condition flags

These instructions do not change the flags.

2.6.3.5 Examples

```
SDIV  R0, R2, R4  ; Signed divide, R0 = R2/R4
UDIV  R8, R8, R1  ; Unsigned divide, R8 = R8/R1
```


© NXP B.V. 2010. All rights reserved.

2.7 Saturating instructions

This section describes the saturating instructions, SSAT and USAT.

2.7.1 SSAT and USAT

Signed Saturate and Unsigned Saturate to any bit position, with optional shift before saturating.

2.7.1.1 Syntax

op{*cond*} *Rd*, #*n*, *Rm* {, *shift* #s}

where:

op is one of:

SSAT Saturates a signed value to a signed range.

USAT Saturates a signed value to an unsigned range.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

n specifies the bit position to saturate to:

- *n* ranges from 1 to 32 for SSAT.
- *n* ranges from 0 to 31 for USAT.

Rm is the register containing the value to saturate.

shift #s is an optional shift applied to *Rm* before saturating. It must be one of the following:

ASR #*s*: where *s* is in the range 1 to 31

LSL #*s*: where *s* is in the range 0 to 31.

2.7.1.2 Operation

These instructions saturate to a signed or unsigned *n*-bit value.

The SSAT instruction applies the specified shift, then saturates to the signed range $-2^{n-1} \le x \le 2^{n-1}-1$.

The USAT instruction applies the specified shift, then saturates to the unsigned range $0 \le x \le 2^{n}-1$.

For signed *n*-bit saturation using SSAT, this means that:

- if the value to be saturated is less than -2^{n-1}, the result returned is -2^{n-1}
- if the value to be saturated is greater than $2^{n-1}-1$, the result returned is $2^{n-1}-1$
- otherwise, the result returned is the same as the value to be saturated.

For unsigned *n*-bit saturation using USAT, this means that:

- if the value to be saturated is less than 0, the result returned is 0
- if the value to be saturated is greater than $2^{n}-1$, the result returned is $2^{n}-1$

© NXP B.V. 2010. All rights reserved.

- otherwise, the result returned is the same as the value to be saturated.

If the returned result is different from the value to be saturated, it is called **saturation**. If saturation occurs, the instruction sets the Q flag to 1 in the APSR. Otherwise, it leaves the Q flag unchanged. To clear the Q flag to 0, you must use the `MSR` instruction, see Section 34–2.10.7.

To read the state of the Q flag, use the `MRS` instruction, see Section 34–2.10.6.

2.7.1.3 Restrictions

Do not use SP and do not use PC.

2.7.1.4 Condition flags

These instructions do not affect the condition code flags.

If saturation occurs, these instructions set the Q flag to 1.

2.7.1.5 Examples

```
SSAT    R7, #16, R7, LSL #4  ; Logical shift left value in R7 by 4, then
                             ; saturate it as a signed 16-bit value and
                             ; write it back to R7
USATNE  R0, #7, R5           ; Conditionally saturate value in R5 as an
                             ; unsigned 7 bit value and write it to R0
```


© NXP B.V. 2010. All rights reserved.

2.8 Bitfield instructions

Table 34–622 shows the instructions that operate on adjacent sets of bits in registers or bitfields:

Table 622. Packing and unpacking instructions

Mnemonic	Brief description	See
BFC	Bit Field Clear	Section 34–2.8.1
BFI	Bit Field Insert	Section 34–2.8.1
SBFX	Signed Bit Field Extract	Section 34–2.8.2
SXTB	Sign extend a byte	Section 34–2.8.3
SXTH	Sign extend a halfword	Section 34–2.8.3
UBFX	Unsigned Bit Field Extract	Section 34–2.8.2
UXTB	Zero extend a byte	Section 34–2.8.3
UXTH	Zero extend a halfword	Section 34–2.8.3

© NXP B.V. 2010. All rights reserved.

2.8.1 BFC and BFI

Bit Field Clear and Bit Field Insert.

2.8.1.1 Syntax

BFC{*cond*} *Rd*, #*lsb*, #*width*

BFI{*cond*} *Rd*, *Rn*, #*lsb*, #*width*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

Rn is the source register.

lsb is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width is the width of the bitfield and must be in the range 1 to 32–*lsb*.

2.8.1.2 Operation

`BFC` clears a bitfield in a register. It clears *width* bits in *Rd*, starting at the low bit position `lsb`. Other bits in *Rd* are unchanged.

`BFI` copies a bitfield into one register from another register. It replaces *width* bits in *Rd* starting at the low bit position `lsb`, with *width* bits from *Rn* starting at bit[0]. Other bits in *Rd* are unchanged.

2.8.1.3 Restrictions

Do not use SP and do not use PC.

2.8.1.4 Condition flags

These instructions do not affect the flags.

2.8.1.5 Examples

```
BFC   R4, #8, #12       ; Clear bit 8 to bit 19 (12 bits) of R4 to 0
BFI   R9, R2, #8, #12   ; Replace bit 8 to bit 19 (12 bits) of R9 with
                        ; bit 0 to bit 11 from R2
```


© NXP B.V. 2010. All rights reserved.

2.8.2 SBFX and UBFX

Signed Bit Field Extract and Unsigned Bit Field Extract.

2.8.2.1 Syntax

SBFX{*cond*} *Rd*, *Rn*, #*lsb*, #*width*

UBFX{*cond*} *Rd*, *Rn*, #*lsb*, #*width*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

Rn is the source register.

lsb is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width is the width of the bitfield and must be in the range 1 to 32–*lsb*.

2.8.2.2 Operation

`SBFX` extracts a bitfield from one register, sign extends it to 32 bits, and writes the result to the destination register.

`UBFX` extracts a bitfield from one register, zero extends it to 32 bits, and writes the result to the destination register.

2.8.2.3 Restrictions

Do not use SP and do not use PC.

2.8.2.4 Condition flags

These instructions do not affect the flags.

2.8.2.5 Examples

```
SBFX  R0, R1, #20, #4  ; Extract bit 20 to bit 23 (4 bits) from R1 and sign
                       ; extend to 32 bits and then write the result to R0.
UBFX  R8, R11, #9, #10 ; Extract bit 9 to bit 18 (10 bits) from R11 and zero
                       ; extend to 32 bits and then write the result to R8
```


© NXP B.V. 2010. All rights reserved.

2.8.3 SXT and UXT

Sign extend and Zero extend.

2.8.3.1 Syntax

SXT*extend*{*cond*} {*Rd*,} *Rm* {, ROR #*n*}

UXT*extend*{*cond*} {*Rd*}, *Rm* {, ROR #*n*}

where:

extend is one of:

- `B`: Extends an 8-bit value to a 32-bit value.
- `H`: Extends a 16-bit value to a 32-bit value.

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

Rm is the register holding the value to extend.

ROR #*n* is one of:

- `ROR #8`: Value from *Rm* is rotated right 8 bits.
- `ROR #16`: Value from *Rm* is rotated right 16 bits.
- `ROR #24`: Value from *Rm* is rotated right 24 bits.

If ROR #*n* is omitted, no rotation is performed.

2.8.3.2 Operation

These instructions do the following:

1. Rotate the value from *Rm* right by 0, 8, 16 or 24 bits.
2. Extract bits from the resulting value:
   - `SXTB` extracts bits[7:0] and sign extends to 32 bits.
   - `UXTB` extracts bits[7:0] and zero extends to 32 bits.
   - `SXTH` extracts bits[15:0] and sign extends to 32 bits.
   - `UXTH` extracts bits[15:0] and zero extends to 32 bits.

2.8.3.3 Restrictions

Do not use SP and do not use PC.

2.8.3.4 Condition flags

These instructions do not affect the flags.

© NXP B.V. 2010. All rights reserved.

2.8.3.5 Examples

```
SXTH  R4, R6, ROR #16  ; Rotate R6 right by 16 bits, then obtain the lower
                       ; halfword of the result and then sign extend to
                       ; 32 bits and write the result to R4.
UXTB  R3, R10          ; Extract lowest byte of the value in R10 and zero
                       ; extend it, and write the result to R3
```


© NXP B.V. 2010. All rights reserved.

2.9 Branch and control instructions

Table 34–623 shows the branch and control instructions:

Table 623. Branch and control instructions

Mnemonic	Brief description	See
B	Branch	Section 34–2.9.1
BL	Branch with Link	Section 34–2.9.1
BLX	Branch indirect with Link	Section 34–2.9.1
BX	Branch indirect	Section 34–2.9.1
CBNZ	Compare and Branch if Non Zero	Section 34–2.9.2
CBZ	Compare and Branch if Non Zero	Section 34–2.9.2
IT	If-Then	Section 34–2.9.3
TBB	Table Branch Byte	Section 34–2.9.4
TBH	Table Branch Halfword	Section 34–2.9.4

© NXP B.V. 2010. All rights reserved.

2.9.1 B, BL, BX, and BLX

Branch instructions.

2.9.1.1 Syntax

B{*cond*} *label*

BL{*cond*} *label*

BX{*cond*} *Rm*

BLX{*cond*} *Rm*

where:

B is branch (immediate).

BL is branch with link (immediate).

BX is branch indirect (register).

BLX is branch indirect with link (register).

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

label is a PC-relative expression. See Section 34–2.3.6 “PC-relative expressions”.

Rm is a register that indicates an address to branch to. Bit[0] of the value in *Rm* must be 1, but the address to branch to is created by changing bit[0] to 0.

2.9.1.2 Operation

All these instructions cause a branch to *label*, or to the address indicated in *Rm*. In addition:

- The `BL` and `BLX` instructions write the address of the next instruction to LR (the link register, R14).
- The `BX` and `BLX` instructions cause a UsageFault exception if bit[0] of *Rm* is 0.

B*cond* label is the only conditional instruction that can be either inside or outside an IT block. All other branch instructions must be conditional inside an IT block, and must be unconditional outside the IT block, see Section 34–2.9.3.

Table 34–624 shows the ranges for the various branch instructions.

Table 624. Branch ranges

Instruction	Branch range
`B label`	–16 MB to +16 MB
`B`*cond* `label` (outside IT block)	–1 MB to +1 MB
`B`*cond* `label` (inside IT block)	–16 MB to +16 MB
`BL`{*cond*} `label`	–16 MB to +16 MB
`BX`{*cond*} `Rm`	Any value in register
`BLX`{*cond*} `Rm`	Any value in register

Remark: You might have to use the .W suffix to get the maximum branch range. See Section 34–2.3.8 “Instruction width selection”.

© NXP B.V. 2010. All rights reserved.

2.9.1.3 Restrictions

The restrictions are:

- do not use PC in the `BLX` instruction
- for `BX` and `BLX`, bit[0] of *Rm* must be 1 for correct execution but a branch occurs to the target address created by changing bit[0] to 0
- when any of these instructions is inside an IT block, it must be the last instruction of the IT block.

B*cond* is the only conditional instruction that is not required to be inside an IT block. However, it has a longer branch range when it is inside an IT block.

2.9.1.4 Condition flags

These instructions do not change the flags.

2.9.1.5 Examples

```
B      loopA  ; Branch to loopA
BLE    ng     ; Conditionally branch to label ng
B.W    target ; Branch to target within 16MB range
BEQ    target ; Conditionally branch to target
BEQ.W  target ; Conditionally branch to target within 1MB
BL     funC   ; Branch with link (Call) to function funC, return address
              ; stored in LR
BX     LR     ; Return from function call
BXNE   R0     ; Conditionally branch to address stored in R0
BLX    R0     ; Branch with link and exchange (Call) to a address stored
              ; in R0
```


© NXP B.V. 2010. All rights reserved.

2.9.2 CBZ and CBNZ

Compare and Branch on Zero, Compare and Branch on Non-Zero.

2.9.2.1 Syntax

CBZ *Rn*, *label*

CBNZ *Rn*, *label*

where:

Rn is the register holding the operand.

label is the branch destination.

2.9.2.2 Operation

Use the `CBZ` or `CBNZ` instructions to avoid changing the condition code flags and to reduce the number of instructions.

`CBZ Rn, label` does not change condition flags but is otherwise equivalent to:

```
CMP     Rn, #0
     BEQ     label
```

`CBNZ Rn, label` does not change condition flags but is otherwise equivalent to:

```
CMP     Rn, #0
     BNE     label
```

2.9.2.3 Restrictions

The restrictions are:

- *Rn* must be in the range of `R0` to `R7`
- the branch destination must be within 4 to 130 bytes after the instruction
- these instructions must not be used inside an IT block.

2.9.2.4 Condition flags

These instructions do not change the flags.

2.9.2.5 Examples

```
CBZ    R5, target ; Forward branch if R5 is zero
CBNZ   R0, target ; Forward branch if R0 is not zero
```


© NXP B.V. 2010. All rights reserved.

2.9.3 IT

If-Then condition instruction.

2.9.3.1 Syntax

IT{*x*{*y*{*z*}}} *cond*

where:

x specifies the condition switch for the second instruction in the IT block.

y specifies the condition switch for the third instruction in the IT block.

z specifies the condition switch for the fourth instruction in the IT block.

cond specifies the condition for the first instruction in the IT block.

The condition switch for the second, third and fourth instruction in the IT block can be either:

- `T`: Then. Applies the condition *cond* to the instruction.
- `E`: Else. Applies the inverse condition of *cond* to the instruction.

Remark: It is possible to use `AL` (the **always** condition) for *cond* in an `IT` instruction. If this is done, all of the instructions in the IT block must be unconditional, and each of *x*, *y*, and *z* must be `T` or omitted but not `E`.

2.9.3.2 Operation

The `IT` instruction makes up to four following instructions conditional. The conditions can be all the same, or some of them can be the logical inverse of the others. The conditional instructions following the `IT` instruction form the **IT block**.

The instructions in the IT block, including any branches, must specify the condition in the *{cond}* part of their syntax.

Remark: Your assembler might be able to generate the required `IT` instructions for conditional instructions automatically, so that you do not need to write them yourself. See your assembler documentation for details.

A `BKPT` instruction in an IT block is always executed, even if its condition fails.

Exceptions can be taken between an `IT` instruction and the corresponding IT block, or within an IT block. Such an exception results in entry to the appropriate exception handler, with suitable return information in LR and stacked PSR.

Instructions designed for use for exception returns can be used as normal to return from the exception, and execution of the IT block resumes correctly. This is the only way that a PC-modifying instruction is permitted to branch to an instruction in an IT block.

2.9.3.3 Restrictions

The following instructions are not permitted in an IT block:

- `IT`
- `CBZ` and `CBNZ`

© NXP B.V. 2010. All rights reserved.

- CPSID and CPSIE.

Other restrictions when using an IT block are:

- a branch or any instruction that modifies the PC must either be outside an IT block or must be the last instruction inside the IT block. These are:
  - ADD PC, PC, Rm
  - MOV PC, Rm
  - B, BL, BX, BLX
  - any LDM, LDR, or POP instruction that writes to the PC
  - TBB and TBH
- do not branch to any instruction inside an IT block, except when returning from an exception handler
- all conditional instructions except B*cond* must be inside an IT block. B*cond* can be either outside or inside an IT block but has a larger branch range if it is inside one
- each instruction inside the IT block must specify a condition code suffix that is either the same or logical inverse as for the other instructions in the block.

Remark: Your assembler might place extra restrictions on the use of IT blocks, such as prohibiting the use of assembler directives within them.

2.9.3.4 Condition flags

This instruction does not change the flags.

2.9.3.5 Example

```
ITTE   NE          ; Next 3 instructions are conditional
ANDNE  R0, R0, R1  ; ANDNE does not update condition flags
ADDSNE R2, R2, #1  ; ADDSNE updates condition flags
MOVEQ  R2, R3      ; Conditional move

CMP    R0, #9      ; Convert R0 hex value (0 to 15) into ASCII
                   ; ('0'-'9', 'A'-'F')
ITE    GT          ; Next 2 instructions are conditional
ADDGT  R1, R0, #55 ; Convert 0xA -> 'A'
ADDLE  R1, R0, #48 ; Convert 0x0 -> '0'


IT     GT          ; IT block with only one conditional instruction

ADDGT  R1, R1, #1  ; Increment R1 conditionally

ITTEE  EQ          ; Next 4 instructions are conditional
MOVEQ  R0, R1      ; Conditional move
ADDEQ  R2, R2, #10 ; Conditional add
ANDNE  R3, R3, #1  ; Conditional AND
BNE.W  dloop       ; Branch instruction can only be used in the last
                   ; instruction of an IT block
```


© NXP B.V. 2010. All rights reserved.

```
IT     NE           ; Next instruction is conditional
ADD    R0, R0, R1   ; Syntax error: no condition code used in IT block
```


© NXP B.V. 2010. All rights reserved.

2.9.4 TBB and TBH

Table Branch Byte and Table Branch Halfword.

2.9.4.1 Syntax

TBB [*Rn*, *Rm*]

TBH [*Rn*, *Rm*, LSL #1]

where:

Rn is the register containing the address of the table of branch lengths.

If *Rn* is PC, then the address of the table is the address of the byte immediately following the `TBB` or `TBH` instruction.

Rm is the index register. This contains an index into the table. For halfword tables, `LSL #1` doubles the value in *Rm* to form the right offset into the table.

2.9.4.2 Operation

These instructions cause a PC-relative forward branch using a table of single byte offsets for `TBB`, or halfword offsets for `TBH`. *Rn* provides a pointer to the table, and *Rm* supplies an index into the table. For `TBB` the branch offset is twice the unsigned value of the byte returned from the table. and for `TBH` the branch offset is twice the unsigned value of the halfword returned from the table. The branch occurs to the address at that offset from the address of the byte immediately after the `TBB` or `TBH` instruction.

2.9.4.3 Restrictions

The restrictions are:

- *Rn* must not be SP
- *Rm* must not be SP and must not be PC
- when any of these instructions is used inside an IT block, it must be the last instruction of the IT block.

2.9.4.4 Condition flags

These instructions do not change the flags.

2.9.4.5 Examples

```
    ADR.W  R0, BranchTable_Byte
    TBB   [R0, R1]         ; R1 is the index, R0 is the base address of the
                           ; branch table
Case1
; an instruction sequence follows
Case2
; an instruction sequence follows
Case3
; an instruction sequence follows
BranchTable_Byte
    DCB    0                   ; Case1 offset calculation
    DCB    ((Case2-Case1)/2)   ; Case2 offset calculation
    DCB    ((Case3-Case1)/2)   ; Case3 offset calculation
```


© NXP B.V. 2010. All rights reserved.

```
    TBH    [PC, R1, LSL #1]   ; R1 is the index, PC is used as base of the
                              ; branch table
BranchTable_H
    DCI    ((CaseA - BranchTable_H)/2)  ; CaseA offset calculation
    DCI    ((CaseB - BranchTable_H)/2)  ; CaseB offset calculation
    DCI    ((CaseC - BranchTable_H)/2)  ; CaseC offset calculation

CaseA
; an instruction sequence follows
CaseB
; an instruction sequence follows
CaseC
; an instruction sequence follows
```


© NXP B.V. 2010. All rights reserved.

2.10 Miscellaneous instructions

Table 34–625 shows the remaining Cortex-M3 instructions:

Table 625. Miscellaneous instructions

Mnemonic	Brief description	See
BKPT	Breakpoint	Section 34–2.10.1
CPSID	Change Processor State, Disable Interrupts	Section 34–2.10.2
CPSIE	Change Processor State, Enable Interrupts	Section 34–2.10.2
DMB	Data Memory Barrier	Section 34–2.10.3
DSB	Data Synchronization Barrier	Section 34–2.10.4
ISB	Instruction Synchronization Barrier	Section 34–2.10.5
MRS	Move from special register to register	Section 34–2.10.6
MSR	Move from register to special register	Section 34–2.10.7
NOP	No Operation	Section 34–2.10.8
SEV	Send Event	Section 34–2.10.9
SVC	Supervisor Call	Section 34–2.10.10
WFE	Wait For Event	Section 34–2.10.11
WFI	Wait For Interrupt	Section 34–2.10.12

© NXP B.V. 2010. All rights reserved.

2.10.1 BKPT

Breakpoint.

2.10.1.1 Syntax

`BKPT` #*imm*

where:

imm is an expression evaluating to an integer in the range 0-255 (8-bit value).

2.10.1.2 Operation

The `BKPT` instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

imm is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The `BKPT` instruction can be placed inside an IT block, but it executes unconditionally, unaffected by the condition specified by the `IT` instruction.

2.10.1.3 Condition flags

This instruction does not change the flags.

2.10.1.4 Examples

```
BKPT 0xAB   ; Breakpoint with immediate value set to 0xAB (debugger can
            ; extract the immediate value by locating it using the PC)
```


© NXP B.V. 2010. All rights reserved.

2.10.2 CPS

Change Processor State.

2.10.2.1 Syntax

CPS*effect iflags*

where:

effect is one of:

- `IE` Clears the special purpose register.
- `ID` Sets the special purpose register

iflags is a sequence of one or more flags:

- `i` Set or clear PRIMASK.
- `f` Set or clear FAULTMASK.

2.10.2.2 Operation

`CPS` changes the PRIMASK and FAULTMASK special register values. See Section 34–3.1.3.6 “Exception mask registers” for more information about these registers.

2.10.2.3 Restrictions

The restrictions are:

- use `CPS` only from privileged software, it has no effect if used in unprivileged software.
- `CPS` cannot be conditional and so must not be used inside an IT block.

2.10.2.4 Condition flags

This instruction does not change the condition flags.

2.10.2.5 Examples

```
CPSID i  ; Disable interrupts and configurable fault handlers (set PRIMASK)
CPSID f  ; Disable interrupts and all fault handlers (set FAULTMASK)
CPSIE i  ; Enable interrupts and configurable fault handlers (clear PRIMASK)
CPSIE f  ; Enable interrupts and fault handlers (clear FAULTMASK)
```


© NXP B.V. 2010. All rights reserved.

2.10.3 DMB

Data Memory Barrier.

2.10.3.1 Syntax

DMB{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.10.3.2 Operation

`DMB` acts as a data memory barrier. It ensures that all explicit memory accesses that appear, in program order, before the `DMB` instruction are completed before any explicit memory accesses that appear, in program order, after the `DMB` instruction. `DMB` does not affect the ordering or execution of instructions that do not access memory.

2.10.3.3 Condition flags

This instruction does not change the flags.

2.10.3.4 Examples

```
DMB  ; Data Memory Barrier
```


© NXP B.V. 2010. All rights reserved.

2.10.4 DSB

Data Synchronization Barrier.

2.10.4.1 Syntax

DSB{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.10.4.2 Operation

`DSB` acts as a special data synchronization memory barrier. Instructions that come after the `DSB`, in program order, do not execute until the `DSB` instruction completes. The `DSB` instruction completes when all explicit memory accesses before it complete.

2.10.4.3 Condition flags

This instruction does not change the flags.

2.10.4.4 Examples

```
DSB ; Data Synchronisation Barrier
```


© NXP B.V. 2010. All rights reserved.

2.10.5 ISB

Instruction Synchronization Barrier.

2.10.5.1 Syntax

ISB{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.10.5.2 Operation

`ISB` acts as an instruction synchronization barrier. It flushes the pipeline of the processor, so that all instructions following the `ISB` are fetched from cache or memory again, after the `ISB` instruction has been completed.

2.10.5.3 Condition flags

This instruction does not change the flags.

2.10.5.4 Examples

```
ISB  ; Instruction Synchronisation Barrier
```


© NXP B.V. 2010. All rights reserved.

2.10.6 MRS

Move the contents of a special register to a general-purpose register.

2.10.6.1 Syntax

`MRS`{*cond*} *Rd*, *spec_reg*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rd is the destination register.

spec_reg can be any of: `APSR`, `IPSR`, `EPSR`, `IEPSR`, `IAPSR`, `EAPSR`, `PSR`, `MSP`, `PSP`, `PRIMASK`, `BASEPRI`, `BASEPRI_MAX`, `FAULTMASK`, or `CONTROL`.

2.10.6.2 Operation

Use `MRS` in combination with `MSR` as part of a read-modify-write sequence for updating a PSR, for example to clear the Q flag.

In process swap code, the programmers model state of the process being swapped out must be saved, including relevant PSR contents. Similarly, the state of the process being swapped in must also be restored. These operations use `MRS` in the state-saving instruction sequence and `MSR` in the state-restoring instruction sequence.

Remark: `BASEPRI_MAX` is an alias of `BASEPRI` when used with the `MRS` instruction.

See Section 34–2.10.7.

2.10.6.3 Restrictions

Rd must not be SP and must not be PC.

2.10.6.4 Condition flags

This instruction does not change the flags.

2.10.6.5 Examples

```
MRS  R0, PRIMASK ; Read PRIMASK value and write it to R0
```


© NXP B.V. 2010. All rights reserved.

2.10.7 MSR

Move the contents of a general-purpose register into the specified special register.

2.10.7.1 Syntax

`MSR`{*cond*} *spec_reg*, *Rn*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

Rn is the source register.

spec_reg can be any of: `APSR`, `IPSR`, `EPSR`, `IEPSR`, `IAPSR`, `EAPSR`, `PSR`, `MSP`, `PSP`, `PRIMASK`, `BASEPRI`, `BASEPRI_MAX`, `FAULTMASK`, or `CONTROL`.

2.10.7.2 Operation

The register access operation in `MSR` depends on the privilege level. Unprivileged software can only access the `APSR`, see Table 34–629 “APSR bit assignments”. Privileged software can access all special registers.

In unprivileged software writes to unallocated or execution state bits in the `PSR` are ignored.

Note

When you write to `BASEPRI_MAX`, the instruction writes to `BASEPRI` only if either:

- *Rn* is non-zero and the current `BASEPRI` value is 0
- *Rn* is non-zero and less than the current `BASEPRI` value.

See Section 34–2.10.6.

2.10.7.3 Restrictions

Rn must not be SP and must not be PC.

2.10.7.4 Condition flags

This instruction updates the flags explicitly based on the value in *Rn*.

2.10.7.5 Examples

```
MSR  CONTROL, R1 ; Read R1 value and write it to the CONTROL register
```


© NXP B.V. 2010. All rights reserved.

2.10.8 NOP

No Operation.

2.10.8.1 Syntax

NOP{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.10.8.2 Operation

`NOP` does nothing. `NOP` is not necessarily a time-consuming `NOP`. The processor might remove it from the pipeline before it reaches the execution stage.

Use `NOP` for padding, for example to place the following instruction on a 64-bit boundary.

2.10.8.3 Condition flags

This instruction does not change the flags.

2.10.8.4 Examples

```
NOP  ; No operation
```


© NXP B.V. 2010. All rights reserved.

2.10.9 SEV

Send Event.

2.10.9.1 Syntax

SEV{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.10.9.2 Operation

`SEV` is a hint instruction that causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register to 1, see Section 34–3.5 “Power management”.

2.10.9.3 Condition flags

This instruction does not change the flags.

2.10.9.4 Examples

```
SEV ; Send Event
```


© NXP B.V. 2010. All rights reserved.

2.10.10 SVC

Supervisor Call.

2.10.10.1 Syntax

`SVC`{*cond*} #*imm*

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

imm is an expression evaluating to an integer in the range 0-255 (8-bit value).

2.10.10.2 Operation

The `SVC` instruction causes the `SVC` exception.

imm is ignored by the processor. If required, it can be retrieved by the exception handler to determine what service is being requested.

2.10.10.3 Condition flags

This instruction does not change the flags.

2.10.10.4 Examples

```
SVC  0x32  ; Supervisor Call (SVC handler can extract the immediate value
           ; by locating it via the stacked PC)
```


© NXP B.V. 2010. All rights reserved.

2.10.11 WFE

Wait For Event.

2.10.11.1 Syntax

WFE{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”

2.10.11.2 Operation

`WFE` is a hint instruction.

If the event register is 0, `WFE` suspends execution until one of the following events occurs:

- an exception, unless masked by the exception mask registers or the current priority level
- an exception enters the Pending state, if `SEVONPEND` in the System Control Register is set
- a Debug Entry request, if Debug is enabled
- an event signaled by a peripheral or another processor in a multiprocessor system using the `SEV` instruction.

If the event register is 1, `WFE` clears it to 0 and returns immediately.

For more information see Section 34–3.5 “Power management”.

2.10.11.3 Condition flags

This instruction does not change the flags.

2.10.11.4 Examples

```
WFE  ; Wait for event
```


© NXP B.V. 2010. All rights reserved.

2.10.12 WFI

Wait for Interrupt.

2.10.12.1 Syntax

WFI{*cond*}

where:

cond is an optional condition code, see Section 34–2.3.7 “Conditional execution”.

2.10.12.2 Operation

`WFI` is a hint instruction that suspends execution until one of the following events occurs:

- an exception
- a Debug Entry request, regardless of whether Debug is enabled.

2.10.12.3 Condition flags

This instruction does not change the flags.

2.10.12.4 Examples

```
WFI ; Wait for interrupt
```


© NXP B.V. 2010. All rights reserved.

3. ARM Cortex-M3 User Guide: Processor

3.1 Programmers model

This section describes the Cortex-M3 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and privilege levels for software execution and stacks.

3.1.1 Processor mode and privilege levels for software execution

The processor **modes** are:

- Thread mode

  Used to execute application software. The processor enters Thread mode when it comes out of reset.
- Handler mode

  Used to handle exceptions. The processor returns to Thread mode when it has finished exception processing.

The **privilege levels** for software execution are:

- Unprivileged

  The software:

  – has limited access to the `MSR` and `MRS` instructions, and cannot use the `CPS` instruction

  – cannot access the system timer, NVIC, or system control block

  – might have restricted access to memory or peripherals.

  Unprivileged software executes at the unprivileged level.
- Privileged

  The software can use all the instructions and has access to all resources.

  Privileged software executes at the privileged level.

In Thread mode, the CONTROL register controls whether software execution is privileged or unprivileged, see Table 34–635. In Handler mode, software execution is always privileged.

Only privileged software can write to the CONTROL register to change the privilege level for software execution in Thread mode. Unprivileged software can use the `SVC` instruction to make a **supervisor call** to transfer control to privileged software.

3.1.2 Stacks

The processor uses a full descending stack. This means the stack pointer indicates the last stacked item on the stack memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the **main stack** and the **process stack**, with independent copies of the stack pointer, see Section 34–3.1.3.2.

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see Table 34–635. In Handler mode, the processor always uses the main stack. The options for processor operations are:

© NXP B.V. 2010. All rights reserved.

Table 626. Summary of processor mode, execution privilege level, and stack use options

Processor mode	Used to execute	Privilege level for software execution	Stack used
Thread	Applications	Privileged or unprivileged[1]	Main stack or process stack[1]
Handler	Exception handlers	Always privileged	Main stack

[1] See Table 34–635.

3.1.3 Core registers

The processor core registers are:

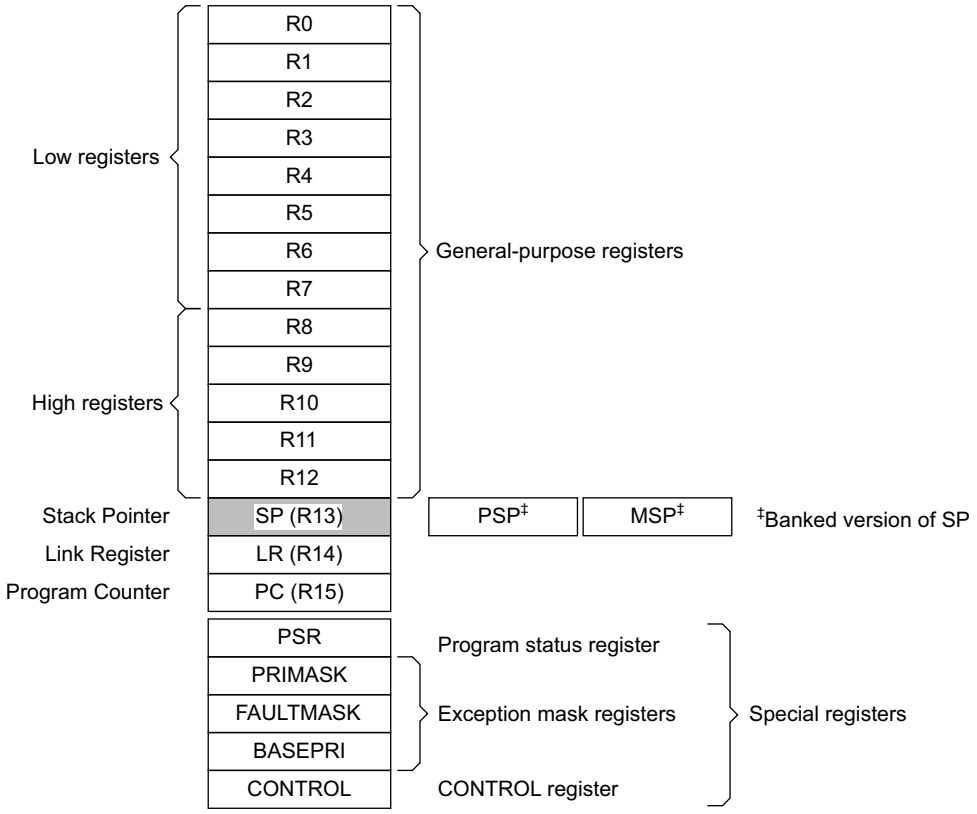


Table 627. Core register set summary

Name	Type[1]	Required privilege[2]	Reset value	Description
R0-R12	RW	Either	Undefined	Section 34–3.1.3.1
MSP	RW	Privileged	See description	Section 34–3.1.3.2
PSP	RW	Either	Undefined	Section 34–3.1.3.2
LR	RW	Either	0xFFFFFFFF	Section 34–3.1.3.3
PC	RW	Either	See description	Section 34–3.1.3.4
PSR	RW	Privileged	0x01000000	Section 34–3.1.3.5
ASPR	RW	Either	0x00000000	Table 34–629
IPSR	RO	Privileged	0x00000000	Table 34–630

© NXP B.V. 2010. All rights reserved.

Table 627. Core register set summary

Name	Type[1]	Required privilege[2]	Reset value	Description
EPSR	RO	Privileged	0x01000000	Table 34–631
PRIMASK	RW	Privileged	0x00000000	Table 34–632
FAULTMASK	RW	Privileged	0x00000000	Table 34–633
BASEPRI	RW	Privileged	0x00000000	Table 34–634
CONTROL	RW	Privileged	0x00000000	Table 34–635

[1] Describes access type during program execution in thread mode and Handler mode. Debug access can differ.

[2] An entry of Either means privileged and unprivileged software can access the register.

3.1.3.1 General-purpose registers

R0-R12 are 32-bit general-purpose registers for data operations.

3.1.3.2 Stack Pointer

The **Stack Pointer** (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

On reset, the processor loads the MSP with the value from address 0x00000000.

- 0 = **Main Stack Pointer** (MSP). This is the reset value.
- 1 = **Process Stack Pointer** (PSP).

3.1.3.3 Link Register

The **Link Register** (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the processor loads the LR value 0xFFFFFFFF.

3.1.3.4 Program Counter

The **Program Counter** (PC) is register R15. It contains the current program address. Bit[0] is always 0 because instruction fetches must be halfword aligned. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004.

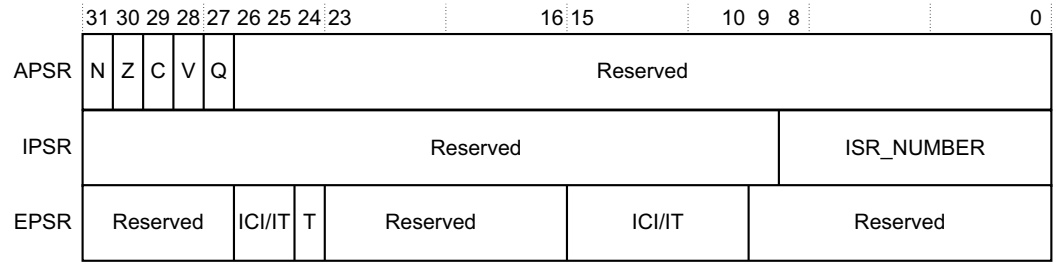
3.1.3.5 Program Status Register

The **Program Status Register** (PSR) combines:

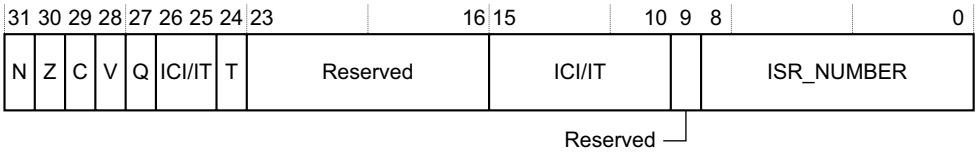
- **Application Program Status Register** (APSR)
- **Interrupt Program Status Register** (IPSR)
- **Execution Program Status Register** (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR. The bit assignments are:

© NXP B.V. 2010. All rights reserved.



