LPC2131/2132/2138

User Manual

Preliminary Release

November 22, 2004

Philips
Semiconductors

PHILIPS

Table of Contents

List of Figures

List of Tables

DOCUMENT REVISION HISTORY

2004 Aug 25:

- Prototype of combined LPC2132/2138 User Manual created from the design specification.

2004 Sep 13:

- Informatoin on counter functionality of the TIMER0/1 added into "Introduction" and "Timer/Counter0 and Timer/Counter1" chapters.
- Reference to the LPC201x in Table 23, “Power Control Register (PCON - 0xE01FC0C0),” ("System Control Block" chapter) replaced with the LPC2132/2138.
- Info on reserved bits in Table 52, “Pin Function Select Register 2 (PINSEL2 - 0xE002C014),” ("Pin Connect Block" chapter) corrected.
- Reference to the PORT2/3 in the "Register Description" section of the "GPIO" chapter removed. Number of PORT0 available pins discussed in this section also updated.

2004 Sep 14:

- RTC related information added into the "Reset" and "Wakeup Timer" sections of the "System Control Block" chapter.
- "RTC Usage Notes" section in the "Real Time Clock" chapter updated.

2004 Sep 15:

- Count Control Register description in the "Timer/Counter0 and Timer/Counter1" chapter uptaded.
- All available CAP and MAT pins listed in the Pin Description section of the "Timer/Counter0 and Timer/Counter1" chapter.
- Details on the counter mode added into the Count Control register description in the "Timer/Counter0 and Timer/Counter1" chapter.

2004 Sep 16:

- Typographic errors in the "SSP Controller (SPI1)" chapter corrected.
- Details on Flash erase/write cycles and data retention added into the "Introduction" chapter.

2004 Nov 22:

- An updated I^2C chapter included in the document.
- Missing chapter on the Memory Accelerator Module (MAM) added to the document.

1. INTRODUCTION

GENERAL DESCRIPTION

The LPC2131/2132/2138 microcontrollers are based on a 32/16 bit ARM7TDMI-S™ CPU with real-time emulation and embedded trace support, that combines the microcontroller with 32 kB, 64 kB and 512 kB of embedded high speed Flash memory. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at maximum clock rate. For critical code size applications, the alternative 16-bit Thumb® Mode reduces code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, these microcontrollers are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. With a wide range of serial communications interfaces and on-chip SRAM options of 8/16/32 kB, they are very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit 8 channel ADC(s), 10-bit DAC, PWM channels and 47 GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.

FEATURES

- 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8/16/32 kB of on-chip static RAM and 32/64/512 kB of on-chip Flash program memory. 128 bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single Flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.
- EmbeddedICE® RT and Embedded Trace interfaces offer real-time debugging with the on-chip RealMonitor™ software and high speed tracing of instruction execution.
- One (LPC2131/2132) or two (LPC2138) 8 channel 10-bit A/D converters provide a total of up to 16 analog inputs, with conversion times as low as 2.44 s per channel.
- Single 10-bit D/A converter provides variable analog output. (LPC2132/2138 only)
- Two 32-bit timers/counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Real-time clock equipped with independent power and clock supply permitting extremely low power consumption in power-save modes.
- Multiple serial interfaces including two UARTs (16C550), two Fast I2C (400 kbit/s), SPI™ and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller with configurable priorities and vector addresses.
- Up to 47 of 5 V tolerant general purpose I/O pins in tiny LQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip Phase-Locked Loop (PLL) with settling time of 100 microseconds.
- On-chip crystal oscillator with an operating range of 1 MHz to 30 MHz.
- Power saving modes include Idle and Power-down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling down for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt.
- Single power supply chip with Power-On Reset (POR) and Brown-Out Detection (BOD) circuits:
  - CPU operating voltage range of 3.0 V to 3.6 V (3.3 V 10 %) with 5 V tolerant I/O pads.

APPLICATIONS

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

DEVICE INFORMATION

Table 1: LPC2131/2132/2138 device information

Device	No. of pins	On-chip RAM	On-chip FLASH	No. of 10-bit AD Channels	No. of 10-bit DA Channels	Note
LPC2131	64	8 kB	32	8	-	-
LPC2132	64	16 kB	64	8	-	-
LPC2138	64	32 kB	512	16	1	-

ARCHITECTURAL OVERVIEW

The LPC2131/2132/2138 consists of an ARM7TDMI-S CPU with emulation support, the ARM7 Local Bus for interface to on-chip memory controllers, the AMBA Advanced High-performance Bus (AHB) for interface to the interrupt controller, and the VLSI Peripheral Bus (VPB, a compatible superset of ARM's AMBA Advanced Peripheral Bus) for connection to on-chip peripheral functions. The LPC2131/2132/2138 configures the ARM7TDMI-S processor in little-endian byte order.

AHB peripherals are allocated a 2 megabyte range of addresses at the very top of the 4 gigabyte ARM memory space. Each AHB peripheral is allocated a 16 kilobyte address space within the AHB address space. LPC2131/2132/2138 peripheral functions (other than the interrupt controller) are connected to the VPB bus. The AHB to VPB bridge interfaces the VPB bus to the AHB bus. VPB peripherals are also allocated a 2 megabyte range of addresses, beginning at the 3.5 gigabyte address point. Each VPB peripheral is allocated a 16 kilobyte address space within the VPB address space.

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

ARM7TDMI-S PROCESSOR

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

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

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

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

- The standard 32-bit ARM instruction set.
- A 16-bit THUMB instruction set.

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

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

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

ON-CHIP FLASH MEMORY SYSTEM

The LPC2131/2132/2138 incorporate a 32 kB, 64 kB and 512 kB Flash memory system respectively. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways: over the serial built-in JTAG interface, using In System Programming (ISP) and UART0, or by means of In Application Programming (IAP) capabilities. The application program, using the In Application Programming (IAP) functions, may also erase and/or program the Flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc. When the LPC2131/2132/2138 on-chip bootloader is used, 32/64/500 kB of Flash memory is available for user code.

The LPC2131/2132/2138 Flash memory provides minimum of 10,000 erase/write cycles and 10 years of data-retention.

ON-CHIP STATIC RAM

On-Chip static RAM (SRAM) may be used for code and/or data storage. The SRAM may be accessed as 8-bits, 16-bits, and 32-bits. The LPC2131/2132/2138 provide 8/16/32 kB of static RAM respectively.

The LPC2131/LPC2132/2138 SRAM is designed to be accessed as a byte-addressed memory. Word and halfword accesses to the memory ignore the alignment of the address and access the naturally-aligned value that is addressed (so a memory access ignores address bits 0 and 1 for word accesses, and ignores bit 0 for halfword accesses). Therefore valid reads and writes require data accessed as halfwords to originate from addresses with address line 0 being 0 (addresses ending with 0, 2, 4, 6, 8, A, C, adnd E) and data accessed as words to originate from adresses with address lines 0 and 1 being 0 (addresses ending with 0, 4, 8, and C). This rule applies to both off and on-chip memory usage.

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

BLOCK DIAGRAM

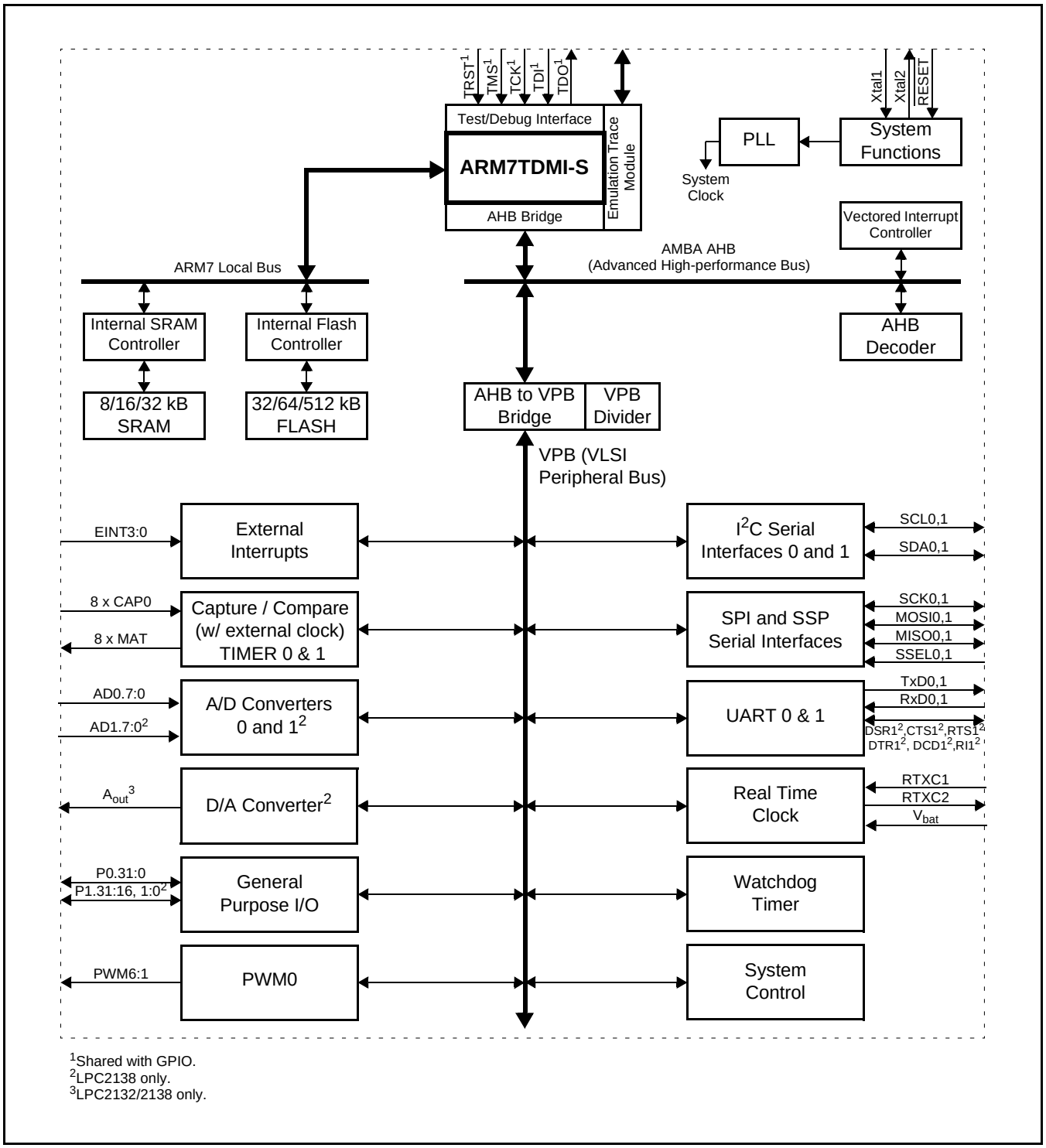


Figure 1: LPC2131/2132/2138 Block Diagram

2. LPC2131/2132/2138 MEMORY ADDRESSING

MEMORY MAPS

The LPC2131/2132/2138 incorporates several distinct memory regions, shown in the following figures. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset. The interrupt vector area supports address re-mapping, which is described later in this section.