The PSR bit assignments are:



Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the `MSR` or `MRS` instructions. For example:

- read all of the registers using `PSR` with the `MRS` instruction
- write to the APSR using `APSR` with the `MSR` instruction.

The PSR combinations and attributes are:

Table 628. PSR register combinations

Register	Type	Combination
PSR	RW[1][2]	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	RW[1]	APSR and IPSR
EAPSR	RW[2]	APSR and EPSR

[1] The processor ignores writes to the IPSR bits.

[2] Reads of the EPSR bits return zero, and the processor ignores writes to the these bits

See the instruction descriptions Section 34–2.10.6 “MRS” and Section 34–2.10.7 “MSR” for more information about how to access the program status registers.

Application Program Status Register: The APSR contains the current state of the condition flags from previous instruction executions. See the register summary in Table 34–627 for its attributes. The bit assignments are:

© NXP B.V. 2010. All rights reserved.

Table 629. APSR bit assignments

Bits	Name	Function
[31]	N	Negative or less than flag: 0 = operation result was positive, zero, greater than, or equal 1 = operation result was negative or less than.
[30]	Z	Zero flag: 0 = operation result was not zero 1 = operation result was zero.
[29]	C	Carry or borrow flag: 0 = add operation did not result in a carry bit or subtract operation resulted in a borrow bit 1 = add operation resulted in a carry bit or subtract operation did not result in a borrow bit.
[28]	V	Overflow flag: 0 = operation did not result in an overflow 1 = operation resulted in an overflow.
[27]	Q	Sticky saturation flag: 0 = indicates that saturation has not occurred since reset or since the bit was last cleared to zero 1 = indicates when an `SSAT` or `USAT` instruction results in saturation. This bit is cleared to zero by software using an `MRS` instruction.
[26:0]	-	Reserved.

Interrupt Program Status Register: The IPSR contains the exception type number of the current **Interrupt Service Routine** (ISR). See the register summary in Table 34–627 for its attributes. The bit assignments are:

© NXP B.V. 2010. All rights reserved.

Table 630. IPSR bit assignments

Bits	Name	Function
[31:9]	-	Reserved
[8:0]	ISR_NUMBER	This is the number of the current exception: 0 = Thread mode 1 = Reserved 2 = NMI 3 = Hard fault 4 = Memory management fault 5 = Bus fault 6 = Usage fault 7-10 = Reserved 11 = SVCall 12 = Reserved for Debug 13 = Reserved 14 = PendSV 15 = SysTick 16 = IRQ0 17 = IRQ1, first device specific interrupt . . 255 = IRQ243 (last implemented interrupt depends on device) see Section 34–3.3.2 for more information.

Execution Program Status Register: The EPSR contains the Thumb state bit, and the execution state bits for either the:

- **If-Then** (IT) instruction
- **Interruptible-Continuable Instruction** (ICI) field for an interrupted load multiple or store multiple instruction.

See the register summary in Table 34–627 for the EPSR attributes. The bit assignments are:

Table 631. EPSR bit assignments

Bits	Name	Function
[31:27]	-	Reserved.
[26:25], [15:10]	ICI	Interruptible-continuable instruction bits, see Section 34–.
[26:25], [15:10]	IT	Indicates the execution state bits of the `IT` instruction, see Section 34–2.9.3 “IT”.
[24]	T	Always set to 1.
[23:16]	-	Reserved.
[9:0]	-	Reserved.

© NXP B.V. 2010. All rights reserved.

Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are ignored. Fault handlers can examine EPSR value in the stacked PSR to indicate the operation that is at fault. See Section 34–3.3.7

Interruptible-continuable instructions: When an interrupt occurs during the execution of an LDM or STM instruction, the processor:

After servicing the interrupt, the processor:

- stops the load multiple or store multiple instruction operation temporarily.
- stores the next register operand in the multiple operation to EPSR bits[15:12].
- returns to the register pointed to by bits[15:12].
- resumes execution of the multiple load or store instruction.

When the EPSR holds ICI execution state, bits[26:25,11:10] are zero.

If-Then block: The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See Section 34–2.9.3 “IT” for more information.

3.1.3.6 Exception mask registers

The exception mask registers disable the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks.

To access the exception mask registers use the MSR and MRS instructions, or the CPS instruction to change the value of PRIMASK or FAULTMASK. See Section 34–2.10.6 “MRS”, Section 34–2.10.7 “MSR”, and Section 34–2.10.2 “CPS” for more information.

Priority Mask Register: The PRIMASK register prevents activation of all exceptions with configurable priority. See the register summary in Table 34–627 for its attributes. The bit assignments are shown in Table 34–632.

Table 632. PRIMASK register bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	PRIMASK	0 = no effect
		1 = prevents the activation of all exceptions with configurable priority.

Fault Mask Register: The FAULTMASK register prevents activation of all exceptions except for **Non-Maskable Interrupt** (NMI). See the register summary in Table 34–627 for its attributes. The bit assignments are shown in Table 34–633.

Table 633. FAULTMASK register bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	FAULTMASK	0 = no effect
		1 = prevents the activation of all exceptions except for NMI.

The processor clears the FAULTMASK bit to 0 on exit from any exception handler except the NMI handler.

© NXP B.V. 2010. All rights reserved.

Base Priority Mask Register: The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with same or lower priority level as the BASEPRI value. See the register summary in Table 34–627 for its attributes. The bit assignments are shown in Table 34–634.

Table 634. BASEPRI register bit assignments

Bits	Name	Function
[31:8]	-	Reserved
[7:0]	BASEPRI[1]	Priority mask bits: 0x0000 = no effect Nonzero = defines the base priority for exception processing. The processor does not process any exception with a priority value greater than or equal to BASEPRI.

[1] This field is similar to the priority fields in the interrupt priority registers. The processor implements only bits[7:M] of this field, bits[M-1:0] read as zero and ignore writes. The value of M depends on the specific device. See Section 34–4.2.7 “Interrupt Priority Registers” for more information. Remember that higher priority field values correspond to lower exception priorities.

3.1.3.7 CONTROL register

The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode. See the register summary in Table 34–627 for its attributes. The bit assignments are shown in Table 34–635.

Table 635. CONTROL register bit assignments

Bits	Name	Function
[31:2]	-	Reserved
[1]	Active stack pointer	Defines the current stack: 0 = MSP is the current stack pointer 1 = PSP is the current stack pointer. In Handler mode this bit reads as zero and ignores writes.
[0]	Thread mode privilege level	Defines the Thread mode privilege level: 0 = privileged 1 = unprivileged.

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms update the CONTROL register.

In an OS environment, ARM recommends that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, use the MSR instruction to set the Active stack pointer bit to 1, see Section 34–2.10.7 “MSR”.

Remark: When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB execute using the new stack pointer. See Section 34–2.10.5 “ISB”

© NXP B.V. 2010. All rights reserved.

3.1.4 Exceptions and interrupts

The Cortex-M3 processor supports interrupts and system exceptions. The processor and the **Nested Vectored Interrupt Controller** (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See Section 34–3.3.7.1 and Section 34–3.3.7.2 for more information.

The NVIC registers control interrupt handling. See Section 34–4.2 “Nested Vectored Interrupt Controller” for more information.

3.1.5 Data types

The processor:

- supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- supports 64-bit data transfer instructions.
- manages all data memory accesses as little-endian. See Section 34–3.2.1 for more information.

3.1.6 The Cortex Microcontroller Software Interface Standard

For a Cortex-M3 microcontroller system, the **Cortex Microcontroller Software Interface Standard** (CMSIS) defines:

- a common way to:
  - access peripheral registers
  - define exception vectors
- the names of:
  - the registers of the core peripherals
  - the core exception vectors
- a device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M3 processor. It also includes optional interfaces for middleware components comprising a TCP/IP stack and a Flash file system.

CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Remark: This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

© NXP B.V. 2010. All rights reserved.

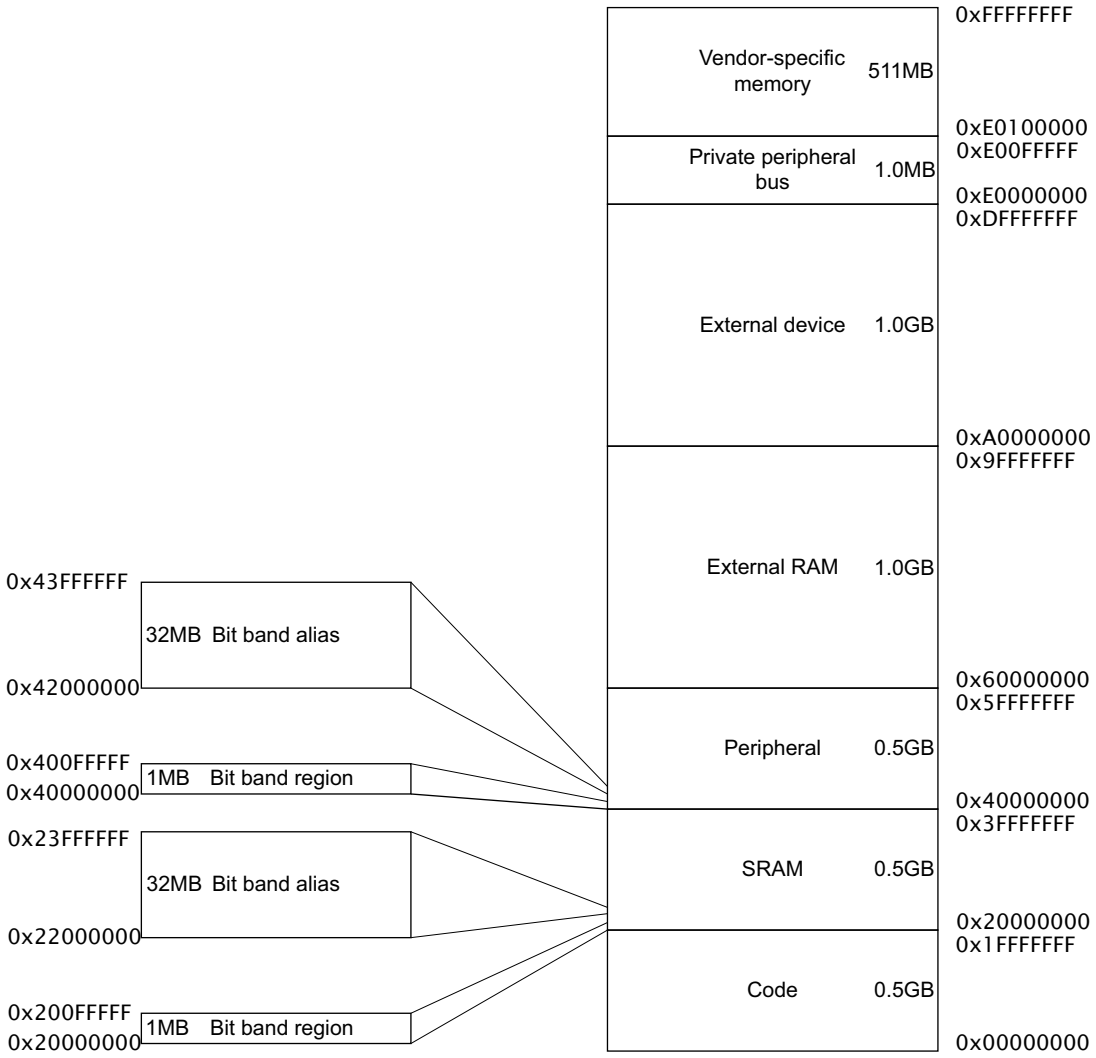
The following sections give more information about the CMSIS:

- Section 34–3.5.4
- Section 34–2.2 “Intrinsic functions”
- Section 34–4.2.1 “The CMSIS mapping of the Cortex-M3 NVIC registers”
- Section 34–4.2.10.1 “NVIC programming hints”.

© NXP B.V. 2010. All rights reserved.

3.2 Memory model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4GB of addressable memory. The memory map is:



The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data, see Section 34–3.2.5.

The processor reserves regions of the **Private peripheral bus** (PPB) address range for core peripheral registers, see Section 34–4.1 “About the Cortex-M3 peripherals”.

3.2.1 Memory regions, types and attributes

The memory map and the programming of the MPU split the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

© NXP B.V. 2010. All rights reserved.

- Normal: The processor can re-order transactions for efficiency, or perform speculative reads.
- Device: The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory.
- Strongly-ordered: The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.

The additional memory attributes include.

- **Shareable**

  For a shareable memory region, the memory system provides data synchronization between bus masters in a system with multiple bus masters, for example, a processor with a DMA controller.

  Strongly-ordered memory is always shareable.

  If multiple bus masters can access a non-shareable memory region, software must ensure data coherency between the bus masters.
- **Execute Never** (XN)

  Means the processor prevents instruction accesses. Any attempt to fetch an instruction from an XN region causes a memory management fault exception.

3.2.2 Memory system ordering of memory accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing this does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions, see Section 34–3.2.4.

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is:

A1 \ A2	Normal access	Device access		Strongly-ordered access
		Non-shareable	Shareable	
Normal access	-	-	-	-
Device access, non-shareable	-	<	-	<
Device access, shareable	-	-	<	<
Strongly-ordered access	-	<	<	<

Where:

‘—’ means that the memory system does not guarantee the ordering of the accesses.

© NXP B.V. 2010. All rights reserved.

'<' means that accesses are observed in program order, that is, A1 is always observed before A2.

3.2.3 Behavior of memory accesses

The behavior of accesses to each region in the memory map is:

Table 636. Memory access behavior

Address range	Memory region	Memory type	XN	Description
0x00000000 - 0x1FFFFFFF	Code	Normal[1]	-	Executable region for program code. You can also put data here.
0x20000000 - 0x3FFFFFFF	SRAM	Normal[1]	-	Executable region for data. You can also put code here. This region includes bit band and bit band alias areas, see Table 34–637.
0x40000000 - 0x5FFFFFFF	Peripheral	Device[1]	XN[1]	This region includes bit band and bit band alias areas, see Table 34–638.
0x60000000 - 0x9FFFFFFF	External RAM	Normal[1]	-	Executable region for data.
0xA0000000 - 0xDFFFFFFF	External device	Device[1]	XN[1]	External Device memory
0xE0000000 - 0xE00FFFFF	Private Peripheral Bus	Strongly-ordered[1]	XN[1]	This region includes the NVIC, System timer, and system control block.
0xE0100000 - 0xFFFFFFFF	Vendor-specific device	Device[1]	XN[1]	Not used for NXP devices.

[1] See Section 34–3.2.1 for more information.

The Code, SRAM, and external RAM regions can hold programs. However, the most efficient access to programs is from the Code region. This is because the processor has separate buses that enable instruction fetches and data accesses to occur simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see Section 34–4.5 "Memory protection unit".

3.2.4 Software ordering of memory accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- the processor has multiple bus interfaces
- memory or devices in the memory map have different wait states
- some memory accesses are buffered or speculative.

Section 34–3.2.2 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

© NXP B.V. 2010. All rights reserved.

- DMB

  The **Data Memory Barrier** (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. See Section 34–2.10.3 “DMB”.

- DSB

  The **Data Synchronization Barrier** (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute. See Section 34–2.10.4 “DSB”.

- ISB

  The **Instruction Synchronization Barrier** (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. See Section 34–2.10.5 “ISB”.

Use memory barrier instructions in, for example:

Memory accesses to Strongly-ordered memory, such as the system control block, do not require the use of DMB instructions.

- MPU programming:
  - Use a DSB instruction to ensure the effect of the MPU takes place immediately at the end of context switching.
  - Use an ISB instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an ISB instruction is not required.
- Vector table. If the program changes an entry in the vector table, and then enables the corresponding exception, use a DMB instruction between the operations. This ensures that if the exception is taken immediately after being enabled the processor uses the new exception vector.
- Self-modifying code. If a program contains self-modifying code, use an ISB instruction immediately after the code modification in the program. This ensures subsequent instruction execution uses the updated program.
- Memory map switching. If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. This ensures subsequent instruction execution uses the updated memory map.
- Dynamic exception priority change. When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. This ensures the change takes effect on completion of the DSB instruction.
- Using a semaphore in multi-master system. If the system contains more than one bus master, for example, if another processor is present in the system, each processor must use a DMB instruction after any semaphore instructions, to ensure other bus masters see the memory transactions in the order in which they were executed.

3.2.5 Bit-banding

A bit-band region maps each word in a **bit-band alias** region to a single bit in the **bit-band region**. The bit-band regions occupy the lowest 1MB of the SRAM and peripheral memory regions.

© NXP B.V. 2010. All rights reserved.

The memory map has two 32MB alias regions that map to two 1MB bit-band regions:

- accesses to the 32MB SRAM alias region map to the 1MB SRAM bit-band region, as shown in Table 34–637
- accesses to the 32MB peripheral alias region map to the 1MB peripheral bit-band region, as shown in Table 34–638.

Table 637. SRAM memory bit-banding regions

Address range	Memory region	Instruction and data accesses
0x20000000 - 0x200FFFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x22000000 - 0x23FFFFF0	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 638. Peripheral memory bit-banding regions

Address range	Memory region	Instruction and data accesses
0x40000000 - 0x400FFFFF	Peripheral bit-band alias	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
0x42000000 - 0x44FFFFF	Peripheral bit-band region	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

Remark: A word access to the SRAM or peripheral bit-band alias regions map to a single bit in the SRAM or peripheral bit-band region.

The following formula shows how the alias region maps onto the bit-band region:

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)

bit_word_addr = bit_band_base + bit_word_offset
```

where:

- Bit_word_offset is the position of the target bit in the bit-band memory region.
- Bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- Bit_band_base is the starting address of the alias region.
- Byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
- Bit_number is the bit position, 0-7, of the targeted bit.

Figure 34–145 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FFFFE0 maps to bit[0] of the bit-band byte at 0x200FFFFF: 0x23FFFFE0 = 0x22000000 + (0xFFFFF*32) + (0*4).

© NXP B.V. 2010. All rights reserved.

- The alias word at 0x23FFFFFC maps to bit[7] of the bit-band byte at 0x200FFFFF: 0x23FFFFFC = 0x22000000 + (0xFFFFF*32) + (7*4).
- The alias word at 0x22000000 maps to bit[0] of the bit-band byte at 0x20000000: 0x22000000 = 0x22000000 + (0*32) + (0 *4).
- The alias word at 0x2200001C maps to bit[7] of the bit-band byte at 0x20000000: 0x2200001C = 0x22000000+ (0*32) + (7*4).

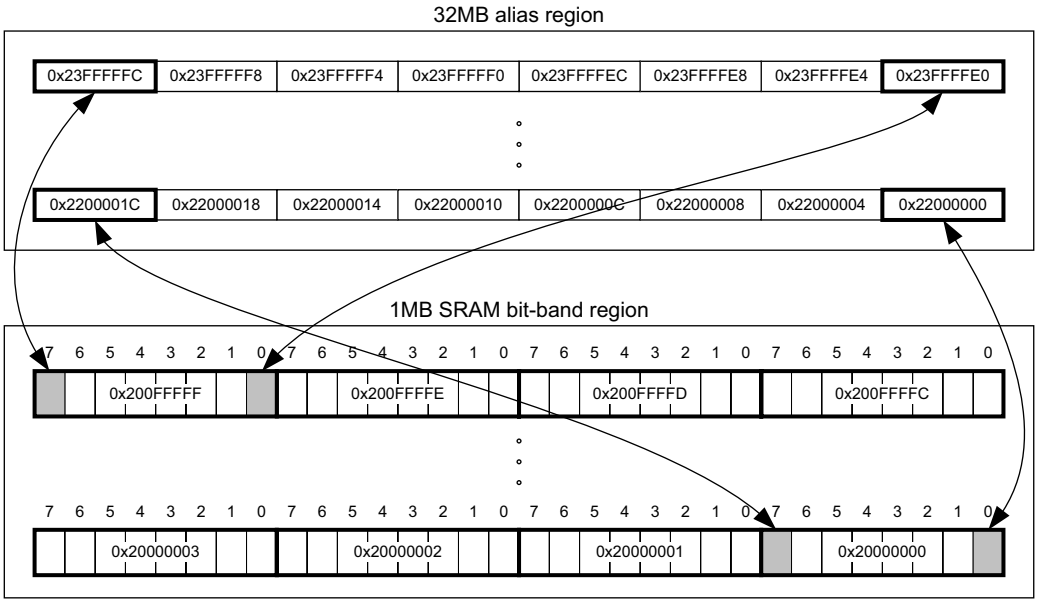


Fig 145. Bit-band mapping

3.2.5.1 Directly accessing an alias region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit[0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit[0] set to 1 writes a 1 to the bit-band bit, and writing a value with bit[0] set to 0 writes a 0 to the bit-band bit.

Bits[31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

Reading a word in the alias region:

- 0x00000000 indicates that the targeted bit in the bit-band region is set to zero
- 0x00000001 indicates that the targeted bit in the bit-band region is set to 1

3.2.5.2 Directly accessing a bit-band region

Section 34–3.2.3 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

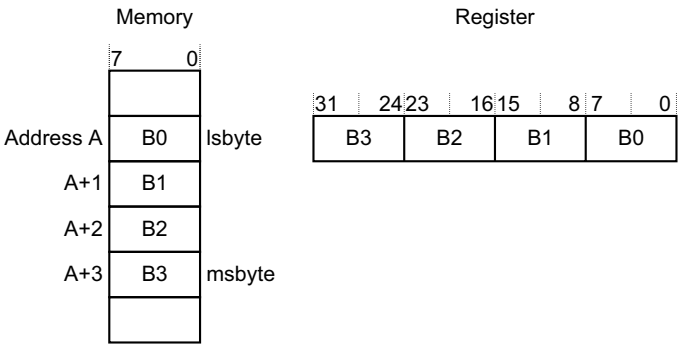
© NXP B.V. 2010. All rights reserved.

3.2.6 Memory endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Section 34–3.2.6.1 describes how words of data are stored in memory.

3.2.6.1 Little-endian format

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. For example:



3.2.7 Synchronization primitives

The Cortex-M3 instruction set includes pairs of **synchronization primitives**. These provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use them to perform a guaranteed read-modify-write memory update sequence, or for a semaphore mechanism.

A pair of synchronization primitives comprises:

- A Load-Exclusive instruction

  Used to read the value of a memory location, requesting exclusive access to that location.
- A Store-Exclusive instruction

  Used to attempt to write to the same memory location, returning a status bit to a register. If this bit is:
  - 0: it indicates that the thread or process gained exclusive access to the memory, and the write succeeds,
  - 1: it indicates that the thread or process did not gain exclusive access to the memory, and no write is performed,

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- the word instructions `LDREX` and `STREX`
- the halfword instructions `LDREXH` and `STREXH`
- the byte instructions `LDREXB` and `STREXB`.

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.

© NXP B.V. 2010. All rights reserved.

To perform a guaranteed read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Update the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location, and tests the returned status bit. If this bit is:
   - 0: The read-modify-write completed successfully,
   - 1: No write was performed. This indicates that the value returned at step 1 might be out of date. The software must retry the read-modify-write sequence,

Software can use the synchronization primitives to implement a semaphores as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

The Cortex-M3 includes an exclusive access monitor, that tags the fact that the processor has executed a Load-Exclusive instruction.

The processor removes its exclusive access tag if:

- It executes a `CLREX` instruction
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs. This means the processor can resolve semaphore conflicts between different threads.

For more information about the synchronization primitive instructions, see Section 34–2.4.8 “LDREX and STREX” and Section 34–2.4.9.

3.2.8 Programming hints for the synchronization primitives

ANSI C cannot directly generate the exclusive access instructions. Some C compilers provide intrinsic functions for generation of these instructions:

Table 639. C compiler intrinsic functions for exclusive access instructions

Instruction	Intrinsic function
`LDREX`, `LDREXH`, or `LDREXB`	`unsigned int __ldrex(volatile void *ptr)`
`STREX`, `STREXH`, or `STREXB`	`int __strex(unsigned int val, volatile void *ptr)`
`CLREX`	`void __clrex(void)`

The actual exclusive access instruction generated depends on the data type of the pointer passed to the intrinsic function. For example, the following C code generates the require `LDREXB` operation:

```
__ldrex((volatile char *) 0xFF);
```


© NXP B.V. 2010. All rights reserved.

3.3 Exception model

This section describes the exception model.

3.3.1 Exception states

Each exception is in one of the following states:

- Inactive

  The exception is not active and not pending.

- Pending

  The exception is waiting to be serviced by the processor.

  An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.

- Active

  An exception that is being serviced by the processor but has not completed.

  Remark: An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.

- Active and pending

  The exception is being serviced by the processor and there is a pending exception from the same source.

3.3.2 Exception types

The exception types are:

- Reset

  Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.

- NMI

  A **NonMaskable Interrupt** (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2. NMIs cannot be:

  – masked or prevented from activation by any other exception

  – preempted by any exception other than Reset.

- Hard fault

  A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.

- Memory management fault

© NXP B.V. 2010. All rights reserved.

A memory management fault is an exception that occurs because of a memory protection related fault. The MPU or the fixed memory protection constraints determines this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to **Execute Never** (XN) memory regions, even if the MPU is disabled.

- Bus fault

  A bus fault is an exception that occurs because of a memory related fault for an instruction or data memory transaction. This might be from an error detected on a bus in the memory system.

- Usage fault

  A usage fault is an exception that occurs because of a fault related to instruction execution. This includes:

  – an undefined instruction
  – an illegal unaligned access
  – invalid state on instruction execution
  – an error on exception return.

  The following can cause a usage fault when the core is configured to report them:

  – an unaligned address on word and halfword memory access
  – division by zero.

- SVCall

  A **supervisor call** (SVC) is an exception that is triggered by the `SVC` instruction. In an OS environment, applications can use `SVC` instructions to access OS kernel functions and device drivers.

- PendSV

  PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active.

- SysTick

  A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick.

- Interrupt (IRQ)

  A interrupt, or IRQ, is an exception signalled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.

Table 640. Properties of the different exception types

Exception number[1]	IRQ number[1]	Exception type	Priority	Vector address or offset[2]	Activation
1	-	Reset	-3, the highest	`0x00000004`	Asynchronous
2	-14	NMI	-2	`0x00000008`	Asynchronous
3	-13	Hard fault	-1	`0x0000000C`	-
4	-12	Memory management fault	Configurable[3]	`0x00000010`	Synchronous

© NXP B.V. 2010. All rights reserved.

Table 640. Properties of the different exception types

Exception number[1]	IRQ number[1]	Exception type	Priority	Vector address or offset[2]	Activation
5	-11	Bus fault	Configurable[3]	0x00000014	Synchronous when precise, asynchronous when imprecise
6	-10	Usage fault	Configurable[3]	0x00000018	Synchronous
7-10	-	-	-	Reserved	-
11	-5	SVCall	Configurable[3]	0x0000002C	Synchronous
12-13	-	-	-	Reserved	-
14	-2	PendSV	Configurable[3]	0x00000038	Asynchronous
15	-1	SysTick	Configurable[3]	0x0000003C	Asynchronous
16 and above	0 and above	Interrupt (IRQ)	Configurable[4]	0x00000040 and above[5]	Asynchronous

[1] To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see Table 34–630.

[2] See Section 34–3.3.4 for more information.

[3] See Section 34–4.3.9 “System Handler Priority Registers”.

[4] See Section 34–4.2.7 “Interrupt Priority Registers”.

[5] Increasing in steps of 4.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that Table 34–640 shows as having configurable priority, see:

- Section 34–4.3.10 “System Handler Control and State Register”
- Section 34–4.2.3 “Interrupt Clear-enable Registers”.

For more information about hard faults, memory management faults, bus faults, and usage faults, see Section 34–3.4.

3.3.3 Exception handlers

The processor handles exceptions using:

- Interrupt Service Routines (ISRs)

  Interrupts IRQ0 and up are the exceptions handled by ISRs.
- Fault handlers

  Hard fault, memory management fault, usage fault, bus fault are fault exceptions handled by the fault handlers.
- System handlers

  NMI, PendSV, SVCall SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

© NXP B.V. 2010. All rights reserved.

3.3.4 Vector table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. Figure 34–146 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code. Note that the upper limit of the IRQ number is device dependent.

Exception number	IRQ number	Offset	Vector
127	111	0x1FC	IRQ111
.		.	.
.		.	.
.		.	.
18		0x004C	IRQ2
17	2	0x0048	IRQ1
16	1	0x0044	IRQ0
15	0	0x0040	Systick
14	-1	0x003C	PendSV
13	-2	0x0038	Reserved
12			Reserved for debug
11			SVCall
10	-5	0x002C	Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value.

Fig 146. Vector table

On system reset, the vector table is fixed at address `0x00000000`. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range `0x00000080` to `0x3FFFFF80`, see Section 34–4.3.5 “Vector Table Offset Register”.

3.3.5 Exception priorities

As Table 34–640 shows, all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset, Hard fault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

© NXP B.V. 2010. All rights reserved.

- Section 34–4.3.9 “System Handler Priority Registers”
- Section 34–4.2.7 “Interrupt Priority Registers”.

Remark: Configurable priority values are in the range 0 to 31. This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

3.3.6 Interrupt priority grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler,

- an upper field that defines the **group priority**
- a lower field that defines a **subpriority** within the group.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see Section 34–4.3.6 “Application Interrupt and Reset Control Register”.

3.3.7 Exception entry and return

Descriptions of exception handling use the following terms:

- Preemption

  When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See Section 34–3.3.6 for more information about preemption by an interrupt.

  When one exception preempts another, the exceptions are called nested exceptions. See Section 34–3.3.7.1 more information.

- Return

  This occurs when the exception handler is completed, and:

© NXP B.V. 2010. All rights reserved.

- there is no pending exception with sufficient priority to be serviced
- the completed exception handler was not handling a late-arriving exception.

The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See Section 34–3.3.7.2 for more information.

- Tail-chaining

  This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

- Late-arriving

  This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

3.3.7.1 Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

- the processor is in Thread mode
- the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers, see Section 34–3.1.3.6. An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred as **stacking** and the structure of eight data words is referred as **stack frame**. The stack frame contains the following information:

- R0-R3, R12
- Return address
- PSR
- LR.

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. Unless stack alignment is disabled, the stack frame is aligned to a double-word address. If the STKALIGN bit of the **Configuration Control Register** (CCR) is set to 1, stack align adjustment is performed during stacking.

© NXP B.V. 2010. All rights reserved.

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the was processor was in before the entry occurred.

If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

3.3.7.2 Exception return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the PC:

- a `POP` instruction that includes the PC
- a `BX` instruction with any register.
- an `LDR` or `LDM` instruction with the PC as the destination.

EXC_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest four bits of this value provide information on the return stack and processor mode. Table 34–641 shows the EXC_RETURN[3:0] values with a description of the exception return behavior.

The processor sets EXC_RETURN bits[31:4] to `0xFFFFFFF`. When this value is loaded into the PC it indicates to the processor that the exception is complete, and the processor initiates the exception return sequence.

Table 641. Exception return behavior

EXC_RETURN[3:0]	Description
bXXX0	Reserved.
b0001	Return to Handler mode. Exception return gets state from MSP. Execution uses MSP after return.
b0011	Reserved.
b01X1	Reserved.

© NXP B.V. 2010. All rights reserved.

Table 641. Exception return behavior

EXC_RETURN[3:0]	Description
b1001	Return to Thread mode. Exception return gets state from MSP. Execution uses MSP after return.
b1101	Return to Thread mode. Exception return gets state from PSP. Execution uses PSP after return.
b1X11	Reserved.

© NXP B.V. 2010. All rights reserved.

3.4 Fault handling

Faults are a subset of the exceptions, see Section 34–3.3. The following generate a fault:

- a bus error on:
  - an instruction fetch or vector table load
  - a data access
- an internally-detected error such as an undefined instruction or an attempt to change state with a `BX` instruction
- attempting to execute an instruction from a memory region marked as **Non-Executable** (XN)
- an MPU fault because of a privilege violation or an attempt to access an unmanaged region

3.4.1 Fault types

Table 34–642 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates that the fault has occurred. See Section 34–4.3.11 “Configurable Fault Status Register” for more information about the fault status registers.

Table 642. Faults

Fault	Handler	Bit name	Fault status register
Bus error on a vector read	Hard fault	VECTTBL	Section 34–4.3.12 “Hard Fault Status Register”
Fault escalated to a hard fault		FORCED	
MPU mismatch:	Memory management fault	-	-
on instruction access		IACCVIOL[1]	Section 34–4.3.13 “Memory Management Fault Address Register”
on data access		DACCVIOL	
during exception stacking		MSTKERR	
during exception unstacking		MUNSKERR	
Bus error:	Bus fault	-	-
during exception stacking		STKERR	Section 34–4.3.14 “Bus Fault Address Register”
during exception unstacking		UNSTKERR	
during instruction prefetch		IBUSERR	
Precise data bus error		PRECISERR	
Imprecise data bus error		IMPRECISERR	
Attempt to access a coprocessor	Usage fault	NOCP	Section 34–4.3.11.3 “Usage Fault Status Register”
Undefined instruction		UNDEFINSTR	
Attempt to enter an invalid instruction set state[2]		INVSTATE	
Invalid EXC_RETURN value		INVPC	
Illegal unaligned load or store		UNALIGNED	
Divide By 0		DIVBYZERO	

[1] Occurs on an access to an XN region even if the MPU is disabled.

[2] Attempting to use an instruction set other than the Thumb instruction set.

© NXP B.V. 2010. All rights reserved.

3.4.2 Fault escalation and hard faults

All faults exceptions except for hard fault have configurable exception priority, see Section 34–4.3.9 “System Handler Priority Registers”. Software can disable execution of the handlers for these faults, see Section 34–4.3.10 “System Handler Control and State Register”.

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler. as described in Section 34–3.3.

In some situations, a fault with configurable priority is treated as a hard fault. This is called **priority escalation**, and the fault is described as **escalated to hard fault**. Escalation to hard fault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This is because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. This means that if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

Remark: Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault.

3.4.3 Fault status registers and fault address registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 34–643.

Table 643. Fault status and fault address registers

Handler	Status register name	Address register name	Register description
Hard fault	HFSR	-	Section 34–4.3.12 “Hard Fault Status Register”
Memory management fault	MMFSR	MMFAR	Section 34–4.3.13 “Memory Management Fault Address Register” Section 34–4.3.11.1 “Memory Management Fault Status Register”
Bus fault	BFSR	BFAR	Section 34–4.3.11.2 “Bus Fault Status Register” Section 34–4.3.14 “Bus Fault Address Register”
Usage fault	UFSR	-	Section 34–4.3.11.3 “Usage Fault Status Register”

© NXP B.V. 2010. All rights reserved.

3.4.4 Lockup

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in lockup state it does not execute any instructions. The processor remains in lockup state until either:

- it is reset
- an NMI occurs.

Remark: If lockup state occurs from the NMI handler a subsequent NMI does not cause the processor to leave lockup state.

© NXP B.V. 2010. All rights reserved.

3.5 Power management

Note: NXP devices based on the Cortex-M3 processor, including the LPC17xx, support additional reduced power modes. See Section 4–8 “Power control” for information on all available reduced power modes.

The Cortex-M3 processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock
- Deep sleep mode stops the system clock and switches off the PLL and flash memory.

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see Section 34–4.3.7 “System Control Register”. For more information about the behavior of the sleep modes see Section 4–8 “Power control”.

This section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode.

3.5.1 Entering sleep mode

This section describes the mechanisms software can use to put the processor into sleep mode.

The system can generate spurious wakeup events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

3.5.1.1 Wait for interrupt

The **wait for interrupt** instruction, `WFI`, causes immediate entry to sleep mode. When the processor executes a `WFI` instruction it stops executing instructions and enters sleep mode. See Section 34–2.10.12 “WFI” for more information.

3.5.1.2 Wait for event

Note: LPC17xx devices based on the Cortex-M3 processor do not implement external events.

The **wait for event** instruction, `WFE`, causes entry to sleep mode conditional on the value of an one-bit event register. When the processor executes a `WFE` instruction, it checks this register:

- if the register is 0 the processor stops executing instructions and enters sleep mode
- if the register is 1 the processor clears the register to 0 and continues executing instructions without entering sleep mode.

See Section 34–2.10.11 “WFE” for more information.

If the event register is 1, this indicate that the processor must not enter sleep mode on execution of a `WFE` instruction. Typically, this is because an external event signal is asserted, or a processor in the system has executed an `SEV` instruction, see Section 34–2.10.9 “SEV”. Software cannot access this register directly.

© NXP B.V. 2010. All rights reserved.

3.5.1.3 Sleep-on-exit

If the SLEEPONEXIT bit of the SCR is set to 1, when the processor completes the execution of an exception handler it returns to Thread mode and immediately enters sleep mode. Use this mechanism in applications that only require the processor to run when an exception occurs.

3.5.2 Wakeup from sleep mode

The conditions for the processor to wakeup depend on the mechanism that cause it to enter sleep mode.

3.5.2.1 Wakeup from WFI or sleep-on-exit

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry.

Some embedded systems might have to execute system restore tasks after the processor wakes up, and before it executes an interrupt handler. To achieve this set the PRIMASK bit to 1 and the FAULTMASK bit to 0. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor sets PRIMASK to zero. For more information about PRIMASK and FAULTMASK see Section 34–3.1.3.6.

3.5.2.2 Wakeup from WFE

The processor wakes up if:

- it detects an exception with sufficient priority to cause exception entry

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about the SCR see Section 34–4.3.7 “System Control Register”.

3.5.3 The Wakeup Interrupt Controller

The **Wakeup Interrupt Controller** (WIC) is a peripheral that can detect an interrupt and wake the processor from deep sleep mode or Power-down mode. The WIC is enabled only when the DEEPSLEEP bit in the SCR is set to 1, see Section 34–4.3.7 “System Control Register”.

Details of wakeup possibilities on the LPC17xx can be found in Section 4–8 “Power control”.

The WIC is not programmable, and does not have any registers or user interface. It operates entirely from hardware signals.

When the WIC is enabled and the processor enters deep sleep mode or Power-down mode, the power management unit in the system can power down most of the Cortex-M3 processor. This has the side effect of stopping the SysTick timer. When the WIC receives an interrupt, it takes a number of clock cycles to wakeup the processor and restore its state, before it can process the interrupt. This means interrupt latency is increased in deep sleep mode. Wakeup from Power-down mode requires startup of many other portions of the device, and takes longer.

Remark: If the processor detects a connection to a debugger it disables the WIC.

© NXP B.V. 2010. All rights reserved.

3.5.4 Power management programming hints

ANSI C cannot directly generate the `WFI` and `WFE` instructions. The CMSIS provides the following intrinsic functions for these instructions:

```
void __WFE(void) // Wait for Event

void __WFE(void) // Wait for Interrupt
```


© NXP B.V. 2010. All rights reserved.

4. ARM Cortex-M3 User Guide: Peripherals

4.1 About the Cortex-M3 peripherals

The address map of the **Private peripheral bus** (PPB) is:

Table 644. Core peripheral register regions

Address	Core peripheral	Description
0xE000E008 - 0xE000E00F	System control block	Table 34–655 “Summary of the system control block registers”
0xE000E010 - 0xE000E01F	System timer	Table 34–675 “System timer registers summary”
0xE000E100 - 0xE000E4EF	Nested Vectored Interrupt Controller	Table 34–645 “NVIC register summary”
0xE000ED00 - 0xE000ED3F	System control block	Table 34–655 “Summary of the system control block registers”
0xE000ED90 - 0xE000EDB8	Memory Protection Unit	Table 34–681 “MPU registers summary”
0xE000EF00 - 0xE000EF03	Nested Vectored Interrupt Controller	Table 34–645 “NVIC register summary”

In register descriptions:

- the register **type** is described as follows:
  - **RW**: Read and write.
  - **RO**: Read-only.
  - **WO**: Write-only.
- the **required privilege** gives the privilege level required to access the register, as follows:
  - **Privileged**: Only privileged software can access the register
  - **Unprivileged**: Both unprivileged and privileged software can access the register.

© NXP B.V. 2010. All rights reserved.

4.2 Nested Vectored Interrupt Controller

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports:

- Up to 112 interrupts. The number of interrupts implemented is device dependent.
- A programmable priority level of 0 to 31 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external **Non-maskable interrupt** (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling. The hardware implementation of the NVIC registers is:

Table 645. NVIC register summary

Address	Name	Type	Required privilege	Reset value	Description
0xE000E100 - 0xE000E10C	ISER0 - ISER3	RW	Privileged	0x00000000	Table 34–647
0XE000E180 - 0xE000E18C	ICER0 - ICER3	RW	Privileged	0x00000000	Table 34–648
0XE000E200 - 0xE000E20C	ISPR0 - ISPR3	RW	Privileged	0x00000000	Table 34–649
0XE000E280 - 0xE000E28C	ICPR0 - ICPR3	RW	Privileged	0x00000000	Table 34–650
0xE000E300 - 0xE000E30C	IABR0 - IABR3	RO	Privileged	0x00000000	Table 34–651
0xE000E400 - 0xE000E46C	IPR0 - IPR27	RW	Privileged	0x00000000	Table 34–652
0xE000EF00	STIR	WO	Configurable[1]	0x00000000	Table 34–653

[1] Each array element corresponds to a single NVIC register, for example the element ICER[1] corresponds to the ICER1 register.

4.2.1 The CMSIS mapping of the Cortex-M3 NVIC registers

To improve software efficiency, the CMSIS simplifies the NVIC register presentation. In the CMSIS:

- the Set-enable, Clear-enable, Set-pending, Clear-pending and Active Bit registers map to arrays of 32-bit integers, so that:
  - the array ISER[0] to ISER[3] corresponds to the registers ISER0 - ISER3
  - the array ICER[0] to ICER[3] corresponds to the registers ICER0 - ICER3
  - the array ISPR[0] to ISPR[3] corresponds to the registers ISPR0 - ISPR3
  - the array ICPR[0] to ICPR[3] corresponds to the registers ICPR0 - ICPR3
  - the array IABR[0] to IABR[3] corresponds to the registers IABR0 - IABR3.
- the 8-bit fields of the Interrupt Priority Registers map to an array of 8-bit integers, so that the array IP[0] to IP[112] corresponds to the registers IPR0 - IPR59, and the array entry IP[n] holds the interrupt priority for interrupt **n**.

© NXP B.V. 2010. All rights reserved.

The CMSIS provides thread-safe code that gives atomic access to the Interrupt Priority Registers. For more information see the description of the `NVIC_SetPriority` function in Section 34–4.2.10.1 “NVIC programming hints”. Table 34–646 shows how the interrupts, or IRQ numbers, map onto the interrupt registers and corresponding CMSIS variables that have one bit per interrupt.

Table 646. Mapping of interrupts to the interrupt variables

Interrupts	CMSIS array elements[1]				
	Set-enable	Clear-enable	Set-pending	Clear-pending	Active Bit
0-31	`ISER[0]`	`ICER[0]`	`ISPR[0]`	`ICPR[0]`	`IABR[0]`
32-63	`ISER[1]`	`ICER[1]`	`ISPR[1]`	`ICPR[1]`	`IABR[1]`
64-95	`ISER[2]`	`ICER[2]`	`ISPR[2]`	`ICPR[2]`	`IABR[2]`
96-127	`ISER[3]`	`ICER[3]`	`ISPR[3]`	`ICPR[3]`	`IABR[3]`

[1] Each array element corresponds to a single NVIC register, for example the element `ICER[1]` corresponds to the ICER1 register.

4.2.2 Interrupt Set-enable Registers

The ISER0-ISER3 registers enable interrupts, and show which interrupts are enabled. See:

- the register summary in Table 34–645 for the register attributes
- Table 34–646 for which interrupts are controlled by each register.

The bit assignments are shown in Table 34–647.

Table 647. ISER bit assignments

Bits	Name	Function
[31:0]	SETENA	Interrupt set-enable bits. Write: 0 = no effect 1 = enable interrupt. Read: 0 = interrupt disabled 1 = interrupt enabled.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

4.2.3 Interrupt Clear-enable Registers

The ICER0-ICER3 registers disable interrupts, and show which interrupts are enabled. See:

- the register summary in Table 34–645 for the register attributes
- Table 34–646 for which interrupts are controlled by each register.

The bit assignments are shown in Table 34–648.

© NXP B.V. 2010. All rights reserved.

Table 648. ICER bit assignments

Bits	Name	Function
[31:0]	CLRENA	Interrupt clear-enable bits. Write: 0 = no effect 1 = disable interrupt. Read: 0 = interrupt disabled 1 = interrupt enabled.

4.2.4 Interrupt Set-pending Registers

The ISPR0-ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. See:

- the register summary in Table 34–645 for the register attributes
- Table 34–646 for which interrupts are controlled by each register.

The bit assignments are shown in Table 34–649.

Table 649. ISPR bit assignments

Bits	Name	Function
[31:0]	SETPEND	Interrupt set-pending bits. Write: 0 = no effect 1 = changes interrupt state to pending. Read: 0 = interrupt is not pending 1 = interrupt is pending.

Remark: Writing 1 to the ISPR bit corresponding to:

- an interrupt that is pending has no effect
- a disabled interrupt sets the state of that interrupt to pending.

4.2.5 Interrupt Clear-pending Registers

The ICPR0-ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. See:

- the register summary in Table 34–645 for the register attributes
- Table 34–646 for which interrupts are controlled by each register.

The bit assignments are shown in Table 34–650.

© NXP B.V. 2010. All rights reserved.

Table 650. ICPR bit assignments

Bits	Name	Function
[31:0]	CLRPEND	Interrupt clear-pending bits. Write: 0 = no effect 1 = removes pending state an interrupt. Read: 0 = interrupt is not pending 1 = interrupt is pending.

Remark: Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt.

4.2.6 Interrupt Active Bit Registers

The IABR0-IABR3 registers indicate which interrupts are active. See:

- the register summary in Table 34–645 for the register attributes
- Table 34–646 for which interrupts are controlled by each register.

The bit assignments are shown in Table 34–651.

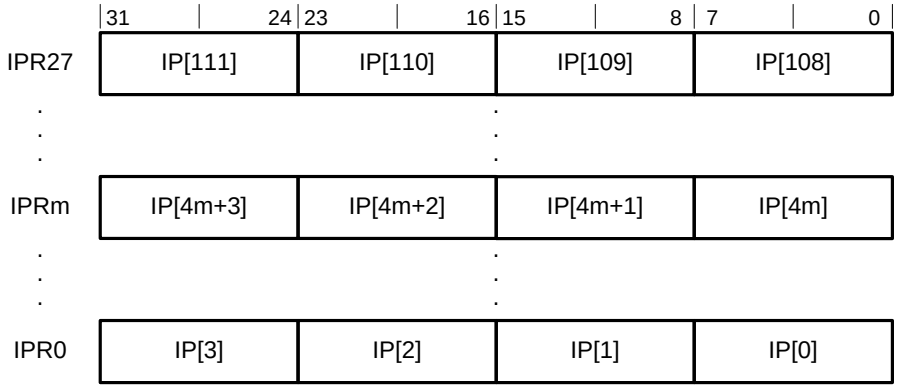
Table 651. IABR bit assignments

Bits	Name	Function
[31:0]	ACTIVE	Interrupt active flags: 0 = interrupt not active 1 = interrupt active.

A bit reads as one if the status of the corresponding interrupt is active or active and pending.

4.2.7 Interrupt Priority Registers

The IPR0-IPR27 registers provide a 5-bit priority field for each interrupt. These registers are byte-accessible. See the register summary in Table 34–645 for their attributes. Each register holds four priority fields, that map to four elements in the CMSIS interrupt priority array `IP[0]` to `IP[111]`, as shown:



© NXP B.V. 2010. All rights reserved.

Table 652. IPR bit assignments

Bits	Name	Function
[31:24]	Priority, byte offset 3	Each priority field holds a priority value, 0-31. The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:3] of each field, bits[2:0] read as zero and ignore writes.
[23:16]	Priority, byte offset 2	
[15:8]	Priority, byte offset 1	
[7:0]	Priority, byte offset 0	

See Table 34–646 for more information about the `IP[0]` to `IP[111]` interrupt priority array, that provides the software view of the interrupt priorities.

Find the IPR number and byte offset for interrupt **N** as follows:

- the corresponding IPR number, **M**, is given by **M** = **N** DIV 4
- the byte offset of the required Priority field in this register is **N** MOD 4, where:
  - byte offset 0 refers to register bits[7:0]
  - byte offset 1 refers to register bits[15:8]
  - byte offset 2 refers to register bits[23:16]
  - byte offset 3 refers to register bits[31:24].

4.2.8 Software Trigger Interrupt Register

Write to the STIR to generate a **Software Generated Interrupt** (SGI). See the register summary in Table 34–645 for the STIR attributes.

When the USERSETMPEND bit in the CCR is set to 1, unprivileged software can access the STIR, see Section 34–4.3.8 “Configuration and Control Register”.

Remark: Only privileged software can enable unprivileged access to the STIR.

The bit assignments are shown in Table 34–653.

Table 653. STIR bit assignments

Bits	Field	Function
[31:9]	-	Reserved.
[8:0]	INTID	Interrupt ID of the required SGI, in the range 0-111. For example, a value of b000000011 specifies interrupt IRQ3.

4.2.9 Level-sensitive and pulse interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

© NXP B.V. 2010. All rights reserved.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see Section 34–4.2.9.1. For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

4.2.9.1 Hardware and software control of interrupts

The Cortex-M3 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- the NVIC detects that the interrupt signal is HIGH and the interrupt is not active
- the NVIC detects a rising edge on the interrupt signal
- software writes to the corresponding interrupt set-pending register bit, see Table 34–649, or to the STIR to make an SGI pending, see Table 34–653.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

    If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit.

  For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.

  For a pulse interrupt, state of the interrupt changes to:
  - inactive, if the state was pending
  - active, if the state was active and pending.

4.2.10 NVIC design hints and tips

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers. See the individual register descriptions for the supported access sizes.

A interrupt can enter pending state even it is disabled.

Before programming VTOR to relocate the vector table, ensure the vector table entries of the new vector table are setup for fault handlers, NMI and all enabled exception like interrupts. For more information see Table 34–659.

© NXP B.V. 2010. All rights reserved.

4.2.10.1 NVIC programming hints

Software uses the CPSIE I and CPSID I instructions to enable and disable interrupts. The CMSIS provides the following intrinsic functions for these instructions:

```
void __disable_irq(void) // Disable Interrupts

void __enable_irq(void) // Enable Interrupts
```

In addition, the CMSIS provides a number of functions for NVIC control, including:

Table 654. CMSIS functions for NVIC control

CMSIS interrupt control function	Description
`void NVIC_SetPriorityGrouping(uint32_t priority_grouping)`	Set the priority grouping
`void NVIC_EnableIRQ(IRQn_t IRQn)`	Enable IRQn
`void NVIC_DisableIRQ(IRQn_t IRQn)`	Disable IRQn
`uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)`	Return true (IRQ-Number) if IRQn is pending
`void NVIC_SetPendingIRQ (IRQn_t IRQn)`	Set IRQn pending
`void NVIC_ClearPendingIRQ (IRQn_t IRQn)`	Clear IRQn pending status
`uint32_t NVIC_GetActive (IRQn_t IRQn)`	Return the IRQ number of the active interrupt
`void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)`	Set priority for IRQn
`uint32_t NVIC_GetPriority (IRQn_t IRQn)`	Read priority of IRQn
`void NVIC_SystemReset (void)`	Reset the system

For more information about these functions see the CMSIS documentation.

© NXP B.V. 2010. All rights reserved.

4.3 System control block

The **System control block** (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The system control block registers are:

Table 655. Summary of the system control block registers

Address	Name	Type	Required privilege	Reset value	Description
0xE000E008	ACTLR	RW	Privileged	0x00000000	Table 34–656
0xE000ED00	CPUID	RO	Privileged	0x412FC230	Table 34–657
0xE000ED04	ICSR	RW[1]	Privileged	0x00000000	Table 34–658
0xE000ED08	VTOR	RW	Privileged	0x00000000	Table 34–659
0xE000ED0C	AIRCR	RW[1]	Privileged	0xFA050000	Table 34–660
0xE000ED10	SCR	RW	Privileged	0x00000000	Table 34–662
0xE000ED14	CCR	RW	Privileged	0x00000200	Table 34–663
0xE000ED18	SHPR1	RW	Privileged	0x00000000	Table 34–665
0xE000ED1C	SHPR2	RW	Privileged	0x00000000	Table 34–666
0xE000ED20	SHPR3	RW	Privileged	0x00000000	Table 34–667
0xE000ED24	SHCRS	RW	Privileged	0x00000000	Table 34–668
0xE000ED28	CFSR	RW	Privileged	0x00000000	Section 34–4.3.11
0xE000ED28	MMSR[2]	RW	Privileged	0x00	Table 34–669
0xE000ED29	BFSR[2]	RW	Privileged	0x00	Table 34–670
0xE000ED2A	UFSR[2]	RW	Privileged	0x0000	Table 34–671
0xE000ED2C	HFSR	RW	Privileged	0x00000000	Table 34–672
0xE000ED34	MMFAR	RW	Privileged	Undefined	Table 34–673
0xE000ED38	BFAR	RW	Privileged	Undefined	Table 34–674

[1] See the register description for more information.

[2] A subregister of the CFSR.

4.3.1 The CMSIS mapping of the Cortex-M3 SCB registers

To improve software efficiency, the CMSIS simplifies the SCB register presentation. In the CMSIS, the byte array `SHP[0]` to `SHP[12]` corresponds to the registers SHPR1-SHPR3.

4.3.2 Auxiliary Control Register

The ACTLR provides disable bits for the following processor functions:

- IT folding
- write buffer use for accesses to the default memory map
- interruption of multi-cycle instructions.

See the register summary in Table 34–655 for the ACTLR attributes. The bit assignments are shown in Table 34–656.

© NXP B.V. 2010. All rights reserved.

Table 656. ACTLR bit assignments

Bits	Name	Function
[31:3]	-	Reserved
[2]	DISFOLD	When set to 1, disables IT folding. see Section 34–4.3.2.1 for more information.
[1]	DISDEFWBUF	When set to 1, disables write buffer use during default memory map accesses. This causes all bus faults to be precise bus faults but decreases performance because any store to memory must complete before the processor can execute the next instruction. **Remark:** This bit only affects write buffers implemented in the Cortex-M3 processor.
[0]	DISMCYCINT	When set to 1, disables interruption of load multiple and store multiple instructions. This increases the interrupt latency of the processor because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler.

4.3.2.1 About IT folding

In some situations, the processor can start executing the first instruction in an IT block while it is still executing the `IT` instruction. This behavior is called IT folding, and improves performance, However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit to 1 before executing the task, to disable IT folding.

4.3.3 CPUID Base Register

The CPUID register contains the processor part number, version, and implementation information. See the register summary in Table 34–655 for its attributes. The bit assignments are shown in Table 34–657.

Table 657. CPUID register bit assignments

Bits	Name	Function
[31:24]	Implementer	Implementer code: `0x41` = ARM
[23:20]	Variant	Variant number, the r value in the r**n**p**n** product revision identifier: 0x2 = r2p0
[19:16]	Constant	Reads as `0xF`
[15:4]	PartNo	Part number of the processor: `0xC23` = Cortex-M3
[3:0]	Revision	Revision number, the p value in the r**n**p**n** product revision identifier: 0x0 = r2p0

4.3.4 Interrupt Control and State Register

The ICSR:

- provides:
  - a set-pending bit for the **Non-Maskable Interrupt** (NMI) exception
  - set-pending and clear-pending bits for the PendSV and SysTick exceptions
- indicates:
  - the exception number of the exception being processed

© NXP B.V. 2010. All rights reserved.

– whether there are preempted active exceptions
– the exception number of the highest priority pending exception
– whether any interrupts are pending.

See the register summary in Table 34–655, and the Type descriptions in Table 34–658, for the ICSR attributes. The bit assignments are shown in Table 34–658.

Table 658. ICSR bit assignments

Bits	Name	Type	Function
[31]	NMIPENDSET	RW	NMI set-pending bit. Write: 0 = no effect 1 = changes NMI exception state to pending. Read: 0 = NMI exception is not pending 1 = NMI exception is pending. Because NMI is the highest-priority exception, normally the processor enter the NMI exception handler as soon as it registers a write of 1 to this bit, and entering the handler clears this bit to 0. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
[30:29]	-	-	Reserved.
[28]	PENDSVSET	RW	PendSV set-pending bit. Write: 0 = no effect 1 = changes PendSV exception state to pending. Read: 0 = PendSV exception is not pending 1 = PendSV exception is pending. Writing 1 to this bit is the only way to set the PendSV exception state to pending.
[27]	PENDSVCLR	WO	PendSV clear-pending bit. Write: 0 = no effect 1 = removes the pending state from the PendSV exception.
[26]	PENDSTSET	RW	SysTick exception set-pending bit. Write: 0 = no effect 1 = changes SysTick exception state to pending. Read: 0 = SysTick exception is not pending 1 = SysTick exception is pending.

© NXP B.V. 2010. All rights reserved.

Table 658. ICSR bit assignments

Bits	Name	Type	Function
[25]	PENDSTCLR	WO	SysTick exception clear-pending bit. Write: 0 = no effect 1 = removes the pending state from the SysTick exception. This bit is WO. On a register read its value is Unknown.
[24]	-	-	Reserved.
[23]	Reserved for Debug use	RO	This bit is reserved for Debug use and reads-as-zero when the processor is not in Debug.
[22]	ISRPENDING	RO	Interrupt pending flag, excluding NMI and Faults: 0 = interrupt not pending 1 = interrupt pending.
[21:18]	-	-	Reserved.
[17:12]	VECTPENDING	RO	Indicates the exception number of the highest priority pending enabled exception: 0 = no pending exceptions Nonzero = the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.
[11]	RETTOBASE	RO	Indicates whether there are preempted active exceptions: 0 = there are preempted active exceptions to execute 1 = there are no active exceptions, or the currently-executing exception is the only active exception.
[10:9]	-	-	Reserved.
[8:0]	VECTACTIVE[1]	RO	Contains the active exception number: 0 = Thread mode Nonzero = The exception number[1] of the currently active exception. **Remark:** Subtract 16 from this value to obtain the IRQ number required to index into the Interrupt Clear-Enable, Set-Enable, Clear-Pending, Set-Pending, or Priority Registers, see Table 34–630 “IPSR bit assignments”.

[1] This is the same value as IPSR bits[8:0], see Table 34–630 “IPSR bit assignments”.

When you write to the ICSR, the effect is unpredictable if you:

- write 1 to the PENDSVSET bit and write 1 to the PENDSVCLR bit
- write 1 to the PENDSTSET bit and write 1 to the PENDSTCLR bit.

4.3.5 Vector Table Offset Register

The VTOR indicates the offset of the vector table base address from memory address `0x00000000`. See the register summary in Table 34–655 for its attributes.

The bit assignments are shown in Table 34–659.

© NXP B.V. 2010. All rights reserved.

Table 659. VTOR bit assignments

Bits	Name	Function
[31:30]	-	Reserved.
[29:8]	TBLOFF	Vector table base offset field. It contains bits[29:8] of the offset of the table base from the bottom of the memory map. **Remark:** Bit[29] determines whether the vector table is in the code or SRAM memory region: Bit[29] is sometimes called the TBLBASE bit. • 0 = code • 1 = SRAM.
[7:0]	-	Reserved.

When setting TBLOFF, you must align the offset to the number of exception entries in the vector table. The recommended alignment is 256 words, allowing for 128 interrupts.

Remark: Table alignment requirements mean that bits[7:0] of the table offset are always zero.

4.3.6 Application Interrupt and Reset Control Register

The AIRCR provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. See the register summary in Table 34–655 and Table 34–660 for its attributes.

To write to this register, you must write `0x5VA` to the VECTKEY field, otherwise the processor ignores the write.

The bit assignments are shown in Table 34–660.

Table 660. AIRCR bit assignments

Bits	Name	Type	Function
[31:16]	Write: VECTKEYSTAT Read: VECTKEY	RW	Register key: Reads as `0x05FA` On writes, write `0x5FA` to VECTKEY, otherwise the write is ignored.
[15]	ENDIANESS	RO	Data endianness bit: 0 = Little-endian.
[14:11]	-	-	Reserved
[10:8]	PRIGROUP	R/W	Interrupt priority grouping field. This field determines the split of group priority from subpriority, see Section 34–4.3.6.1.
[7:3]	-	-	Reserved.

© NXP B.V. 2010. All rights reserved.

Table 660. AIRCR bit assignments

Bits	Name	Type	Function
[2]	SYSRESETREQ	WO	System reset request: 0 = no system reset request 1 = asserts a signal to the outer system that requests a reset. This is intended to force a large system reset of all major components except for debug. **Note: support for SYSRESETREQ is not included in LPC17xx devices.** This bit reads as 0.
[1]	VECTCLRACTIVE	WO	Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable.
[0]	VECTRESET	WO	Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable.

4.3.6.1 Binary point

The PRIGROUP field indicates the position of the binary point that splits the PRI_**n** fields in the Interrupt Priority Registers into separate **group priority** and **subpriority** fields. Table 34–661 shows how the PRIGROUP value controls this split.

Table 661. Priority grouping

	Interrupt priority level value, PRI_N[7:0]			Number of	
PRIGROUP	Binary point[1]	Group priority bits	Subpriority bits	Group priorities	Subpriorities
b010	bxxxxxy000	[7:3]	none	32	1
b011	bxxxx.y000	[7:4]	[3]	16	2
b100	bxxx.yy000	[7:5]	[4:3]	8	3
b101	bxx.yyy000	[7:6]	[5:3]	4	8
b110	bx.yyyy000	[7]	[6:3]	2	16
b111	b.yyyyy000	None	[7:3]	1	32

[1] PRI_**n**[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit. Bits [2:0] are not used in LPC17xx devices.

Remark: Determining preemption of an exception uses only the group priority field, see Section 34–3.3.6 “Interrupt priority grouping”.

4.3.7 System Control Register

The SCR controls features of entry to and exit from low power state. See the register summary in Table 34–655 for its attributes. The bit assignments are shown in Table 34–662.

© NXP B.V. 2010. All rights reserved.

Table 662. SCR bit assignments

Bits	Name	Function
[31:5]	-	Reserved.
[4]	SEVONPEND	Send Event on Pending bit: 0 = only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded 1 = enabled events and all interrupts, including disabled interrupts, can wakeup the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an `SEV` instruction or an external event.
[3]	-	Reserved.
[2]	SLEEPDEEP	Controls whether the processor uses sleep or deep sleep as its low power mode: 0 = sleep 1 = deep sleep.
[1]	SLEEPONEXIT	Indicates sleep-on-exit when returning from Handler mode to Thread mode: 0 = do not sleep when returning to Thread mode. 1 = enter sleep, or deep sleep, on return from an ISR. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.
[0]	-	Reserved.

4.3.8 Configuration and Control Register

The CCR controls entry to Thread mode and enables:

- the handlers for NMI, hard fault and faults escalated by FAULTMASK to ignore bus faults
- trapping of divide by zero and unaligned accesses
- access to the STIR by unprivileged software, see Table 34–653.

See the register summary in Table 34–655 for the CCR attributes.

The bit assignments are shown in Table 34–663.

© NXP B.V. 2010. All rights reserved.

Table 663. CCR bit assignments

Bits	Name	Function
[31:10]	-	Reserved.
[9]	STKALIGN	Indicates stack alignment on exception entry: 0 = 4-byte aligned 1 = 8-byte aligned. On exception entry, the processor uses bit[9] of the stacked PSR to indicate the stack alignment. On return from the exception it uses this stacked bit to restore the correct stack alignment.
[8]	BFHFNMIGN	Enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. This applies to the hard fault, NMI, and FAULTMASK escalated handlers: 0 = data bus faults caused by load and store instructions cause a lock-up 1 = handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions. Set this bit to 1 only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.
[7:5]	-	Reserved.
[4]	DIV_0_TRP	Enables faulting or halting when the processor executes an `SDIV` or `UDIV` instruction with a divisor of 0: 0 = do not trap divide by 0 1 = trap divide by 0. When this bit is set to 0,a divide by zero returns a quotient of 0.
[3]	UNALIGN_TRP	Enables unaligned access traps: 0 = do not trap unaligned halfword and word accesses 1 = trap unaligned halfword and word accesses. If this bit is set to 1, an unaligned access generates a usage fault. Unaligned `LDM`, `STM`, `LDRD`, and `STRD` instructions always fault irrespective of whether UNALIGN_TRP is set to 1.
[2]	-	Reserved.
[1]	USERSETMPEND	Enables unprivileged software access to the STIR, see Table 34–653: 0 = disable 1 = enable.
[0]	NONEBASETHRDENA	Indicates how the processor enters Thread mode: 0 = processor can enter Thread mode only when no exception is active. 1 = processor can enter Thread mode from any level under the control of an EXC_RETURN value, see Section 34–3.3.7.2 “Exception return”.

4.3.9 System Handler Priority Registers

The SHPR1-SHPR3 registers set the priority level, 0 to 31 of the exception handlers that have configurable priority.

SHPR1-SHPR3 are byte accessible. See the register summary in Table 34–655 for their attributes.

The system fault handlers and the priority field and register for each handler are:

© NXP B.V. 2010. All rights reserved.

Table 664. System fault handler priority fields

Handler	Field	Register description
Memory management fault	PRI_4	Table 34–665
Bus fault	PRI_5	
Usage fault	PRI_6	
SVCall	PRI_11	Table 34–666
PendSV	PRI_14	Table 34–667
SysTick	PRI_15	

Each PRI_N field is 8 bits wide, but the processor implements only bits[7:3] of each field, and bits[2:0] read as zero and ignore writes.

4.3.9.1 System Handler Priority Register 1

The bit assignments are shown in Table 34–665.

Table 665. SHPR1 register bit assignments

Bits	Name	Function
[31:24]	PRI_7	Reserved
[23:16]	PRI_6	Priority of system handler 6, usage fault
[15:8]	PRI_5	Priority of system handler 5, bus fault
[7:0]	PRI_4	Priority of system handler 4, memory management fault

4.3.9.2 System Handler Priority Register 2

The bit assignments are shown in Table 34–666.

Table 666. SHPR2 register bit assignments

Bits	Name	Function
[31:24]	PRI_11	Priority of system handler 11, SVCall
[23:0]	-	Reserved

4.3.9.3 System Handler Priority Register 3

The bit assignments are shown in Table 34–667.

Table 667. SHPR3 register bit assignments

Bits	Name	Function
[31:24]	PRI_15	Priority of system handler 15, SysTick exception
[23:16]	PRI_14	Priority of system handler 14, PendSV
[15:0]	-	Reserved

4.3.10 System Handler Control and State Register

The SHCSR enables the system handlers, and indicates:

- the pending status of the bus fault, memory management fault, and SVC exceptions
- the active status of the system handlers.

See the register summary in Table 34–655 for the SHCSR attributes. The bit assignments are shown in Table 34–668.

© NXP B.V. 2010. All rights reserved.

Table 668. SHCSR bit assignments

Bits	Name	Function
[31:19]	-	Reserved
[18]	USGFAULTENA	Usage fault enable bit, set to 1 to enable[1]
[17]	BUSFAULTENA	Bus fault enable bit, set to 1 to enable[1]
[16]	MEMFAULTENA	Memory management fault enable bit, set to 1 to enable[1]
[15]	SVCALLPENDED	SVC call pending bit, reads as 1 if exception is pending[2]
[14]	BUSFAULTPENDED	Bus fault exception pending bit, reads as 1 if exception is pending[2]
[13]	MEMFAULTPENDED	Memory management fault exception pending bit, reads as 1 if exception is pending[2]
[12]	USGFAULTPENDED	Usage fault exception pending bit, reads as 1 if exception is pending[2]
[11]	SYSTICKACT	SysTick exception active bit, reads as 1 if exception is active[3]
[10]	PENDSVACT	PendSV exception active bit, reads as 1 if exception is active
[9]	-	Reserved
[8]	MONITORACT	Debug monitor active bit, reads as 1 if Debug monitor is active
[7]	SVCALLACT	SVC call active bit, reads as 1 if SVC call is active
[6:4]	-	Reserved
[3]	USGFAULTACT	Usage fault exception active bit, reads as 1 if exception is active
[2]	-	Reserved
[1]	BUSFAULTACT	Bus fault exception active bit, reads as 1 if exception is active
[0]	MEMFAULTACT	Memory management fault exception active bit, reads as 1 if exception is active

[1] Enable bits, set to 1 to enable the exception, or set to 0 to disable the exception.

[2] Pending bits, read as 1 if the exception is pending, or as 0 if it is not pending. You can write to these bits to change the pending status of the exceptions.

[3] Active bits, read as 1 if the exception is active, or as 0 if it is not active. You can write to these bits to change the active status of the exceptions, but see the Caution in this section.

If you disable a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault.

You can write to this register to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

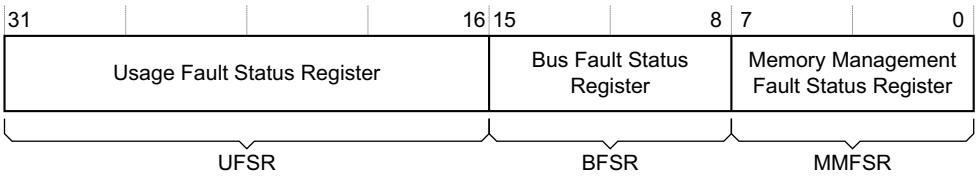
Caution

- Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.
- After you have enabled the system handlers, if you have to change the value of a bit in this register you must use a read-modify-write procedure to ensure that you change only the required bit.

© NXP B.V. 2010. All rights reserved.

4.3.11 Configurable Fault Status Register

The CFSR indicates the cause of a memory management fault, bus fault, or usage fault. See the register summary in Table 34–655 for its attributes. The bit assignments are:



The following subsections describe the subregisters that make up the CFSR:

- Table 34–669 “MMFSR bit assignments”
- Table 34–674 “BFAR bit assignments”
- Table 34–671 “UFSR bit assignments”.

The CFSR is byte accessible. You can access the CFSR or its subregisters as follows:

- access the complete CFSR with a word access to `0xE000ED28`
- access the MMFSR with a byte access to `0xE000ED28`
- access the MMFSR and BFSR with a halfword access to `0xE000ED28`
- access the BFSR with a byte access to `0xE000ED29`
- access the UFSR with a halfword access to `0xE000ED2A`.

4.3.11.1 Memory Management Fault Status Register

The flags in the MMFSR indicate the cause of memory access faults. The bit assignments are shown in Table 34–669.

Table 669. MMFSR bit assignments

Bits	Name	Function
[7]	MMARVALID	**Memory Management Fault Address Register** (MMAR) valid flag: 0 = value in MMAR is not a valid fault address 1 = MMAR holds a valid fault address. If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems on return to a stacked active memory management fault handler whose MMAR value has been overwritten.
[6:5]	-	Reserved.
[4]	MSTKERR	Memory manager fault on stacking for exception entry: 0 = no stacking fault 1 = stacking for an exception entry has caused one or more access violations. When this bit is 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor has not written a fault address to the MMAR.

© NXP B.V. 2010. All rights reserved.

Table 669. MMFSR bit assignments

Bits	Name	Function
[3]	MUNSTKERR	Memory manager fault on unstacking for a return from exception: 0 = no unstacking fault 1 = unstack for an exception return has caused one or more access violations. This fault is chained to the handler. This means that when this bit is 1, the original return stack is still present. The processor has not adjusted the SP from the failing return, and has not performed a new save. The processor has not written a fault address to the MMAR.
[2]	-	Reserved
[1]	DACCVIOL	Data access violation flag: 0 = no data access violation fault 1 = the processor attempted a load or store at a location that does not permit the operation. When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has loaded the MMAR with the address of the attempted access.
[0]	IACCVIOL	Instruction access violation flag: 0 = no instruction access violation fault 1 = the processor attempted an instruction fetch from a location that does not permit execution. This fault occurs on any access to an XN region, even when the MPU is disabled. When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has not written a fault address to the MMAR.

4.3.11.2 Bus Fault Status Register

The flags in the BFSR indicate the cause of a bus access fault. The bit assignments are shown in Table 34–670.

Table 670. BFSR bit assignments

Bits	Name	Function
[7]	BFARVALID	**Bus Fault Address Register** (BFAR) valid flag: 0 = value in BFAR is not a valid fault address 1 = BFAR holds a valid fault address. The processor sets this bit to 1 after a bus fault where the address is known. Other faults can set this bit to 0, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems if returning to a stacked active bus fault handler whose BFAR value has been overwritten.
[6:5]	-	Reserved.

© NXP B.V. 2010. All rights reserved.

Table 670. BFSR bit assignments

Bits	Name	Function
[4]	STKERR	Bus fault on stacking for exception entry: 0 = no stacking fault 1 = stacking for an exception entry has caused one or more bus faults. When the processor sets this bit to 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor does not write a fault address to the BFAR.
[3]	UNSTKERR	Bus fault on unstacking for a return from exception: 0 = no unstacking fault 1 = unstack for an exception return has caused one or more bus faults. This fault is chained to the handler. This means that when the processor sets this bit to 1, the original return stack is still present. The processor does not adjust the SP from the failing return, does not performed a new save, and does not write a fault address to the BFAR.
[2]	IMPRECISERR	Imprecise data bus error: 0 = no imprecise data bus error 1 = a data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error. When the processor sets this bit to 1, it does not write a fault address to the BFAR. This is an asynchronous fault. Therefore, if it is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects both IMPRECISERR set to 1 and one of the precise fault status bits set to 1.
[1]	PRECISERR	Precise data bus error: 0 = no precise data bus error 1 = a data bus error has occurred, and the PC value stacked for the exception return points to the instruction that caused the fault. When the processor sets this bit is 1, it writes the faulting address to the BFAR.
[0]	IBUSERR	Instruction bus error: 0 = no instruction bus error 1 = instruction bus error. The processor detects the instruction bus error on prefetching an instruction, but it sets the IBUSERR flag to 1 only if it attempts to issue the faulting instruction. When the processor sets this bit is 1, it does not write a fault address to the BFAR.

4.3.11.3 Usage Fault Status Register

The UFSR indicates the cause of a usage fault. The bit assignments are shown in Table 34–671.

© NXP B.V. 2010. All rights reserved.

Table 671. UFSR bit assignments

Bits	Name	Function
[15:10]	-	Reserved.
[9]	DIVBYZERO	Divide by zero usage fault: 0 = no divide by zero fault, or divide by zero trapping not enabled 1 = the processor has executed an `SDIV` or `UDIV` instruction with a divisor of 0. When the processor sets this bit to 1, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Enable trapping of divide by zero by setting the DIV_0_TRP bit in the CCR to 1, see Table 34–663.
[8]	UNALIGNED	Unaligned access usage fault: 0 = no unaligned access fault, or unaligned access trapping not enabled 1 = the processor has made an unaligned memory access. Enable trapping of unaligned accesses by setting the UNALIGN_TRP bit in the CCR to 1, see Table 34–663. Unaligned `LDM`, `STM`, `LDRD`, and `STRD` instructions always fault irrespective of the setting of UNALIGN_TRP.
[7:4]	-	Reserved.
[3]	NOCP	No coprocessor usage fault. The processor does not support coprocessor instructions: 0 = no usage fault caused by attempting to access a coprocessor 1 = the processor has attempted to access a coprocessor.
[2]	INVPC	Invalid PC load usage fault, caused by an invalid PC load by EXC_RETURN: 0 = no invalid PC load usage fault 1 = the processor has attempted an illegal load of EXC_RETURN to the PC, as a result of an invalid context, or an invalid EXC_RETURN value. When this bit is set to 1, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC.
[1]	INVSTATE	Invalid state usage fault: 0 = no invalid state usage fault 1 = the processor has attempted to execute an instruction that makes illegal use of the EPSR. When this bit is set to 1, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the EPSR. This bit is not set to 1 if an undefined instruction uses the EPSR.
[0]	UNDEFINSTR	Undefined instruction usage fault: 0 = no undefined instruction usage fault 1 = the processor has attempted to execute an undefined instruction. When this bit is set to 1, the PC value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode.

Remark: The UFSR bits are sticky. This means as one or more fault occurs, the associated bits are set to 1. A bit that is set to 1 is cleared to 0 only by writing 1 to that bit, or by a reset.

© NXP B.V. 2010. All rights reserved.

4.3.12 Hard Fault Status Register

The HFSR gives information about events that activate the hard fault handler. See the register summary in Table 34–655 for its attributes.

This register is read, write to clear. This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0. The bit assignments are shown in Table 34–672.

Table 672. HFSR bit assignments

Bits	Name	Function
[31]	DEBUGEVT	Reserved for Debug use. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable.
[30]	FORCED	Indicates a forced hard fault, generated by escalation of a fault with configurable priority that cannot be handles, either because of priority or because it is disabled: 0 = no forced hard fault 1 = forced hard fault. When this bit is set to 1, the hard fault handler must read the other fault status registers to find the cause of the fault.
[29:2]	-	Reserved.
[1]	VECTTBL	Indicates a bus fault on a vector table read during exception processing: 0 = no bus fault on vector table read 1 = bus fault on vector table read. This error is always handled by the hard fault handler. When this bit is set to 1, the PC value stacked for the exception return points to the instruction that was preempted by the exception.
[0]	-	Reserved.

Remark: The HFSR bits are sticky. This means as one or more fault occurs, the associated bits are set to 1. A bit that is set to 1 is cleared to 0 only by writing 1 to that bit, or by a reset.

4.3.13 Memory Management Fault Address Register

The MMFAR contains the address of the location that generated a memory management fault. See the register summary in Table 34–655 for its attributes. The bit assignments are:

Table 673. MMFAR bit assignments

Bits	Name	Function
[31:0]	ADDRESS	When the MMARVALID bit of the MMFSR is set to 1, this field holds the address of the location that generated the memory management fault

When an unaligned access faults, the address is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size.

Flags in the MMFSR indicate the cause of the fault, and whether the value in the MMFAR is valid. See Table 34–669.

4.3.14 Bus Fault Address Register

The BFAR contains the address of the location that generated a bus fault. See the register summary in Table 34–655 for its attributes. The bit assignments are:

© NXP B.V. 2010. All rights reserved.

Table 674. BFAR bit assignments

Bits	Name	Function
[31:0]	ADDRESS	When the BFARVALID bit of the BFSR is set to 1, this field holds the address of the location that generated the bus fault

When an unaligned access faults the address in the BFAR is the one requested by the instruction, even if it is not the address of the fault.

Flags in the BFSR indicate the cause of the fault, and whether the value in the BFAR is valid. See Table 34–670.

4.3.15 System control block design hints and tips

Ensure software uses aligned accesses of the correct size to access the system control block registers:

- except for the CFSR and SHPR1-SHPR3, it must use aligned word accesses
- for the CFSR and SHPR1-SHPR3 it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to system control block registers.

In a fault handler. to determine the true faulting address:

1. Read and save the MMFAR or BFAR value.
2. Read the MMARVALID bit in the MMFSR, or the BFARVALID bit in the BFSR. The MMFAR or BFAR address is valid only if this bit is 1.

Software must follow this sequence because another higher priority exception might change the MMFAR or BFAR value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the MMFAR or BFAR value.

© NXP B.V. 2010. All rights reserved.

4.4 System timer, SysTick

The processor has a 24-bit system timer, SysTick, that counts down from the reload value to zero, reloads (wraps to) the value in the LOAD register on the next clock edge, then counts down on subsequent clocks.

Note: refer to the separate chapter in the LPC17xx User Manual Section 23–1 for device specific information on the System Timer.

Remark: when the processor is halted for debugging the counter does not decrement.

The system timer registers are:

Table 675. System timer registers summary

Address	Name	Type	Required privilege	Reset value	Description
0xE000E010	CTRL	RW	Privileged	0x00000004	Table 34–676
0xE000E014	LOAD	RW	Privileged	0x00000000	Table 34–677
0xE000E018	VAL	RW	Privileged	0x00000000	Table 34–678
0xE000E01C	CALIB	RO	Privileged	0x000F423F[1]	Table 34–679

[1] SysTick calibration value. This value is specific to LPC17xx devices.

4.4.1 SysTick Control and Status Register

The SysTick CTRL register enables the SysTick features. See the register summary in Table 34–675 for its attributes. The bit assignments are shown in Table 34–676.

Table 676. SysTick CTRL register bit assignments

Bits	Name	Function
[31:17]	-	Reserved.
[16]	COUNTFLAG	Returns 1 if timer counted to 0 since last time this was read.
[15:3]	-	Reserved.
[2]	CLKSOURCE	Indicates the clock source: 0 = external clock 1 = processor clock.
[1]	TICKINT	Enables SysTick exception request: 0 = counting down to zero does not assert the SysTick exception request 1 = counting down to zero to asserts the SysTick exception request. Software can use COUNTFLAG to determine if SysTick has ever counted to zero.
[0]	ENABLE	Enables the counter: 0 = counter disabled 1 = counter enabled.

When ENABLE is set to 1, the counter loads the RELOAD value from the LOAD register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.

© NXP B.V. 2010. All rights reserved.

4.4.2 SysTick Reload Value Register

The LOAD register specifies the start value to load into the VAL register. See the register summary in Table 34–675 for its attributes. The bit assignments are shown in Table 34–677.

Table 677. LOAD register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	RELOAD	Value to load into the VAL register when the counter is enabled and when it reaches 0, see Section 34–4.4.2.1.

4.4.2.1 Calculating the RELOAD value

The RELOAD value can be any value in the range `0x00000001-0x00FFFFFF`. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

The RELOAD value is calculated according to its use:

- To generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. For example, if the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.
- To deliver a single SysTick interrupt after a delay of N processor clock cycles, use a RELOAD of value N. For example, if a SysTick interrupt is required after 400 clock pulses, set RELOAD to 400.

4.4.3 SysTick Current Value Register

The VAL register contains the current value of the SysTick counter. See the register summary in Table 34–675 for its attributes. The bit assignments are shown in Table 34–678.

Table 678. VAL register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	CURRENT	Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SysTick CTRL.COUNTFLAG bit to 0.

4.4.4 SysTick Calibration Value Register

The CALIB register indicates the SysTick calibration properties. See the register summary in Table 34–675 for its attributes. The bit assignments are shown in Table 34–679.

Table 679. CALIB register bit assignments

Bits	Name	Function
[31]	NOREF	Indicates whether a separate reference clock is available. This value is factory preset as described in Section 23–1.

© NXP B.V. 2010. All rights reserved.

Table 679. CALIB register bit assignments

Bits	Name	Function
[30]	SKEW	Indicates whether the value of TENMS is precise. This can affect the suitability of SysTick as a software real time clock. This value is factory preset as described in Section 23–1.
[29:24]	-	Reserved.
[23:0]	TENMS	This value is factory preset as described in Section 23–1.

If a different frequency is used than that intended by the factory preset value, calculate the calibration value required from the frequency of the processor clock or external clock.

4.4.5 SysTick design hints and tips

The SysTick counter runs on the processor clock. If this clock signal is stopped for low power mode while the SysTick counter is running from it, the SysTick counter will stop.

Ensure software uses aligned word accesses to access the SysTick registers.

© NXP B.V. 2010. All rights reserved.

4.5 Memory protection unit

This section describes the **Memory protection unit** (MPU).

The MPU divides the memory map into a number of regions, and defines the location, size, access permissions, and memory attributes of each region. It supports:

- independent attribute settings for each region
- overlapping regions
- export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M3 MPU defines:

- eight separate memory regions, 0-7
- a background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M3 MPU memory map is unified. This means instruction accesses and data accesses have same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault. This causes a fault exception, and might cause termination of the process in an OS environment.

In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types, see Section 34–3.2.1 “Memory regions, types and attributes”.

Table 34–680 shows the possible MPU region attributes. These include Shareability and cache behavior attributes that are not relevant to most microcontroller implementations. See Table 34–691 for guidelines for programming such an implementation.

Table 680. Memory attributes summary

Memory type	Shareability	Other attributes	Description
Strongly- ordered	-	-	All accesses to Strongly-ordered memory occur in program order. All Strongly-ordered regions are assumed to be shared.
Device	Shared	-	Memory-mapped peripherals that several processors share.

© NXP B.V. 2010. All rights reserved.

Table 680. Memory attributes summary

Memory type	Shareability	Other attributes	Description
	Non-shared	-	Memory-mapped peripherals that only a single processor uses.
Normal	Shared	Non-cacheable Write-through Cacheable Write-back Cacheable	Normal memory that is shared between several processors.
	Non-shared	Non-cacheable Write-through Cacheable Write-back Cacheable	Normal memory that only a single processor uses.

Use the MPU registers to define the MPU regions and their attributes. The MPU registers are:

Table 681. MPU registers summary

Address	Name	Type	Required privilege	Reset value	Description
0xE000ED90	TYPE	RO	Privileged	0x00000800	Table 34–682
0xE000ED94	CTRL	RW	Privileged	0x00000000	Table 34–683
0xE000ED98	RNR	RW	Privileged	0x00000000	Table 34–684
0xE000ED9C	RBAR	RW	Privileged	0x00000000	Table 34–685
0xE000EDA0	RASR	RW	Privileged	0x00000000	Table 34–686
0xE000EDA4	RBAR_A1	RW	Privileged	0x00000000	Alias of RBAR, see Table 34–685
0xE000EDA8	RASR_A1	RW	Privileged	0x00000000	Alias of RASR, see Table 34–686
0xE000EDAC	RBAR_A2	RW	Privileged	0x00000000	Alias of RBAR, see Table 34–685
0xE000EDB0	RASR_A2	RW	Privileged	0x00000000	Alias of RASR, see Table 34–686
0xE000EDB4	RBAR_A3	RW	Privileged	0x00000000	Alias of RBAR, see Table 34–685
0xE000EDB8	RASR_A3	RW	Privileged	0x00000000	Alias of RASR, see Table 34–686

4.5.1 MPU Type Register

The TYPE register indicates whether the MPU is present, and if so, how many regions it supports. See the register summary in Table 34–681 for its attributes. The bit assignments are shown in Table 34–682.

Table 682. TYPE register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:16]	IREGION	Indicates the number of supported MPU instruction regions. Always contains 0x00. The MPU memory map is unified and is described by the DREGION field.

© NXP B.V. 2010. All rights reserved.

Table 682. TYPE register bit assignments

Bits	Name	Function
[15:8]	DREGION	Indicates the number of supported MPU data regions: 0x08 = Eight MPU regions.
[7:0]	-	Reserved.
[0]	SEPARATE	Indicates support for unified or separate instruction and date memory maps: 0 = unified.

4.5.2 MPU Control Register

The MPU CTRL register:

- enables the MPU
- enables the default memory map background region
- enables use of the MPU when in the hard fault, **Non-maskable Interrupt** (NMI), and FAULTMASK escalated handlers.

See the register summary in Table 34–681 for the MPU CTRL attributes. The bit assignments are shown in Table 34–683.

Table 683. MPU CTRL register bit assignments

Bits	Name	Function
[31:3]	-	Reserved.
[2]	PRIVDEFENA	Enables privileged software access to the default memory map: 0 = If the MPU is enabled, disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault. 1 = If the MPU is enabled, enables use of the default memory map as a background region for privileged software accesses. When enabled, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map. If the MPU is disabled, the processor ignores this bit.
[1]	HFNMIENA	Enables the operation of MPU during hard fault, NMI, and FAULTMASK handlers. When the MPU is enabled: 0 = MPU is disabled during hard fault, NMI, and FAULTMASK handlers, regardless of the value of the ENABLE bit 1 = the MPU is enabled during hard fault, NMI, and FAULTMASK handlers. When the MPU is disabled, if this bit is set to 1 the behavior is Unpredictable.
[0]	ENABLE	Enables the MPU: 0 = MPU disabled 1 = MPU enabled.

When ENABLE and PRIVDEFENA are both set to 1:

© NXP B.V. 2010. All rights reserved.

- For privileged accesses, the **default memory map** is as described in Section 34–3.2 “Memory model”. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

XN and Strongly-ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set to 1, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFENA bit is set to 1. If the PRIVDEFENA bit is set to 1 and no regions are enabled, then only privileged software can operate.

When the ENABLE bit is set to 0, the system uses the default memory map. This has the same memory attributes as if the MPU is not implemented, see Table 34–636 “Memory access behavior”. The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFENA is set to 1.

Unless HFNMIENA is set to 1, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception, or when FAULTMASK is enabled. Setting the HFNMIENA bit to 1 enables the MPU when operating with these two priorities.

4.5.3 MPU Region Number Register

The RNR selects which memory region is referenced by the RBAR and RASR registers. See the register summary in Table 34–681 for its attributes. The bit assignments are shown in Table 34–684.

Table 684. RNR bit assignments

Bits	Name	Function
[31:8]	-	Reserved.
[7:0]	REGION	Indicates the MPU region referenced by the RBAR and RASR registers. The MPU supports 8 memory regions, so the permitted values of this field are 0-7.

Normally, you write the required region number to this register before accessing the RBAR or RASR. However you can change the region number by writing to the RBAR with the VALID bit set to 1, see Table 34–685. This write updates the value of the REGION field.

4.5.4 MPU Region Base Address Register

The RBAR defines the base address of the MPU region selected by the RNR, and can update the value of the RNR. See the register summary in Table 34–681 for its attributes.

Write RBAR with the VALID bit set to 1 to change the current region number and update the RNR. The bit assignments are shown in Table 34–685.

© NXP B.V. 2010. All rights reserved.

Table 685. RBAR bit assignments

[31:N]	ADDR	Region base address field. The value of N depends on the region size. For more information see Section 34–4.5.4.1.
[(N-1):5]	-	Reserved.
[4]	VALID	MPU Region Number valid bit: Write: 0 = RNR not changed, and the processor: • updates the base address for the region specified in the RNR • ignores the value of the REGION field 1 = the processor: • updates the value of the RNR to the value of the REGION field • updates the base address for the region specified in the REGION field. Always reads as zero.
[3:0]	REGION	MPU region field: For the behavior on writes, see the description of the VALID field. On reads, returns the current region number, as specified by the RNR.

4.5.4.1 The ADDR field

The ADDR field is bits[31:N] of the RBAR. The region size, as specified by the SIZE field in the RASR, defines the value of N:

$N = Log_2(\text{Region size in bytes})$,

If the region size is configured to 4GB, in the RASR, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is `0x00000000`.

The base address is aligned to the size of the region. For example, a 64KB region must be aligned on a multiple of 64KB, for example, at `0x00010000` or `0x00020000`.

4.5.5 MPU Region Attribute and Size Register

The RASR defines the region size and memory attributes of the MPU region specified by the RNR, and enables that region and any subregions. See the register summary in Table 34–681 for its attributes.

RASR is accessible using word or halfword accesses:

The bit assignments are shown in Table 34–686.

- the most significant halfword holds the region attributes
- the least significant halfword holds the region size and the region and subregion enable bits.

© NXP B.V. 2010. All rights reserved.

Table 686. RASR bit assignments

Bits	Name	Function
[31:29]	-	Reserved.
[28]	XN	Instruction access disable bit: 0 = instruction fetches enabled 1 = instruction fetches disabled.
[27]	-	Reserved.
[26:24]	AP	Access permission field, see Table 34–690.
[23:22]	-	Reserved.
[21:19, 17, 16]	TEX, C, B	Memory access attributes, see Table 34–688.
[18]	S	Shareable bit, see Table 34–688.
[15:8]	SRD	Subregion disable bits. For each bit in this field: 0 = corresponding sub-region is enabled 1 = corresponding sub-region is disabled See Section 34–4.5.8.3 for more information. Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, write the SRD field as `0x00`.
[7:6]	-	Reserved.
[5:1]	SIZE	Specifies the size of the MPU protection region. The minimum permitted value is 3 (b00010), see See Section 34–4.5.5.1 for more information.
[0]	ENABLE	Region enable bit.

For information about access permission, see Section 34–4.5.6.

4.5.5.1 SIZE field values

The SIZE field defines the size of the MPU memory region specified by the RNR. as follows:

$$\text{(Region size in bytes)} = 2^{(\text{SIZE}+1)}$$

The smallest permitted region size is 32B, corresponding to a SIZE value of 4. Table 34–687 gives example SIZE values, with the corresponding region size and value of N in the RBAR.

Table 687. Example SIZE field values

SIZE value	Region size	Value of N[1]	Note
b00100 (4)	32B	5	Minimum permitted size
b01001 (9)	1KB	10	-
b10011 (19)	1MB	20	-
b11101 (29)	1GB	30	-
b11111 (31)	4GB	b01100	Maximum possible size

[1] In the RBAR, see Table 34–685.

© NXP B.V. 2010. All rights reserved.

4.5.6 MPU access permission attributes

This section describes the MPU access permission attributes. The access permission bits, TEX, C, B, S, AP, and XN, of the RASR, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

Table 34–688 shows the encodings for the TEX, C, B, and S access permission bits.

Table 688. TEX, C, B, and S encoding

TEX	C	B	S	Memory type	Shareability	Other attributes
b000	0	0	x[1]	Strongly-ordered	Shareable	-
		1	x[1]	Device	Shareable	-
	1	0	0	Normal	Not shareable	Outer and inner write-through. No write allocate.
			1		Shareable	
		1	0	Normal	Not shareable	Outer and inner write-back. No write allocate.
			1		Shareable	
b001	0	0	0	Normal	Not shareable	Outer and inner noncacheable.
			1		Shareable	
		1	x[1]	Reserved encoding		-
	1	0	x[1]	Implementation defined attributes.		-
		1	0	Normal	Not shareable	Outer and inner write-back. Write and read allocate.
			1		Shareable	
b010	0	0	x[1]	Device	Not shareable	Nonshared Device.
		1	x[1]	Reserved encoding		-
	1	x[1]	x[1]	Reserved encoding		-
b1BB	A	A	0	Normal	Not shareable	Cached memory[2], BB = outer policy, AA = inner policy.
			1		Shareable	

[1] The MPU ignores the value of this bit.

[2] See Table 34–689 for the encoding of the AA and BB bits.

Table 34–689 shows the cache policy for memory attribute encodings with a TEX value is in the range 4-7.

Table 689. Cache policy for memory attribute encoding

Encoding, AA or BB	Corresponding cache policy
00	Non-cacheable
01	Write back, write and read allocate
10	Write through, no write allocate
11	Write back, no write allocate

Table 34–690 shows the AP encodings that define the access permissions for privileged and unprivileged software.

© NXP B.V. 2010. All rights reserved.

Table 690. AP encoding

AP[2:0]	Privileged permissions	Unprivileged permissions	Description
000	No access	No access	All accesses generate a permission fault
001	RW	No access	Access from privileged software only
010	RW	RO	Writes by unprivileged software generate a permission fault
011	RW	RW	Full access
100	Unpredictable	Unpredictable	Reserved
101	RO	No access	Reads by privileged software only
110	RO	RO	Read only, by privileged or unprivileged software
111	RO	RO	Read only, by privileged or unprivileged software

4.5.7 MPU mismatch

When an access violates the MPU permissions, the processor generates a memory management fault, see Section 34–3.1.4 “Exceptions and interrupts”. The MMFSR indicates the cause of the fault. See Table 34–669 for more information.

4.5.8 Updating an MPU region

To update the attributes for an MPU region, update the RNR, RBAR and RASR registers. You can program each register separately, or use a multiple-word write to program all of these registers. You can use the RBAR and RASR aliases to program up to four regions simultaneously using an `STM` instruction.

4.5.8.1 Updating an MPU region using separate words

Simple code to configure one region:

```
; R1 = region number

; R2 = size/enable

; R3 = attributes

; R4 = address

LDR R0,=MPU_RNR         ; 0xE000ED98, MPU region number register

STR R1, [R0, #0x0]      ; Region Number

STR R4, [R0, #0x4]      ; Region Base Address

STRH R2, [R0, #0x8]     ; Region Size and Enable

STRH R3, [R0, #0xA]     ; Region Attribute
```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```
; R1 = region number

; R2 = size/enable
```


© NXP B.V. 2010. All rights reserved.

```
; R3 = attributes

; R4 = address

LDR R0,=MPU_RNR          ; 0xE000ED98, MPU region number register

STR R1, [R0, #0x0]       ; Region Number

BIC R2, R2, #1           ; Disable

STRH R2, [R0, #0x8]      ; Region Size and Enable

STR R4, [R0, #0x4]       ; Region Base Address

STRH R3, [R0, #0xA]      ; Region Attribute

ORR R2, #1               ; Enable

STRH R2, [R0, #0x8]      ; Region Size and Enable
```

Software must use memory barrier instructions:

- before MPU setup if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings
- after MPU setup if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not need any memory barrier instructions during MPU setup, because it accesses the MPU through the PPB, which is a Strongly-Ordered memory region.

For example, if you want all of the memory access behavior to take effect immediately after the programming sequence, use a `DSB` instruction and an `ISB` instruction. A `DSB` is required after changing MPU settings, such as at the end of context switch. An `ISB` is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then you do not require an `ISB`.

4.5.8.2 Updating an MPU region using multi-word writes

You can program directly using multi-word writes, depending on how the information is divided. Consider the following reprogramming:

```
; R1 = region number

; R2 = address

; R3 = size, attributes in one

LDR R0, =MPU_RNR    ; 0xE000ED98, MPU region number register

STR R1, [R0, #0x0]  ; Region Number

STR R2, [R0, #0x4]  ; Region Base Address

STR R3, [R0, #0x8]  ; Region Attribute, Size and Enable
```


© NXP B.V. 2010. All rights reserved.

Use an STM instruction to optimize this:

```
; R1 = region number

; R2 = address

; R3 = size, attributes in one

LDR R0, =MPU_RNR    ; 0xE000ED98, MPU region number register

STM R0, {R1-R3}     ; Region Number, address, attribute, size and enable
```

You can do this in two words for pre-packed information. This means that the RBAR contains the required region number and had the VALID bit set to 1, see Table 34–685. Use this when the data is statically packed, for example in a boot loader:

```
; R1 = address and region number in one

; R2 = size and attributes in one

LDR R0, =MPU_RBAR   ; 0xE000ED9C, MPU Region Base register

STR R1, [R0, #0x0]  ; Region base address and

                    ; region number combined with VALID (bit 4) set to 1

STR R2, [R0, #0x4]  ; Region Attribute, Size and Enable
```

Use an STM instruction to optimize this:

```
; R1 = address and region number in one

; R2 = size and attributes in one

LDR R0,=MPU_RBAR    ; 0xE000ED9C, MPU Region Base register

STM R0, {R1-R2}     ; Region base address, region number and VALID bit,

                    ; and Region Attribute, Size and Enable
```

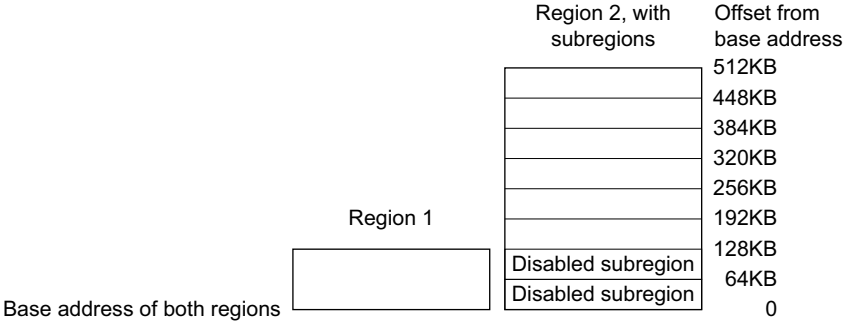
4.5.8.3 Subregions

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the RASR to disable a subregion, see Table 34–686. The least significant bit of SRD controls the first subregion, and the most significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions, With regions of these sizes, you must set the SRD field to 0x00, otherwise the MPU behavior is Unpredictable.

Example of SRD use: Two regions with the same base address overlap. Region one is 128KB, and region two is 512KB. To ensure the attributes from region one apply to the first128KB region, set the SRD field for region two to b00000011 to disable the first two subregions, as the figure shows.

© NXP B.V. 2010. All rights reserved.



4.5.9 MPU design hints and tips

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- except for the RASR, it must use aligned word accesses
- for the RASR it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

4.5.9.1 MPU configuration for a microcontroller

Usually, a microcontroller system has only a single processor and no caches. In such a system, program the MPU as follows:

Table 691. Memory region attributes for a microcontroller

Memory region	TEX	C	B	S	Memory type and attributes
Flash memory	b000	1	0	0	Normal memory, Non-shareable, write-through
Internal SRAM	b000	1	0	1	Normal memory, Shareable, write-through
External SRAM	b000	1	1	1	Normal memory, Shareable, write-back, write-allocate
Peripherals	b000	0	1	1	Device memory, Shareable

In most microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations. In special systems, such as multiprocessor designs or designs with a separate DMA engine, the shareability attribute might be important. In these cases refer to the recommendations of the memory device manufacturer.

© NXP B.V. 2010. All rights reserved.

5. ARM Cortex-M3 User Guide: Glossary

Abort — A mechanism that indicates to a processor that the value associated with a memory access is invalid. An abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction or data memory.

Aligned — A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

Banked register — A register that has multiple physical copies, where the state of the processor determines which copy is used. The Stack Pointer, SP (R13) is a banked register.

Base register — In instruction descriptions, a register specified by a load or store instruction that is used to hold the base value for the instruction's address calculation. Depending on the instruction and its addressing mode, an offset can be added to or subtracted from the base register value to form the address that is sent to memory. See also *Index register.*

Big-endian (BE) — Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory. See also *Byte-invariant, Endianness, Little-endian.*

Big-endian memory — Memory in which:

- a byte or halfword at a word-aligned address is the most significant byte or halfword within the word at that address
- a byte at a halfword-aligned address is the most significant byte within the halfword at that address.

See also *Little-endian memory*.

Breakpoint — A breakpoint is a mechanism provided by debuggers to identify an instruction at which program execution is to be halted. Breakpoints are inserted by the programmer to enable inspection of register contents, memory locations, variable values at fixed points in the program execution to test that the program is operating correctly. Breakpoints are removed after the program is successfully tested.

Byte-invariant — In a byte-invariant system, the address of each byte of memory remains unchanged when switching between little-endian and big-endian operation. When a data item larger than a byte is loaded from or stored to memory, the bytes making up that data item are arranged into the correct order depending on the endianness of the memory access.
An ARM byte-invariant implementation also supports unaligned halfword and word memory accesses. It expects multi-word accesses to be word-aligned.

Cache — A block of on-chip or off-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used instructions, data, or instructions and data. This is done to greatly increase the average speed of memory accesses and so improve processor performance.

© NXP B.V. 2010. All rights reserved.

Condition field — A four-bit field in an instruction that specifies a condition under which the instruction can execute.

Context — The environment that each process operates in for a multitasking operating system. In ARM processors, this is limited to mean the physical address range that it can access in memory and the associated memory access permissions.

Coprocessor — A processor that supplements the main processor. Cortex-M3 does not support any coprocessors.

Debugger — A debugging system that includes a program, used to detect, locate, and correct software faults, together with custom hardware that supports software debugging.

Direct Memory Access (DMA) — An operation that accesses main memory directly, without the processor performing any accesses to the data concerned.

Doubleword — A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.

Doubleword-aligned — A data item having a memory address that is divisible by eight.

Endianness — Byte ordering. The scheme that determines the order that successive bytes of a data word are stored in memory. An aspect of the system's memory mapping. See also *Little-endian* and *Big-endian*.

Exception — An event that interrupts program execution. When an exception occurs, the processor suspends the normal program flow and starts execution at the address indicated by the corresponding exception vector. The indicated address contains the first instruction of the handler for the exception.

An exception can be an interrupt request, a fault, or a software-generated system exception. Faults include attempting an invalid memory access, attempting to execute an instruction in an invalid processor state, and attempting to execute an undefined instruction.

Exception service routine — See also *Interrupt handler*.

Exception vector — See also *Interrupt vector.*

Flat address mapping — A system of organizing memory in which each physical address in the memory space is the same as the corresponding virtual address.

Halfword — A 16-bit data item.

Illegal instruction — An instruction that is architecturally Undefined.

Implementation-defined — The behavior is not architecturally defined, but is defined and documented by individual implementations.

Implementation-specific — The behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.

© NXP B.V. 2010. All rights reserved.

Index register — In some load and store instruction descriptions, the value of this register is used as an offset to be added to or subtracted from the base register value to form the address that is sent to memory. Some addressing modes optionally enable the index register value to be shifted prior to the addition or subtraction. See also *Base register*.

Instruction cycle count — The number of cycles that an instruction occupies the Execute stage of the pipeline.

Interrupt handler — A program that control of the processor is passed to when an interrupt occurs.

Interrupt vector — One of a number of fixed addresses in low memory, or in high memory if high vectors are configured, that contains the first instruction of the corresponding interrupt handler.

Little-endian (LE) — Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory. See also *Big-endian, Byte-invariant, Endianness.*

Little-endian memory — Memory in which:

- a byte or halfword at a word-aligned address is the least significant byte or halfword within the word at that address
- a byte at a halfword-aligned address is the least significant byte within the halfword at that address.

See also *Big-endian memory.*

Load/store architecture — A processor architecture where data-processing operations only operate on register contents, not directly on memory contents.

Memory Protection Unit (MPU) — Hardware that controls access permissions to blocks of memory. An MPU does not perform any address translation.

Prefetching — In pipelined processors, the process of fetching instructions from memory to fill up the pipeline before the preceding instructions have finished executing. Prefetching an instruction does not mean that the instruction has to be executed.

Read — Reads are defined as memory operations that have the semantics of a load. Reads include the Thumb instructions `LDM`, `LDR`, `LDRSH`, `LDRH`, `LDRSB`, `LDRB`, and `POP`.

Region — A partition of memory space.

Reserved — A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

Should Be One (SBO) — Write as 1, or all 1s for bit fields, by software. Writing as 0 produces Unpredictable results.

Should Be Zero (SBZ) — Write as 0, or all 0s for bit fields, by software. Writing as 1 produces Unpredictable results.

© NXP B.V. 2010. All rights reserved.

Should Be Zero or Preserved (SBZP) — Write as 0, or all 0s for bit fields, by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.

Thread-safe — In a multi-tasking environment, thread-safe functions use safeguard mechanisms when accessing shared resources, to ensure correct operation without the risk of shared access conflicts.

Thumb instruction — One or two halfwords that specify an operation for a processor to perform. Thumb instructions must be halfword-aligned.

Unaligned — A data item stored at an address that is not divisible by the number of bytes that defines the data size is said to be unaligned. For example, a word stored at an address that is not divisible by four.

Unpredictable (UNP) — You cannot rely on the behavior. Unpredictable behavior must not represent security holes. Unpredictable behavior must not halt or hang the processor, or any parts of the system.

Warm reset — Also known as a core reset. Initializes the majority of the processor excluding the debug controller and debug logic. This type of reset is useful if you are using the debugging features of a processor.

WA — See *Write-allocate*.

WB — See *Write-back*.

Word — A 32-bit data item.

Write — Writes are defined as operations that have the semantics of a store. Writes include the Thumb instructions `STM`, `STR`, `STRH`, `STRB`, and `PUSH`.

Write-allocate (WA) — In a write-allocate cache, a cache miss on storing data causes a cache line to be allocated into the cache.

Write-back (WB) — In a write-back cache, data is only written to main memory when it is forced out of the cache on line replacement following a cache miss. Otherwise, writes by the processor only update the cache. This is also known as copyback.

Write buffer — A block of high-speed memory, arranged as a FIFO buffer, between the data cache and main memory, whose purpose is to optimize stores to main memory.

Write-through (WT) — In a write-through cache, data is written to main memory at the same time as the cache is updated.

© NXP B.V. 2010. All rights reserved.