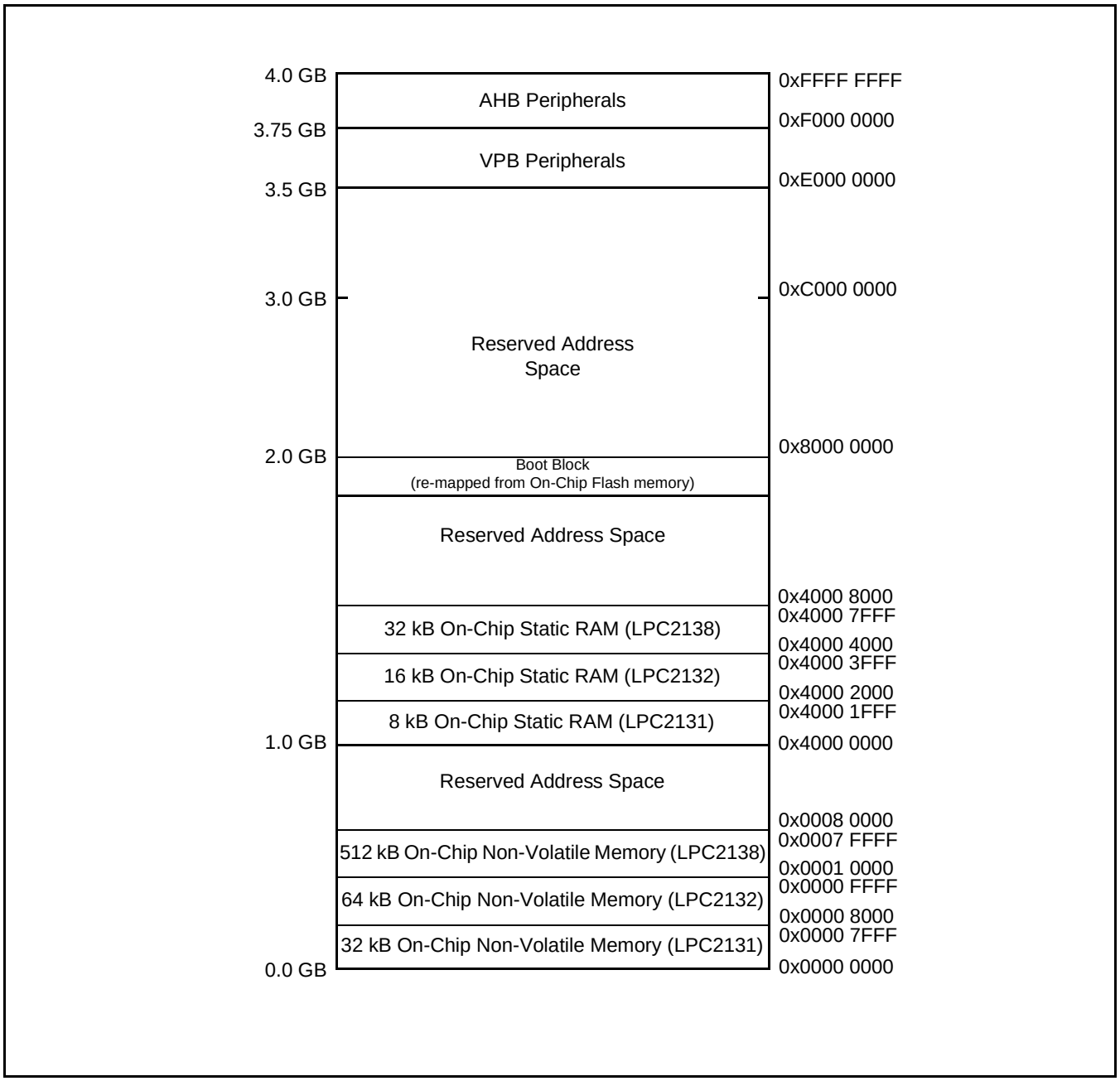


Figure 2: System Memory Map

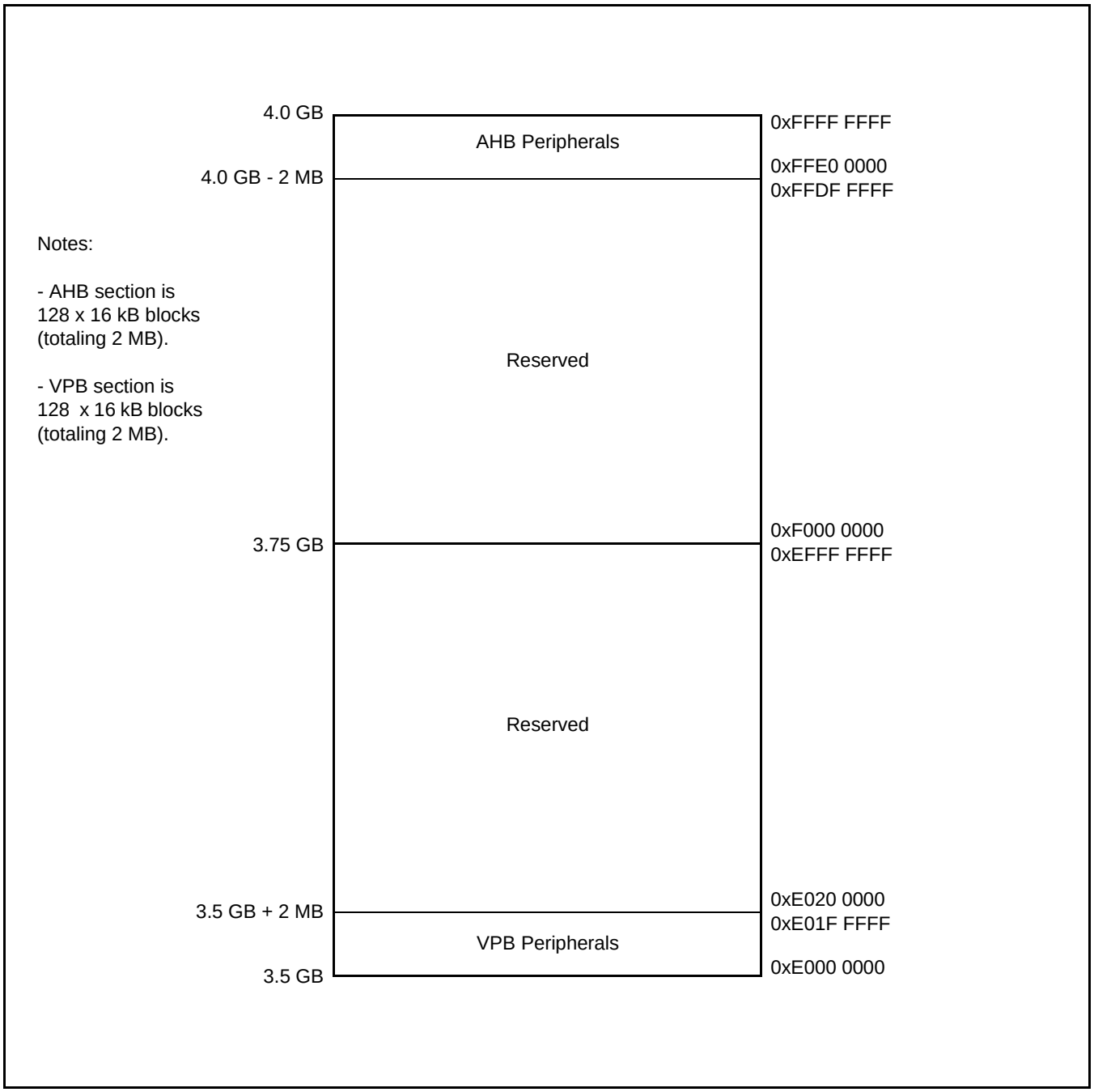


Figure 3: Peripheral Memory Map

Figures 3 through 5 show different views of the peripheral address space. Both the AHB and VPB peripheral areas are 2 megabyte spaces which are divided up into 128 peripherals. Each peripheral space is 16 kilobytes in size. This allows simplifying the address decoding for each peripheral. All peripheral register addresses are word aligned (to 32-bit boundaries) regardless of their size. This eliminates the need for byte lane mapping hardware that would be required to allow byte (8-bit) or half-word (16-bit) accesses to occur at smaller boundaries. An implication of this is that word and half-word registers must be accessed all at once. For example, it is not possible to read or write the upper byte of a word register separately.

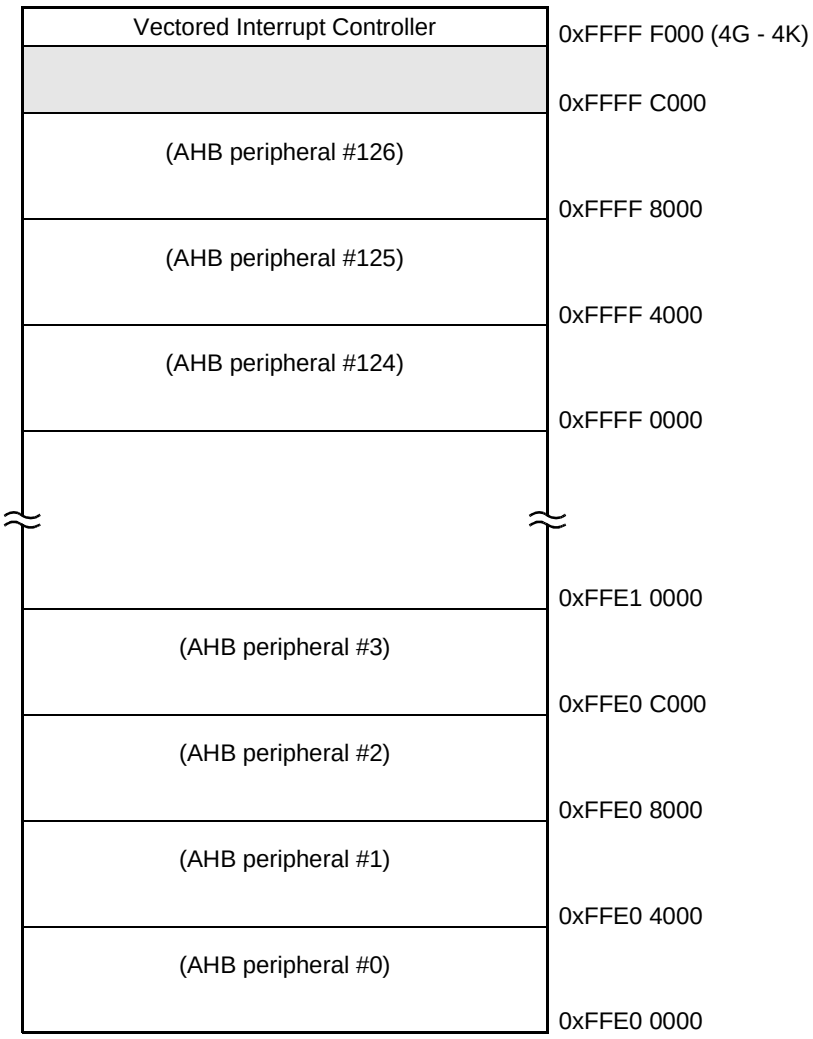


Figure 4: AHB Peripheral Map

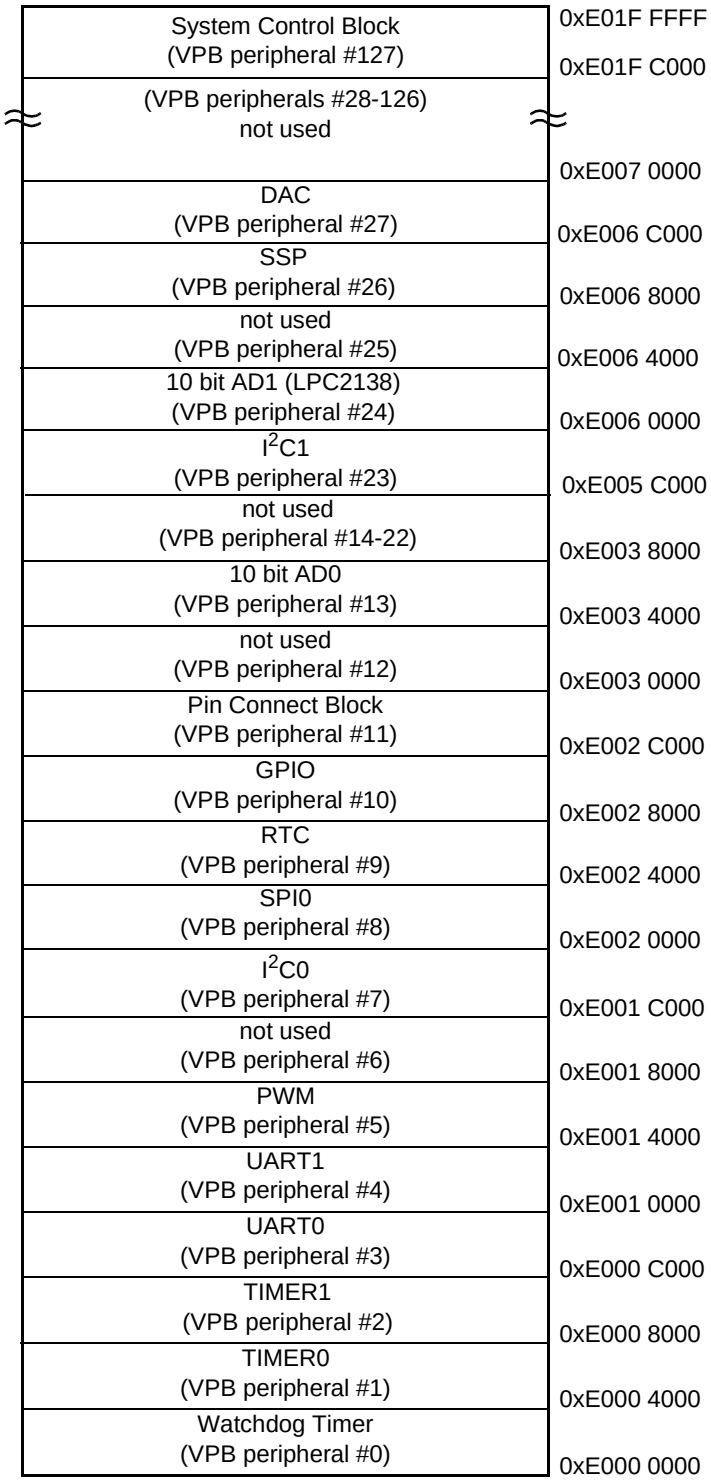


Figure 5: VPB Peripheral Map

LPC2131/2132/2138 MEMORY RE-MAPPING AND BOOT BLOCK

Memory Map Concepts and Operating Modes

The basic concept on the LPC2131/2132/2138 is that each memory area has a "natural" location in the memory map. This is the address range for which code residing in that area is written. The bulk of each memory space remains permanently fixed in the same location, eliminating the need to have portions of the code designed to run in different address ranges.

Because of the location of the interrupt vectors on the ARM7 processor (at addresses 0x0000 0000 through 0x0000 001C, as shown in Table 2 below), a small portion of the Boot Block and SRAM spaces need to be re-mapped in order to allow alternative uses of interrupts in the different operating modes described in Table 3. Re-mapping of the interrupts is accomplished via the Memory Mapping Control feature described in the System Control Block section.

Table 2: ARM Exception Vector Locations

Address	Exception
0x0000 0000	Reset
0x0000 0004	Undefined Instruction
0x0000 0008	Software Interrupt
0x0000 000C	Prefetch Abort (instruction fetch memory fault)
0x0000 0010	Data Abort (data access memory fault)
0x0000 0014	Reserved *
0x0000 0018	IRQ
0x0000 001C	FIQ

*: Identified as reserved in ARM documentation, this location is used by the Boot Loader as the Valid User Program key. This is descibed in detail in Flash Memory System and Programming on page 225.

Table 3: LPC2131/2132/2138 Memory Mapping Modes

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

Memory Re-Mapping

In order to allow for compatibility with future derivatives, the entire Boot Block is mapped to the top of the on-chip memory space. In this manner, the use of larger or smaller flash modules will not require changing the location of the Boot Block (which would require changing the Boot Loader code itself) or changing the mapping of the Boot Block interrupt vectors. Memory spaces other than the interrupt vectors remain in fixed locations. Figure 6 shows the on-chip memory mapping in the modes defined above.

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

There are three reasons this configuration was chosen:

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

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

Details on re-mapping and examples can be found in System Control Block on page 29.

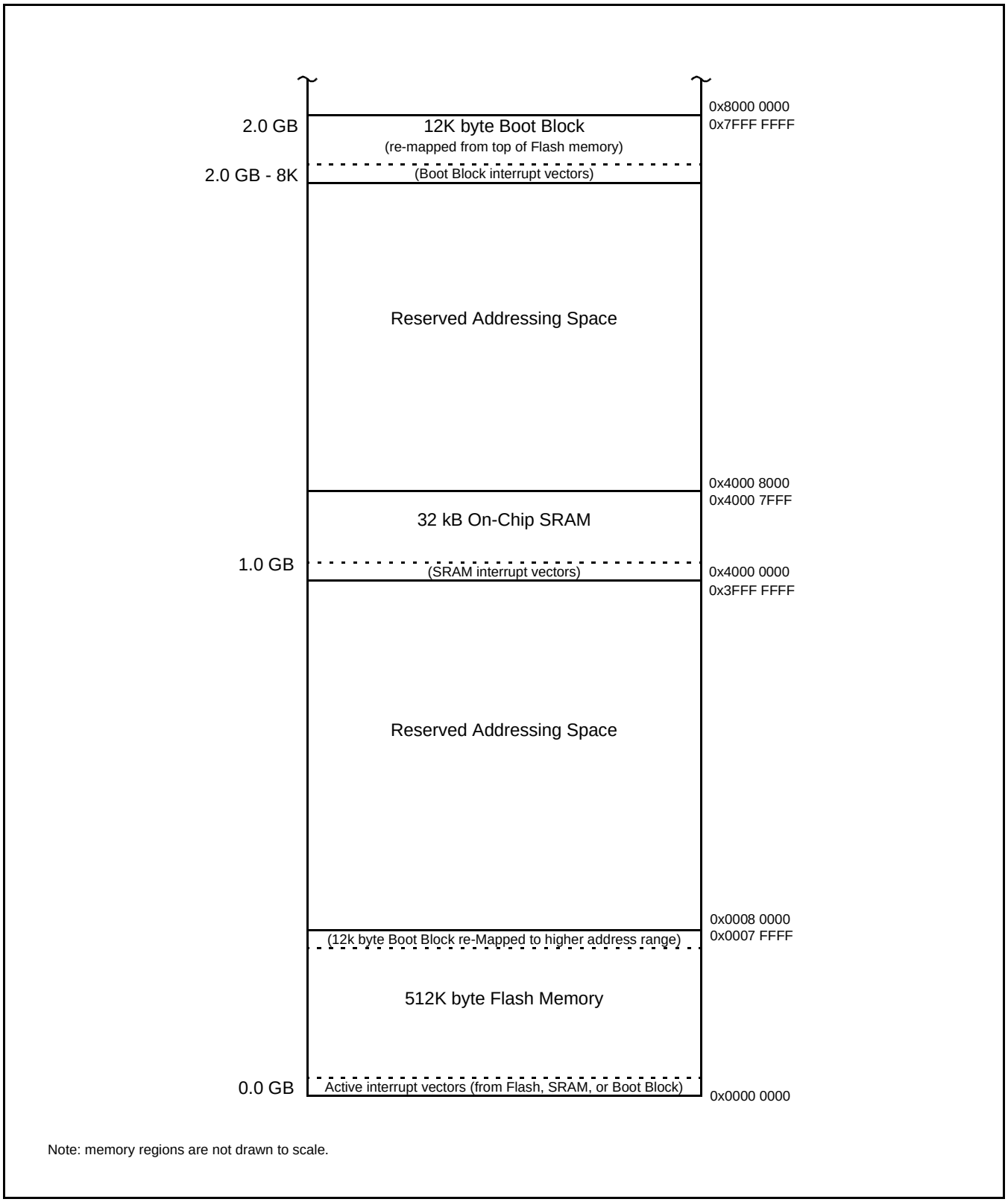


Figure 6: Map of lower memory is showing re-mapped and re-mappable areas (LPC2138 with 512 kB Flash).

PREFETCH ABORT AND DATA ABORT EXCEPTIONS

The LPC2131/2132/2138 generates the appropriate bus cycle abort exception if an access is attempted for an address that is in a reserved or unassigned address region. The regions are:

- Areas of the memory map that are not implemented for a specific ARM derivative. For the LPC2131/2132/2138, this is:
  - Address space between On-Chip Non-Volatile Memory and On-Chip SRAM, labelled "Reserved Addressing Space" in Figure 2 and Figure 6. For 32 kB Flash device this is memory address range from 0x0000 8000 to 0x3FFF FFFF, for 64 kB Flash device this is memory address range from 0x0001 0000 to 0x3FFF FFFF, while for 512 kB Flash device this range is from 0x0008 0000 to 0x3FFF FFFF.
  - Address space between On-Chip Static RAM and External Memory. Labelled "Reserved Addressing Space" in Figure 2. For 8 kB SRAM device this is memory address range from 0x4000 1FFF to 0x7FFF DFFFF, for 16 kB SRAM device this is memory address range from 0x4000 3FFF to 0x7FFF DFFF, while for 32 kB SRAM device this range is from 0x4000 7FFF to 0x7FFF D000. This is an address range from 0x4000 3FFF to 0x7FFF D000.
  - Reserved regions of the AHB and VPB spaces. See Figure 3.
- Unassigned AHB peripheral spaces. See Figure 4.
- Unassigned VPB peripheral spaces. See Figure 5.

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

Within the address space of an existing VPB peripheral, a data abort exception is not generated in response to an access to an undefined address. Address decoding within each peripheral is limited to that needed to distinguish defined registers within the peripheral itself. For example, an access to address 0xE000D000 (an undefined address within the UART0 space) may result in an access to the register defined at address 0xE000C000. Details of such address aliasing within a peripheral space are not defined in the LPC2131/2132/2138 documentation and are not a supported feature.

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

3. SYSTEM CONTROL BLOCK

SUMMARY OF SYSTEM CONTROL BLOCK FUNCTIONS

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

- Crystal Oscillator.
- External Interrupt Inputs.
- Memory Mapping Control.
- PLL.
- Power Control.
- Reset.
- VPB Divider.
- Wakeup Timer.

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

PIN DESCRIPTION

Table 4 shows pins that are associated with System Control block functions.

Table 4: Pin summary

Pin name	Pin direction	Pin Description
X1	Input	**Crystal Oscillator Input-** Input to the oscillator and internal clock generator circuits.
X2	Output	**Crystal Oscillator Output-** Output from the oscillator amplifier.
EINT0	Input	**External Interrupt Input 0-** An active low general purpose interrupt input. This pin may be used to wake up the processor from Idle or Power down modes. Pins P0.1 and P0.16 can be selected to perform EINT0 function.
EINT1	Input	**External Interrupt Input 1-** See the EINT0 description above. Pins P0.3 and P0.14 can be selected to perform EINT1 function. LOW level on pin P0.14 immediately after reset is considered as an external hardware request to start the ISP command handler. More details on ISP and Serial Boot Loader can be found in "Flash Memory System and Programming" chapter.
EINT2	Input	**External Interrupt Input 2-** See the EINT0 description above. Pins P0.7 and P0.15 can be selected to perform EINT2 function.
EINT3	Input	**External Interrupt Input 3-** See the EINT0 description above. Pins P0.9, P0.20 and P0.30 can be selected to perform EINT3 function.
$\overline{\text{RESET}}$	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 0.

REGISTER DESCRIPTION

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

Table 5: Summary of System Control Registers

Name	Description	Access	Reset Value*	Address
External Interrupts				
EXTINT	External Interrupt Flag Register.	R/W	0	0xE01FC140
EXTWAKE	External Interrupt Wakeup Register.	R/W	0	0xE01FC144
EXTMODE	External Interrupt Flag Register.	R/W	0	0xE01FC148
EXTPOLAR	External Interrupt Wakeup Register.	R/W	0	0xE01FC14C
Memory Mapping Control				
MEMMAP	Memory Mapping Control.	R/W	0	0xE01FC040
Phase Locked Loop				
PLLCON	PLL Control Register.	R/W	0	0xE01FC080
PLLCFG	PLL Configuration Register.	R/W	0	0xE01FC084
PLLSTAT	PLL Status Register.	RO	0	0xE01FC088
PLLFEED	PLL Feed Register.	WO	NA	0xE01FC08C
Power Control				
PCON	Power Control Register.	R/W	0	0xE01FC0C0
PCONP	Power Control for Peripherals.	R/W	0x3BE	0xE01FC0C4
VPB Divider				
VPBDIV	VPB Divider Control.	R/W	0	0xE01FC100
Reset				
RSID	Reset Source Identification Register	R/W	0	0xE01FC180
Code Security/Debugging				
CSPR	Code Security Protection Register	RO	0	0xE01FC184

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

CRYSTAL OSCILLATOR

While an input signal of 50-50 duty cycle within a frequency range from 1 MHz to 50 MHz can be used by LPC2131/2132/2138 if supplied to its input XTAL1 pin, this microcontroller's onboard oscillator circuit supports external crystals in the range of 1 MHz to 30 MHz only. If on-chip PLL system or boot-loader is used, input clock frequency is limited to exclusive range of 10 MHz to 25 MHz.

The oscillator output frequency is called F_{osc} and the ARM processor clock frequency is referred to as cclk for purposes of rate equations, etc. elsewhere in this document. F_{osc} and cclk are the same value unless the PLL is running and connected. Refer to the PLL description in this chapter for details and frequency limitations.

Onboard oscillator in LPC2131/2132/2138 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 (Cc in Figure 7, drawing a), with an amplitude of at least 200 mVrms. X2 pin in this configuration can be left not connected. If slave mode is selected, F_{osc} signal of 50-50 duty cycle can range from 1 MHz to 50 MHz.

External components and models used in oscillation mode are shown in Figure 7, drawings b and c, and in Table 6. 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 Cp in Figure 7, drawing c, represents the parallel package capacitance and should not be larger than 7 pF. Parameters F_C, C_L, R_S and C_P are supplied by the crystal manufacturer.

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

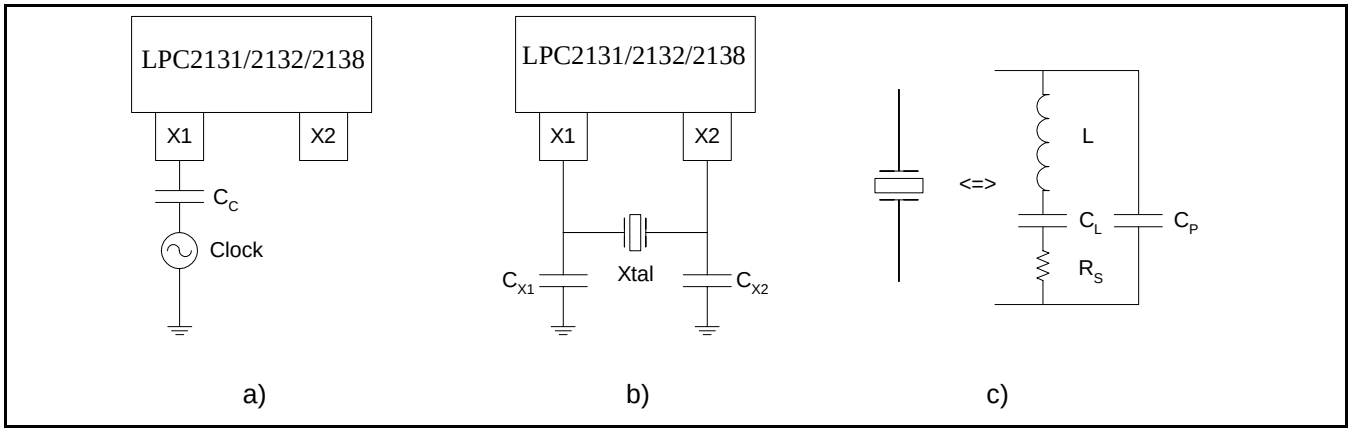


Figure 7: Oscillator modes and models: a) *slave mode* of operation, b) *oscillation mode* of operation, c) external crystal model used for $C_{X1/X2}$ evaluation

Table 6: Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters)

Fundamental Oscillation Frequency F_C	Crystal Load Capacitance C_L	Max. Crystal Series Resistance R_S	External Load Capacitors C_{X1}, C_{X2}
1 - 5 MHz	10 pF	n.a.	n.a.
	20 pF	n.a.	n.a.
	30 pF	< 300 Ω	58 pF, 58 pF

Table 6: Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters)

Fundamental Oscillation Frequency F_C	Crystal Load Capacitance C_L	Max. Crystal Series Resistance R_S	External Load Capacitors C_{X1}, C_{X2}
5 - 10 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 300 Ω	38 pF, 38 pF
	30 pF	< 300 Ω	58 pF, 58 pF
10 - 15 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 220 Ω	38 pF, 38 pF
	30 pF	< 140 Ω	58 pF, 58 pF
15 - 20 MHz	10 pF	< 220 Ω	18 pF, 18 pF
	20 pF	< 140 Ω	38 pF, 38 pF
	30 pF	< 80 Ω	58 pF, 58 pF
20 - 25 MHz	10 pF	< 160 Ω	18 pF, 18 pF
	20 pF	< 90 Ω	38 pF, 38 pF
	30 pF	< 50 Ω	58 pF, 58 pF
25 - 30 MHz	10 pF	<130 Ω	18 pF, 18 pF
	20 pF	<50 Ω	38 pF, 38 pF
	30 pF	n.a.	n.a.

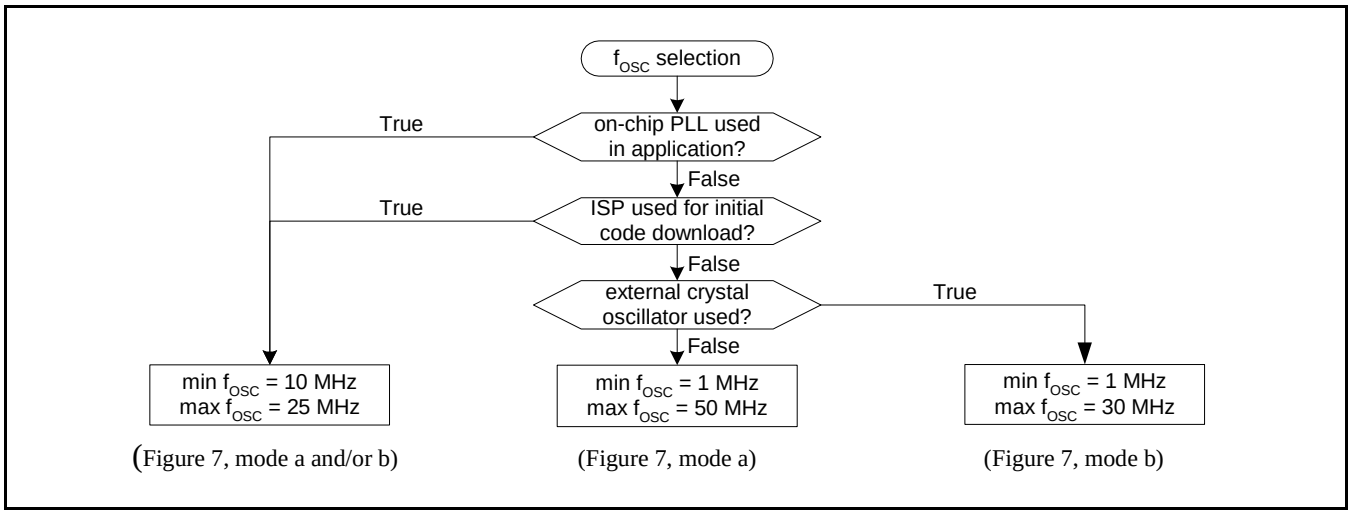


Figure 8: F_{OSC} selection algorithm

EXTERNAL INTERRUPT INPUTS

The LPC2131/2132/2138 includes four External Interrupt Inputs as selectable pin functions. The External Interrupt Inputs can optionally be used to wake up the processor from the Power Down mode.

Register Description

The external interrupt function has four registers associated with it. The EXTINT register contains the interrupt flags, and the EXTWAKEUP register contains bits that enable individual external interrupts to wake up the LPC2131/2132/2138 from Power Down mode. The EXTMODE and EXTPOLAR registers specify the level and edge sensitivity parameters.