1. Abbreviations

Table 692. Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
CAN	Controller Area Network
DAC	Digital-to-Analog Converter
DCC	Debug Communication Channel
DMA	Direct Memory Access
DSP	Digital Signal Processing
EOP	End Of Packet
ETM	Embedded Trace Macrocell
GPIO	General Purpose Input/Output
IrDA	Infrared Data Association
JTAG	Joint Test Action Group
MIIM	Media Independent Interface Management
PHY	Physical Layer
PLL	Phase-Locked Loop
PWM	Pulse Width Modulator
RMII	Reduced Media Independent Interface
SE0	Single Ended Zero
SPI	Serial Peripheral Interface
SSI	Serial Synchronous Interface
SSP	Synchronous Serial Port
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

© NXP B.V. 2010. All rights reserved.

2. Legal information

2.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

2.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

2.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^2C-bus — logo is a trademark of NXP B.V.

© NXP B.V. 2010. All rights reserved.

Notes

continued >>

© NXP B.V. 2010. All rights reserved.

3. Tables

Table 1. Ordering information 7
Table 2. Ordering options for LPC17xx parts 7
Table 3. LPC17xx memory usage and details 12
Table 4. APB0 peripherals and base addresses 14
Table 5. APB1 peripherals and base addresses 15
Table 6. Pin summary. 17
Table 7. Summary of system control registers 18
Table 8. Reset Source Identification register (RSID - address 0x400F C180) bit description 21
Table 9. External Interrupt registers 24
Table 10. External Interrupt Flag register (EXTINT - address 0x400F C140) bit description 25
Table 11. External Interrupt Mode register (EXTMODE - address 0x400F C148) bit description 26
Table 12. External Interrupt Polarity register (EXTPOLAR - address 0x400F C14C) bit description 26
Table 13. System Controls and Status register (SCS - address 0x400F C1A0) bit description 28
Table 14. Summary of system control registers 30
Table 15. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) low frequency mode (OSCRANGE = 0, see Table 3–13) 32
Table 16. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters) high frequency mode (OSCRANGE = 1, see Table 3–13) 32
Table 17. Clock Source Select register (CLKSRCSEL - address 0x400F C10C) bit description 34
Table 18. PLL0 registers 36
Table 19. PLL Control register (PLL0CON - address 0x400F C080) bit description 37
Table 20. PLL0 Configuration register (PLL0CFG - address 0x400F C084) bit description 37
Table 21. Multiplier values for PLL0 with a 32 kHz input . 38
Table 22. PLL Status register (PLL0STAT - address 0x400F C088) bit description 39
Table 23. PLL control bit combinations 40
Table 24. PLL Feed register (PLL0FEED - address 0x400F C08C) bit description 40
Table 25. PLL frequency parameter 41
Table 26. Additional Multiplier Values for use with a Low Frequency Clock Input 42
Table 27. Summary of PLL0 examples 43
Table 28. Potential values for PLL example 45
Table 29. PLL1 registers 47
Table 30. PLL1 Control register (PLL1CON - address 0x400F C0A0) bit description 50
Table 31. PLL Configuration register (PLL1CFG - address 0x400F C0A4) bit description. 50
Table 32. PLL1 Status register (PLL1STAT - address 0x400F C0A8) bit description. 51
Table 33. PLL1 control bit combinations 51
Table 34. PLL1 Feed register (PLL1FEED - address 0x400F C0AC) bit description 52
Table 35. Elements determining PLL frequency 53
Table 36. PLL1 Divider values 54
Table 37. PLL1 Multiplier values 54
Table 38. CPU Clock Configuration register (CCLKCFG - address 0x400F C104) bit description 55
Table 39. USB Clock Configuration register (USBCLKCFG - address 0x400F C108) bit description 56
Table 40. Peripheral Clock Selection register 0 (PCLKSEL0 - address 0x400F C1A8) bit description. 57
Table 41. Peripheral Clock Selection register 1 (PCLKSEL1 - address 0x400F C1AC) bit description 57
Table 42. Peripheral Clock Selection register bit values . 58
Table 43. Power Control registers 61
Table 44. Power Mode Control register (PCON - address 0x400F C0C0) bit description 62
Table 45. Encoding of reduced power modes 63
Table 46. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description. 64
Table 47. Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8) bit description 67
Table 48. Summary of flash accelerator registers 70
Table 49. Flash Accelerator Configuration register (FLASHCFG - address 0x400F C000) bit description 71
Table 50. Connection of interrupt sources to the Vectored Interrupt Controller 74
Table 51. NVIC register map 77
Table 52. Interrupt Set-Enable Register 0 register (ISER0 - 0xE000 E100) 78
Table 53. Interrupt Set-Enable Register 1 register (ISER1 - 0xE000 E104) 79
Table 54. Interrupt Clear-Enable Register 0 (ICER0 - 0xE000 E180) 80
Table 55. Interrupt Clear-Enable Register 1 register (ICER1 - 0xE000 E184) 81
Table 56. Interrupt Set-Pending Register 0 register (ISPR0 - 0xE000 E200) 82
Table 57. Interrupt Set-Pending Register 1 register (ISPR1 - 0xE000 E204) 83
Table 58. Interrupt Clear-Pending Register 0 register (ICPR0 - 0xE000 E280) 84

continued >>

© NXP B.V. 2010. All rights reserved.

Table 59. Interrupt Set-Pending Register 1 register (ISPR1 - 0xE000 E204) 85
Table 60. Interrupt Active Bit Register 0 (IABR0 - 0xE000 E300) 86
Table 61. Interrupt Active Bit Register 1 (IABR1 - 0xE000 E304) 87
Table 62. Interrupt Priority Register 0 (IPR0 - 0xE000 E400) 88
Table 63. Interrupt Priority Register 1 (IPR1 - 0xE000 E404) 88
Table 64. Interrupt Priority Register 2 (IPR2 - 0xE000 E408) 88
Table 65. Interrupt Priority Register 3 (IPR3 - 0xE000 E40C) 89
Table 66. Interrupt Priority Register 4 (IPR4 - 0xE000 E410) 89
Table 67. Interrupt Priority Register 5 (IPR5 - 0xE000 E414) 89
Table 68. Interrupt Priority Register 6 (IPR6 - 0xE000 E418) 90
Table 69. Interrupt Priority Register 7 (IPR7 - 0xE000 E41C) 90
Table 70. Interrupt Priority Register 8 (IPR8 - 0xE000 E420) 90
Table 71. Software Trigger Interrupt Register (STIR - 0xE000 EF00) 91
Table 72. Pin description 93
Table 73. Summary of PINSEL registers 102
Table 74. Pin function select register bits 102
Table 75. Pin Mode Select register Bits 103
Table 76. Open Drain Pin Mode Select register Bits . . .104
Table 77. Pin Connect Block Register Map 105
Table 78. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description 106
Table 79. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description 106
Table 80. Pin function select register 2 (PINSEL2 - address 0x4002 C008) bit description 107
Table 81. Pin function select register 3 (PINSEL3 - address 0x4002 C00C) bit description 107
Table 82. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description 108
Table 83. Pin function select register 7 (PINSEL7 - address 0x4002 C01C) bit description 109
Table 84. Pin function select register 9 (PINSEL9 - address 0x4002 C024) bit description 109
Table 85. Pin function select register 10 (PINSEL10 - address 0x4002 C028) bit description 109
Table 86. Pin Mode select register 0 (PINMODE0 - address 0x4002 C040) bit description 110
Table 87. Pin Mode select register 1 (PINMODE1 - address 0x4002 C044) bit description 110
Table 88. Pin Mode select register 2 (PINMODE2 - address 0x4002 C048) bit description 111
Table 89. Pin Mode select register 3 (PINMODE3 - address 0x4002 C04C) bit description. 111
Table 90. Pin Mode select register 4 (PINMODE4 - address 0x4002 C050) bit description 112
Table 91. Pin Mode select register 7 (PINMODE7 - address 0x4002 C05C) bit description. 113
Table 92. Pin Mode select register 9 (PINMODE9 - address 0x4002 C064) bit description 113
Table 93. Open Drain Pin Mode select register 0 (PINMODE_OD0 - address 0x4002 C068) bit description 113
Table 94. Open Drain Pin Mode select register 1 (PINMODE_OD1 - address 0x4002 C06C) bit description 114
Table 95. Open Drain Pin Mode select register 2 (PINMODE_OD2 - address 0x4002 C070) bit description 115
Table 96. Open Drain Pin Mode select register 3 (PINMODE_OD3 - address 0x4002 C074) bit description 116
Table 97. Open Drain Pin Mode select register 4 (PINMODE_OD4 - address 0x4002 C078) bit description 116
Table 98. I2C Pin Configuration register (I2CPADCFG - address 0x4002 C07C) bit description. 117
Table 99. GPIO pin description 119
Table 100. GPIO register map (local bus accessible registers - enhanced GPIO features) 120
Table 101. GPIO interrupt register map. 121
Table 102. Fast GPIO port Direction register FIO0DIR to FIO4DIR - addresses 0x2009 C000 to 0x2009 C080) bit description 121
Table 103. Fast GPIO port Direction control byte and half-word accessible register description. . . . 122
Table 104. Fast GPIO port output Set register (FIO0SET to FIO4SET - addresses 0x2009 C018 to 0x2009 C098) bit description 123
Table 105. Fast GPIO port output Set byte and half-word accessible register description. 123
Table 106. Fast GPIO port output Clear register (FIO0CLR to FIO4CLR- addresses 0x2009 C01C to 0x2009 C09C) bit description 124
Table 107. Fast GPIO port output Clear byte and half-word accessible register description. 124
Table 108. Fast GPIO port Pin value register (FIO0PIN to FIO4PIN- addresses 0x2009 C014 to 0x2009 C094) bit description 125
Table 109. Fast GPIO port Pin value byte and half-word

continued >>

© NXP B.V. 2010. All rights reserved.

accessible register description 126
Table 110. Fast GPIO port Mask register (FIO0MASK to FIO4MASK - addresses 0x2009 C010 to 0x2009 C090) bit description. 127
Table 111. Fast GPIO port Mask byte and half-word accessible register description 127
Table 112. GPIO overall Interrupt Status register (IOIntStatus - address 0x4002 8080) bit description 129
Table 113. GPIO Interrupt Enable for port 0 Rising Edge (IO0IntEnR - 0x4002 8090) bit description . . .129
Table 114. GPIO Interrupt Enable for port 2 Rising Edge (IO2IntEnR - 0x4002 80B0) bit description. . .130
Table 115. GPIO Interrupt Enable for port 0 Falling Edge (IO0IntEnF - address 0x4002 8094) bit description 131
Table 116. GPIO Interrupt Enable for port 2 Falling Edge (IO2IntEnF - 0x4002 80B4) bit description . . .132
Table 117. GPIO Interrupt Status for port 0 Rising Edge Interrupt (IO0IntStatR - 0x4002 8084) bit description 133
Table 118. GPIO Interrupt Status for port 2 Rising Edge Interrupt (IO2IntStatR - 0x4002 80A4) bit description 134
Table 119. GPIO Interrupt Status for port 0 Falling Edge Interrupt (IO0IntStatF - 0x4002 8088) bit description 134
Table 120. GPIO Interrupt Status for port 2 Falling Edge Interrupt (IO2IntStatF - 0x4002 80A8) bit description 135
Table 121. GPIO Interrupt Clear register for port 0 (IO0IntClr - 0x4002 808C)) bit description 136
Table 122. GPIO Interrupt Clear register for port 0 (IO2IntClr - 0x4002 80AC) bit description 137
Table 123. Ethernet acronyms, abbreviations, and definitions 139
Table 124. Example PHY Devices 145
Table 125. Ethernet RMII pin descriptions 145
Table 126. Ethernet MIIM pin descriptions 145
Table 127. Ethernet register definitions 146
Table 128. MAC Configuration register 1 (MAC1 - address 0x5000 0000) bit description 148
Table 129. MAC Configuration register 2 (MAC2 - address 0x5000 0004) bit description 149
Table 130. Pad operation 150
Table 131. Back-to-back Inter-packet-gap register (IPGT - address 0x5000 0008) bit description. 150
Table 132. Non Back-to-back Inter-packet-gap register (IPGR - address 0x5000 000C) bit description 150
Table 133. Collision Window / Retry register (CLRT - address 0x5000 0010) bit description 151
Table 134. Maximum Frame register (MAXF - address 0x5000 0014) bit description 151
Table 135. PHY Support register (SUPP - address 0x5000 0018) bit description 151
Table 136. Test register (TEST - address 0x5000) bit description 152
Table 137. MII Mgmt Configuration register (MCFG - address 0x5000 0020) bit description 152
Table 138. Clock select encoding 152
Table 139. MII Mgmt Command register (MCMD - address 0x5000 0024) bit description 153
Table 140. MII Mgmt Address register (MADR - address 0x5000 0028) bit description 153
Table 141. MII Mgmt Write Data register (MWTD - address 0x5000 002C) bit description 154
Table 142. MII Mgmt Read Data register (MRDD - address 0x5000 0030) bit description 154
Table 143. MII Mgmt Indicators register (MIND - address 0x5000 0034) bit description 154
Table 144. Station Address register (SA0 - address 0x5000 0040) bit description 155
Table 145. Station Address register (SA1 - address 0x5000 0044) bit description 155
Table 146. Station Address register (SA2 - address 0x5000 0048) bit description 155
Table 147. Command register (Command - address 0x5000 0100) bit description 156
Table 148. Status register (Status - address 0x5000 0104) bit description 156
Table 149. Receive Descriptor Base Address register (RxDescriptor - address 0x5000 0108) bit description 157
Table 150. receive Status Base Address register (RxStatus - address 0x5000 010C) bit description 157
Table 151. Receive Number of Descriptors register (RxDescriptor - address 0x5000 0110) bit description 158
Table 152. Receive Produce Index register (RxProduceIndex - address 0x5000 0114) bit description 158
Table 153. Receive Consume Index register (RxConsumeIndex - address 0x5000 0118) bit description 158
Table 154. Transmit Descriptor Base Address register (TxDescriptor - address 0x5000 011C) bit description 159
Table 155. Transmit Status Base Address register (TxStatus - address 0x5000 0120) bit description 159
Table 156. Transmit Number of Descriptors register (TxDescriptorNumber - address 0x5000 0124) bit description 159
Table 157. Transmit Produce Index register (TxProduceIndex

continued >>

© NXP B.V. 2010. All rights reserved.

- address 0x5000 0128) bit description 160
Table 158. Transmit Consume Index register (TxConsumeIndex - address 0x5000 012C) bit description 160
Table 159. Transmit Status Vector 0 register (TSV0 - address 0x5000 0158) bit description 161
Table 160. Transmit Status Vector 1 register (TSV1 - address 0x5000 015C) bit description 162
Table 161. Receive Status Vector register (RSV - address 0x5000 0160) bit description 162
Table 162. Flow Control Counter register (FlowControlCounter - address 0x5000 0170) bit description 163
Table 163. Flow Control Status register (FlowControlStatus - address 0x5000 8174) bit description. 163
Table 164. Receive Filter Control register (RxFilterCtrl - address 0x5000 0200) bit description. 164
Table 165. Receive Filter WoL Status register (RxFilterWoLStatus - address 0x5000 0204) bit description 164
Table 166. Receive Filter WoL Clear register (RxFilterWoLClear - address 0x5000 0208) bit description 165
Table 167. Hash Filter Table LSBs register (HashFilterL - address 0x5000 0210) bit description. 165
Table 168. Hash Filter MSBs register (HashFilterH - address 0x5000 0214) bit description 166
Table 169. Interrupt Status register (IntStatus - address 0x5000 0FE0) bit description 166
Table 170. Interrupt Enable register (intEnable - address 0x5000 0FE4) bit description 167
Table 171. Interrupt Clear register (IntClear - address 0x5000 0FE8) bit description 168
Table 172. Interrupt Set register (IntSet - address 0x5000 0FEC) bit description 168
Table 173. Power-Down register (PowerDown - address 0x5000 0FF4) bit description 169
Table 174. Receive Descriptor Fields. 171
Table 175. Receive Descriptor Control Word 171
Table 176. Receive Status Fields 171
Table 177. Receive Status HashCRC Word 172
Table 178. Receive status information word. 172
Table 179. Transmit descriptor fields 174
Table 180. Transmit descriptor control word 174
Table 181. Transmit status fields 174
Table 182. Transmit status information word 175
Table 183. USB related acronyms, abbreviations, and definitions used in this chapter 212
Table 184. Fixed endpoint configuration. 213
Table 185. USB external interface 216
Table 186. USB device controller clock sources 217
Table 187. USB device register map 218
Table 188. USBClkCtrl register (USBClkCtrl - address 0x5000 CFF4) bit description. 219
Table 189. USB Clock Status register (USBClkSt - address 0x5000 CFF8) bit description. 220
Table 190. USB Interrupt Status register (USBIntSt - address 0x5000 C1C0) bit description. 220
Table 191. USB Device Interrupt Status register (USBDevIntSt - address 0x5000 C200) bit allocation 221
Table 192. USB Device Interrupt Status register (USBDevIntSt - address 0x5000 C200) bit description 221
Table 193. USB Device Interrupt Enable register (USBDevIntEn - address 0x5000 C204) bit allocation 222
Table 194. USB Device Interrupt Enable register (USBDevIntEn - address 0x5000 C204) bit description 222
Table 195. USB Device Interrupt Clear register (USBDevIntClr - address 0x5000 C208) bit allocation 222
Table 196. USB Device Interrupt Clear register (USBDevIntClr - address 0x5000 C208) bit description 223
Table 197. USB Device Interrupt Set register (USBDevIntSet - address 0x5000 C20C) bit allocation 223
Table 198. USB Device Interrupt Set register (USBDevIntSet - address 0x5000 C20C) bit description 223
Table 199. USB Device Interrupt Priority register (USBDevIntPri - address 0x5000 C22C) bit description 224
Table 200. USB Endpoint Interrupt Status register (USBEpIntSt - address 0x5000 C230) bit allocation 224
Table 201. USB Endpoint Interrupt Status register (USBEpIntSt - address 0x5000 C230) bit description 224
Table 202. USB Endpoint Interrupt Enable register (USBEpIntEn - address 0x5000 C234) bit allocation 225
Table 203. USB Endpoint Interrupt Enable register (USBEpIntEn - address 0x5000 C234) bit description 226
Table 204. USB Endpoint Interrupt Clear register (USBEpIntClr - address 0x5000 C238) bit allocation 226
Table 205. USB Endpoint Interrupt Clear register (USBEpIntClr - address 0x5000 C238) bit description 227
Table 206. USB Endpoint Interrupt Set register (USBEpIntSet

continued >>

© NXP B.V. 2010. All rights reserved.

- address 0x5000 C23C) bit allocation 227
Table 207. USB Endpoint Interrupt Set register (USBEpIntSet - address 0x5000 C23C) bit description 227
Table 208. USB Endpoint Interrupt Priority register (USBEpIntPri - address 0x5000 C240) bit allocation 227
Table 209. USB Endpoint Interrupt Priority register (USBEpIntPri - address 0x5000 C240) bit description 228
Table 210. USB Realize Endpoint register (USBReEp - address 0x5000 C244) bit allocation 229
Table 211. USB Realize Endpoint register (USBReEp - address 0x5000 C244) bit description. 229
Table 212. USB Endpoint Index register (USBEpIn - address 0x5000 C248) bit description. 230
Table 213. USB MaxPacketSize register (USBMaxPSize - address 0x5000 C24C) bit description. 230
Table 214. USB Receive Data register (USBRxData - address 0x5000 C218) bit description. 231
Table 215. USB Receive Packet Length register (USBRxPlen - address 0x5000 C220) bit description. 231
Table 216. USB Transmit Data register (USBTxData - address 0x5000 C21C) bit description. 231
Table 217. USB Transmit Packet Length register (USBTxPLen - address 0x5000 C224) bit description 232
Table 218. USB Control register (USBCtrl - address 0x5000 C228) bit description 232
Table 219. USB Command Code register (USBCmdCode - address 0x5000 C210) bit description. 233
Table 220. USB Command Data register (USBCmdData - address 0x5000 C214) bit description. 233
Table 221. USB DMA Request Status register (USBDMARSt - address 0x5000 C250) bit allocation 233
Table 222. USB DMA Request Status register (USBDMARSt - address 0x5000 C250) bit description. 234
Table 223. USB DMA Request Clear register (USBDMARClr - address 0x5000 C254) bit description. 234
Table 224. USB DMA Request Set register (USBDMARSet - address 0x5000 C258) bit description. 235
Table 225. USB UDCA Head register (USBUDCAH - address 0x5000 C280) bit description. 235
Table 226. USB EP DMA Status register (USBEpDMASt - address 0x5000 C284) bit description. 235
Table 227. USB EP DMA Enable register (USBEpDMAEn - address 0x5000 C288) bit description. 236
Table 228. USB EP DMA Disable register (USBEpDMADis - address 0x5000 C28C) bit description. 236
Table 229. USB DMA Interrupt Status register (USBDMAIntSt - address 0x5000 C290) bit description. 236
Table 230. USB DMA Interrupt Enable register (USBDMAIntEn - address 0x5000 C294) bit description 237
Table 231. USB End of Transfer Interrupt Status register (USBEoTIntSt - address 0x5000 C2A0s) bit description 237
Table 232. USB End of Transfer Interrupt Clear register (USBEoTIntClr - address 0x5000 C2A4) bit description 238
Table 233. USB End of Transfer Interrupt Set register (USBEoTIntSet - address 0x5000 C2A8) bit description 238
Table 234. USB New DD Request Interrupt Status register (USBNDDRIntSt - address 0x5000 C2AC) bit description 238
Table 235. USB New DD Request Interrupt Clear register (USBNDDRIntClr - address 0x5000 C2B0) bit description 238
Table 236. USB New DD Request Interrupt Set register (USBNDDRIntSet - address 0x5000 C2B4) bit description 239
Table 237. USB System Error Interrupt Status register (USBSysErrIntSt - address 0x5000 C2B8) bit description 239
Table 238. USB System Error Interrupt Clear register (USBSysErrIntClr - address 0x5000 C2BC) bit description 239
Table 239. USB System Error Interrupt Set register (USBSysErrIntSet - address 0x5000 C2C0) bit description 239
Table 240. SIE command code table. 243
Table 241. Set Address command bit description 243
Table 242. Configure Device command bit description . . 244
Table 243. Set Mode command bit description 244
Table 244. Set Device Status command bit description. . 245
Table 245. Get Error Code command bit description. . . . 247
Table 246. Read Error Status command bit description . 247
Table 247. Select Endpoint command bit description . . . 248
Table 248. Set Endpoint Status command bit description 249
Table 249. Clear Buffer command bit description 250
Table 250. DMA descriptor 255
Table 251. USB (OHCI) related acronyms and abbreviations used in this chapter 267
Table 252. USB Host port pins 269
Table 253. USB Host register address definitions 269
Table 254. USB OTG port pins 273
Table 255. USB OTG and I2C register address definitions. 275
Table 256. USB Interrupt Status register - (USBIntSt - address 0x5000 C1C0) bit description. 275
Table 257. OTG Interrupt Status register (OTGIntSt - address 0x5000 C100) bit description 276

continued >>

© NXP B.V. 2010. All rights reserved.

Table 258. OTG Status Control register (OTGStCtrl - address 0x5000 C110) bit description 277
Table 259. OTG Timer register (OTGTmr - address 0x5000 C114) bit description 278
Table 260. OTG clock control register (OTG_clock_control - address 0x5000 CFF4) bit description 278
Table 261. OTG clock status register (OTGClkSt - address 0x5000 CFF8) bit description 279
Table 262. I2C Receive register (I2C_RX - address 0x5000 C300) bit description 280
Table 263. I2C Transmit register (I2C_TX - address 0x5000 C300) bit description 280
Table 264. I2C status register (I2C_STS - address 0x5000 C304) bit description 280
Table 265. I2C Control register (I2C_CTL - address 0x5000 C308) bit description 282
Table 266. I2C_CLKHI register (I2C_CLKHI - address 0x5000 C30C) bit description 283
Table 267. I2C_CLKLO register (I2C_CLKLO - address 0x5000 C310) bit description 283
Table 268: UARTn Pin description 297
Table 269. UART0/2/3 Register Map 298
Table 270: UARTn Receiver Buffer Register (U0RBR - address 0x4000 C000, U2RBR - 0x4009 8000, U3RBR - 04009 C000 when DLAB = 0) bit description 299
Table 271: UARTn Transmit Holding Register (U0THR - address 0x4000 C000, U2THR - 0x4009 8000, U3THR - 0x4009 C000 when DLAB = 0) bit description 299
Table 272: UARTn Divisor Latch LSB register (U0DLL - address 0x4000 C000, U2DLL - 0x4009 8000, U3DLL - 0x4009 C000 when DLAB = 1) bit description 300
Table 273: UARTn Divisor Latch MSB register (U0DLM - address 0x4000 C004, U2DLM - 0x4009 8004, U3DLM - 0x4009 C004 when DLAB = 1) bit description 300
Table 274: UARTn Interrupt Enable Register (U0IER - address 0x4000 C004, U2IER - 0x4009 8004, U3IER - 0x4009 C004 when DLAB = 0) bit description 300
Table 275: UARTn Interrupt Identification Register (U0IIR - address 0x4000 C008, U2IIR - 0x4009 8008, U3IIR - 0x4009 C008) bit description 301
Table 276: UARTn Interrupt Handling 302
Table 277: UARTn FIFO Control Register (U0FCR - address 0x4000 C008, U2FCR - 0x4009 8008, U3FCR - 0x4007 C008) bit description 303
Table 278: UARTn Line Control Register (U0LCR - address 0x4000 C00C, U2LCR - 0x4009 800C, U3LCR - 0x4009 C00C) bit description. 304
Table 279: UARTn Line Status Register (U0LSR - address 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014) bit description 305
Table 280: UARTn Scratch Pad Register (U0SCR - address 0x4000 C01C, U2SCR - 0x4009 801C, U3SCR - 0x4009 C01C) bit description. 306
Table 281: UARTn Auto-baud Control Register (U0ACR - address 0x4000 C020, U2ACR - 0x4009 8020, U3ACR - 0x4009 C020) bit description 306
Table 282: UARTn IrDA Control Register (U0ICR - 0x4000 C024, U2ICR - 0x4009 8024, U3ICR - 0x4009 C024) bit description 309
Table 283: IrDA Pulse Width 310
Table 284: UARTn Fractional Divider Register (U0FDR - address 0x4000 C028, U2FDR - 0x4009 8028, U3FDR - 0x4009 C028) bit description 310
Table 285. Fractional Divider setting look-up table 313
Table 286: UARTn Transmit Enable Register (U0TER - address 0x4000 C030, U2TER - 0x4009 8030, U3TER - 0x4009 C030) bit description 314
Table 287. UARTn FIFO Level register (U0FIFOLVL - 0x4000 C058, U2FIFOLVL - 0x4009 8058, U3FIFOLVL - 0x4009 C058) bit description 314
Table 288: UART1 Pin Description 317
Table 289: UART1 register map 318
Table 290: UART1 Receiver Buffer Register (U1RBR - address 0x4001 0000 when DLAB = 0) bit description 319
Table 291: UART1 Transmitter Holding Register (U1THR - address 0x4001 0000 when DLAB = 0) bit description 319
Table 292: UART1 Divisor Latch LSB Register (U1DLL - address 0x4001 0000 when DLAB = 1) bit description 320
Table 293: UART1 Divisor Latch MSB Register (U1DLM - address 0x4001 0004 when DLAB = 1) bit description 320
Table 294: UART1 Interrupt Enable Register (U1IER - address 0x4001 0004 when DLAB = 0) bit description 320
Table 295: UART1 Interrupt Identification Register (U1IIR - address 0x4001 0008) bit description 321
Table 296: UART1 Interrupt Handling 322
Table 297: UART1 FIFO Control Register (U1FCR - address 0x4001 0008) bit description 323
Table 298: UART1 Line Control Register (U1LCR - address 0x4001 000C) bit description 324
Table 299: UART1 Modem Control Register (U1MCR - address 0x4001 0010) bit description 325
Table 300: Modem status interrupt generation 326

continued >>

© NXP B.V. 2010. All rights reserved.

Table 301: UART1 Line Status Register (U1LSR - address 0x4001 0014) bit description 327
Table 302: UART1 Modem Status Register (U1MSR - address 0x4001 0018) bit description. 328
Table 303: UART1 Scratch Pad Register (U1SCR - address 0x4001 0014) bit description 329
Table 304: Auto-baud Control Register (U1ACR - address 0x4001 0020) bit description 329
Table 305: UART1 Fractional Divider Register (U1FDR - address 0x4001 0028) bit description. 333
Table 306. Fractional Divider setting look-up table. 335
Table 307: UART1 Transmit Enable Register (U1TER - address 0x4001 0030) bit description. 336
Table 308: UART1 RS485 Control register (U1RS485CTRL - address 0x4001 004C) bit description 336
Table 309. UART1 RS-485 Address Match register (U1RS485ADRMATCH - address 0x4001 0050) bit description 337
Table 310. UART1 RS-485 Delay value register (U1RS485DLY - address 0x4001 0054) bit description 337
Table 311. UART1 FIFO Level register (U1FIFOLVL - address 0x4001 0058) bit description. 339
Table 312. CAN Pin descriptions 342
Table 313. Memory map of the CAN block. 347
Table 314. CAN acceptance filter and central CAN registers 347
Table 315. CAN1 and CAN2 controller register map 347
Table 316. CAN1 and CAN2 controller register summary 349
Table 317. CAN Wake and Sleep registers 349
Table 318. CAN Mode register (CAN1MOD - address 0x4004 4000, CAN2MOD - address 0x4004 8000) bit description 350
Table 319. CAN Command Register (CAN1CMR - address 0x4004 4004, CAN2CMR - address 0x4004 8004) bit description 351
Table 320. CAN Global Status Register (CAN1GSR - address 0x4004 4008, CAN2GSR - address 0x4004 8008) bit description 353
Table 321. CAN Interrupt and Capture Register (CAN1ICR - address 0x4004 400C, CAN2ICR - address 0x4004 800C) bit description 355
Table 322. CAN Interrupt Enable Register (CAN1IER - address 0x4004 4010, CAN2IER - address 0x4004 8010) bit description 358
Table 323. CAN Bus Timing Register (CAN1BTR - address 0x4004 4014, CAN2BTR - address 0x4004 8014) bit description 359
Table 324. CAN Error Warning Limit register (CAN1EWL - address 0x4004 4018, CAN2EWL - address 0x4004 8018) bit description 361
Table 325. CAN Status Register (CAN1SR - address 0x4004 401C, CAN2SR - address 0x4004 801C) bit description. 361
Table 326. CAN Receive Frame Status register (CAN1RFS - address 0x4004 4020, CAN2RFS - address 0x4004 8020) bit description 363
Table 327. CAN Receive Identifier register (CAN1RID - address 0x4004 4024, CAN2RID - address 0x4004 8024) bit description 364
Table 328. RX Identifier register when FF = 1 364
Table 329. CAN Receive Data register A (CAN1RDA - address 0x4004 4028, CAN2RDA - address 0x4004 8028) bit description 364
Table 330. CAN Receive Data register B (CAN1RDB - address 0x4004 402C, CAN2RDB - address 0x4004 802C) bit description 364
Table 331. CAN Transmit Frame Information register (CAN1TFI[1/2/3] - address 0x4004 40[30/40/50], CAN2TFI[1/2/3] - 0x4004 80[30/40/50]) bit description 365
Table 332. CAN Transfer Identifier register (CAN1TID[1/2/3] - address 0x4004 40[34/44/54], CAN2TID[1/2/3] - address 0x4004 80[34/44/54]) bit description 366
Table 333. Transfer Identifier register when FF = 1. 367
Table 334. CAN Transmit Data register A (CAN1TDA[1/2/3] - address 0x4004 40[38/48/58], CAN2TDA[1/2/3] - address 0x4004 80[38/48/58]) bit description 367
Table 335. CAN Transmit Data register B (CAN1TDB[1/2/3] - address 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3] - address 0x4004 80[3C/4C/5C]) bit description 367
Table 336. CAN Sleep Clear register (CANSLEEPCLR - address 0x400F C110) bit description 368
Table 337. CAN Wake-up Flags register (CANWAKEFLAGS - address 0x400F C114) bit description 368
Table 338. Central Transit Status Register (CANTxSR - address 0x4004 0000) bit description 370
Table 339. Central Receive Status Register (CANRxSR - address 0x4004 0004) bit description 370
Table 340. Central Miscellaneous Status Register (CANMSR - address 0x4004 0008) bit description 371
Table 341. Acceptance filter modes and access control . 371
Table 342. Section configuration register settings. 372
Table 343. Acceptance Filter Mode Register (AFMR - address 0x4003 C000) bit description 375
Table 344. Standard Frame Individual Start Address register (SFF_sa - address 0x4003 C004) bit description 376
Table 345. Standard Frame Group Start Address register (SFF_GRP_sa - address 0x4003 C008) bit description 376

continued >>

© NXP B.V. 2010. All rights reserved.

Table 346. Extended Frame Start Address register (EFF_sa - address 0x4003 C00C) bit description 376
Table 347. Extended Frame Group Start Address register (EFF_GRP_sa - address 0x4003 C010) bit description 377
Table 348. End of AF Tables register (ENDofTable - address 0x4003 C014) bit description 377
Table 349. LUT Error Address register (LUTerrAd - address 0x4003 C018) bit description 378
Table 350. LUT Error register (LUTerr - address 0x4003 C01C) bit description 378
Table 351. Global FullCAN Enable register (FCANIE - address 0x4003 C020) bit description 378
Table 352. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0x4003 C024) bit description 378
Table 353. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0x4003 C028) bit description 379
Table 354. Format of automatically stored Rx messages. 382
Table 355. FullCAN semaphore operation 382
Table 356. Example of Acceptance Filter Tables and ID index Values 392
Table 357. Used ID-Look-up Table sections. 394
Table 358. Used ID-Look-up Table sections. 395
Table 359. SPI pin description 400
Table 360. SPI Data To Clock Phase Relationship. 401
Table 361. SPI register map 404
Table 362: SPI Control Register (S0SPCR - address 0x4002 0000) bit description 405
Table 363: SPI Status Register (S0SPSR - address 0x4002 0004) bit description 406
Table 364: SPI Data Register (S0SPDR - address 0x4002 0008) bit description 406
Table 365: SPI Clock Counter Register (S0SPCCR - address 0x4002 000C) bit description 407
Table 366: SPI Test Control Register (SPTCR - address 0x4002 0010) bit description 407
Table 367: SPI Test Status Register (SPTSR - address 0x4002 0014) bit description 407
Table 368: SPI Interrupt Register (S0SPINT - address 0x4002 001C) bit description 408
Table 369. SSP pin descriptions 411
Table 370. SSP Register Map 419
Table 371: SSPn Control Register 0 (SSP0CR0 - address 0x4008 8000, SSP1CR0 - 0x4003 0000) bit description 420
Table 372: SSPn Control Register 1 (SSP0CR1 - address 0x4008 8004, SSP1CR1 - 0x4003 0004) bit description 421
Table 373: SSPn Data Register (SSP0DR - address 0x4008 8008, SSP1DR - 0x4003 0008) bit description 421
Table 374: SSPn Status Register (SSP0SR - address 0x4008 800C, SSP1SR - 0x4003 000C) bit description 422
Table 375: SSPn Clock Prescale Register (SSP0CPSR - address 0x4008 8010, SSP1CPSR - 0x4003 0010) bit description 422
Table 376: SSPn Interrupt Mask Set/Clear register (SSP0IMSC - address 0x4008 8014, SSP1IMSC - 0x4003 0014) bit description 423
Table 377: SSPn Raw Interrupt Status register (SSP0RIS - address 0x4008 8018, SSP1RIS - 0x4003 0018) bit description. 423
Table 378: SSPn Masked Interrupt Status register (SSPnMIS -address 0x4008 801C, SSP1MIS - 0x4003 001C) bit description 424
Table 379: SSPn interrupt Clear Register (SSP0ICR - address 0x4008 8020, SSP1ICR - 0x4003 0020) bit description. 424
Table 380: SSPn DMA Control Register (SSP0DMACR - address 0x4008 8024, SSP1DMACR - 0x4003 0024) bit description 424
Table 381. I^2C Pin Description. 427
Table 382. I2C0CONSET and I2C1CONSET used to configure Master mode 428
Table 383. I2C0CONSET and I2C1CONSET used to configure Slave mode 430
Table 384. I^2C register map. 436
Table 385. I^2C Control Set register (I2CONSET: I^2C0, I2C0CONSET - address 0x4001 C000, I^2C1, I2C1CONSET - address 0x4005 C000, I^2C2, I2C2CONSET - address 0x400A 0000) bit description 438
Table 386. I^2C Control Clear register (I2CONCLR: I^2C0, I2C0CONCLR - 0x4001 C018; I^2C1, I2C1CONCLR - 0x4005 C018; I^2C2, I2C2CONCLR - 0x400A 0018) bit description 439
Table 387. I^2C Status register (I2STAT: I^2C0, I2C0STAT - 0x4001 C004; I^2C1, I2C1STAT - 0x4005 C004; I^2C2, I2C2STAT - 0x400A 0004) bit description. . 440
Table 388. I^2C Data register (I2DAT: I^2C0, I2C0DAT - 0x4001 C008; I^2C1, I2C1DAT - 0x4005 C008; I^2C2, I2C2DAT - 0x400A 0008) bit description441
Table 389. I^2C Monitor mode control register (I2MMCTRL: I^2C0, I2C0MMCTRL - 0x4001 C01C; I^2C1, I2C1MMCTRL- 0x4005 C01C; I^2C2, I2C2MMCTRL- 0x400A 001C) bit description 441
Table 390. I^2C Data buffer register (I2DATA_BUFFER: I^2C0, I2CDATA_BUFFER - 0x4001 C02C; I^2C1,

continued >>

© NXP B.V. 2010. All rights reserved.