Table 7: External Interrupt Registers

Address	Name	Description	Access
0xE01FC140	EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, and EINT2. See Table 8.	R/W
0xE01FC144	EXTWAKE	The External Interrupt Wakeup Register contains three enable bits that control whether each external interrupt will cause the processor to wake up from Power Down mode. See Table 9.	R/W
0xE01FC148	EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level-sensitive.	R/W
0xE01FC14C	EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt.	R/W

External Interrupt Flag Register (EXTINT - 0xE01FC140)

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

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

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

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

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

Table 8: External Interrupt Flag Register (EXTINT - 0xE01FC140)

EXTINT	Function	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. Up to two pins can be selected to perform EINT0 function (see P0.1 and P0.16 description in "Pin Configuration" chapter.) This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	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. Up to two pins can be selected to perform EINT1 function (see P0.3 and P0.14 description in "Pin Configuration" chapter.) This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	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. Up to two pins can be selected to perform EINT2 function (see P0.7 and P0.15 description in "Pin Configuration" chapter.) This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	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. Up to three pins can be selected to perform EINT3 function (see P0.9, P0.20 and P0.30 description in "Pin Configuration" chapter.) This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Interrupt Wakeup Register (INTWAKE - 0xE01FC144)

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

Table 9: Interrupt Wakeup Register (INTWAKE - 0xE01FC144)

EXTWAKE	Function	Description	Reset Value
0	EXTWAKE0	When one, assertion of $\overline{\text{EINT0}}$ will wake up the processor from Power Down mode.	0
1	EXTWAKE1	When one, assertion of $\overline{\text{EINT1}}$ will wake up the processor from Power Down mode.	0
2	EXTWAKE2	When one, assertion of $\overline{\text{EINT2}}$ will wake up the processor from Power Down mode.	0
3	EXTWAKE3	When one, assertion of $\overline{\text{EINT3}}$ will wake up the processor from Power Down mode.	0
13:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
14	BODWAKE	When one, BOD interrupt will wake up the processor from Power Down mode.	
15	RTCWAKE	When one, assertion of an RTC interrupt will wake the processor from Power Down mode.	

External Interrupt Mode Register (EXTMODE - 0xE01FC148)

The bits in this register select whether each EINT pin is level- or edge-sensitive. Only pins that are selected for the EINT function (chapter Pin Connect Block on page 81) and enabled via the VICIntEnable register (chapter Vectored Interrupt Controller (VIC) on page 61) can cause interrupts from the External Interrupt function (though of course pins selected for) other functions may cause interrupts from those functions).

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

Table 10: External Interrupt Mode Register (EXTMODE - 0xE01FC148)

EXTMODE	Function	Description	Reset Value
0	EXTMODE0	When 0, level-sensitivity is selected for EINT0. When 1, EINT0 is edge-sensitive.	0
1	EXTMODE1	When 0, level-sensitivity is selected for EINT1. When 1, EINT1 is edge-sensitive.	0
2	EXTMODE2	When 0, level-sensitivity is selected for EINT2. When 1, EINT2 is edge-sensitive.	0
3	EXTMODE3	When 0, level-sensitivity is selected for EINT3. When 1, EINT3 is edge-sensitive.	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

External Interrupt Polarity Register (EXTPOLAR - 0xE01FC14C)

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 (chapter Pin Connect Block on page 81) and enabled in the VICIntEnable register (chapter Vectored Interrupt Controller (VIC) on page 61) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

Table 11: External Interrupt Polarity Register (EXTPOLAR - 0xE01FC14C)

EXTPOLAR	Function	Description	Reset Value
0	EXTPOLAR0	When 0, EINT0 is low-active or falling-edge sensitive (depending on EXTMODE0). When 1, EINT0 is high-active or rising-edge sensitive (depending on EXTMODE0).	0
1	EXTPOLAR1	When 0, EINT1 is low-active or falling-edge sensitive (depending on EXTMODE1). When 1, EINT1 is high-active or rising-edge sensitive (depending on EXTMODE1).	0
2	EXTPOLAR2	When 0, EINT2 is low-active or falling-edge sensitive (depending on EXTMODE2). When 1, EINT2 is high-active or rising-edge sensitive (depending on EXTMODE2).	0
3	EXTPOLAR3	When 0, EINT3 is low-active or falling-edge sensitive (depending on EXTMODE3). When 1, EINT3 is high-active or rising-edge sensitive (depending on EXTMODE3).	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Multiple External Interrupt Pins

Software can select multiple pins for each of EINT3:0 in the Pin Select registers, which are described in chapter Pin Connect Block on page 81. The external interrupt logic for each of EINT3:0 receives the state of all of its associated pins from the pins' receivers, along with signals that indicate whether each pin is selected for the EINT function. The external interrupt logic handles the case when more than one pin is so selected, differently according to the state of its Mode and Polarity bits:

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

The signal derived by this logic is the EINTi signal in the following logic schematic (Figure 9).

When more than one EINT pin is logically ORed, the interrupt service routine can read the states of the pins from GPIO port using IO0PIN and IO1PIN registers, to determine which pin(s) caused the interrupt.

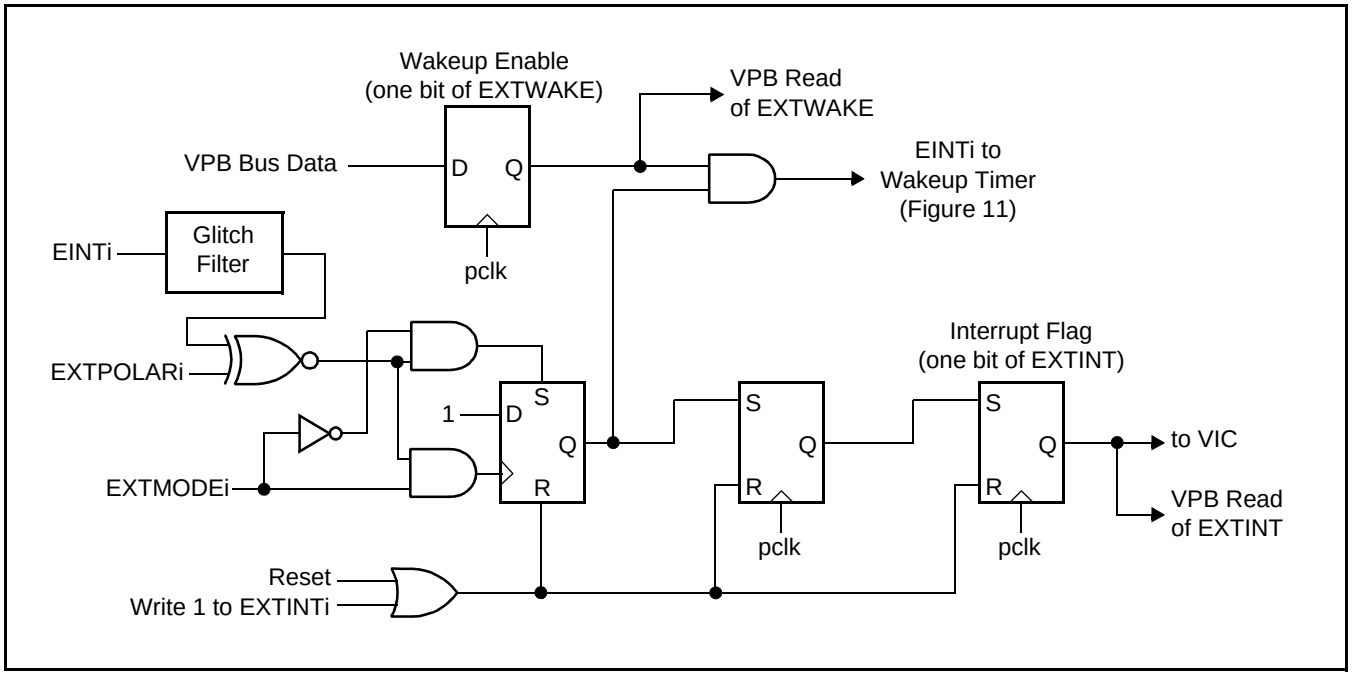


Figure 9: External Interrupt Logic

MEMORY MAPPING CONTROL

The Memory Mapping Control alters the mapping of the interrupt vectors that appear beginning at address 0x00000000. This allows code running in different memory spaces to have control of the interrupts.

Memory Mapping Control Register (MEMMAP - 0xE01FC040)

Table 12: MEMMAP Register

Address	Name	Description	Access
0xE01FC040	MEMMAP	Memory mapping control. Selects whether the ARM interrupt vectors are read from the Flash Boot Block, User Flash, or RAM.	R/W

Table 13: Memory Mapping Control Register (MEMMAP - 0xE01FC040)

MEMMAP	Function	Description	Reset Value*
1:0	MAP1:0	00: Boot Loader Mode. Interrupt vectors are re-mapped to Boot Block. 01: User Flash Mode. Interrupt vectors are not re-mapped and reside in Flash. 10: User RAM Mode. Interrupt vectors are re-mapped to Static RAM. 11: Reserved. Do not use this option. **Warning:** Improper setting of this value may result in incorrect operation of the device.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

*: The hardware reset value of the MAP bits is 00 for LPC2131/2132/2138 parts. The apparent reset value that the user will see will be altered by the Boot Loader code, which always runs initially at reset. User documentation will reflect this difference.

Memory Mapping Control Usage Notes

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

For example, whenever a Software Interrupt request is generated, ARM core will always fetch 32-bit data "residing" on 0x0000 0008 (see Table 2, “ARM Exception Vector Locations,” on page 25). This means that when MEMMAP[1:0]=10 (User RAM Mode), read/fetch from 0x0000 0008 will provide data stored in 0x4000 0008. In case of MEMMAP[1:0]=00 (Boot Loader Mode), read/fetch from 0x0000 0008 will provide data available also at 0x7FFF E008 (Boot Block remapped from on-chip ROM). MEMMAP[1:1]=11 (User External Memory Mode) will result in fetching data from off-chip memory at location 0x8000 0008.

PLL (PHASE LOCKED LOOP)

The PLL accepts an input clock frequency in the range of 10 MHz to 25 MHz only. The input frequency is multiplied up into the cclk with the range of 10 MHz to 60 MHz using a Current Controlled Oscillator (CCO). The multiplier can be an integer value from 1 to 32 (in practice, the multiplier value cannot be higher than 6 on the LPC2131/2132/2138 due to the upper frequency limit of the CPU). The CCO operates in the range of 156 MHz to 320 MHz, so there is an additional divider in the loop to keep the CCO within its frequency range while the PLL is providing the desired output frequency. The output divider may be set to divide by 2, 4, 8, or 16 to produce the output clock. Since the minimum output divider value is 2, it is insured that the PLL output has a 50% duty cycle. A block diagram of the PLL is shown in Figure 10.

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

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

Register Description

The PLL is controlled by the registers shown in Table 14. More detailed descriptions follow.

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

Table 14: PLL Registers

Address	Name	Description	Access
0xE01FC080	PLLCON	PLL Control Register. Holding register for updating PLL control bits. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W
0xE01FC084	PLLCFG	PLL Configuration Register. Holding register for updating PLL configuration values. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W
0xE01FC088	PLLSTAT	PLL Status Register. Read-back register for PLL control and configuration information. If PLLCON or PLLCFG have been written to, but a PLL feed sequence has not yet occurred, they will not reflect the current PLL state. Reading this register provides the actual values controlling the PLL, as well as the status of the PLL.	RO
0xE01FC08C	PLLFEED	PLL Feed Register. This register enables loading of the PLL control and configuration information from the PLLCON and PLLCFG registers into the shadow registers that actually affect PLL operation.	WO

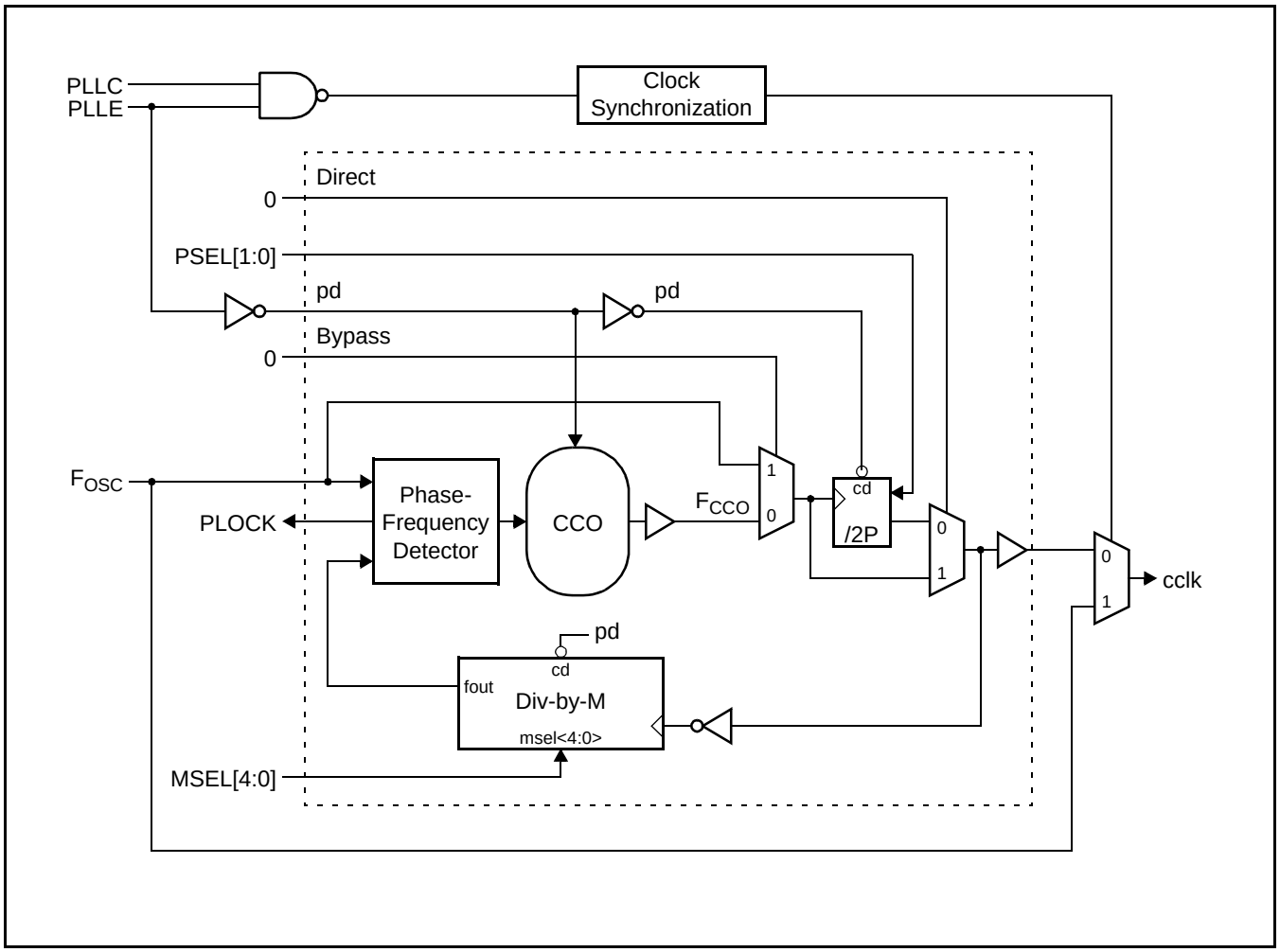


Figure 10: PLL Block Diagram

PLL Control Register (PLLCON - 0xE01FC080)

The PLLCON register contains the bits that enable and connect the PLL. Enabling the PLL allows it to attempt to lock to the current settings of the multiplier and divider values. Connecting the PLL causes the processor and all chip functions to run from the PLL output clock. Changes to the PLLCON register do not take effect until a correct PLL feed sequence has been given (see PLL Feed Register (PLLFEED - 0xE01FC08C) description).

Table 15: PLL Control Register (PLLCON - 0xE01FC080)

PLLCON	Function	Description	Reset Value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 17.	0
1	PLLC	PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the LPLPC2131/2132/2138. Otherwise, the oscillator clock is used directly by the LPLPC2131/2132/2138. See PLLSTAT register, Table 17.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

PLL Configuration Register (PLLCFG - 0xE01FC084)

The PLLCFG register contains the PLL multiplier and divider values. Changes to the PLLCFG register do not take effect until a correct PLL feed sequence has been given (see PLL Feed Register (PLLFEED - 0xE01FC08C) description). Calculations for the PLL frequency, and multiplier and divider values are found in the PLL Frequency Calculation section.

Table 16: PLL Configuration Register (PLLCFG - 0xE01FC084)

PLLCFG	Function	Description	Reset Value
4:0	MSEL4:0	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations. **Note:** For details on selecting the right value for MSEL4:0 see section "PLL Frequency Calculation" on page 43.	0
6:5	PSEL1:0	PLL Divider value. Supplies the value "P" in the PLL frequency calculations. **Note:** For details on selecting the right value for PSEL1:0 see section "PLL Frequency Calculation" on page 43.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL Status Register (PLLSTAT - 0xE01FC088)

The read-only PLLSTAT register provides the actual PLL parameters that are in effect at the time it is read, as well as the PLL status. PLLSTAT may disagree with values found in PLLCON and PLLCFG because changes to those registers do not take effect until a proper PLL feed has occurred (see PLL Feed Register (PLLFEED - 0xE01FC08C) description).

Table 17: PLL Status Register (PLLSTAT - 0xE01FC088)

PLLSTAT	Function	Description	Reset Value
4:0	MSEL4:0	Read-back for the PLL Multiplier value. This is the value currently used by the PLL.	0
6:5	PSEL1:0	Read-back for the PLL Divider value. This is the value currently used by the PLL.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	PLLE	Read-back for the PLL Enable bit. When one, the PLL is currently activated. When zero, the PLL is turned off. This bit is automatically cleared when Power Down mode is activated.	0
9	PLLC	Read-back for the PLL Connect bit. When PLLC and PLLE are both one, the PLL is connected as the clock source for the LPLPC2131/2132/2138. When either PLLC or PLLE is zero, the PLL is bypassed and the oscillator clock is used directly by the LPC2131/2132/2138. This bit is automatically cleared when Power Down mode is activated.	0
10	PLOCK	Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency.	0
15:11	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL Interrupt

The PLOCK bit in the PLLSTAT register is connected to the interrupt controller. This allows for software to turn on the PLL and continue with other functions without having to wait for the PLL to achieve lock. When the interrupt occurs (PLOCK = 1), the PLL may be connected, and the interrupt disabled.

PLL Modes

The combinations of PLLE and PLLC are shown in Table 18.

Table 18: PLL Control Bit Combinations

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

PLL Feed Register (PLLFEED - 0xE01FC08C)

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

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

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

Table 19: PLL Feed Register (PLLFEED - 0xE01FC08C)

PLLFEED	Function	Description	Reset Value
7:0	PLLFEED	The PLL feed sequence must be written to this register in order for PLL configuration and control register changes to take effect.	undefined

PLL and Power Down Mode

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

PLL Frequency Calculation

The PLL equations use the following parameters:

F_{OSC} the frequency from the crystal oscillator
F_{CCO} the frequency of the PLL current controlled oscillator
cclk the PLL output frequency (also the processor clock frequency)
M PLL Multiplier value from the MSEL bits in the PLLCFG register
P PLL Divider value from the PSEL bits in the PLLCFG register

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

$$cclk = M * F_{osc} \quad \text{or} \quad cclk = \frac{F_{cco}}{2 * P}$$

The CCO frequency can be computed as:

$$F_{cco} = cclk * 2 * P \quad \text{or} \quad F_{cco} = F_{osc} * M * 2 * P$$

The PLL inputs and settings must meet the following:

- F_{osc} is in the range of 10 MHz to 25 MHz.
- cclk is in the range of 10 MHz to F_{max} (the maximum allowed frequency for the LPLPC2131/2132/2138).
- F_{cco} is in the range of 156 MHz to 320 MHz.

Procedure for Determining PLL Settings

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

1. Choose the desired processor operating frequency (cclk). This may be based on processor throughput requirements, need to support a specific set of UART baud rates, etc. Bear in mind that peripheral devices may be running from a lower clock than the processor (see the VPB Divider description in this chapter).
2. Choose an oscillator frequency (F_{osc}). cclk must be the whole (non-fractional) multiple of F_{osc}.
3. Calculate the value of M to configure the MSEL bits. M = cclk / F_{osc}. M must be in the range of 1 to 32. The value written to the MSEL bits in PLLCFG is M - 1 (see Table 21).
4. Find a value for P to configure the PSEL bits, such that F_{cco} is within its defined frequency limits. F_{cco} is calculated using the equation given above. P must have one of the values 1, 2, 4, or 8. The value written to the PSEL bits in PLLCFG is 00 for P = 1; 01 for P = 2; 10 for P = 4; 11 for P = 8 (see Table 20).

Table 20: PLL Divider Values

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

Table 21: PLL Multiplier Values

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

PLL Example

System design asks for F_{osc}= 10 MHz and requires cclk = 60 MHz.

Based on these specifications, M = cclk / F_{osc} = 60 MHz / 10 MHz = 6. Consequently, M-1 = 5 will be written as PLLCFG 4:0.

Value for P can be derived from P = F_{cco} / (cclk * 2), using condition that F_{cco} must be in range of 156 MHz to 320 MHz. Assuming the lowest allowed frequency for F_{cco} = 156 MHz, P = 156 MHz / (2*60 MHz) = 1.3. The highest F_{cco} frequency criteria produces P = 2.67. The only solution for P that satisfies both of these requirements and is listed in Table 20 is P = 2. Therefore, PLLCFG 6:5 = 1 will be used.

POWER CONTROL

The LPLPC2131/2132/2138 supports two reduced power modes: Idle mode and Power Down mode. In Idle mode, execution of instructions is suspended until either a Reset or interrupt occurs. Peripheral functions continue operation during Idle mode and may generate interrupts to cause the processor to resume execution. Idle mode eliminates power used by the processor itself, memory systems and related controllers, and internal buses.

In Power Down mode, the oscillator is shut down and the chip receives no internal clocks. The processor state and registers, peripheral registers, and internal SRAM values are preserved throughout Power Down mode and the logic levels of chip pins remain static. The Power Down mode can be terminated and normal operation resumed by either a Reset or certain specific interrupts that are able to function without clocks. Since all dynamic operation of the chip is suspended, Power Down mode reduces chip power consumption to nearly zero.

Entry to Power Down and Idle modes must be coordinated with program execution. Wakeup from Power Down or Idle modes via an interrupt resumes program execution in such a way that no instructions are lost, incomplete, or repeated. Wake up from Power Down mode is discussed further in the description of the Wakeup Timer later in this chapter.

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.

Register Description

The Power Control function contains two registers, as shown in Table 22. More detailed descriptions follow.

Table 22: Power Control Registers

Address	Name	Description	Access
0xE01FC0C0	PCON	Power Control Register. This register contains control bits that enable the two reduced power operating modes of the LPLPC2131/2132/2138. See Table 23.	R/W
0xE01FC0C4	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

Power Control Register (PCON - 0xE01FC0C0)

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

Table 23: Power Control Register (PCON - 0xE01FC0C0)

PCON	Function	Description	Reset Value
0	IDL	Idle mode - when 1, this bit causes the processor clock to be stopped, while on-chip peripherals remain active. Any enabled interrupt from a peripheral or an external interrupt source will cause the processor to resume execution.	0
1	PD	Power Down mode - when 1, this bit causes the oscillator and all on-chip clocks to be stopped. A wakeup condition from an external interrupt can cause the oscillator to re-start, the PD bit to be cleared, and the processor to resume execution.	0
2	$\overline{\text{PDBOD}}$	When PD is 1 and this bit is 0, Brown Out Detection remains operative during power down mode, such that its Reset can release the LPC2131/2132/2138 from power down mode (see Note). When PD and this bit are both 1, the BOD circuit is disabled during power down mode to conserve power. When PD is 0, the state of this bit has no effect.	
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Note: since execution is delayed until after the Wakeup Timer has allowed the main oscillator to resume stable operation, there is no guarantee that execution will resume before Vdd has fallen below the lower BOD threshhold, which prevents execution. If execution does resume, there is no guarantee of how long the LPC2131/2132/2138 will continue execution before the lower BOD threshhold terminates execution. These issues depend on the slope of the decline of Vdd. High decoupling capacitance (between Vdd and ground) in the vicinity of the LPC2131/2132/2138 will improve the likelihood that software will be able to do what needs to be done when power is being lost.

Power Control for Peripherals Register (PCONP - 0xE01FC0C4)

The PCONP register allows turning off selected peripheral functions for the purpose of saving power. This is accomplished by gating off the clock source to the specified peripheral blocks. A few peripheral functions cannot be turned off (i.e. the Watchdog timer, GPIO, the Pin Connect block, and the System Control block). Some peripherals, particularly those that include analog functions, may consume power that is not clock dependent. These peripherals may contain a separate disable control that turns off additional circuitry to reduce power. Each bit in PCONP controls one of the peripherals. The bit numbers correspond to the related peripheral number as shown in the VPB peripheral map in the LPC2131/2132/2138 Memory Addressing section.

Table 24: Power Control for Peripherals Register for LPC2131/2132/2138 (PCONP - 0xE01FC0C4)

PCONP	Function	Description	Reset Value
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
1	PCTIM0	When 1, TIMER0 is enabled. When 0, TIMER0 is disabled to conserve power.	1
2	PCTIM1	When 1, TIMER1 is enabled. When 0, TIMER1 is disabled to conserve power.	1
3	PCURT0	When 1, UART0 is enabled. When 0, UART0 is disabled to conserve power.	1
4	PCURT1	When 1, UART1 is enabled. When 0, UART1 is disabled to conserve power.	1
5	PCPWM0	When 1, PWM0 is enabled. When 0, PWM0 is disabled to conserve power.	1
6	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

Table 24: Power Control for Peripherals Register for LPC2131/2132/2138 (PCONP - 0xE01FC0C4)

PCONP	Function	Description	Reset Value
7	PCI2C0	When 1, the I^2C0 interface is enabled. When 0, the I^2C0 interface is disabled to conserve power.	1
8	PCSPI0	When 1, the SPI0 interface is enabled. When 0, the SPI0 is disabled to conserve power.	1
9	PCRTC	When 1, the RTC is enabled. When 0, the RTC is disabled to conserve power.	1
10	PCSPI1	When 1, the SSP interface is enabled. When 0, the SSP is disabled to conserve power.	1
11	Reserved	User software should write 0 here to reduce power consumption.	1
12	PCAD0	When 1, A/D converter 0 is enabled. When 0, A/D 0 is disabled to conserve power. Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	1
18:13	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
19	PCI2C1	When 1, the I^2C1 interface is enabled. When 0, the I^2C1 interface is disabled to conserve power.	1
20	PCAD1	When 1, A/D converter 1 is enabled. When 0, A/D 1 is disabled to conserve power. Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	1
31:21	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

Power Control Usage Notes

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

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

RESET

Reset has two sources on the LPC2131/2132/2138: the $\overline{\text{RESET}}$ pin and Watchdog Reset. The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin with an additional glitch filter. Assertion of chip Reset by any source starts the Wakeup Timer (see Wakeup Timer description later in this chapter), causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, a fixed number of clocks have passed, and the on-chip circuitry has completed its initialization. The relationship between Reset, the oscillator, and the Wakeup Timer are shown in Figure 11.

The Reset glitch filter allows the processor to ignore external reset pulses that are very short, and also determines the minimum duration of $\overline{\text{RESET}}$ that must be asserted in order to guarantee a chip reset. Once asserted, $\overline{\text{RESET}}$ pin can be deasserted only when crystal oscillator is fully running and an adequate signal is present on the X1 pin of the LPC2131/2132/2138. Assuming that an external crystal is used in the crystal oscillator subsystem, after power on, the $\overline{\text{RESET}}$ pin should be asserted for 10 ms. For all subsequent resets when crystal oscillator is already running and stable signal is on the X1 pin, the $\overline{\text{RESET}}$ pin needs to be asserted for 300 ns only.

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

External and internal Resets have some small differences. An external Reset causes the value of certain pins to be latched to configure the part. External circuitry cannot determine when an internal Reset occurs in order to allow setting up those special pins, so those latches are not reloaded during an internal Reset. Pins that are examined during an external Reset for various purposes are: P1.20/TRACESYNC, P1.26/RTCK (see chapters Pin Configuration on page 75 and Pin Connect Block on page 81). Pin P0.14 (see Flash Memory System and Programming on page 225) is examined by on-chip bootloader when this code is executed after reset.