I2C1DATA_BUFFER- 0x4005 C02C; I²C2, I2C2DATA_BUFFER- 0x400A 002C) bit description 443
Table 391. I²C Slave Address registers (I2ADR0 to 3: I²C0, I2C0ADR[0, 1, 2, 3]- 0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3] - address 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3] - address 0x400A 00[0C, 20, 24, 28]) bit description 443
Table 392. I²C Mask registers (I2MASK0 to 3: I²C0, I2C0MASK[0, 1, 2, 3] - 0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3] - address 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3] - address 0x400A 00[30, 34, 38, 3C]) bit description 444
Table 393. I²C SCL HIGH Duty Cycle register (I2SCLH: I²C0, I2C0SCLH - address 0x4001 C010; I²C1, I2C1SCLH - address 0x4005 C010; I²C2, I2C2SCLH - 0x400A 0010) bit description 444
Table 394. I²C SCL Low duty cycle register (I2SCLL: I²C0 - I2C0SCLL: 0x4001 C014; I²C1 - I2C1SCLL: 0x4005 C014; I²C2 - I2C2SCLL: 0x400A 0014) bit description 444
Table 395. Example I²C clock rates 445
Table 396. Abbreviations used to describe an I²C operation. 446
Table 397. I2CONSET used to initialize Master Transmitter mode. 447
Table 398. I2CONSET used to initialize Slave Receiver mode. 451
Table 399. Master Transmitter mode 454
Table 400. Master Receiver mode 455
Table 401. Slave Receiver mode 456
Table 402. Slave Transmitter mode 458
Table 403. Miscellaneous States 459
Table 404. Pin descriptions 472
Table 405. I2S register map 473
Table 406: Digital Audio Output register (I2SDAO - address 0x400A 8000) bit description 473
Table 407: Digital Audio Input register (I2SDAI - address 0x400A 8004) bit description 474
Table 408: Transmit FIFO register (I2STXFIFO - address 0x400A 8008) bit description 474
Table 409: Receive FIFO register (I2RXFIFO - address 0x400A 800C) bit description 475
Table 410: Status Feedback register (I2SSTATE - address 0x400A 8010) bit description 475
Table 411: DMA Configuration register 1 (I2SDMA1 - address 0x400A 8014) bit description 475
Table 412: DMA Configuration register 2 (I2SDMA2 - address 0x400A 8018) bit description 476
Table 413: Interrupt Request Control register (I2SIRQ - address 0x400A 801C) bit description 476
Table 414: Transmit Clock Rate register (I2TXRATE - address 0x400A 8020) bit description 477
Table 415: Receive Clock Rate register (I2SRXRATE - address 0x400A 8024) bit description 478
Table 416: Transmit Clock Rate register (I2TXBITRATE - address 0x400A 8028) bit description 478
Table 417: Receive Clock Rate register (I2SRXBITRATE - address 0x400A 802C) bit description 478
Table 418: Transmit Mode Control register (I2STXMODE - 0x400A 8030) bit description 479
Table 419: Receive Mode Control register (I2SRXMODE - 0x400A 8034) bit description 479
Table 420: I2S transmit modes 481
Table 421: I2S receive modes 483
Table 422. Conditions for FIFO level comparison 485
Table 423. DMA and interrupt request generation. 485
Table 424. Status feedback in the I2SSTATE register . . . 485
Table 425. Timer/Counter pin description 488
Table 426. TIMER/COUNTER0-3 register map. 489
Table 427. Interrupt Register (T[0/1/2/3]IR - addresses 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000) bit description 490
Table 428. Timer Control Register (TCR, TIMERn: TnTCR - addresses 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004) bit description . . 491
Table 429. Count Control Register (T[0/1/2/3]CTCR - addresses 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) bit description . . 491
Table 430. Match Control Register (T[0/1/2/3]MCR - addresses 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) bit description . . 493
Table 431. Capture Control Register (T[0/1/2/3]CCR - addresses 0x4000 4028, 0x4000 8020, 0x4009 0028, 0x4009 4028) bit description . . 494
Table 432. External Match Register (T[0/1/2/3]EMR - addresses 0x4000 403C, 0x4000 803C, 0x4009 003C, 0x4009 403C) bit description . 495
Table 433. External Match Control 495
Table 434. Repetitive Interrupt Timer register map 498
Table 435. RI Compare Value register (RICOMPVAL - address 0x400B 0000) bit description 498
Table 436. RI Compare Value register (RICOMPVAL - address 0x400B 0004) bit description 498
Table 437. RI Control register (RICTRL - address 0x400B 0008) bit description. 499
Table 438. RI Counter register (RICOUNTER - address 0x400B 000C) bit description. 499
Table 439. System Tick Timer register map. 502
Table 440. System Timer Control and status register

continued >>

© NXP B.V. 2010. All rights reserved.

(STCTRL - 0xE000 E010) bit description 502
Table 441. System Timer Reload value register (STRELOAD - 0xE000 E014) bit description 503
Table 442. System Timer Current value register (STCURR - 0xE000 E018) bit description 503
Table 443. System Timer Calibration value register (STCALIB - 0xE000 E01C) bit description . . .504
Table 444. Set and reset inputs for PWM Flip-Flops 509
Table 445. Pin summary. 510
Table 446. PWM1 register map 511
Table 447: PWM Interrupt Register (PWM1IR - address 0x4001 8000) bit description 512
Table 448: PWM Timer Control Register (PWM1TCR address 0x4001 8004) bit description 513
Table 449: PWM Count control Register (PWM1CTCR - address 0x4001 8004) bit description. 513
Table 450: Match Control Register (PWM1MCR - address 0x4000 4014) bit description 514
Table 451: PWM Capture Control Register (PWM1CCR - address 0x4001 8028) bit description. 515
Table 452: PWM Control Register (PWM1PCR - address 0x4001 804C) bit description 516
Table 453: PWM Latch Enable Register (PWM1LER - address 0x4001 8050) bit description. 517
Table 454. Pin summary. 519
Table 455. Motor Control Pulse Width Modulator (MCPWM) register map 522
Table 456. MCPWM Control read address (MCCON - 0x400B 8000) bit description 523
Table 457. MCPWM Control set address (MCCON_SET - 0x400B 8004) bit description 524
Table 458. MCPWM Control clear address (MCCON_CLR - 0x400B 8008) bit description 525
Table 459. MCPWM Capture Control read address (MCCAPCON - 0x400B 800C) bit description 525
Table 460. MCPWM Capture Control set address (MCCAPCON_SET - 0x400B 8010) bit description 526
Table 461. MCPWM Capture control clear register (MCCAPCON_CLR - address 0x400B 8014) bit description 526
Table 462. Motor Control PWM interrupts 526
Table 463. Interrupt sources bit allocation table. 526
Table 464. MCPWM Interrupt Enable read address (MCINTEN - 0x400B 8050) bit description . . .526
Table 465. PWM interrupt enable set register (MCINTEN_SET - address 0x400B 8054) bit description 527
Table 466. PWM interrupt enable clear register (MCINTEN_CLR - address 0x400B 8058) bit description 527
Table 467. MCPWM Interrupt Flags read address (MCINTF - 0x400B 8068) bit description 527
Table 468. MCPWM Interrupt Flags set address (PWMINTF_SET - 0x400B 806C) bit description 527
Table 469. MCPWM Interrupt Flags clear address (PWMINTF_CLR - 0x400B 8070) bit description 528
Table 470. MCPWM Count Control read address (MCCNTCON - 0x400B 805C) bit description 528
Table 471. MCPWM Count Control set address (MCCNTCON_SET - 0x400B 8060) bit description 529
Table 472. MCPWM Count Control clear address (MCCAPCON_CLR - 0x400B 8064) bit description 529
Table 473. MCPWM Timer/Counter 0-2 registers (MCTC0-2 - 0x400B 8018, 0x400B 801C, 0x400B 8020) bit description 530
Table 474. MCPWM Limit 0-2 registers (MCLIM0-2 - 0x400B 8024, 0x400B 8028, 0x400B 802C) bit description 530
Table 475. MCPWM Match 0-2 registers (MCMAT0-2 - addresses 0x400B 8030, 0x400B 8034, 0x400B 8038) bit description 531
Table 476. MCPWM Dead-time register (MCDT - address 0x400B 803C) bit description. 532
Table 477. MCPWM Commutation Pattern register (MCCP - address 0x400B 8040) bit description 532
Table 478. MCPWM Capture read addresses (MCCAP0/1/2 - 0x400B 8044, 0x400B 8048, 0x400B 804C) bit description 532
Table 479. MCPWM Capture clear address (CAP_CLR - 0x400B 8074) bit description 533
Table 480. Encoder states. 542
Table 481. Encoder state transitions[1] 542
Table 482. Encoder direction 543
Table 483. QEI pin description. 545
Table 484. QEI Register summary. 546
Table 485: QEI Control register (QEICON - address 0x400B C000) bit description. 547
Table 486: QEI Configuration register (QEICONF - address 0x400B C008) bit description. 547
Table 487: QEI Interrupt Status register (QEISTAT - address 0x400B C004) bit description. 547
Table 488: QEI Position register (QEIPOS - address 0x400B C00C) bit description 548
Table 489: QEI Maximum Position register (QEIMAXPOS - address 0x400B C010) bit description. 548
Table 490: QEI Position Compare register 0 (CMPOS0 - address 0x400B C014) bit description. 548

continued >>

© NXP B.V. 2010. All rights reserved.

Table 491: QEI Position Compare register 1 (CMPOS1 - address 0x400B C018) bit description 548
Table 492: QEI Position Compare register 2 (CMPOS2 - address 0x400B C01C) bit description 549
Table 493: QEI Index Count register (CMPOS - address 0x400B C020) bit description 549
Table 494: QEI Index Compare register (CMPOS - address 0x400B C024) bit description 549
Table 495: QEI Timer Load register (QEILOAD - address 0x400B C028) bit description 549
Table 496: QEI Timer register (QEITIME - address 0x400B C02C) bit description 549
Table 497: QEI Velocity register (QEIVEL - address 0x400B C030) bit description 550
Table 498: QEI Velocity Capture register (QEICAP - address 0x400B C034) bit description 550
Table 499: QEI Velocity Compare register (VELCOMP - address 0x400B C038) bit description 550
Table 500: QEI Digital Filter register (FILTER - address 0x400B C03C) bit description 550
Table 501: QEI Interrupt Status register (QEIINTSTAT - address 0x400B CFE0) bit description 551
Table 502: QEI Interrupt Set register (QEISET - address 0x400B CFEC) bit description 551
Table 503: QEI Interrupt Clear register (QEICLR - 0x400B CFE8) bit description 552
Table 504: QEI Interrupt Enable register (QEIIE - address 0x400B CFE4) bit description 552
Table 505: QEI Interrupt Enable Set register (QEIIES - address 0x400B CFDC) bit description 553
Table 506: QEI Interrupt Enable Clear register (QEIIEC - address 0x400B CFD8) bit description 554
Table 507. RTC pin description 557
Table 508. Real-Time Clock register map 558
Table 509. Interrupt Location Register (ILR - address 0x4002 4000) bit description 559
Table 510. Clock Control Register (CCR - address 0x4002 4008) bit description 559
Table 511. Counter Increment Interrupt Register (CIIR - address 0x4002 400C) bit description 560
Table 512. Alarm Mask Register (AMR - address 0x4002 4010) bit description 561
Table 513. RTC Auxiliary control register (RTC_AUX - address 0x4002 405C) bit description 561
Table 514. RTC Auxiliary Enable register (RTC_AUXEN - address 0x4002 4058) bit description 561
Table 515. Consolidated Time register 0 (CTIME0 - address 0x4002 4014) bit description 562
Table 516. Consolidated Time register 1 (CTIME1 - address 0x4002 4018) bit description 562
Table 517. Consolidated Time register 2 (CTIME2 - address 0x4002 401C) bit description 563
Table 518. Time Counter relationships and values 563
Table 519. Time Counter registers. 563
Table 520. Calibration register (CALIBRATION - address 0x4002 4040) bit description 564
Table 521. General purpose registers 0 to 4 (GPREG0 to GPREG4 - addresses 0x4002 4044 to 0x4002 4054) bit description. 565
Table 522. Alarm registers. 565
Table 523. Watchdog register map 567
Table 524: Watchdog Mode register (WDMOD - address 0x4000 0000) bit description 568
Table 525. Watchdog operating modes selection 568
Table 526: Watchdog Constant register (WDTC - address 0x4000 0004) bit description 569
Table 527: Watchdog Feed register (WDFEED - address 0x4000 0008) bit description 569
Table 528: Watchdog Timer Value register (WDTV - address 0x4000 000C) bit description 569
Table 529: Watchdog Timer Clock Source Selection register (WDCLKSEL - address 0x4000 0010) bit description 570
Table 530. ADC pin description 572
Table 531. ADC registers. 573
Table 532: A/D Control Register (AD0CR - address 0x4003 4000) bit description 574
Table 533: A/D Global Data Register (AD0GDR - address 0x4003 4004) bit description 575
Table 534: A/D Status register (AD0INTEN - address 0x4003 400C) bit description 575
Table 535: A/D Data Registers (AD0DR0 to AD0DR7 - 0x4003 4010 to 0x4003 402C) bit description 576
Table 536: A/D Status register (AD0STAT - address 0x4003 4030) bit description 577
Table 537: A/D Trim register (ADTRM - address 0x4003 4034) bit description 577
Table 538. D/A Pin Description 579
Table 539. DAC registers. 580
Table 540: D/A Converter Register (DACR - address 0x4008 C000) bit description 580
Table 541. D/A Control register (DACCTRL - address 0x4008 C004) bit description 581
Table 542: D/A Converter register (DACR - address 0x4008 C000) bit description 581
Table 543. Endian behavior 586
Table 544. DMA Connections 589
Table 545. GPDMA register map. 590
Table 546. DMA Interrupt Status register (DMACIntStat - 0x5000 4000) 592
Table 547. DMA Interrupt Terminal Count Request Status register (DMACIntTCStat - 0x5000 4004) . . . 592

continued >>

© NXP B.V. 2010. All rights reserved.

Table 548. DMA Interrupt Terminal Count Request Clear register (DMACIntTCClear - 0x5000 4008) . . 592
Table 549. DMA Interrupt Error Status register (DMACIntErrStat - 0x5000 400C) 593
Table 550. DMA Interrupt Error Clear register (DMACIntErrClr - 0x5000 4010) 593
Table 551. DMA Raw Interrupt Terminal Count Status register (DMACRawIntTCStat - 0x5000 4014) 593
Table 552. DMA Raw Error Interrupt Status register (DMACRawIntErrStat - 0x5000 4018) 594
Table 553. DMA Enabled Channel register (DMACEnbldChns - 0x5000 401C) 594
Table 554. DMA Software Burst Request register (DMACSoftBReq - 0x5000 4020) 594
Table 555. DMA Software Single Request register (DMACSoftSReq - 0x5000 4024) 595
Table 556. DMA Software Last Burst Request register (DMACSoftLBReq - 0x5000 4028) 595
Table 557. DMA Software Last Single Request register (DMACSoftLSReq - 0x5000 402C) 596
Table 558. DMA Configuration register (DMACConfig - 0x5000 4030) 596
Table 559. DMA Synchronization register (DMACSync - 0x5000 4034) 596
Table 560. DMA Request Select register (DMAReqSel - 0x400F C1C4) 597
Table 561. DMA Channel Source Address registers (DMACCxSrcAddr - 0x5000 41x0) 598
Table 562. DMA Channel Destination Address registers (DMACCxDestAddr - 0x5000 41x4) 598
Table 563. DMA Channel Linked List Item registers (DMACCxLLI - 0x5000 41x8) 599
Table 564. DMA channel control registers (DMACCxControl - 0x5000 41xC) 600
Table 565. DMA Channel Configuration registers (DMACCxConfig - 0x5000 41x0) 602
Table 566. Transfer type bits 603
Table 567. DMA request signal usage 606
Table 568. Sectors in a LPC17xx device 617
Table 569. Code Read Protection options 618
Table 570. Code Read Protection hardware/software interaction 619
Table 571. ISP command summary 620
Table 572. ISP Unlock command 620
Table 573. ISP Set Baud Rate command 621
Table 574. Correlation between possible ISP baudrates and CCLK frequency (in MHz) 621
Table 575. ISP Echo command 621
Table 576. ISP Write to RAM command 622
Table 577. ISP Read Memory command 622
Table 578. ISP Prepare sector(s) for write operation command 623
Table 579. ISP Copy command 623
Table 580. ISP Go command 624
Table 581. ISP Erase sector command 624
Table 582. ISP Blank check sector command 625
Table 583. ISP Read Part Identification command 625
Table 584. LPC17xx part identification numbers 625
Table 585. ISP Read Boot Code version number command 626
Table 586. ISP Read device serial number command . . . 626
Table 587. ISP Compare command 626
Table 588. ISP Return Codes Summary 627
Table 589. IAP Command Summary 629
Table 590. IAP Prepare sector(s) for write operation command 630
Table 591. IAP Copy RAM to Flash command 630
Table 592. IAP Erase Sector(s) command 631
Table 593. IAP Blank check sector(s) command 631
Table 594. IAP Read part identification number command 631
Table 595. IAP Read Boot Code version number command 632
Table 596. IAP Read device serial number command . . . 632
Table 597. IAP Compare command 632
Table 598. Re-invoke ISP 633
Table 599. IAP Status Codes Summary 633
Table 600. Register overview: FMC (base address 0x2020 0000) 634
Table 601. Flash Module Signature Start register (FMSSTART - 0x4008 4020) bit description . 635
Table 602. Flash Module Signature Stop register (FMSSTOP - 0x4008 4024) bit description 635
Table 603. FMSW0 register bit description (FMSW0, address: 0x2020 002C) 635
Table 604. FMSW1 register bit description (FMSW1, address: 0x2020 0030) 635
Table 605. FMSW2 register bit description (FMSW2, address: 0x2020 0034) 636
Table 606. FMSW3 register bit description (FMSW3, address: 0x2020 0038) 636
Table 607. Flash module Status register (FMSTAT - 0x4008 4FE0) bit description 636
Table 608. Flash Module Status Clear register (FMSTATCLR - 0x0x4008 4FE8) bit description 636
Table 609. JTAG pin description 639
Table 610. Serial Wire Debug pin description 639
Table 611. Parallel Trace pin description 639
Table 612. Memory Mapping Control register (MEMMAP - 0x400F C040) bit description 640
Table 613. Cortex-M3 instructions 644
Table 614. CMSIS intrinsic functions to generate some

continued >>

© NXP B.V. 2010. All rights reserved.

Cortex-M3 instructions 647
Table 615. CMSIS intrinsic functions to access the special registers 647
Table 616. Condition code suffixes. 654
Table 617. Memory access instructions 657
Table 618. Offset ranges 660
Table 619. Offset ranges 666
Table 620. Data processing instructions. 674
Table 621. Multiply and divide instructions. 689
Table 622. Packing and unpacking instructions 697
Table 623. Branch and control instructions 702
Table 624. Branch ranges 703
Table 625. Miscellaneous instructions 711
Table 626. Summary of processor mode, execution privilege level, and stack use options 725
Table 627. Core register set summary 725
Table 628. PSR register combinations 727
Table 629. APSR bit assignments 728
Table 630. IPSR bit assignments 729
Table 631. EPSR bit assignments 729
Table 632. PRIMASK register bit assignments. 730
Table 633. FAULTMASK register bit assignments 730
Table 634. BASEPRI register bit assignments 731
Table 635. CONTROL register bit assignments 731
Table 636. Memory access behavior 736
Table 637. SRAM memory bit-banding regions 738
Table 638. Peripheral memory bit-banding regions 738
Table 639. C compiler intrinsic functions for exclusive access instructions 741
Table 640. Properties of the different exception types . . . 743
Table 641. Exception return behavior. 748
Table 642. Faults 750
Table 643. Fault status and fault address registers 751
Table 644. Core peripheral register regions 756
Table 645. NVIC register summary 757
Table 646. Mapping of interrupts to the interrupt variables 758
Table 647. ISER bit assignments 758
Table 648. ICER bit assignments 759
Table 649. ISPR bit assignments 759
Table 650. ICPR bit assignments 760
Table 651. IABR bit assignments 760
Table 652. IPR bit assignments 761
Table 653. STIR bit assignments 761
Table 654. CMSIS functions for NVIC control 763
Table 655. Summary of the system control block registers 764
Table 656. ACTLR bit assignments 765
Table 657. CPUID register bit assignments 765
Table 658. ICSR bit assignments 766
Table 659. VTOR bit assignments 768
Table 660. AIRCR bit assignments 768
Table 661. Priority grouping. 769
Table 662. SCR bit assignments 770
Table 663. CCR bit assignments 771
Table 664. System fault handler priority fields 772
Table 665. SHPR1 register bit assignments 772
Table 666. SHPR2 register bit assignments 772
Table 667. SHPR3 register bit assignments 772
Table 668. SHCSR bit assignments 773
Table 669. MMFSR bit assignments 774
Table 670. BFSR bit assignments 775
Table 671. UFSR bit assignments 777
Table 672. HFSR bit assignments 778
Table 673. MMFAR bit assignments 778
Table 674. BFAR bit assignments 779
Table 675. System timer registers summary 780
Table 676. SysTick CTRL register bit assignments 780
Table 677. LOAD register bit assignments 781
Table 678. VAL register bit assignments 781
Table 679. CALIB register bit assignments 781
Table 680. Memory attributes summary 783
Table 681. MPU registers summary 784
Table 682. TYPE register bit assignments. 784
Table 683. MPU CTRL register bit assignments 785
Table 684. RNR bit assignments 786
Table 685. RBAR bit assignments 787
Table 686. RASR bit assignments 788
Table 687. Example SIZE field values 788
Table 688. TEX, C, B, and S encoding 789
Table 689. Cache policy for memory attribute encoding . 789
Table 690. AP encoding. 790
Table 691. Memory region attributes for a microcontroller 793
Table 692. Abbreviations 798

continued >>

© NXP B.V. 2010. All rights reserved.

4. Figures

Fig 1. LPC1768 simplified block diagram. 8
Fig 2. LPC1768 block diagram, CPU and buses 11
Fig 3. LPC17xx system memory map 13
Fig 4. Reset block diagram including the wake-up timer19
Fig 5. Example of start-up after reset. 20
Fig 6. External interrupt logic 23
Fig 7. Clock generation for the LPC17xx 29
Fig 8. Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for $C_{X1}/_{X2}$ evaluation32
Fig 9. PLL0 block diagram 36
Fig 10. PLL1 block diagram 49
Fig 11. PLLs and clock dividers 55
Fig 12. CLKOUT selection 67
Fig 13. Simplified block diagram of the flash accelerator showing potential bus connections 69
Fig 14. LPC176x LQFP100 pin configuration 92
Fig 15. LPC175x LQFP80 pin configuration 92
Fig 16. Ethernet block diagram 141
Fig 17. Ethernet packet fields 143
Fig 18. Receive descriptor memory layout. 170
Fig 19. Transmit descriptor memory layout 173
Fig 20. Transmit example memory and registers. 184
Fig 21. Receive Example Memory and Registers 190
Fig 22. Transmit Flow Control 195
Fig 23. Receive filter block diagram. 197
Fig 24. Receive Active/Inactive state machine 201
Fig 25. Transmit Active/Inactive state machine 202
Fig 26. USB device controller block diagram. 214
Fig 27. USB MaxPacketSize register array indexing . . .230
Fig 28. Interrupt event handling 241
Fig 29. UDCA Head register and DMA Descriptors 254
Fig 30. Isochronous OUT endpoint operation example .261
Fig 31. Data transfer in ATLE mode. 262
Fig 32. USB Host controller block diagram 268
Fig 33. USB OTG controller block diagram 272
Fig 34. USB OTG port configuration 273
Fig 35. USB host port configuration. 274
Fig 36. USB device port configuration 274
Fig 37. USB OTG interrupt handling 284
Fig 38. USB OTG controller with software stack 285
Fig 39. Hardware support for B-device switching from peripheral state to host state 286
Fig 40. State transitions implemented in software during B-device switching from peripheral to host 287
Fig 41. Hardware support for A-device switching from host state to peripheral state 289
Fig 42. State transitions implemented in software during A-device switching from host to peripheral 290
Fig 43. Clocking and power control. 293
Fig 44. Auto-baud a) mode 0 and b) mode 1 waveform 309
Fig 45. Algorithm for setting UART dividers 312
Fig 46. UART0, 2 and 3 block diagram 315
Fig 47. Auto-RTS Functional Timing 326
Fig 48. Auto-CTS Functional Timing 327
Fig 49. Auto-baud a) mode 0 and b) mode 1 waveform 332
Fig 50. Algorithm for setting UART dividers 334
Fig 51. UART1 block diagram 340
Fig 52. CAN controller block diagram 343
Fig 53. Transmit buffer layout for standard and extended frame format configurations 344
Fig 54. Receive buffer layout for standard and extended frame format configurations 345
Fig 55. Global Self-Test (high-speed CAN Bus example) 346
Fig 56. Local self test (high-speed CAN Bus example). 346
Fig 57. Entry in FullCAN and individual standard identifier tables. 373
Fig 58. Entry in standard identifier range table 373
Fig 59. Entry in either extended identifier table. 374
Fig 60. ID Look-up table example explaining the search algorithm 380
Fig 61. Semaphore procedure for reading an auto-stored message 383
Fig 62. FullCAN section example of the ID look-up table 385
Fig 63. FullCAN message object layout 385
Fig 64. Normal case, no messages lost 387
Fig 65. Message lost. 387
Fig 66. Message gets overwritten 388
Fig 67. Message overwritten indicated by semaphore bits and message lost 389
Fig 68. Message overwritten indicated by message lost390
Fig 69. Clearing message lost. 391
Fig 70. Detailed example of acceptance filter tables and ID index values 393
Fig 71. ID Look-up table configuration example (no FullCAN) 395
Fig 72. ID Look-up table configuration example (FullCAN activated and enabled) 397
Fig 73. SPI data transfer format (CPHA = 0 and CPHA = 1). 401
Fig 74. SPI block diagram. 409
Fig 75. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer 412

continued >>

© NXP B.V. 2010. All rights reserved.

Fig 76. SPI frame format with CPOL=0 and CPHA=0 (a) Single and b) Continuous Transfer) 413
Fig 77. SPI frame format with CPOL=0 and CPHA=1 . . 414
Fig 78. SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer) 415
Fig 79. SPI Frame Format with CPOL = 1 and CPHA = 1. 416
Fig 80. Microwire frame format (single transfer) 417
Fig 81. Microwire frame format (continuos transfers) . . 418
Fig 82. Microwire frame format setup and hold details . 418
Fig 83. I2C-bus configuration 427
Fig 84. Format in the Master Transmitter mode. 429
Fig 85. Format of Master Receiver mode 430
Fig 86. A Master Receiver switches to Master Transmitter after sending repeated START 430
Fig 87. Format of Slave Receiver mode 431
Fig 88. Format of Slave Transmitter mode 431
Fig 89. I2C serial interface block diagram 432
Fig 90. Arbitration procedure 434
Fig 91. Serial clock synchronization. 434
Fig 92. Format and states in the Master Transmitter mode 448
Fig 93. Format and states in the Master Receiver mode 450
Fig 94. Format and states in the Slave Receiver mode . 452
Fig 95. Format and states in the Slave Transmitter mode 453
Fig 96. Simultaneous repeated START conditions from two masters 461
Fig 97. Forced access to a busy I2C-bus. 461
Fig 98. Recovering from a bus obstruction caused by a LOW level on SDA. 461
Fig 99. Simple I2S configurations and bus timing 472
Fig 100. Typical transmitter master mode, with or without MCLK output 482
Fig 101. Transmitter master mode sharing the receiver reference clock 482
Fig 102. 4-wire transmitter master mode sharing the receiver bit clock and WS 482
Fig 103. Typical transmitter slave mode 482
Fig 104. Transmitter slave mode sharing the receiver reference clock 482
Fig 105. 4-wire transmitter slave mode sharing the receiver bit clock and WS 483
Fig 106. Typical receiver master mode, with or without MCLK output. 484
Fig 107. Receiver master mode sharing the transmitter reference clock 484
Fig 108. 4-wire receiver master mode sharing the transmitter bit clock and WS 484
Fig 109. Typical receiver slave mode. 484
Fig 110. Receiver slave mode sharing the transmitter reference clock 484
Fig 111. 4-wire receiver slave mode sharing the transmitter bit clock and WS 485
Fig 112. FIFO contents for various I2S modes 486
Fig 113. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled. 496
Fig 114. A timer Cycle in Which PR=2, MRx=6, and both interrupt and stop on match are enabled 496
Fig 115. Timer block diagram 497
Fig 116. RI timer block diagram 500
Fig 117. System Tick Timer block diagram 502
Fig 118. PWM block diagram 508
Fig 119. Sample PWM waveforms 509
Fig 120. MCPWM Block Diagram 520
Fig 121. Edge-aligned PWM waveform without dead time, POLA = 0. 534
Fig 122. Center-aligned PWM waveform without dead time, POLA = 0. 535
Fig 123. Edge-aligned PWM waveform with dead time, POLA = 0. 535
Fig 124. Center-aligned waveform with dead time, POLA = 0. 536
Fig 125. Three-phase DC mode sample waveforms. . . . 538
Fig 126. Three-phase AC mode sample waveforms, edge aligned PWM mode. 539
Fig 127. Encoder interface block diagram. 541
Fig 128. Quadrature Encoder Basic Operation 543
Fig 129. RTC domain conceptual diagram 556
Fig 130. RTC functional block diagram 556
Fig 131. Watchdog block diagram. 570
Fig 132. DAC control with DMA interrupt and timer 582
Fig 133. DMA controller block diagram 584
Fig 134. LLI example. 610
Fig 135. Map of lower memory 613
Fig 136. Boot process flowchart 616
Fig 137. IAP parameter passing 629
Fig 138. Algorithm for generating a 128 bit signature . . . 637
Fig 139. Typical Cortex-M3 implementation 641
Fig 140. ASR #3 650
Fig 141. LSR#3 651
Fig 142. LSL#3 651
Fig 143. ROR#3 652
Fig 144. RRX. 652
Fig 145. Bit-band mapping 739
Fig 146. Vector table 745

continued >>

© NXP B.V. 2010. All rights reserved.

5. Contents

Chapter 1: LPC17xx Introductory information

1 Introduction 3
2 Features 4
3 Applications 6
4 Ordering information 7
4.1 Part options summary 7
5 Simplified block diagram 8
6 Architectural overview 9
7 ARM Cortex-M3 processor 9
7.1 Cortex-M3 Configuration Options 9
8 On-chip flash memory system 10
9 On-chip Static RAM 10
10 Block diagram 11

Chapter 2: LPC17xx Memory map

1 Memory map and peripheral addressing 12
2 Memory maps 12
3 APB peripheral addresses 14
4 Memory re-mapping 15
5 AHB arbitration 15
6 Bus fault exceptions 16

Chapter 3: LPC17xx System control

1 Introduction 17
2 Pin description 17
3 Register description 18
4 Reset 18
4.1 Reset Source Identification Register (RSID - 0x400F C180) 21
5 Brown-out detection 22
6 External interrupt inputs 23
6.1 Register description 24
6.2 External Interrupt flag register (EXTINT - 0x400F C140) 24
6.3 External Interrupt Mode register (EXTMODE - 0x400F C148) 25
6.4 External Interrupt Polarity register (EXTPOLAR - 0x400F C14C) 26
7 Other system controls and status flags 28
7.1 System Controls and Status register (SCS - 0x400F C1A0) 28

Chapter 4: LPC17xx Clocking and power control

1 Summary of clocking and power control functions 29
2 Register description 30
3 Oscillators 31
3.1 Internal RC oscillator 31
3.2 Main oscillator 31
3.3 RTC oscillator 33
4 Clock source selection multiplexer 34
4.1 Clock Source Select register (CLKSRCSEL - 0x400F C10C) 34
5 PLL0 (Phase Locked Loop 0) 35
5.1 PLL0 operation 35
5.1.1 PLL0 and startup/boot code interaction 35
5.2 PLL0 register description 36
5.3 PLL0 Control register (PLL0CON - 0x400F C080) 36
5.4 PLL0 Configuration register (PLL0CFG - 0x400F C084) 37
5.5 PLL0 Status register (PLL0STAT - 0x400F C088) 39
5.6 PLL0 Interrupt: PLOCK0 39
5.7 PLL0 Modes 40
5.8 PLL0 Feed register (PLL0FEED - 0x400F C08C) 40
5.9 PLL0 and Power-down mode 40
5.10 PLL0 frequency calculation 40
5.11 Procedure for determining PLL0 settings 42
5.12 Examples of PLL0 settings 43
5.13 PLL0 setup sequence 46
6 PLL1 (Phase Locked Loop 1) 47
6.1 PLL1 register description 47
6.2 PLL1 Control register (PLL1CON - 0x400F C0A0) 49

continued >>

© NXP B.V. 2010. All rights reserved.

6.3 PLL1 Configuration register (PLL1CFG - 0x400F C0A4) . . . 50
6.4 PLL1 Status register (PLL1STAT - 0x400F C0A8) . . . 50
6.4.1 PLL1 modes . . . 51
6.5 PLL1 Interrupt: PLOCK1 . . . 51
6.6 PLL1 Feed register (PLL1FEED - 0x400F C0AC) . . . 52
6.7 PLL1 and Power-down mode . . . 52
6.8 PLL1 frequency calculation . . . 53
6.9 Procedure for determining PLL1 settings . . . 53
7 Clock dividers . . . 55
7.1 CPU Clock Configuration register (CCLKCFG - 0x400F C104) . . . 55
7.2 USB Clock Configuration register (USBCLKCFG - 0x400F C108) . . . 56
7.3 Peripheral Clock Selection registers 0 and 1 (PCLKSEL0 - 0x400F C1A8 and PCLKSEL1 - 0x400F C1AC) . . . 56
8 Power control . . . 59
8.1 Sleep mode . . . 59
8.2 Deep Sleep mode . . . 59
8.3 Power-down mode . . . 60
8.4 Deep Power-down mode . . . 61
8.5 Peripheral power control . . . 61
8.6 Register description . . . 61
8.7 Power Mode Control register (PCON - 0x400F C0C0) . . . 62
8.7.1 Encoding of Reduced Power Modes . . . 63
8.8 Wake-up from Reduced Power Modes . . . 63
8.9 Power Control for Peripherals register (PCONP - 0x400F C0C4) . . . 63
8.10 Power control usage notes . . . 65
8.11 Power domains . . . 65
9 Wake-up timer . . . 66
10 External clock output pin . . . 67
10.1 Clock Output Configuration register (CLKOUTCFG - 0x400F C1C8) . . . 67

Chapter 5: LPC17xx Flash accelerator

1 Introduction . . . 69
2 Flash accelerator blocks . . . 69
2.1 Flash memory bank . . . 69
2.2 Flash programming Issues . . . 70
3 Register description . . . 70
4 Flash Accelerator Configuration register (FLASHCFG - 0x400F C000) . . . 71
5 Operation . . . 71

Chapter 6: LPC17xx Nested Vectored Interrupt Controller (NVIC)

1 Features . . . 73
2 Description . . . 73
3 Interrupt sources . . . 73
4 Vector table remapping . . . 76
5 Register description . . . 77
5.1 Interrupt Set-Enable Register 0 register (ISER0 - 0xE000 E100) . . . 78
5.2 Interrupt Set-Enable Register 1 register (ISER1 - 0xE000 E104) . . . 79
5.3 Interrupt Clear-Enable Register 0 (ICER0 - 0xE000 E180) . . . 80
5.4 Interrupt Clear-Enable Register 1 register (ICER1 - 0xE000 E184) . . . 81
5.5 Interrupt Set-Pending Register 0 register (ISPR0 - 0xE000 E200) . . . 82
5.6 Interrupt Set-Pending Register 1 register (ISPR1 - 0xE000 E204) . . . 83
5.7 . . . . . Interrupt Clear-Pending Register 0 register (ICPR0 - 0xE000 E280) . . . 84
5.8 . . . . . Interrupt Clear-Pending Register 1 register (ICPR1 - 0xE000 E284) . . . 85
5.9 Interrupt Active Bit Register 0 (IABR0 - 0xE000 E300) . . . 86
5.10 Interrupt Active Bit Register 1 (IABR1 - 0xE000 E304) . . . 87
5.11 Interrupt Priority Register 0 (IPR0 - 0xE000 E400) . . . 88
5.12 Interrupt Priority Register 1 (IPR1 - 0xE000 E404) . . . 88
5.13 Interrupt Priority Register 2 (IPR2 - 0xE000 E408) . . . 88
5.14 Interrupt Priority Register 3 (IPR3 - 0xE000 E40C) . . . 89
5.15 Interrupt Priority Register 4 (IPR4 - 0xE000 E410) . . . 89
5.16 Interrupt Priority Register 5 (IPR5 - 0xE000 E414) . . . 89
5.17 Interrupt Priority Register 6 (IPR6 - 0xE000 E418) . . . 90
5.18 Interrupt Priority Register 7 (IPR7 - 0xE000 E41C) . . . 90

continued >>

© NXP B.V. 2010. All rights reserved.

5.19 Interrupt Priority Register 8 (IPR8 - 0xE000 E420) 90
5.20 Software Trigger Interrupt Register (STIR - 0xE000 EF00) 91

Chapter 7: LPC17xx Pin configuration

1 LPC17xx pin configuration 92
1.1 LPC17xx pin description 92

Chapter 8: LPC17xx Pin connect block

1 How to read this chapter 102
2 Description 102
3 Pin function select register values 102
4 Pin mode select register values 103
5 Register description 105
5.1 Pin Function Select register 0 (PINSEL0 - 0x4002 C000) 106
5.2 Pin Function Select Register 1 (PINSEL1 - 0x4002 C004) 106
5.3 Pin Function Select register 2 (PINSEL2 - 0x4002 C008) 107
5.4 Pin Function Select Register 3 (PINSEL3 - 0x4002 C00C) 107
5.5 Pin Function Select Register 4 (PINSEL4 - 0x4002 C010) 108
5.6 Pin Function Select Register 7 (PINSEL7 - 0x4002 C01C) 109
5.7 Pin Function Select Register 9 (PINSEL9 - 0x4002 C024) 109
5.8 Pin Function Select Register 10 (PINSEL10 - 0x4002 C028) 109
5.9 Pin Mode select register 0 (PINMODE0 - 0x4002 C040) 110
5.10 Pin Mode select register 1 (PINMODE1 - 0x4002 C044) 110
5.11 Pin Mode select register 2 (PINMODE2 - 0x4002 C048) 111
5.12 Pin Mode select register 3 (PINMODE3 - 0x4002 C04C) 111
5.13 Pin Mode select register 4 (PINMODE4 - 0x4002 C050) 112
5.14 Pin Mode select register 7 (PINMODE7 - 0x4002 C05C) 113
5.15 Pin Mode select register 9 (PINMODE9 - 0x4002 C064) 113
5.16 Open Drain Pin Mode select register 0 (PINMODE_OD0 - 0x4002 C068) 113
5.17 Open Drain Pin Mode select register 1 (PINMODE_OD1 - 0x4002 C06C) 114
5.18 Open Drain Pin Mode select register 2 (PINMODE_OD2 - 0x4002 C070) 115
5.19 Open Drain Pin Mode select register 3 (PINMODE_OD3 - 0x4002 C074) 116
5.20 Open Drain Pin Mode select register 4 (PINMODE_OD4 - 0x4002 C078) 116
5.21 I^2C Pin Configuration register (I2CPADCFG - 0x4002 C07C) 117

Chapter 9: LPC17xx General Purpose Input/Output (GPIO)

1 Basic configuration 118
2 Features 118
2.1 Digital I/O ports 118
2.2 Interrupt generating digital ports 118
3 Applications 119
4 Pin description 119
5 Register description 120
5.1 GPIO port Direction register FIOxDIR (FIO0DIR to FIO4DIR- 0x2009 C000 to 0x2009 C080) . . 121
5.2 GPIO port output Set register FIOxSET (FIO0SET to FIO4SET - 0x2009 C018 to 0x2009 C098) 122
5.3 GPIO port output Clear register FIOxCLR (FIO0CLR to FIO4CLR- 0x2009 C01C to 0x2009 C09C) 124
5.4 GPIO port Pin value register FIOxPIN (FIO0PIN to FIO4PIN- 0x2009 C014 to 0x2009 C094) . . 125
5.5 Fast GPIO port Mask register FIOxMASK (FIO0MASK to FIO4MASK - 0x2009 C010 to 0x2009 C090) 126
5.6 GPIO interrupt registers 129
5.6.1 GPIO overall Interrupt Status register (IOIntStatus - 0x4002 8080) 129
5.6.2 GPIO Interrupt Enable for port 0 Rising Edge (IO0IntEnR - 0x4002 8090) 129
5.6.3 GPIO Interrupt Enable for port 2 Rising Edge (IO2IntEnR - 0x4002 80B0) 130
5.6.4 GPIO Interrupt Enable for port 0 Falling Edge (IO0IntEnF - 0x4002 8094) 131
5.6.5 GPIO Interrupt Enable for port 2 Falling Edge (IO2IntEnF - 0x4002 80B4) 132

continued >>

© NXP B.V. 2010. All rights reserved.

5.6.6 GPIO Interrupt Status for port 0 Rising Edge Interrupt (IO0IntStatR - 0x4002 8084) 133
5.6.7 GPIO Interrupt Status for port 2 Rising Edge Interrupt (IO2IntStatR - 0x4002 80A4) 134
5.6.8 GPIO Interrupt Status for port 0 Falling Edge Interrupt (IO0IntStatF - 0x4002 8088). 134
5.6.9 GPIO Interrupt Status for port 2 Falling Edge Interrupt (IO2IntStatF - 0x4002 80A8) 135
5.6.10 GPIO Interrupt Clear register for port 0 (IO0IntClr - 0x4002 808C) 136
5.6.11 GPIO Interrupt Clear register for port 0 (IO2IntClr - 0x4002 80AC) 137
6 GPIO usage notes 138
6.1 Example: An instantaneous output of 0s and 1s on a GPIO port 138
6.2 Writing to FIOSET/FIOCLR vs. FIOPIN 138

Chapter 10: LPC17xx Ethernet

1 Basic configuration 139
2 Introduction 139
3 Features 140
4 Architecture and operation 141
5 DMA engine functions 142
6 Overview of DMA operation 142
7 Ethernet Packet 143
8 Overview 144
8.1 Partitioning 144
8.2 Example PHY Devices 145
9 Pin description 145
10 Registers and software interface 146
10.1 Register map 146
11 Ethernet MAC register definitions 148
11.1 MAC Configuration Register 1 (MAC1 - 0x5000 0000) 148
11.2 MAC Configuration Register 2 (MAC2 - 0x5000 0004) 148
11.3 Back-to-Back Inter-Packet-Gap Register (IPGT - 0x5000 0008) 150
11.4 Non Back-to-Back Inter-Packet-Gap Register (IPGR - 0x5000 000C) 150
11.5 Collision Window / Retry Register (CLRT - 0x5000 0010) 151
11.6 Maximum Frame Register (MAXF - 0x5000 0014) 151
11.7 PHY Support Register (SUPP - 0x5000 0018) 151
11.8 Test Register (TEST - 0x5000 001C) 151
11.9 MII Mgmt Configuration Register (MCFG - 0x5000 0020) 152
11.10 MII Mgmt Command Register (MCMD - 0x5000 0024) 153
11.11 MII Mgmt Address Register (MADR - 0x5000 0028) 153
11.12 MII Mgmt Write Data Register (MWTD - 0x5000 002C) 153
11.13 MII Mgmt Read Data Register (MRDD - 0x5000 0030) 154
11.14 MII Mgmt Indicators Register (MIND - 0x5000 0034) 154
11.15 Station Address 0 Register (SA0 - 0x5000 0040) 155
11.16 Station Address 1 Register (SA1 - 0x5000 0044) 155
11.17 Station Address 2 Register (SA2 - 0x5000 0048) 155
12 Control register definitions 156
12.1 Command Register (Command - 0x5000 0100) 156
12.2 Status Register (Status - 0x5000 0104) 156
12.3 Receive Descriptor Base Address Register (RxDescriptor - 0x5000 0108) 157
12.4 Receive Status Base Address Register (RxStatus - 0x5000 010C) 157
12.5 Receive Number of Descriptors Register (RxDescriptor - 0x5000 0110) 157
12.6 Receive Produce Index Register (RxProduceIndex - 0x5000 0114) 158
12.7 Receive Consume Index Register (RxConsumeIndex - 0x5000 0118) 158
12.8 Transmit Descriptor Base Address Register (TxDescriptor - 0x5000 011C) 159
12.9 Transmit Status Base Address Register (TxStatus - 0x5000 0120). 159
12.10 Transmit Number of Descriptors Register (TxDescriptorNumber - 0x5000 0124) 159
12.11 Transmit Produce Index Register (TxProduceIndex - 0x5000 0128) 160
12.12 Transmit Consume Index Register (TxConsumeIndex - 0x5000 012C) 160
12.13 Transmit Status Vector 0 Register (TSV0 - 0x5000 0158) 160
12.14 Transmit Status Vector 1 Register (TSV1 - 0x5000 015C) 161

continued >>

© NXP B.V. 2010. All rights reserved.

12.15 Receive Status Vector Register (RSV - 0x5000 0160) 162
12.16 Flow Control Counter Register (FlowControlCounter - 0x5000 0170) 163
12.17 Flow Control Status Register (FlowControlStatus - 0x5000 0174) 163
13 Receive filter register definitions 164
13.1 Receive Filter Control Register (RxFilterCtrl - 0x5000 0200) 164
13.2 Receive Filter WoL Status Register (RxFilterWoLStatus - 0x5000 0204) 164
13.3 Receive Filter WoL Clear Register (RxFilterWoLClear - 0x5000 0208) 165
13.4 Hash Filter Table LSBs Register (HashFilterL - 0x5000 0210) 165
13.5 Hash Filter Table MSBs Register (HashFilterH - 0x5000 0214) 166
14 Module control register definitions 166
14.1 Interrupt Status Register (IntStatus - 0x5000 0FE0) 166
14.2 Interrupt Enable Register (IntEnable - 0x5000 0FE4) 167
14.3 Interrupt Clear Register (IntClear - 0x5000 0FE8) 167
14.4 Interrupt Set Register (IntSet - 0x5000 0FEC) 168
14.5 Power-Down Register (PowerDown - 0x5000 0FF4) 169
15 Descriptor and status formats 170
15.1 Receive descriptors and statuses 170
15.2 Transmit descriptors and statuses 173
16 Ethernet block functional description 175
16.1 Overview 175
16.2 AHB interface 176
17 Interrupts 176
17.1 Direct Memory Access (DMA) 176
17.2 Initialization 179
17.3 Transmit process 180
17.4 Receive process 186
17.5 Transmission retry 192
17.6 Status hash CRC calculations 192
17.7 Duplex modes 193
17.8 IEE 802.3/Clause 31 flow control 193
17.9 Half-Duplex mode backpressure 195
17.10 Receive filtering 196
17.11 Power management 198
17.12 Wake-up on LAN 198
17.13 Enabling and disabling receive and transmit 200
17.14 Transmission padding and CRC 202
17.15 Huge frames and frame length checking . . . 203
17.16 Statistics counters 203
17.17 MAC status vectors 203
17.18 Reset 204
17.19 Ethernet errors 205
18 AHB bandwidth 206
18.1 DMA access 206
18.2 Types of CPU access 207
18.3 Overall bandwidth 207
19 CRC calculation 209

Chapter 11: LPC17xx USB device controller

1 How to read this chapter 211
2 Basic configuration 211
3 Introduction 211
4 Features 212
5 Fixed endpoint configuration 212
6 Functional description 213
6.1 Analog transceiver 214
6.2 Serial Interface Engine (SIE) 214
6.3 Endpoint RAM (EP_RAM) 214
6.4 EP_RAM access control 214
6.5 DMA engine and bus master interface 215
6.6 Register interface 215
6.7 SoftConnect 215
6.8 GoodLink 215
7 Operational overview 215
8 Pin description 216
9 Clocking and power management 216
9.1 Power requirements 216
9.2 Clocks 216
9.3 Power management support 217
9.4 Remote wake-up 218
10 Register description 218
10.1 Clock control registers 219
10.1.1 USB Clock Control register (USBClkCtrl - 0x5000 CFF4) 219
10.1.2 USB Clock Status register (USBClkSt - 0x5000 CFF8) 220
10.2 Device interrupt registers 220
10.2.1 USB Interrupt Status register (USBIntSt - 0x5000 C1C0) 220
10.2.2 USB Device Interrupt Status register (USBDevIntSt - 0x5000 C200) 221

continued >>

© NXP B.V. 2010. All rights reserved.

10.2.3 USB Device Interrupt Enable register (USBDevIntEn - 0x5000 C204) 222
10.2.4 USB Device Interrupt Clear register (USBDevIntClr - 0x5000 C208) 222
10.2.5 USB Device Interrupt Set register (USBDevIntSet - 0x5000 C20C) 223
10.2.6 USB Device Interrupt Priority register (USBDevIntPri - 0x5000 C22C) 223
10.3 Endpoint interrupt registers 224
10.3.1 USB Endpoint Interrupt Status register (USBEpIntSt - 0x5000 C230) 224
10.3.2 USB Endpoint Interrupt Enable register (USBEpIntEn - 0x5000 C234) 225
10.3.3 USB Endpoint Interrupt Clear register (USBEpIntClr - 0x5000 C238) 226
10.3.4 USB Endpoint Interrupt Set register (USBEpIntSet - 0x5000 C23C) 227
10.3.5 USB Endpoint Interrupt Priority register (USBEpIntPri - 0x5000 C240) 227
10.4 Endpoint realization registers 228
10.4.1 EP RAM requirements 228
10.4.2 USB Realize Endpoint register (USBReEp - 0x5000 C244) 228
10.4.3 USB Endpoint Index register (USBEpIn - 0x5000 C248) 230
10.4.4 USB MaxPacketSize register (USBMaxPSize - 0x5000 C24C) 230
10.5 USB transfer registers 230
10.5.1 USB Receive Data register (USBRxData - 0x5000 C218) 231
10.5.2 USB Receive Packet Length register (USBRxPLen - 0x5000 C220) 231
10.5.3 USB Transmit Data register (USBTxData - 0x5000 C21C) 231
10.5.4 USB Transmit Packet Length register (USBTxPLen - 0x5000 C224) 232
10.5.5 USB Control register (USBCtrl - 0x5000 C228) 232
10.6 SIE command code registers 232
10.6.1 USB Command Code register (USBCmdCode - 0x5000 C210) 233
10.6.2 USB Command Data register (USBCmdData - 0x5000 C214) 233
10.7 DMA registers 233
10.7.1 USB DMA Request Status register (USBDMARSt - 0x5000 C250) 233
10.7.2 USB DMA Request Clear register (USBDMARClr - 0x5000 C254) 234
10.7.3 USB DMA Request Set register (USBDMARSet - 0x5000 C258) 234
10.7.4 USB UDCA Head register (USBUDCAH - 0x5000 C280) 235
10.7.5 USB EP DMA Status register (USBEpDMASt - 0x5000 C284) 235
10.7.6 USB EP DMA Enable register (USBEpDMAEn - 0x5000 C288) 236
10.7.7 USB EP DMA Disable register (USBEpDMADis - 0x5000 C28C) 236
10.7.8 USB DMA Interrupt Status register (USBDMAIntSt - 0x5000 C290) 236
10.7.9 USB DMA Interrupt Enable register (USBDMAIntEn - 0x5000 C294) 237
10.7.10 USB End of Transfer Interrupt Status register (USBEoTIntSt - 0x5000 C2A0) 237
10.7.11 USB End of Transfer Interrupt Clear register (USBEoTIntClr - 0x5000 C2A4) 237
10.7.12 USB End of Transfer Interrupt Set register (USBEoTIntSet - 0x5000 C2A8) 238
10.7.13 USB New DD Request Interrupt Status register (USBNDDRIntSt - 0x5000 C2AC) 238
10.7.14 USB New DD Request Interrupt Clear register (USBNDDRIntClr - 0x5000 C2B0) 238
10.7.15 USB New DD Request Interrupt Set register (USBNDDRIntSet - 0x5000 C2B4) 238
10.7.16 USB System Error Interrupt Status register (USBSysErrIntSt - 0x5000 C2B8) 239
10.7.17 USB System Error Interrupt Clear register (USBSysErrIntClr - 0x5000 C2BC) 239
10.7.18 USB System Error Interrupt Set register (USBSysErrIntSet - 0x5000 C2C0) 239
11 Interrupt handling 239
12 Serial interface engine command description 242
12.1 Set Address (Command: 0xD0, Data: write 1 byte) 243
12.2 Configure Device (Command: 0xD8, Data: write 1 byte) 243
12.3 Set Mode (Command: 0xF3, Data: write 1 byte) 244
12.4 Read Current Frame Number (Command: 0xF5, Data: read 1 or 2 bytes) 244
12.5 Read Test Register (Command: 0xFD, Data: read 2 bytes) 245
12.6 Set Device Status (Command: 0xFE, Data: write 1 byte) 245
12.7 Get Device Status (Command: 0xFE, Data: read 1 byte) 246
12.8 Get Error Code (Command: 0xFF, Data: read 1 byte) 246

continued >>

© NXP B.V. 2010. All rights reserved.

12.9 Read Error Status (Command: 0xFB, Data: read 1 byte) . . . 247
12.10 Select Endpoint (Command: 0x00 - 0x1F, Data: read 1 byte (optional)) . . . 247
12.11 Select Endpoint/Clear Interrupt (Command: 0x40 - 0x5F, Data: read 1 byte) . . . 248
12.12 Set Endpoint Status (Command: 0x40 - 0x55, Data: write 1 byte (optional)) . . . 249
12.13 Clear Buffer (Command: 0xF2, Data: read 1 byte (optional)) . . . 249
12.14 Validate Buffer (Command: 0xFA, Data: none) . . . 250
13 USB device controller initialization . . . 250
14 Slave mode operation . . . 251
14.1 Interrupt generation . . . 251
14.2 Data transfer for OUT endpoints . . . 252
14.3 Data transfer for IN endpoints . . . 252
15 DMA operation . . . 252
15.1 Transfer terminology . . . 253
15.2 USB device communication area . . . 253
15.3 Triggering the DMA engine . . . 254
15.4 The DMA descriptor . . . 254
15.4.1 Next_DD_pointer . . . 255
15.4.2 DMA_mode . . . 256
15.4.3 Next_DD_valid . . . 256
15.4.4 Isochronous_endpoint . . . 256
15.4.5 Max_packet_size . . . 256
15.4.6 DMA_buffer_length . . . 256
15.4.7 DMA_buffer_start_addr . . . 256
15.4.8 DD_retired . . . 256
15.4.9 DD_status . . . 256
15.4.10 Packet_valid . . . 257
15.4.11 LS_byte_extracted . . . 257
15.4.12 MS_byte_extracted . . . 257
15.4.13 Present_DMA_count . . . 257
15.4.14 Message_length_position . . . 257
15.4.15 Isochronous_packetsize_memory_address . . . 257
15.5 Non-isochronous endpoint operation . . . 258
15.5.1 Setting up DMA transfers . . . 258
15.5.2 Finding DMA Descriptor . . . 258
15.5.3 Transferring the data . . . 258
15.5.4 Optimizing descriptor fetch . . . 258
15.5.5 Ending the packet transfer . . . 259
15.5.6 No_Packet DD . . . 259
15.6 Isochronous endpoint operation . . . 259
15.6.1 Setting up DMA transfers . . . 259
15.6.2 Finding the DMA Descriptor . . . 260
15.6.3 Transferring the Data . . . 260
15.6.4 DMA descriptor completion . . . 260
15.6.5 Isochronous OUT Endpoint Operation Example . . . 260
15.7 Auto Length Transfer Extraction (ATLE) mode operation . . . 261
15.7.1 Setting up the DMA transfer . . . 263
15.7.2 Finding the DMA Descriptor . . . 263
15.7.3 Transferring the Data . . . 263
15.7.4 Ending the packet transfer . . . 264
16 Double buffered endpoint operation . . . 264
16.1 Bulk endpoints . . . 264
16.2 Isochronous endpoints . . . 266

Chapter 12: LPC17xx USB Host controller

1 How to read this chapter . . . 267
2 Basic configuration . . . 267
3 Introduction . . . 267
3.1 Features . . . 268
3.2 Architecture . . . 268
4 Interfaces . . . 268
4.1 Pin description . . . 269
4.1.1 USB host usage note . . . 269
4.2 Software interface . . . 269
4.2.1 Register map . . . 269
4.2.2 USB Host Register Definitions . . . 270

Chapter 13: LPC17xx USB OTG controller

1 How to read this chapter . . . 271
2 Basic configuration . . . 271
3 Introduction . . . 271
4 Features . . . 271
5 Architecture . . . 272
6 Modes of operation . . . 272
7 Pin configuration . . . 273
7.1 Connecting the USB port to an external OTG transceiver . . . 273
7.2 Connecting USB as a host . . . 274
7.3 Connecting USB as device . . . 274
8 Register description . . . 275
8.1 USB Interrupt Status Register (USBIntSt - 0x5000 C1C0) . . . 275

continued >>

© NXP B.V. 2010. All rights reserved.

8.2 OTG Interrupt Status Register (OTGIntSt - 0x5000 C100) 276
8.3 OTG Interrupt Enable Register (OTGIntEn - 0x5000 C104) 276
8.4 OTG Interrupt Set Register (OTGIntSet - 0x5000 C20C) 276
8.5 OTG Interrupt Clear Register (OTGIntClr - 0x5000 C10C) 277
8.6 OTG Status and Control Register (OTGStCtrl - 0x5000 C110) 277
8.7 OTG Timer Register (OTGTmr - 0x5000 C114) 278
8.8 OTG Clock Control Register (OTGClkCtrl - 0x5000 CFF4) 278
8.9 OTG Clock Status Register (OTGClkSt - 0x5000 CFF8) 279
8.10 I2C Receive Register (I2C_RX - 0x5000 C300) 279
8.11 I2C Transmit Register (I2C_TX - 0x5000 C300) 280
8.12 I2C Status Register (I2C_STS - 0x5000 C304) 280
8.13 I2C Control Register (I2C_CTL - 0x5000 C308) 282
8.14 I2C Clock High Register (I2C_CLKHI - 0x5000 C30C) 283
8.15 I2C Clock Low Register (I2C_CLKLO - 0x5000 C310) 283
8.16 Interrupt handling 283
9 HNP support 284
9.1 B-device: peripheral to host switching 285
9.2 A-device: host to peripheral HNP switching . 288
10 Clocking and power management 292
10.1 Device clock request signals 293
10.1.1 Host clock request signals 294
10.2 Power-down mode support 294
11 USB OTG controller initialization 294

Chapter 14: LPC17xx UART0/2/3

1 Basic configuration 296
2 Features 296
3 Pin description 297
4 Register description 297
14.4.1 UARTn Receiver Buffer Register (U0RBR - 0x4000 C000, U2RBR - 0x4009 8000, U3RBR - 0x4009 C000 when DLAB = 0) 299
4.2 UARTn Transmit Holding Register (U0THR - 0x4000 C000, U2THR - 0x4009 8000, U3THR - 0x4009 C000 when DLAB = 0) 299
4.3 UARTn Divisor Latch LSB register (U0DLL - 0x4000 C000, U2DLL - 0x4009 8000, U3DLL - 0x4009 C000 when DLAB = 1) and UARTn Divisor Latch MSB register (U0DLM - 0x4000 C004, U2DLL - 0x4009 8004, U3DLL - 0x4009 C004 when DLAB = 1) 299
4.4 UARTn Interrupt Enable Register (U0IER - 0x4000 C004, U2IER - 0x4009 8004, U3IER - 0x4009 C004 when DLAB = 0) 300
4.5 UARTn Interrupt Identification Register (U0IIR - 0x4000 C008, U2IIR - 0x4009 8008, U3IIR - 0x4009 C008) 301
4.6 UARTn FIFO Control Register (U0FCR - 0x4000 C008, U2FCR - 0x4009 8008, U3FCR - 0x4009 C008) 303
4.6.1 DMA Operation 303
4.7 UARTn Line Control Register (U0LCR - 0x4000 C00C, U2LCR - 0x4009 800C, U3LCR - 0x4009 C00C) 304
4.8 UARTn Line Status Register (U0LSR - 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014) 304
4.9 UARTn Scratch Pad Register (U0SCR - 0x4000 C01C, U2SCR - 0x4009 801C U3SCR - 0x4009 C01C) 306
4.10 UARTn Auto-baud Control Register (U0ACR - 0x4000 C020, U2ACR - 0x4009 8020, U3ACR - 0x4009 C020) 306
14.4.10.1 Auto-baud 307
14.4.10.2 Auto-baud modes 308
4.11 UARTn IrDA Control Register (U0ICR - 0x4000 C024, U2ICR - 0x4009 8024, U3ICR - 0x4009 C024) 309
4.12 UARTn Fractional Divider Register (U0FDR - 0x4000 C028, U2FDR - 0x4009 8028, U3FDR - 0x4009 C028) 310
4.12.1 Baud rate calculation 311
4.13 UARTn Transmit Enable Register (U0TER - 0x4000 C030, U2TER - 0x4009 8030, U3TER - 0x4009 C030) 313
4.14 UARTn FIFO Level register (U0FIFOLVL - 0x4000 C058, U2FIFOLVL - 0x4009 8058, U3FIFOLVL - 0x4009 C058) 314
5 Architecture 314

continued >>

© NXP B.V. 2010. All rights reserved.

Chapter 15: LPC17xx UART1

1 Basic configuration 316
2 Features 316
3 Pin description 317
4 Register description 318
4.1 UART1 Receiver Buffer Register (U1RBR - 0x4001 0000, when DLAB = 0) 319
4.2 UART1 Transmitter Holding Register (U1THR - 0x4001 0000 when DLAB = 0) 319
4.3 UART1 Divisor Latch LSB and MSB Registers (U1DLL - 0x4001 0000 and U1DLM - 0x4001 0004, when DLAB = 1) 319
4.4 UART1 Interrupt Enable Register (U1IER - 0x4001 0004, when DLAB = 0) 320
4.5 UART1 Interrupt Identification Register (U1IIR - 0x4001 0008) 321
4.6 UART1 FIFO Control Register (U1FCR - 0x4001 0008) 323
4.6.1 DMA Operation 323
4.7 UART1 Line Control Register (U1LCR - 0x4001 000C) 324
4.8 UART1 Modem Control Register (U1MCR - 0x4001 0010) 324
4.9 Auto-flow control 325
15.4.9.1 Auto-RTS 325
15.4.9.2 Auto-CTS 326
4.10 UART1 Line Status Register (U1LSR - 0x4001 0014) 327
4.11 UART1 Modem Status Register (U1MSR - 0x4001 0018) 328
4.12 UART1 Scratch Pad Register (U1SCR - 0x4001 001C) 329
4.13 UART1 Auto-baud Control Register (U1ACR - 0x4001 0020) 329
4.14 Auto-baud 330
4.15 Auto-baud modes 331
4.16 UART1 Fractional Divider Register (U1FDR - 0x4001 0028) 332
4.16.1 Baud rate calculation 333
4.17 UART1 Transmit Enable Register (U1TER - 0x4001 0030) 335
4.18 UART1 RS485 Control register (U1RS485CTRL - 0x4001 004C) 336
4.19 UART1 RS-485 Address Match register (U1RS485ADRMATCH - 0x4001 0050) 337
4.20 UART1 RS-485 Delay value register (U1RS485DLY - 0x4001 0054) 337
4.21 RS-485/EIA-485 modes of operation 337
4.22 UART1 FIFO Level register (U1FIFOLVL - 0x4001 0058) 339
5 Architecture 339

Chapter 16: LPC17xx CAN1/2

1 Basic configuration 341
2 CAN controllers 341
3 Features 341
3.1 General CAN features 341
3.2 CAN controller features 342
3.3 Acceptance filter features 342
4 Pin description 342
5 CAN controller architecture 342
5.1 APB Interface Block (AIB) 343
5.2 Interface Management Logic (IML) 343
5.3 Transmit Buffers (TXB) 343
5.4 Receive Buffer (RXB) 344
5.5 Error Management Logic (EML) 345
5.6 Bit Timing Logic (BTL) 345
5.7 Bit Stream Processor (BSP) 345
5.8 CAN controller self-tests 345
6 Memory map of the CAN block 347
7 CAN controller registers 347
7.1 CAN Mode register (CAN1MOD - 0x4004 4000, CAN2MOD - 0x4004 8000) 349
7.2 CAN Command Register (CAN1CMR - 0x4004 x004, CAN2CMR - 0x4004 8004) . . 351
7.3 CAN Global Status Register (CAN1GSR - 0x4004 x008, CAN2GSR - 0x4004 8008) . . 352
7.4 CAN Interrupt and Capture Register (CAN1ICR - 0x4004 400C, CAN2ICR - 0x4004 800C) . . 355
7.5 CAN Interrupt Enable Register (CAN1IER - 0x4004 4010, CAN2IER - 0x4004 8010) . . . 358
7.6 CAN Bus Timing Register (CAN1BTR - 0x4004 4014, CAN2BTR - 0x4004 8014) . . 359
7.7 CAN Error Warning Limit register (CAN1EWL - 0x4004 4018, CAN2EWL - 0x4004 8018) . . 360
7.8 CAN Status Register (CAN1SR - 0x4004 401C, CAN2SR - 0x4004 801C) 361
7.9 CAN Receive Frame Status register (CAN1RFS - 0x4004 4020, CAN2RFS - 0x4004 8020) . . 363
7.9.1 ID index field 363

continued >>

© NXP B.V. 2010. All rights reserved.

7.10 CAN Receive Identifier register (CAN1RID - 0x4004 4024, CAN2RID - 0x4004 8024) . . . 363
7.11 CAN Receive Data register A (CAN1RDA - 0x4004 4028, CAN2RDA - 0x4004 8028) . . 364
7.12 CAN Receive Data register B (CAN1RDB - 0x4004 402C, CAN2RDB - 0x4004 802C) . . 364
7.13 CAN Transmit Frame Information register (CAN1TFI[1/2/3] - 0x4004 40[30/ 40/50], CAN2TFI[1/2/3] - 0x4004 80[30/40/50]) 365
7.14 CAN Transmit Identifier register (CAN1TID[1/2/3] - 0x4004 40[34/44/54], CAN2TID[1/2/3] - 0x4004 80[34/44/54]) 366
7.15 CAN Transmit Data register A (CAN1TDA[1/2/3] - 0x4004 40[38/48/58], CAN2TDA[1/2/3] - 0x4004 80[38/48/58]) 367
7.16 CAN Transmit Data register B (CAN1TDB[1/2/3] - 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3] - 0x4004 80[3C/4C/5C]) 367
7.17 CAN Sleep Clear register (CANSLEEPCLR - 0x400F C110) 367
7.18 CAN Wake-up Flags register (CANWAKEFLAGS - 0x400F C114) 368
8 CAN controller operation 368
8.1 Error handling 368
8.2 Sleep mode 369
8.3 Interrupts 369
8.4 Transmit priority 369
9 Centralized CAN registers 370
9.1 Central Transmit Status Register (CANTxSR - 0x4004 0000) 370
9.2 Central Receive Status Register (CANRxSR - 0x4004 0004) 370
9.3 Central Miscellaneous Status Register (CANMSR - 0x4004 0008) 371
10 Global acceptance filter 371
11 Acceptance filter modes 371
11.1 Acceptance filter Off mode 372
11.2 Acceptance filter Bypass mode 372
11.3 Acceptance filter Operating mode 372
11.4 FullCAN mode 372
12 Sections of the ID look-up table RAM 372
13 ID look-up table RAM 373
14 Acceptance filter registers 375
14.1 Acceptance Filter Mode Register (AFMR - 0x4003 C000) 375
14.2 Section configuration registers 375
14.3 Standard Frame Individual Start Address register (SFF_sa - 0x4003 C004) 376
14.4 Standard Frame Group Start Address register (SFF_GRP_sa - 0x4003 C008) 376
14.5 Extended Frame Start Address register (EFF_sa - 0x4003 C00C) 376
14.6 Extended Frame Group Start Address register (EFF_GRP_sa - 0x4003 C010) 377
14.7 End of AF Tables register (ENDofTable - 0x4003 C014) 377
14.8 Status registers 377
14.9 LUT Error Address register (LUTerrAd - 0x4003 C018) 378
14.10 LUT Error register (LUTerr - 0x4003 C01C) . 378
14.11 Global FullCANInterrupt Enable register (FCANIE - 0x4003 C020) 378
14.12 FullCAN Interrupt and Capture registers (FCANIC0 - 0x4003 C024 and FCANIC1 - 0x4003 C028) 378
15 Configuration and search algorithm 379
15.1 Acceptance filter search algorithm 379
16 FullCAN mode 380
16.1 FullCAN message layout 382
16.2 FullCAN interrupts 384
16.2.1 FullCAN message interrupt enable bit 384
16.2.2 Message lost bit and CAN channel number . 385
16.2.3 Setting the interrupt pending bits (IntPnd 63 to 0) 386
16.2.4 Clearing the interrupt pending bits (IntPnd 63 to 0) 386
16.2.5 Setting the message lost bit of a FullCAN message object (MsgLost 63 to 0) 386
16.2.6 Clearing the message lost bit of a FullCAN message object (MsgLost 63 to 0) 386
16.3 Set and clear mechanism of the FullCAN interrupt 386
16.3.1 Scenario 1: Normal case, no message lost . 386
16.3.2 Scenario 2: Message lost 387
16.3.3 Scenario 3: Message gets overwritten indicated by Semaphore bits 388
16.3.4 Scenario 3.1: Message gets overwritten indicated by Semaphore bits and Message Lost 388
16.3.5 Scenario 3.2: Message gets overwritten indicated by Message Lost 389
16.3.6 Scenario 4: Clearing Message Lost bit 390
17 Examples of acceptance filter tables and ID index values 391
17.1 Example 1: only one section is used 391
17.2 Example 2: all sections are used 391
17.3 Example 3: more than one but not all sections are used 391
17.4 Configuration example 4 392

continued >>

© NXP B.V. 2010. All rights reserved.