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

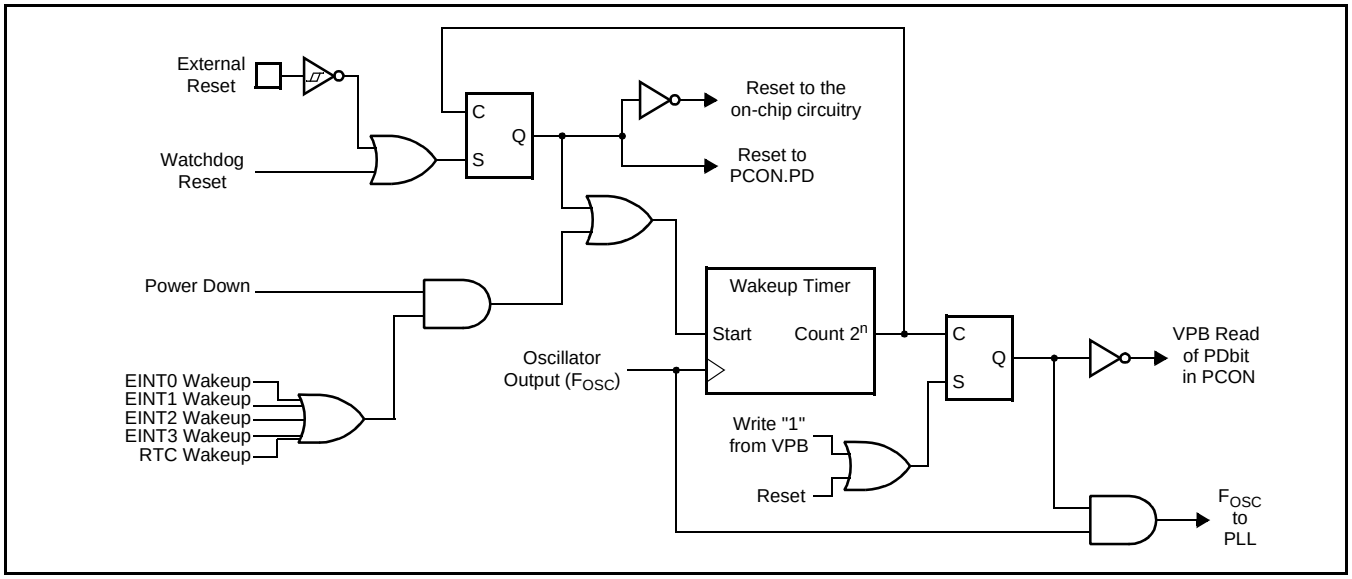


Figure 11: Reset Block Diagram including Wakeup Timer

Reset Source Identification Register (RSIR - 0xE01FC180)

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 25: Power Control for Peripherals Register for LPC2131/2132/2138 (PCONP - 0xE01FC0C4)

RSIR	Function	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. Ths bit is cleared by POR, but is not affected by WDT or BOD reset.	
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.	
3	BODR	This bit is set when the 3.3V power falls below 2.6V. If the voltage continues to decline o the level at which POR is asserted (nominally 1V), this bit is cleared, but if the voltage comes back up without reaching that level, this bit remains 1. This bit is not affected by External Reset nor Watchdog Reset.	
7:4	Reserved	Reserved, user software should not write ones to reserved bits.	0

VPB DIVIDER

The VPB Divider determines the relationship between the processor clock (cclk) and the clock used by peripheral devices (pclk). The VPB Divider serves two purposes. The first is to provides peripherals with desired pclk via VPB bus so that they can operate at the speed chosen for the ARM processor. In order to achieve this, the VPB bus may be slowed down to one half or one fourth of the processor clock rate. Because the VPB bus must work properly at power up (and its timing cannot be altered if it does not work since the VPB divider control registers reside on the VPB bus), the default condition at reset is for the VPB bus to run at one quarter speed. The second purpose of the VPB Divider is to allow power savings when an application does not require any peripherals to run at the full processor rate.

The connection of the VPB Divider relative to the oscillator and the processor clock is shown in Figure 12. Because the VPB Divider is connected to the PLL output, the PLL remains active (if it was running) during Idle mode.

VPBDIV Register (VPBDIV - 0xE01FC100)

The VPB Divider register contains two bits, allowing three divider values, as shown in Table 27.

Table 26: VPBDIV Register Map

Address	Name	Description	Access
0xE01FC100	VPBDIV	Controls the rate of the VPB clock in relation to the processor clock.	R/W

Table 27: VPB Divider Register (VPBDIV - 0xE01FC100)

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

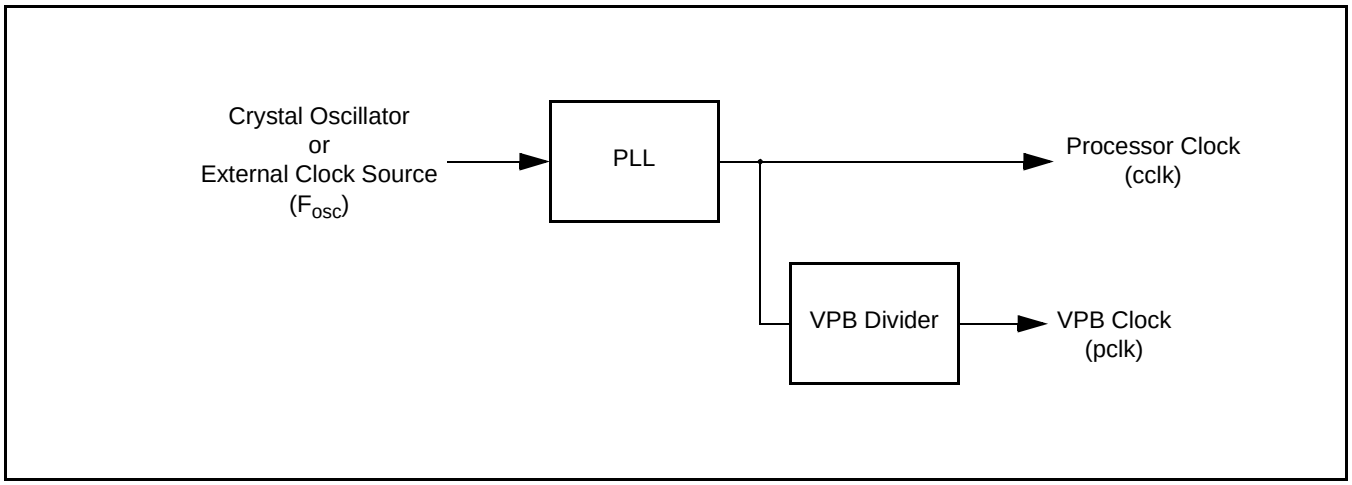


Figure 12: VPB Divider Connections

WAKEUP TIMER

The purpose of the wakeup timer is to ensure that the oscillator and other analog functions required for chip operation are fully functional before the processor is allowed to execute instructions. This is important at power on, all types of Reset, and whenever any of the aforementioned functions are turned off for any reason. Since the oscillator and other functions are turned off during Power Down mode, any wakeup of the processor from Power Down mode makes use of the Wakeup Timer.

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

Once a clock is detected, the Wakeup Timer counts 4096 clocks, then enables the on-chip circuitry to initialize. When the on-board modules initialization is complete, the processor is released to execute instructions if the external Reset has been de-asserted. In the case where an external clock source is used in the system (as opposed to a crystal connected to the oscillator pins), the possibility that there could be little or no delay for oscillator start-up must be considered. The Wakeup Timer design then ensures that any other required chip functions will be operational prior to the beginning of program execution.

Any of the various Resets can bring the LPC2131/2132/2138 out of power-down mode, as can the external interrupts EINT3:0, plus the RTC interrupt if the RTC is operating from its own oscillator on the RTCX1-2 pins. When one of these interrupts is enabled for wakeup and its selected event occurs, an oscillator wakeup cycle is started. The actual interrupt (if any) occurs after the wakeup timer expires, and is handled by the Vectored Interrupt Controller.

However, the pin multiplexing on the LPC2131/2132/2138 (see Pin Configuration on page 75 and Pin Connect Block on page 81) was designed to allow other peripherals to, in effect, bring the device out of power down mode. The following pin-function pairings allow interrupts from events relating to UART0 or 1, SPI 0 or 1, or the I^2C: RxD0 / EINT0, SDA / EINT1, SSEL0 / EINT2, RxD1 / EINT3, DCD1 / EINT1, RI1 / EINT2, SSEL1 / EINT3.

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

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

The only flaw in this scheme is that the time to restart the oscillator prevents the LPC2131/2132/2138 from capturing the bus or line activity that wakes it up. Idle mode is more appropriate than power-down mode for devices that must capture and respond to external activity in a timely manner.

To summarize: on the LPC2131/2132/2138, the Wakeup Timer enforces a minimum reset duration based on the crystal oscillator, and is activated whenever there is a wakeup from Power Down mode or any type of Reset.

BROWN-OUT DETECTION

The LPC2131/2132/2138 includes 2-stage monitoring of the voltage on the Vdd pins. If this voltage falls below 2.9V, the Brown-Out Detector (BOD) asserts an interrupt signal to the Vectored Interrupt Controller. This signal can be enabled for interrupt in the Interrupt Enable Register (VICIntEnable - 0xFFFFF010, Read/Write) on page 65; if not, software can monitor the signal by reading the Raw Interrupt Status Register (VICRawIntr - 0xFFFFF008, Read Only) on page 64.

The second stage of low-voltage detection asserts Reset to inactivate the LPC2131/2132/2138 when the voltage on the V3 pins falls below 2.6V. 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 1V, at which point the Power-On Reset circuitry maintains the overall Reset.

Both the 2.9V and 2.6V threshholds include some hysteresis. In normal operation, this hysteresis allows the 2.9V detection to reliably interrupt, or a regularly-executed event loop to sense the condition.

But when Brown-Out Detection is enabled to bring the LPC2131/2132/2138 out of Power-Down mode (which is itself not a guaranteed operation -- see page 50), the supply voltage may recover from a transient before the Wakeup Timer has completed its delay. In this case, the net result of the transient BOD is that the part wakes up and continues operation after the instructions that set Power-Down Mode, without any interrupt occurring and with the BOD bit in the RISR being 0. Since all other wakeup conditions have latching flags (see the EXTINT register on page 33 and the RTC ILR on page 209), a wakeup of this type, without any apparent cause, can be assumed to be a Brown-Out that has gone away.

CODE SECURITY VS. DEBUGGING

Applications in development typically need the debugging and tracing facilities in the LPC2131/2132/2138. Later in the life cycle of an application, it may be more important to protect the application code from observation by hostile or competitive eyes. The following feature of the LPC2131/2132/2138 allows an application to control whether it can be debugged or protected from observation.

Details on the way Code Read Protection works can be found in Flash Memory System and Programming chapter.

4. MEMORY ACCELERATOR MODULE (MAM)

INTRODUCTION

The MAM block in the LPC2131/2132/2138 maximizes the performance of the ARM processor when it is running code in Flash memory, but does so using a single Flash bank.

OPERATION

Simply put, the Memory Accelerator Module (MAM) attempts to have the next ARM instruction that will be needed in its latches in time to prevent CPU fetch stalls. The LPC2131/2132/2138 uses one bank of Flash memory, compared to the two banks used on predecessor devices. It includes three 128-bit buffers called the Prefetch buffer, the Branch Trail Buffer and the data buffer. When an Instruction Fetch is not satisfied by either the Prefetch or Branch Trail buffer, nor has a prefetch been initiated for that line, the ARM is stalled while a fetch is initiated for the 128-bit line. If a prefetch has been initiated but not yet completed, the ARM is stalled for a shorter time. Unless aborted by a data access, a prefetch is initiated as soon as the Flash has completed the previous access. The prefetched line is latched by the Flash module, but the MAM does not capture the line in its prefetch buffer until the ARM core presents the address from which the prefetch has been made. If the core presents a different address from the one from which the prefetch has been made, the prefetched line is discarded.

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

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

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

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

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

Memory Accelerator Module Blocks

The Memory Accelerator Module is divided into several functional blocks:

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

- Control logic
- Wait logic

Figure 13 shows a simplified block diagram of the Memory Accelerator Module data paths.

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

Flash Memory Banks

There is one bank of Flash memory with the LPC2131/2132/2138 MAM.

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

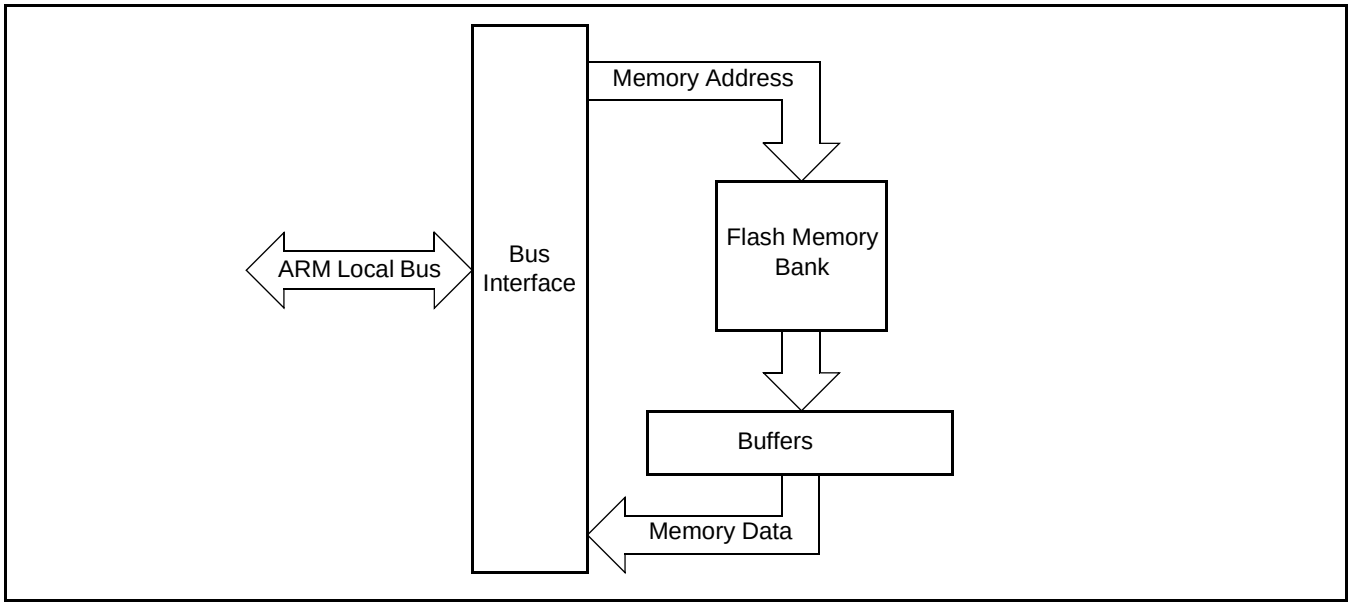


Figure 13: Simplified Block Diagram of the Memory Accelerator Module

Instruction Latches and Data Latches

Code and Data accesses are treated separately by the Memory Accelerator Module. There is a 128-bit Latch, a 15-bit Address Latch, and a 15-bit comparator associated with each buffer (prefetch, branch trail, and data). Each 128-bit latch holds 4 words (4 ARM instructions, or 8 Thumb instructions). Also associated with each buffer are 32 4:1 Multiplexers that select the requested word from the 128-bit line.

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

Flash Programming Issues

Since the Flash memory does not allow accesses during programming and erase operations, it is necessary for the MAM to force the CPU to wait if a memory access to a Flash address is requested while the Flash module is busy. (This is accomplished by

asserting the ARM7TDMI-S local bus signal CLKEN.) Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the Flash memory.

In order to preclude the possibility of stale data being read from the Flash memory, the LPC2131/2132/2138 MAM holding latches are automatically invalidated at the beginning of any Flash programming or erase operation. Any subsequent read from a Flash address will cause a new fetch to be initiated after the Flash operation has completed.

MEMORY ACCELERATOR MODULE OPERATING MODES

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

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

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

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

Table 28: MAM Responses to Program Accesses of Various Types

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

Table 29: MAM Responses to Data and DMA Accesses of Various Types

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

1. Instruction prefetch is enabled in modes 1 and 2.
2. The MAM actually uses latched data if it is available, but mimics the timing of a Flash read operation. This saves power while resulting in the same execution timing. The MAM can truly be turned off by setting the fetch timing value in MAMTIM to one clock.

MAM CONFIGURATION

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

REGISTER DESCRIPTION

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

Table 30: Summary of System Control Registers

Name	Description	Access	Reset Value*	Address
MAM				
MAMCR	Memory Accelerator Module Control Register. Determines the MAM functional mode, that is, to what extent the MAM performance enhancements are enabled. See Table 31.	R/W	0	0xE01FC000
MAMTIM	Memory Accelerator Module Timing control. Determines the number of clocks used for Flash memory fetches (1 to 7 processor clocks).	R/W	0x07	0xE01FC004

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

MAM Control Register (MAMCR - 0xE01FC000)

Two configuration bits select the three MAM operating modes, as shown in Table 31. Following Reset, MAM functions are disabled. Changing the MAM operating mode causes the MAM to invalidate all of the holding latches, resulting in new reads of Flash information as required.

Table 31: MAM Control Register (MAMCR - 0xE01FC000)

MAMCR	Function	Description	Reset Value
1:0	MAM mode control	These bits determine the operating mode of the MAM as follows: 0 0 - MAM functions disabled. 0 1 - MAM functions partially enabled. 1 0 - MAM functions fully enabled. 1 1 - reserved	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

MAM Timing Register (MAMTIM - 0xE01FC004)

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

Table 32: MAM Timing Register (MAMTIM - 0xE01FC004)

MAMTIM	Function	Description	Reset Value
2:0	MAM Fetch Cycle timing	These bits set the duration of MAM Flash fetch operations as follows: 0 0 0 = 0 - Reserved. 0 0 1 = 1 - MAM fetch cycles are 1 processor clock (cclk) in duration. 0 1 0 = 2 - MAM fetch cycles are 2 processor clocks (cclks) in duration. 0 1 1 = 3 - MAM fetch cycles are 3 processor clocks (cclks) in duration. 1 0 0 = 4 - MAM fetch cycles are 4 processor clocks (cclks) in duration. 1 0 1 = 5 - MAM fetch cycles are 5 processor clocks (cclks) in duration. 1 1 0 = 6 - MAM fetch cycles are 6 processor clocks (cclks) in duration. 1 1 1 = 7 - MAM fetch cycles are 7 processor clocks (cclks) in duration. **Warning:** Improper setting of this value may result in incorrect operation of the device.	0x07
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

MAM USAGE NOTES

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

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

5. VECTORED INTERRUPT CONTROLLER (VIC)

FEATURES

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

DESCRIPTION

The Vectored Interrupt Controller (VIC) takes 32 interrupt request inputs and programmably assigns them into 3 categories, FIQ, vectored IRQ, and non-vectored IRQ. The programmable assignment scheme means that priorities of interrupts from the various peripherals can be dynamically assigned and adjusted.

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

Vectored IRQs have the middle priority, but only 16 of the 32 requests can be assigned to this category. Any of the 32 requests can be assigned to any of the 16 vectored IRQ slots, among which slot 0 has the highest priority and slot 15 has the lowest.

Non-vectored IRQs have the lowest priority.

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

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

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

REGISTER DESCRIPTION

The VIC implements the registers shown in Table 33. More detailed descriptions follow.

Table 33: VIC Register Map

Name	Description	Access	Reset Value*	Address
VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.	RO	0	0xFFFF F000
VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ.	RO	0	0xFFFF F004
VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	0	0xFFFF F008
VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0	0xFFFF F00C
VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ.	R/W	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	W	0	0xFFFF F014
VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0	0xFFFF F018
VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register.	W	0	0xFFFF F01C
VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode.	R/W	0	0xFFFF F020
VICVectAddr	Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.	R/W	0	0xFFFF F030
VICDefVectAddr	Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.	R/W	0	0xFFFF F034
VICVectAddr0	Vector address 0 register. Vector Address Registers 0-15 hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.	R/W	0	0xFFFF F100
VICVectAddr1	Vector address 1 register	R/W	0	0xFFFF F104
VICVectAddr2	Vector address 2 register	R/W	0	0xFFFF F108
VICVectAddr3	Vector address 3 register	R/W	0	0xFFFF F10C
VICVectAddr4	Vector address 4 register	R/W	0	0xFFFF F110
VICVectAddr5	Vector address 5 register	R/W	0	0xFFFF F114
VICVectAddr6	Vector address 6 register	R/W	0	0xFFFF F118
VICVectAddr7	Vector address 7 register	R/W	0	0xFFFF F11C
VICVectAddr8	Vector address 8 register	R/W	0	0xFFFF F120
VICVectAddr9	Vector address 9 register	R/W	0	0xFFFF F124

Table 33: VIC Register Map

Name	Description	Access	Reset Value*	Address
VICVectAddr10	Vector address 10 register	R/W	0	0xFFFF F128
VICVectAddr11	Vector address 11 register	R/W	0	0xFFFF F12C
VICVectAddr12	Vector address 12 register	R/W	0	0xFFFF F130
VICVectAddr13	Vector address 13 register	R/W	0	0xFFFF F134
VICVectAddr14	Vector address 14 register	R/W	0	0xFFFF F138
VICVectAddr15	Vector address 15 register	R/W	0	0xFFFF F13C
VICVectCntl0	Vector control 0 register. Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest.	R/W	0	0xFFFF F200
VICVectCntl1	Vector control 1 register	R/W	0	0xFFFF F204
VICVectCntl2	Vector control 2 register	R/W	0	0xFFFF F208
VICVectCntl3	Vector control 3 register	R/W	0	0xFFFF F20C
VICVectCntl4	Vector control 4 register	R/W	0	0xFFFF F210
VICVectCntl5	Vector control 5 register	R/W	0	0xFFFF F214
VICVectCntl6	Vector control 6 register	R/W	0	0xFFFF F218
VICVectCntl7	Vector control 7 register	R/W	0	0xFFFF F21C
VICVectCntl8	Vector control 8 register	R/W	0	0xFFFF F220
VICVectCntl9	Vector control 9 register	R/W	0	0xFFFF F224
VICVectCntl10	Vector control 10 register	R/W	0	0xFFFF F228
VICVectCntl11	Vector control 11 register	R/W	0	0xFFFF F22C
VICVectCntl12	Vector control 12 register	R/W	0	0xFFFF F230
VICVectCntl13	Vector control 13 register	R/W	0	0xFFFF F234
VICVectCntl14	Vector control 14 register	R/W	0	0xFFFF F238
VICVectCntl15	Vector control 15 register	R/W	0	0xFFFF F23C

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

VIC REGISTERS

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

Software Interrupt Register (VICSoftInt - 0xFFFFF018, Read/Write)

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

Table 34: Software Interrupt Register (VICSoftInt - 0xFFFFF018, Read/Write)

VICSoftInt	Function	Reset Value
31:0	1:force the interrupt request with this bit number. 0: do not force the interrupt request with this bit number. Writing zeroes to bits in VICSoftInt has no effect, see VICSoftIntClear.	0

Software Interrupt Clear Register (VICSoftIntClear - 0xFFFFF01C, Write Only)

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

Table 35: Software Interrupt Clear Register (VICSoftIntClear - 0xFFFFF01C, Write Only)

VICSoftIntClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Software Interrupt register, thus releasing the forcing of this request. 0: writing a 0 leaves the corresponding bit in VICSoftInt unchanged.	0

Raw Interrupt Status Register (VICRawIntr - 0xFFFFF008, Read Only)

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

Table 36: Raw Interrupt Status Register (VICRawIntr - 0xFFFFF008, Read-Only)

VICRawIntr	Function	Reset Value
31:0	1: the interrupt request or software interrupt with this bit number is asserted. 0: the interrupt request or software interrupt with this bit number is negated.	0

Interrupt Enable Register (VICIntEnable - 0xFFFFF010, Read/Write)

This register controls which of the 32 interrupt requests and software interrupts contribute to FIQ or IRQ.

Table 37: Interrupt Enable Register (VICINtEnable - 0xFFFFF010, Read/Write)

VICIntEnable	Function	Reset Value
31:0	When this register is read, 1s indicate interrupt requests or software interrupts that are enabled to contribute to FIQ or IRQ. When this register is written, ones enable interrupt requests or software interrupts to contribute to FIQ or IRQ, zeroes have no effect. See the VICIntEnClear register (Table 38 below), for how to disable interrupts.	0

Interrupt Enable Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)

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

Table 38: Software Interrupt Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)

VICIntEnClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Interrupt Enable register, thus disabling interrupts for this request. 0: writing a 0 leaves the corresponding bit in VICIntEnable unchanged.	0

Interrupt Select Register (VICIntSelect - 0xFFFFF00C, Read/Write)

This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.

Table 39: Interrupt Select Register (VICIntSelect - 0xFFFFF00C, Read/Write)

VICIntSelect	Function	Reset Value
31:0	1: the interrupt request with this bit number is assigned to the FIQ category. 0: the interrupt request with this bit number is assigned to the IRQ category.	0

IRQ Status Register (VICIRQStatus - 0xFFFFF000, Read Only)

This register reads out the state of those interrupt requests that are enabled and classified as IRQ. It does not differentiate between vectored and non-vectored IRQs.

Table 40: IRQ Status Register (VICIRQStatus - 0xFFFFF000, Read-Only)

VICIRQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as IRQ, and asserted.	0

FIQ Status Register (VICFIQStatus - 0xFFFFF004, Read Only)

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

Table 41: IRQ Status Register (VICFIQStatus - 0xFFFFF004, Read-Only)

VICFIQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as FIQ, and asserted.	0

Vector Control Registers 0-15 (VICVectCntl0-15 - 0xFFFFF200-23C, Read/Write)

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

Table 42: Vector Control Registers (VICVectCntl0-15 - 0xFFFFF200-23C, Read/Write)

VICVectCntl0-15	Function	Reset Value
5	1: this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
4:0	The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower-numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0

Vector Address Registers 0-15 (VICVectAddr0-15 - 0xFFFFF100-13C, Read/Write)

These registers hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.

Table 43: Vector Address Registers (VICVectAddr0-15 - 0xFFFFF100-13C, Read/Write)

VICVectAddr0-15	Function	Reset Value
31:0	When one or more interrupt request or software interrupt is (are) enabled, classified as IRQ, asserted, and assigned to an enabled vectored IRQ slot, the value from this register for the highest-priority such slot will be provided when the IRQ service routine reads the Vector Address register (VICVectAddr).	0

Default Vector Address Register (VICDefVectAddr - 0xFFFFF034, Read/Write)

This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.

Table 44: Default Vector Address Register (VICDefVectAddr - 0xFFFFF034, Read/Write)

VICDefVectAddr	Function	Reset Value
31:0	When an IRQ service routine reads the Vector Address register (VICVectAddr), and no IRQ slot responds as described above, this address is returned.	0

Vector Address Register (VICVectAddr - 0xFFFFF030, Read/Write)

When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.

Table 45: Vector Address Register (VICVectAddr - 0xFFFFF030, Read/Write)

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

Protection Enable Register (VICProtection - 0xFFFFF020, Read/Write)

This one-bit register controls access to the VIC registers by software running in User mode.

Table 46: Protection Enable Register (VICProtection - 0xFFFFF020, Read/Write)

VICProtection	Function	Reset Value
0	1: the VIC registers can only be accessed in privileged mode. 0: VIC registers can be accessed in User or privileged mode.	0

INTERRUPT SOURCES

Table 47 lists the interrupt sources for each peripheral function. Each peripheral device has one interrupt line connected to the Vectored Interrupt Controller, but may have several internal interrupt flags. Individual interrupt flags may also represent more than one interrupt source.