17.5 Configuration example 5 392
17.6 Configuration example 6 393
17.7 Configuration example 7 395
17.8 Look-up table programming guidelines 397

Chapter 17: LPC17xx SPI

1 Basic configuration 399
2 Features 399
3 SPI overview 399
4 Pin description 400
5 SPI data transfers 400
6 SPI peripheral details 402
6.1 General information 402
6.2 Master operation 402
6.3 Slave operation 403
6.4 Exception conditions 403
7 Register description 404
7.1 SPI Control Register (S0SPCR - 0x4002 0000) 404
7.2 SPI Status Register (S0SPSR - 0x4002 0004) 406
7.3 SPI Data Register (S0SPDR - 0x4002 0008) 406
7.4 SPI Clock Counter Register (S0SPCCR - 0x4002 000C) 406
7.5 SPI Test Control Register (SPTCR - 0x4002 0010) 407
7.6 SPI Test Status Register (SPTSR - 0x4002 0014) 407
7.7 SPI Interrupt Register (S0SPINT - 0x4002 001C) 408
8 Architecture 409

Chapter 18: LPC17xx SSP0/1 interface

1 Basic configuration 410
2 Features 410
3 Description 410
4 Pin descriptions 411
5 Bus description 411
5.1 Texas Instruments synchronous serial frame format 411
5.2 SPI frame format 412
5.2.1 Clock Polarity (CPOL) and Phase (CPHA) control 412
5.2.2 SPI format with CPOL=0,CPHA=0 413
5.2.3 SPI format with CPOL=0,CPHA=1 414
5.2.4 SPI format with CPOL = 1,CPHA = 0 414
5.2.5 SPI format with CPOL = 1,CPHA = 1 416
5.3 National Semiconductor Microwire frame format 416
5.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode 418
6 Register description 419
6.1 SSPn Control Register 0 (SSP0CR0 - 0x4008 8000, SSP1CR0 - 0x4003 0000) . . . 419
6.2 SSPn Control Register 1 (SSP0CR1 - 0x4008 8004, SSP1CR1 - 0x4003 0004) . . . 420
6.3 SSPn Data Register (SSP0DR - 0x4008 8008, SSP1DR - 0x4003 0008) 421
6.4 SSPn Status Register (SSP0SR - 0x4008 800C, SSP1SR - 0x4003 000C) 422
6.5 SSPn Clock Prescale Register (SSP0CPSR - 0x4008 8010, SSP1CPSR - 0x4003 0010) . 422
6.6 SSPn Interrupt Mask Set/Clear Register (SSP0IMSC - 0x4008 8014, SSP1IMSC - 0x4003 0014) 422
6.7 SSPn Raw Interrupt Status Register (SSP0RIS - 0x4008 8018, SSP1RIS - 0x4003 0018) . . . 423
6.8 SSPn Masked Interrupt Status Register (SSP0MIS - 0x4008 801C, SSP1MIS - 0x4003 001C) 423
6.9 SSPn Interrupt Clear Register (SSP0ICR - 0x4008 8020, SSP1ICR - 0x4003 0020) . . . 424
6.10 SSPn DMA Control Register (SSP0DMACR - 0x4008 8024, SSP1DMACR - 0x4003 0024) 424

Chapter 19: LPC17xx I2C0/1/2 interface

1 Basic configuration 425
2 Features 425
3 Applications 426
4 Description 426
4.1 I^2C FAST Mode Plus 427
5 Pin description 427
6 I^2C operating modes 428
6.1 Master Transmitter mode 428

continued >>

© NXP B.V. 2010. All rights reserved.

6.2 Master Receiver mode 429
6.3 Slave Receiver mode 430
6.4 Slave Transmitter mode 431
7 I²C implementation and operation 431
7.1 Input filters and output stages. 432
7.2 Address Registers, I2ADR0 to I2ADR3 433
7.3 Address mask registers, I2MASK0 to I2MASK3 433
7.4 Comparator. 433
7.5 Shift register, I2DAT 433
7.6 Arbitration and synchronization logic 433
7.7 Serial clock generator. 434
7.8 Timing and control 435
7.9 Control register, I2CONSET and I2CONCLR 435
7.10 Status decoder and status register 435
8 Register description 436
8.1 I²C Control Set register (I2CONSET: I²C0, I2C0CONSET - 0x4001 C000; I²C1, I2C1CONSET - 0x4005 C000; I²C2, I2C2CONSET - 0x400A 0000) 437
8.2 I²C Control Clear register (I2CONCLR: I²C0, I2C0CONCLR - 0x4001 C018; I²C1, I2C1CONCLR - 0x4005 C018; I²C2, I2C2CONCLR - 0x400A 0018) 439
8.3 I²C Status register (I2STAT: I²C0, I2C0STAT - 0x4001 C004; I²C1, I2C1STAT - 0x4005 C004; I²C2, I2C2STAT - 0x400A 0004). 440
8.4 I²C Data register (I2DAT: I²C0, I2C0DAT - 0x4001 C008; I²C1, I2C1DAT - 0x4005 C008; I²C2, I2C2DAT - 0x400A 0008) 440
8.5 I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C) 441
8.5.1 Interrupt in Monitor mode 442
8.5.2 Loss of arbitration in Monitor mode 442
8.6 I²C Data buffer register (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER - 0x4001 C02C; I²C1, I2C1DATA_BUFFER- 0x4005 C02C; I²C2, I2C2DATA_BUFFER- 0x400A 002C) 443
8.7 I²C Slave Address registers (I2ADR0 to 3: I²C0, I2C0ADR[0, 1, 2, 3]- 0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3] - address 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3] - address 0x400A 00[0C, 20, 24, 28]). . . . 443
8.8 I²C Mask registers (I2MASK0 to 3: I²C0, I2C0MASK[0, 1, 2, 3] - 0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3] - address 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3] - address 0x400A 00[30, 34, 38, 3C]). . 444
8.9 I²C SCL HIGH duty cycle register (I2SCLH: I²C0, I2C0SCLH - 0x4001 C010; I²C1, I2C1SCLH - 0x4005 C010; I²C2, I2C2SCLH - 0x400A 0010). 444
8.10 I²C SCL Low duty cycle register (I2SCLL: I²C0 - I2C0SCLL: 0x4001 C014; I²C1 - I2C1SCLL: 0x4005 C014; I²C2 - I2C2SCLL: 0x400A 0014). 444
8.11 Selecting the appropriate I²C data rate and duty cycle. 445
9 Details of I²C operating modes 446
9.1 Master Transmitter mode. 447
9.2 Master Receiver mode. 449
9.3 Slave Receiver mode. 451
9.4 Slave Transmitter mode. 453
9.5 Detailed state tables. 454
9.6 Miscellaneous states 459
9.6.1 I2STAT = 0xF8 459
9.6.2 I2STAT = 0x00 459
9.7 Some special cases. 459
9.7.1 Simultaneous repeated START conditions from two masters 459
9.7.2 Data transfer after loss of arbitration 460
9.7.3 Forced access to the I²C-bus. 460
9.7.4 I²C-bus obstructed by a LOW level on SCL or SDA 460
9.7.5 Bus error 460
9.8 I²C state service routines. 462
9.8.1 Initialization 462
9.8.2 I²C interrupt service 462
9.8.3 The state service routines 462
9.8.4 Adapting state services to an application. . . 462
10 Software example 463
10.1 Initialization routine 463
10.2 Start Master Transmit function 463
10.3 Start Master Receive function 463
10.4 I²C interrupt routine 463
10.5 Non mode specific states. 463
10.5.1 State: 0x00 463
10.5.2 Master States 464
10.5.3 State: 0x08 464
10.5.4 State: 0x10 464
10.6 Master Transmitter states 464
10.6.1 State: 0x18 464
10.6.2 State: 0x20 465
10.6.3 State: 0x28 465
10.6.4 State: 0x30 465
10.6.5 State: 0x38 465
10.7 Master Receive states 465
10.7.1 State: 0x40 465

continued >>

© NXP B.V. 2010. All rights reserved.

10.7.2 State: 0x48 466
10.7.3 State: 0x50 466
10.7.4 State: 0x58 466
10.8 Slave Receiver states 466
10.8.1 State: 0x60 466
10.8.2 State: 0x68 467
10.8.3 State: 0x70 467
10.8.4 State: 0x78 467
10.8.5 State: 0x80 467
10.8.6 State: 0x88 468
10.8.7 State: 0x90 468
10.8.8 State: 0x98 468
10.8.9 State: 0xA0 468
10.9 Slave Transmitter states 468
10.9.1 State: 0xA8 468
10.9.2 State: 0xB0 469
10.9.3 State: 0xB8 469
10.9.4 State: 0xC0 469
10.9.5 State: 0xC8 469

Chapter 20: LPC17xx I2S interface

1 Basic configuration 470
2 Features 470
3 Description 471
4 Pin descriptions 472
5 Register description 473
5.1 Digital Audio Output register (I2SDAO - 0x400A 8000) 473
5.2 Digital Audio Input register (I2SDAI - 0x400A 8004) 474
5.3 Transmit FIFO register (I2STXFIFO - 0x400A 8008) 474
5.4 Receive FIFO register (I2SRXFIFO - 0x400A 800C) 474
5.5 Status Feedback register (I2SSTATE - 0x400A 8010) 475
5.6 DMA Configuration Register 1 (I2SDMA1 - 0x400A 8014) 475
5.7 DMA Configuration Register 2 (I2SDMA2 - 0x400A 8018) 476
5.8 Interrupt Request Control register (I2SIRQ - 0x400A 801C) 476
5.9 Transmit Clock Rate register (I2STXRATE - 0x400A 8020) 476
5.9.1 Notes on fractional rate generators 477
5.10 Receive Clock Rate register (I2SRXRATE - 0x400A 8024) 477
5.11 Transmit Clock Bit Rate register (I2STXBITRATE - 0x400A 8028) 478
5.12 Receive Clock Bit Rate register (I2SRXBITRATE - 0x400A 802C) 478
5.13 Transmit Mode Control register (I2STXMODE - 0x400A 8030) 478
5.14 Receive Mode Control register (I2SRXMODE - 0x400A 8034) 479
6 I^2S transmit and receive interfaces 480
7 I^2S operating modes 481
8 FIFO controller 485

Chapter 21: LPC17xx Timer 0/1/2/3

1 Basic configuration 487
2 Features 487
3 Applications 488
4 Description 488
5 Pin description 488
5.1 Multiple CAP and MAT pins 488
6 Register description 489
6.1 Interrupt Register (T[0/1/2/3]IR - 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000) . 490
6.2 Timer Control Register (T[0/1/2/3]CR - 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004) 490
6.3 Count Control Register (T[0/1/2/3]CTCR - 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) 491
6.4 Timer Counter . . . . . registers (T0TC - T3TC, 0x4000 4008, 0x4000 8008, 0x4009 0008, 0x4009 4008) 492
6.5 Prescale register (T0PR - T3PR, 0x4000 400C, 0x4000 800C, 0x4009 000C, 0x4009 400C) 492
6.6 Prescale Counter register (T0PC - T3PC, 0x4000 4010, 0x4000 8010, 0x4009 0010, 0x4009 4010) 492
6.7 Match Registers (MR0 - MR3) 493
6.8 Match Control Register (T[0/1/2/3]MCR - 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) 493
6.9 Capture Registers (CR0 - CR1) 494

continued >>

© NXP B.V. 2010. All rights reserved.

6.10 Capture Control Register (T[0/1/2/3]CCR - 0x4000 4028, 0x4000 8028, 0x4009 0028, 0x4009 4028) 494
6.11 External Match Register (T[0/1/2/3]EMR - 0x4000 403C, 0x4000 803C, 0x4009 003C, 0x4009 403C) 494
6.12 DMA operation 495
7 Example timer operation 496
8 Architecture 497

Chapter 22: LPC17xx Repetitive Interrupt Timer (RIT)

1 Features 498
2 Description 498
3 Register description 498
3.1 RI Compare Value register (RICOMPVAL - 0x400B 0000) 498
3.2 RI Mask register (RIMASK - 0x400B 0004) . 498
3.3 RI Control register (RICTRL - 0x400B 0008) 499
3.4 RI Counter register (RICOUNTER - 0x400B 000C) 499
4 RI timer operation 499

Chapter 23: LPC17xx System Tick Timer

1 Basic configuration 501
2 Features 501
3 Description 501
4 Operation 501
5 Register description 502
5.1 System Timer Control and status register (STCTRL - 0xE000 E010) 502
5.2 System Timer Reload value register (STRELOAD - 0xE000 E014) 503
5.3 System Timer Current value register (STCURR - 0xE000 E018) 503
5.4 System Timer Calibration value register (STCALIB - 0xE000 E01C) 503
6 Example timer calculations 505

Chapter 24: LPC17xx Pulse Width Modulator (PWM)

1 Basic configuration 506
2 Features 506
3 Description 507
4 Sample waveform with rules for single and double edge control. 509
4.1 Rules for Single Edge Controlled PWM Outputs 510
4.2 Rules for Double Edge Controlled PWM Outputs 510
5 Pin description 510
6 Register description 511
6.1 PWM Interrupt Register (PWM1IR - 0x4001 8000) 512
6.2 PWM Timer Control Register (PWM1TCR 0x4001 8004) 513
6.3 PWM Count Control Register (PWM1CTCR - 0x4001 8070) 513
6.4 PWM Match Control Register (PWM1MCR - 0x4001 8014) 514
6.5 PWM Capture Control Register (PWM1CCR - 0x4001 8028) 515
6.6 PWM Control Register (PWM1PCR - 0x4001 804C) 516
6.7 PWM Latch Enable Register (PWM1LER - 0x4001 8050) 517

Chapter 25: LPC17xx Motor Control PWM

1 Introduction 519
2 Description 519
3 Pin description 519
4 Block Diagram 520
5 Configuring other modules for MCPWM use 521
6 General Operation 521
7 Register description 522
7.1 MCPWM Control register 523
7.1.1 MCPWM Control read address (MCCON - 0x400B 8000) 523
7.1.2 MCPWM Control set address (MCCON_SET - 0x400B 8004) 524

continued >>

© NXP B.V. 2010. All rights reserved.

7.1.3 MCPWM Control clear address (MCCON_CLR - 0x400B 8008) 524
7.2 MCPWM Capture Control register 525
7.2.1 MCPWM Capture Control read address (MCCAPCON - 0x400B 800C) 525
7.2.2 MCPWM Capture Control set address (MCCAPCON_SET - 0x400B 8010) 525
7.2.3 MCPWM Capture control clear address (MCCAPCON_CLR - 0x400B 8014) 526
7.3 MCPWM Interrupt registers 526
7.3.1 MCPWM Interrupt Enable read address (MCINTEN - 0x400B 8050) 526
7.3.2 MCPWM Interrupt Enable set address (MCINTEN_SET - 0x400B 8054) 527
7.3.3 MCPWM Interrupt Enable clear address (MCINTEN_CLR - 0x400B 8058) 527
7.3.4 MCPWM Interrupt Flags read address (MCINTF - 0x400B 8068) 527
7.3.5 MCPWM Interrupt Flags set address (MCINTF_SET - 0x400B 806C) 527
7.3.6 MCPWM Interrupt Flags clear address (MCINTF_CLR - 0x400B 8070) 527
7.4 MCPWM Count Control register 528
7.4.1 MCPWM Count Control read address (MCCNTCON - 0x400B 805C) 528
7.4.2 MCPWM Count Control set address (MCCNTCON_SET - 0x400B 8060) 529
7.4.3 MCPWM Count Control clear address (MCCNTCON_CLR - 0x400B 8064) 529
7.5 MCPWM Timer/Counter 0-2 registers (MCTC0-2 - 0x400B 8018, 0x400B 801C, 0x400B 8020) 529
7.6 MCPWM Limit 0-2 registers (MCLIM0-2 - 0x400B 8024, 0x400B 8028, 0x400B 802C) 530
7.7 MCPWM Match 0-2 registers (MCMAT0-2 - 0x400B 8030, 0x400B 8034, 0x400B 8038) 530
7.7.1 Match register in Edge-Aligned mode 531
7.7.2 Match register in Center-Aligned mode 531
7.7.3 0 and 100% duty cycle 531
7.8 MCPWM Dead-time register (MCDT - 0x400B 803C) 531
7.9 MCPWM Commutation Pattern register (MCCP - 0x400B 8040) 532
7.10 MCPWM Capture Registers 532
7.10.1 MCPWM Capture read addresses (MCCAP0-2 - 0x400B 8044, 0x400B 8048, 0x400B 804C) 532
7.10.2 MCPWM Capture clear address (MCCAP_CLR - 0x400B 8074) 532
8 PWM operation 534
8.1 Pulse-width modulation 534
8.2 Shadow registers and simultaneous updates 536
8.3 Fast Abort (ABORT) 536
8.4 Capture events 536
8.5 External event counting (Counter mode) . . . 537
8.6 Three-phase DC mode 537
8.7 Three phase AC mode 538
8.8 Interrupts 539

Chapter 26: LPC17xx Quadrature Encoder Interface (QEI)

1 Basic configuration 540
2 Features 540
3 Introduction 540
4 Functional description 542
4.1 Input signals 542
4.1.1 Quadrature input signals 542
4.1.2 Digital input filtering 543
4.2 Position capture 543
4.3 Velocity capture 543
4.4 Velocity compare 544
5 Pin description 545
6 Register description 546
6.1 Register summary 546
6.2 Control registers 547
6.2.1 QEI Control register (QEICON - 0x400B C000) 547
6.2.2 QEI Configuration register (QEICONF - 0x400B C008) 547
6.2.3 QEI Status register (QEISTAT - 0x400B C004) 547
6.3 Position, index and timer registers 548
6.3.1 QEI Position register (QEIPOS - 0x400B C00C) 548
6.3.2 QEI Maximum Position register (QEIMAXPOS - 0x400B C010) 548
6.3.3 QEI Position Compare register 0 (CMPOS0 - 0x400B C014) 548
6.3.4 QEI Position Compare register 1 (CMPOS1 - 0x400B C018) 548
6.3.5 QEI Position Compare register 2 (CMPOS2 - 0x400B C01C) 549
6.3.6 QEI Index Count register (INXCNT - 0x400B C020) 549
6.3.7 QEI Index Compare register (INXCMP - 0x400B C024) 549
6.3.8 QEI Timer Reload register (QEILOAD - 0x400B C028) 549

continued >>

© NXP B.V. 2010. All rights reserved.

6.3.9 QEI Timer register (QEITIME - 0x400B C02C) . . . 549
6.3.10 QEI Velocity register (QEIVEL - 0x400B C030) . . . 550
6.3.11 QEI Velocity Capture register (QEICAP - 0x400B C034) . . . 550
6.3.12 QEI Velocity Compare register (VELCOMP - 0x400B C038) . . . 550
6.3.13 QEI Digital Filter register (FILTER - 0x400B C03C) . . . 550
6.4 Interrupt registers . . . 551
6.4.1 QEI Interrupt Status register (QEIINTSTAT) 551
6.4.2 QEI Interrupt Set register (QEISET - 0x400B CFEC) . . . 551
6.4.3 QEI Interrupt Clear register (QEICLR - 0x400B CFE8) . . . 552
6.4.4 QEI Interrupt Enable register (QEIIE - 0x400B CFE4) . . . 552
6.4.5 QEI Interrupt Enable Set register (QEIIES - 0x400B CFDC) . . . 553
6.4.6 QEI Interrupt Enable Clear register (QEIIEC - 0x400B CFD8) . . . 554

Chapter 27: LPC17xx Real-Time Clock (RTC) and backup registers

1 Basic configuration . . . 555
2 Features . . . 555
3 Description . . . 555
4 Architecture . . . 556
5 Pin description . . . 557
6 Register description . . . 557
6.1 RTC interrupts . . . 559
6.2 Miscellaneous register group . . . 559
6.2.1 Interrupt Location Register (ILR - 0x4002 4000) . . . 559
6.2.2 Clock Control Register (CCR - 0x4002 4008) 559
6.2.3 Counter Increment Interrupt Register (CIIR - 0x4002 400C) . . . 560
6.2.4 Alarm Mask Register (AMR - 0x4002 4010) . 560
6.2.5 RTC Auxiliary control register (RTC_AUX - 0x4002 405C) . . . 561
6.2.6 RTC Auxiliary Enable register (RTC_AUXEN - 0x4002 4058) . . . 561
6.3 Consolidated time registers . . . 562
6.3.1 Consolidated Time Register 0 (CTIME0 - 0x4002 4014) . . . 562
6.3.2 Consolidated Time Register 1 (CTIME1 - 0x4002 4018) . . . 562
6.3.3 Consolidated Time Register 2 (CTIME2 - 0x4002 401C) . . . 562
6.4 Time Counter Group . . . 563
6.4.1 Leap year calculation . . . 563
6.4.2 Calibration register (CALIBRATION - address 0x4002 4040) . . . 563
6.5 Calibration procedure . . . 564
6.6 General purpose registers . . . 565
6.6.1 General purpose registers 0 to 4 (GPREG0 to GPREG4 - addresses 0x4002 4044 to 0x4002 4054) . . . 565
6.7 Alarm register group . . . 565
7 RTC usage notes . . . 565

Chapter 28: LPC17xx Watchdog Timer (WDT)

1 Features . . . 566
2 Applications . . . 566
3 Description . . . 567
4 Register description . . . 567
4.1 Watchdog Mode register (WDMOD - 0x4000 0000) . . . 568
4.2 Watchdog Timer Constant register (WDTC - 0x4000 0004) . . . 569
4.3 Watchdog Feed register (WDFEED - 0x4000 0008) . . . 569
4.4 Watchdog Timer Value register (WDTV - 0x4000 000C) . . . 569
4.5 Watchdog Timer Clock Source Selection register (WDCLKSEL - 0x4000 0010) . . . 569
5 Block diagram . . . 570

Chapter 29: LPC17xx Analog-to-Digital Converter (ADC)

1 Basic configuration . . . 571
2 Features . . . 571
3 Description . . . 571
4 Pin description . . . 572
5 Register description . . . 573
5.1 A/D Control Register (AD0CR - 0x4003 4000) . . . 574

continued >>

© NXP B.V. 2010. All rights reserved.

5.2 A/D Global Data Register (AD0GDR - 0x4003 4004) 575
5.3 A/D Interrupt Enable register (AD0INTEN - 0x4003 400C) 575
5.4 A/D Data Registers (AD0DR0 to AD0DR7 - 0x4003 4010 to 0x4003 402C) 576
5.5 A/D Status register (ADSTAT - 0x4003 4030) 577
5.6 A/D Trim register (ADTRIM - 0x4003 4034). 577
6 Operation 578
6.1 Hardware-triggered conversion 578
6.2 Interrupts 578
6.3 Accuracy vs. digital receiver 578
6.4 DMA control 578

Chapter 30: LPC17xx Digital-to-Analog Converter (DAC)

1 Basic configuration 579
2 Features 579
3 Pin description 579
4 Register description 580
4.1 D/A Converter Register (DACR - 0x4008 C000) 580
4.2 D/A Converter Control register (DACCTRL - 0x4008 C004) 580
4.3 D/A Converter Counter Value register (DACCNTVAL - 0x4008 C008). 581
5 Operation 581
5.1 DMA counter 581
5.2 Double buffering. 581

Chapter 31: LPC17xx General Purpose DMA (GPDMA) controller

1 Basic configuration 583
2 Introduction 583
3 Features 583
4 Functional description 584
4.1 DMA controller functional description 584
4.1.1 AHB slave interface 584
4.1.2 Control logic and register bank. 585
4.1.3 DMA request and response interface 585
4.1.4 Channel logic and channel register bank . . . 585
4.1.5 Interrupt request 585
4.1.6 AHB master interface 585
4.1.7 Channel hardware 588
4.1.8 DMA request priority. 588
4.1.9 Interrupt generation 588
4.2 DMA system connections 588
4.2.1 DMA request signals 588
4.2.2 DMA response signals 588
4.2.3 DMA request connections 589
5 Register description 590
5.1 DMA Interrupt Status register (DMACIntStat - 0x5000 4000) 592
5.2 DMA Interrupt Terminal Count Request Status register (DMACIntTCStat - 0x5000 4004). . . 592
5.3 DMA Interrupt Terminal Count Request Clear register (DMACIntTCClear - 0x5000 4008) . 592
5.4 DMA Interrupt Error Status register (DMACIntErrStat - 0x5000 400C) 592
5.5 DMA Interrupt Error Clear register (DMACIntErrClr - 0x5000 4010) 593
5.6 DMA Raw Interrupt Terminal Count Status register (DMACRawIntTCStat - 0x5000 4014) 593
5.7 DMA Raw Error Interrupt Status register (DMACRawIntErrStat - 0x5000 4018) 593
5.8 DMA Enabled Channel register (DMACEnbldChns - 0x5000 401C) 594
5.9 DMA Software Burst Request register (DMACSoftBReq - 0x5000 4020). 594
5.10 DMA Software Single Request register (DMACSoftSReq - 0x5000 4024). 595
5.11 DMA Software Last Burst Request register (DMACSoftLBReq - 0x5000 4028). 595
5.12 DMA Software Last Single Request register (DMACSoftLSReq - 0x5000 402C) 595
5.13 DMA Configuration register (DMACConfig - 0x5000 4030) 596
5.14 DMA Synchronization register (DMACSync - 0x5000 4034) 596
5.15 DMA Request Select register (DMAReqSel - 0x400F C1C4) 597
5.16 DMA Channel registers 597
5.17 DMA Channel Source Address registers (DMACCxSrcAddr - 0x5000 41x0). 598
5.18 DMA Channel Destination Address registers (DMACCxDestAddr - 0x5000 41x4). 598
5.19 DMA Channel Linked List Item registers (DMACCxLLI - 0x5000 41x8). 598
5.20 DMA channel control registers (DMACCxControl - 0x5000 41xC). 599
5.20.1 Protection and access information. 599

continued >>

© NXP B.V. 2010. All rights reserved.

5.21 DMA Channel Configuration registers (DMACCxConfig - 0x5000 41x0) 601
5.21.1 Lock control 603
5.21.2 Transfer type 603
6 Using the DMA controller 604
6.1 Programming the DMA controller 604
6.1.1 Enabling the DMA controller 604
6.1.2 Disabling the DMA controller 604
6.1.3 Enabling a DMA channel 604
6.1.4 Disabling a DMA channel 604
6.1.5 Setting up a new DMA transfer 604
6.1.6 Halting a DMA channel 605
6.1.7 Programming a DMA channel 605
6.2 Flow control 605
6.2.1 Peripheral-to-memory or memory-to-peripheral DMA flow 606
6.2.2 Peripheral-to-peripheral DMA flow 606
6.2.3 Memory-to-memory DMA flow 607
6.3 Interrupt requests 607
6.3.1 Hardware interrupt sequence flow 608
6.4 Address generation 608
6.4.1 Word-aligned transfers across a boundary . 608
6.5 Scatter/gather 608
6.5.1 Linked list items 609

Chapter 32: LPC17xx Flash memory interface and programming

1 Introduction 612
2 Features 612
3 Description 612
3.1 Memory map after any reset 613
3.1.1 Criterion for Valid User Code 613
3.2 Communication protocol 614
3.2.1 ISP command format 614
3.2.2 ISP response format 614
3.2.3 ISP data format 614
3.2.4 ISP flow control 615
3.2.5 ISP command abort 615
3.2.6 Interrupts during IAP 615
3.2.7 RAM used by ISP command handler 615
3.2.8 RAM used by IAP command handler 615
4 Boot process flowchart 616
5 Sector numbers 617
6 Code Read Protection (CRP) 618
7 ISP commands 620
7.1 Unlock <Unlock code> 620
7.2 Set Baud Rate <Baud Rate> <stop bit> 621
7.3 Echo <setting> 621
7.4 Write to RAM <start address> <number of bytes> 621
7.5 Read Memory <address> <no. of bytes> . . . 622
7.6 Prepare sector(s) for write operation <start sector number> <end sector number> 623
7.7 Copy RAM to Flash <flash address> <RAM address> <no of bytes> 623
7.8 Go <address> <mode> 624
7.9 Erase sector(s) <start sector number> <end sector number> 624
7.10 Blank check sector(s) <sector number> <end sector number> 625
7.11 Read Part Identification number 625
7.12 Read Boot Code version number 626
7.13 Read device serial number 626
7.14 Compare <address1> <address2> <no of bytes> 626
7.15 ISP Return Codes 627
8 IAP commands 628
8.1 Prepare sector(s) for write operation 630
8.2 Copy RAM to Flash 630
8.3 Erase Sector(s) 631
8.4 Blank check sector(s) 631
8.5 Read part identification number 631
8.6 Read Boot Code version number 632
8.7 Read device serial number 632
8.8 Compare <address1> <address2> <no of bytes> 632
8.9 Re-invoke ISP 633
8.10 IAP Status Codes 633
9 JTAG flash programming interface 633
10 Flash signature generation 634
10.1 Register description for signature generation 634
10.1.1 Signature generation address and control registers 635
10.1.2 Signature generation result registers 635
10.1.3 Flash Module Status register (FMSTAT - 0x0x4008 4FE0) 636
10.1.4 Flash Module Status Clear register (FMSTATCLR - 0x0x4008 4FE8) 636
10.2 Algorithm and procedure for signature generation 637

continued >>

© NXP B.V. 2010. All rights reserved.

Chapter 33: LPC17xx JTAG, Serial Wire Debug, and Trace

1 Features 638
2 Introduction 638
3 Description 638
4 Pin Description 638
5 Debug Notes 639
6 Debug memory re-mapping 640
6.1 Memory Mapping Control register (MEMMAP - 0x400F C040) 640
7 JTAG TAP Identification 640

Chapter 34: Appendix: Cortex-M3 User Guide

1 ARM Cortex-M3 User Guide: Introduction . . 641
1.1 About the processor and core peripherals . . 641
1.1.1 System level interface 642
1.1.2 Integrated configurable debug 642
1.1.3 Cortex-M3 processor features and benefits summary 643
1.1.4 Cortex-M3 core peripherals 643
2 ARM Cortex-M3 User Guide: Instruction Set 644
2.1 Instruction set summary 644
2.2 Intrinsic functions 647
2.3 About the instruction descriptions 647
2.3.1 Operands 648
2.3.2 Restrictions when using PC or SP 648
2.3.3 Flexible second operand 648
2.3.4 Shift Operations 649
2.3.5 Address alignment 652
2.3.6 PC-relative expressions 653
2.3.7 Conditional execution 653
2.3.8 Instruction width selection 655
2.4 Memory access instructions 657
2.4.1 ADR 658
2.4.2 LDR and STR, immediate offset 659
2.4.3 LDR and STR, register offset 662
2.4.4 LDR and STR, unprivileged 664
2.4.5 LDR, PC-relative 666
2.4.6 LDM and STM 668
2.4.7 PUSH and POP 670
2.4.8 LDREX and STREX 671
2.4.9 CLREX 673
2.5 General data processing instructions 674
2.5.1 ADD, ADC, SUB, SBC, and RSB 675
2.5.2 AND, ORR, EOR, BIC, and ORN 678
2.5.3 ASR, LSL, LSR, ROR, and RRX 680
2.5.4 CLZ 682
2.5.5 CMP and CMN 683
2.5.6 MOV and MVN 684
2.5.7 MOVT 686
2.5.8 REV, REV16, REVSH, and RBIT 687
2.5.9 TST and TEQ 688
2.6 Multiply and divide instructions 689
2.6.1 MUL, MLA, and MLS 690
2.6.2 UMULL, UMLAL, SMULL, and SMLAL 692
2.6.3 SDIV and UDIV 694
2.7 Saturating instructions 695
2.7.1 SSAT and USAT 695
2.8 Bitfield instructions 697
2.8.1 BFC and BFI 698
2.8.2 SBFX and UBFX 699
2.8.3 SXT and UXT 700
2.9 Branch and control instructions 702
2.9.1 B, BL, BX, and BLX 703
2.9.2 CBZ and CBNZ 705
2.9.3 IT 706
2.9.4 TBB and TBH 709
2.10 Miscellaneous instructions 711
2.10.1 BKPT 712
2.10.2 CPS 713
2.10.3 DMB 714
2.10.4 DSB 715
2.10.5 ISB 716
2.10.6 MRS 717
2.10.7 MSR 718
2.10.8 NOP 719
2.10.9 SEV 720
2.10.10 SVC 721
2.10.11 WFE 722
2.10.12 WFI 723
3 ARM Cortex-M3 User Guide: Processor . . . 724
3.1 Programmers model 724
3.1.1 Processor mode and privilege levels for software execution 724
3.1.2 Stacks 724
3.1.3 Core registers 725
3.1.4 Exceptions and interrupts 732
3.1.5 Data types 732
3.1.6 The Cortex Microcontroller Software Interface Standard 732
3.2 Memory model 734
3.2.1 Memory regions, types and attributes 734

continued >>

© NXP B.V. 2010. All rights reserved.

3.2.2 Memory system ordering of memory accesses . . . 735
3.2.3 Behavior of memory accesses . . . 736
3.2.4 Software ordering of memory accesses . . . 736
3.2.5 Bit-banding . . . 737
3.2.6 Memory endianness . . . 740
3.2.7 Synchronization primitives . . . 740
3.2.8 Programming hints for the synchronization primitives . . . 741
3.3 Exception model . . . 742
3.3.1 Exception states . . . 742
3.3.2 Exception types . . . 742
3.3.3 Exception handlers . . . 744
3.3.4 Vector table . . . 745
3.3.5 Exception priorities . . . 745
3.3.6 Interrupt priority grouping . . . 746
3.3.7 Exception entry and return . . . 746
3.4 Fault handling . . . 750
3.4.1 Fault types . . . 750
3.4.2 Fault escalation and hard faults . . . 751
3.4.3 Fault status registers and fault address registers . . . 751
3.4.4 Lockup . . . 752
3.5 Power management . . . 753
3.5.1 Entering sleep mode . . . 753
3.5.2 Wakeup from sleep mode . . . 754
3.5.3 The Wakeup Interrupt Controller . . . 754
3.5.4 Power management programming hints . . . 755
4 ARM Cortex-M3 User Guide: Peripherals . . . 756
4.1 About the Cortex-M3 peripherals . . . 756
4.2 Nested Vectored Interrupt Controller . . . 757
4.2.1 The CMSIS mapping of the Cortex-M3 NVIC registers . . . 757
4.2.2 Interrupt Set-enable Registers . . . 758
4.2.3 Interrupt Clear-enable Registers . . . 758
4.2.4 Interrupt Set-pending Registers . . . 759
4.2.5 Interrupt Clear-pending Registers . . . 759
4.2.6 Interrupt Active Bit Registers . . . 760
4.2.7 Interrupt Priority Registers . . . 760
4.2.8 Software Trigger Interrupt Register . . . 761
4.2.9 Level-sensitive and pulse interrupts . . . 761
4.2.10 NVIC design hints and tips . . . 762
4.3 System control block . . . 764
4.3.1 The CMSIS mapping of the Cortex-M3 SCB registers . . . 764
4.3.2 Auxiliary Control Register . . . 764
4.3.3 CPUID Base Register . . . 765
4.3.4 Interrupt Control and State Register . . . 765
4.3.5 Vector Table Offset Register . . . 767
4.3.6 Application Interrupt and Reset Control Register . . . 768
4.3.7 System Control Register . . . 769
4.3.8 Configuration and Control Register . . . 770
4.3.9 System Handler Priority Registers . . . 771
4.3.10 System Handler Control and State Register 772
4.3.11 Configurable Fault Status Register . . . 774
4.3.12 Hard Fault Status Register . . . 778
4.3.13 Memory Management Fault Address Register . . . 778
4.3.14 Bus Fault Address Register . . . 778
4.3.15 System control block design hints and tips . 779
4.4 System timer, SysTick . . . 780
4.4.1 SysTick Control and Status Register . . . 780
4.4.2 SysTick Reload Value Register . . . 781
4.4.3 SysTick Current Value Register . . . 781
4.4.4 SysTick Calibration Value Register . . . 781
4.4.5 SysTick design hints and tips . . . 782
4.5 Memory protection unit . . . 783
4.5.1 MPU Type Register . . . 784
4.5.2 MPU Control Register . . . 785
4.5.3 MPU Region Number Register . . . 786
4.5.4 MPU Region Base Address Register . . . 786
4.5.5 MPU Region Attribute and Size Register . . . 787
4.5.6 MPU access permission attributes . . . 789
4.5.7 MPU mismatch . . . 790
4.5.8 Updating an MPU region . . . 790
4.5.9 MPU design hints and tips . . . 793
5 ARM Cortex-M3 User Guide: Glossary . . . 794

Chapter 35: LPC17xx Supplementary information

1 Abbreviations . . . 798
2 Legal information . . . 799
2.1 Definitions . . . 799
2.2 Disclaimers . . . 799
2.3 Trademarks . . . 799
3 Tables . . . 801
4 Figures . . . 814
5 Contents . . . 816

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

founded by PHILIPS

© NXP B.V. 2010. All rights reserved.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 4 January 2010
Document identifier: UM10360_1