Table 47: Connection of Interrupt Sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
TIMER0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4
TIMER1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)[1]	7
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8
I^2C0	SI (state change)	9
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10
SPI1 (SSP)	Tx FIFO at least half empty (TXRIS) Rx FIFO at least half full (RXRIS) Receive Timeout condition (RTRIS) Receive overrun (RORRIS)	11
PLL	PLL Lock (PLOCK)	12
RTC	Counter Increment (RTCCIF) Alarm (RTCALF)	13
System Control	External Interrupt 0 (EINT0)	14
	External Interrupt 1 (EINT1)	15
	External Interrupt 2 (EINT2)	16
	External Interrupt 3 (EINT3)	17
A/D0	A/D Converter 0	18
I^2C1	SI (state change)	19
BOD	Brown Out Detect	20

Table 47: Connection of Interrupt Sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Channel #
A/D1	A/D Converter 1[1]	21

[1]LPC2138 only

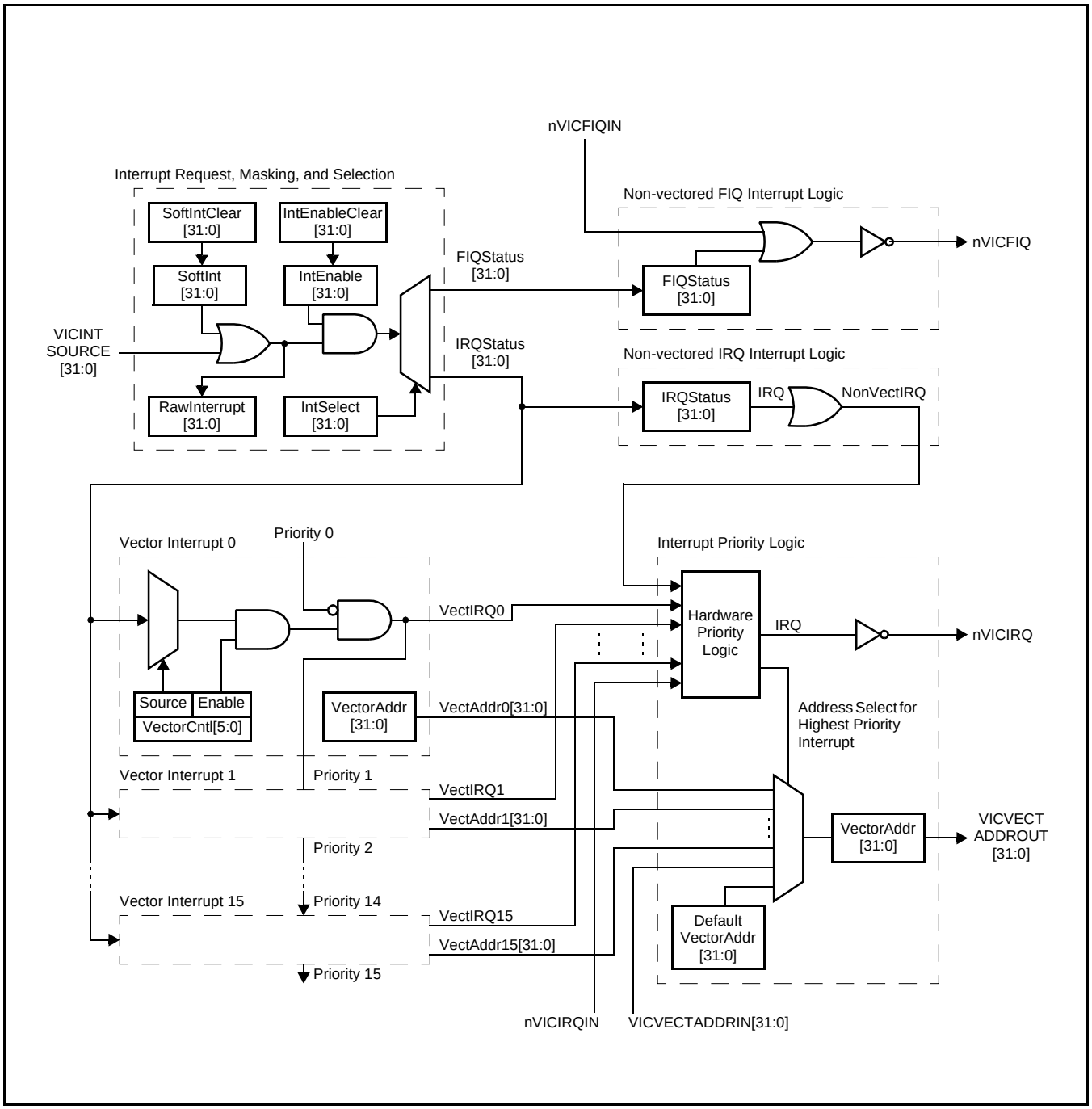


Figure 14: Block Diagram of the Vectored Interrupt Controller

SPURIOUS INTERRUPTS

Spurious interrupts are possible in the ARM7TDMI based microcontrollers such as the LPC2131/2132/2138 due to asynchronous interrupt handling. The asynchronous character of the interrupt processing has its roots in the interaction of the core and the VIC. If the VIC state is changed between the moments when the core detects an interrupt, and the core actually processes an interrupt, problems may be generated.

Real-life applications may experience the following scenarios:

1) VIC decides there is an IRQ interrupt and sends the IRQ signal to the core.

2) Core latches the IRQ state.

3) Processing continues for a few cycles due to pipelining.

4) Core loads IRQ address from VIC.

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

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

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

2. VIC default handler should be set up and tested properly.

Details and Case Studies on Spurious Interrupts

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

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

Applies to: ARM7TDMI

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

For example, consider the follow instruction sequence:

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

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

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

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

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

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

the SPSR_IRQ is restored to the CPSR. The CPSR will now have the I bit and F bit set, and therefore execution will continue with all interrupts disabled.

However, this can cause problems in the following cases:

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

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

Workaround:

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

Solution 1: Add code similar to the following at the start of the interrupt routine.

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

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

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

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

Solution 2: Disable IRQs and FIQs using separate writes to the CPSR, e.g.:

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

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

Solution 3: Re-enable FIQs at the beginning of the IRQ handler. As the required state of all bits in the c field of the CPSR are known, this can be most efficiently be achieved by writing an immediate value to CPSR_c, for example:

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

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

VIC USAGE NOTES

If user code is running from an on-chip RAM and an application uses interrupts, interrupt vectors must be re-mapped to on-chip address 0x0. This is necessary because all the exception vectors are located at addresses 0x0 and above. This is easily achieved by configuring the MEMMAP register (see System Control Block chapter) to User RAM mode. Application code should be linked such that at 0x4000 0000 the Interrupt Vector Table (IVT) will reside.

Although multiple sources can be selected (VICIntSelect) to generate FIQ request, only one interrupt service routine should be dedicated to service all available/present FIQ request(s). Therefore, if more than one interrupt sources are classified as FIQ the FIQ interrupt service routine must read VICFIQStatus to decide based on this content what to do and how to process the interrupt request. However, it is recommended that only one interrupt source should be classified as FIQ. Classifying more than one interrupt sources as FIQ will increase the interrupt latency.

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

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

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

Example:

Assuming that UART0 and SPI0 are generating interrupt requests that are classified as vectored IRQs (UART0 being on the higher level than SPI0), while UART1 and I^2C are generating non-vectored IRQs, the following could be one possibility for VIC setup:

```
VICIntSelect = 0x0000 0000(SPI0, I2C, UART1 and UART0 are IRQ => bit10, bit9, bit7 and bit6=0)
VICIntEnable = 0x0000 06C0(SPI0, I2C, UART1 and UART0 are enabled interrupts => bit10, bit9, bit 7 and bit6=1)
VICDefVectAddr = 0x…       (holds address at what routine for servicing non-vectored IRQs (i.e. UART1 and I2C) starts)
VICVectAddr0 = 0x…         (holds address where UART0 IRQ service routine starts)
VICVectAddr1 = 0x…         (holds address where SPI0 IRQ service routine starts)
VICVectCntl0 = 0x0000 0026(interrupt source with index 6 (UART0) is enabled as the one with priority 0 (the highest))
VICVectCntl1 = 0x0000 002A(interrupt source with index 10 (SPI0) is enabled as the one with priority 1)
```

After any of IRQ requests (SPI0, I2C, UART0 or UART1) is made, microcontroller will redirect code execution to the address specified at location 0x00000018. For vectored and non-vectored IRQ's the following instruction could be placed at 0x18:

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

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

In case UART0 request has been made, VICVectAddr will be identical to VICVectAddr0, while in case SPI0 request has been made value from VICVectAddr1 will be found here. If neither UART0 nor SPI0 have generated IRQ request but UART1 and/or I^2C were the reason, content of VICVectAddr will be identical to VICDefVectAddr.

6. PIN CONFIGURATION

LPC2131/2132/2138 PINOUT

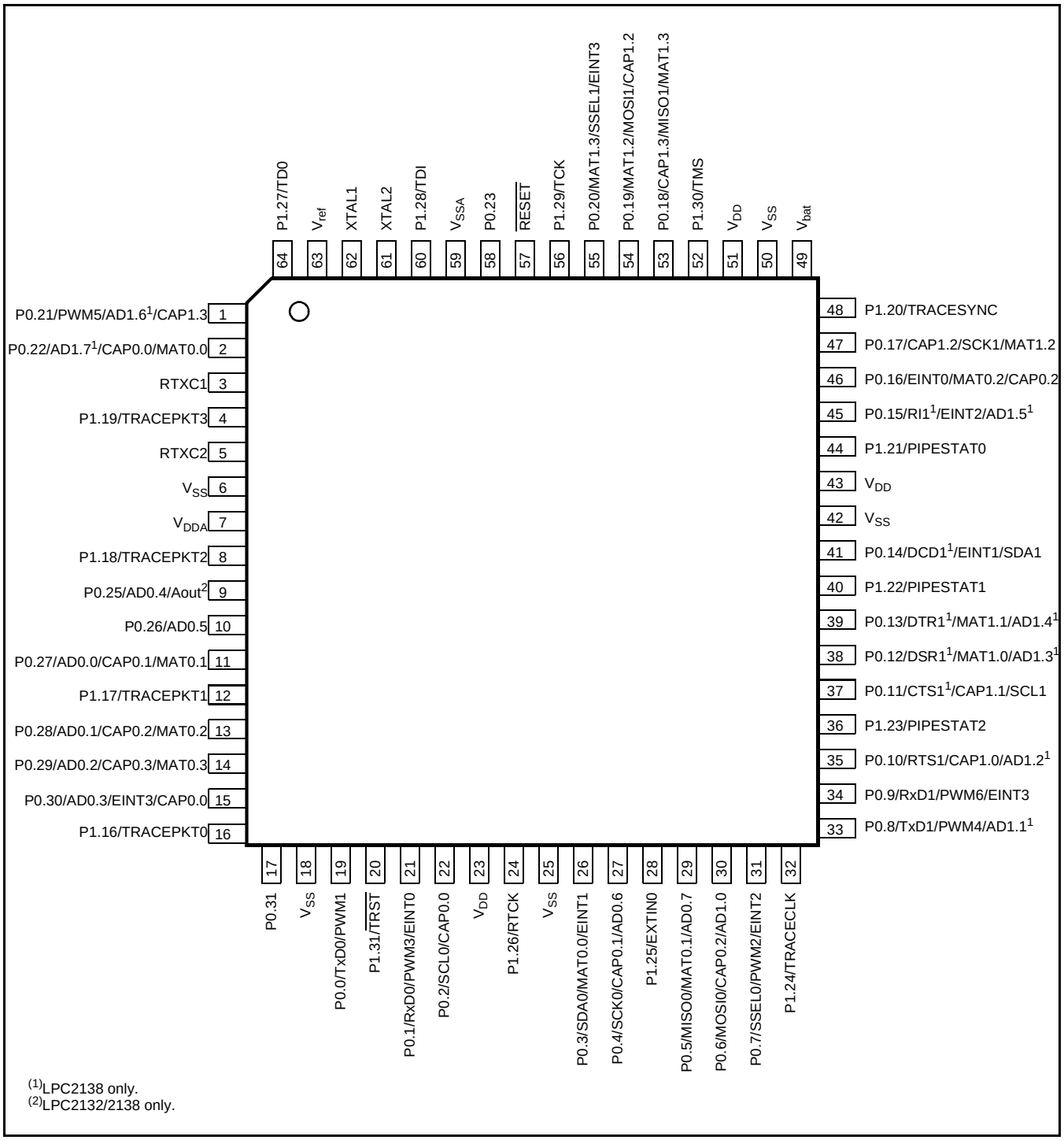


Figure 15: LPC2131/2132/2138 64-pin package

PIN DESCRIPTION FOR LPC2131/2132/2138

Pin description for LPC2131/2132/2138 and a brief explanation of corresponding functions are shown in the following table.

Table 48: Pin description for LPC2131/2132/2138

Pin Name	LQFP64 Pin #	Type	Description		
P0.0 to P0.31		I/O	**Port 0:** Port 0 is a 32-bit I/O port with individual direction controls for each bit. Total of 31 pins of the Port 0 can be used as a general purpose bi-directional digital I/Os while P0.31 is output only pin. The operation of port 0 pins depends upon the pin function selected via the Pin Connect Block. Pin P0.24 is not available. **Note:** All Port 0 pins excluding those that can be used as A/D inputs are functionally 5V tolerant. If the A/D converter is not used at all, pins associated with A/D inputs can be used as 5V tolerant digital IO pins. See "A/D Converter" chapter of the User Manual for A/D input pin voltage considerations.		
	19	O O	**P0.0**	**TxD0** **PWM1**	Transmitter output for UART 0. Pulse Width Modulator output 1.
	21	I O I	**P0.1**	**RxD0** **PWM3** **EINT0**	Receiver input for UART 0. Pulse Width Modulator output 3. External interrupt 0 input.
	22	I/O I	**P0.2**	**SCL0** **CAP0.0**	I^2C0 clock input/output. Open drain output (for I^2C compliance). Capture input for Timer 0, channel 0.
	26	I/O O I	**P0.3**	**SDA0** **MAT0.0** **EINT1**	I^2C0 data input/output. Open drain output (for I^2C compliance). Match output for Timer 0, channel 0. External interrupt 1 input.
	27	I/O I I	**P0.4**	**SCK0** **CAP0.1** **AD0.6**	Serial Clock for SPI0. SPI clock output from master or input to slave. Capture input for Timer 0, channel 1. A/D converter 0, input 6. This analog input is always connected to its pin.
	29	I/O O I	**P0.5**	**MISO0** **MAT0.1** **AD0.7**	Master In Slave Out for SPI0. Data input to SPI master or data output from SPI slave. Match output for Timer 0, channel 1. A/D converter 0, input 7. This analog input is always connected to its pin.
	30	I/O I I	**P0.6**	**MOSI0** **CAP0.2** **AD1.0**	Master Out Slave In for SPI0. Data output from SPI master or data input to SPI slave. Capture input for Timer 0, channel 2. A/D converter 1, input 0. This analog input is always connected to its pin. (LPC2138 only)
	31	I O I	**P0.7**	**SSEL0** **PWM2** **EINT2**	Slave Select for SPI0. Selects the SPI interface as a slave. Pulse Width Modulator output 2. External interrupt 2 input.

Table 48: Pin description for LPC2131/2132/2138

Pin Name	LQFP64 Pin #	Type	Description		
	33	O O I	**P0.8**	**TxD1** **PWM4** **AD1.1**	Transmitter output for UART 1. Pulse Width Modulator output 4. A/D converter 1, input 1. This analog input is always connected to its pin. (LPC2138 only)
	34	I O I	**P0.9**	**RxD1** **PWM6** **EINT3**	Receiver input for UART 1. Pulse Width Modulator output 6. External interrupt 3 input.
	35	O I I	**P0.10**	**RTS1** **CAP1.0** **AD1.2**	Request to Send output for UART 1. (LPC2138 only) Capture input for Timer 1, channel 0. A/D converter 1, input 2. This analog input is always connected to its pin. (LPC2138 only)
	37	I I I/O	**P0.11**	**CTS1** **CAP1.1** **SCL1**	Clear to Send input for UART 1. (LPC2138 only) Capture input for Timer 1, channel 1. I^2C1 clock input/output. Open drain output (for I^2C compliance).
	38	I O I	**P0.12**	**DSR1** **MAT1.0** **AD1.3**	Data Set Ready input for UART 1. (LPC2138 only) Match output for Timer 1, channel 0. A/D converter 1, input 3. This analog input is always connected to its pin. (LPC2138 only)
	39	O O I	**P0.13**	**DTR1** **MAT1.1** **AD1.4**	Data Terminal Ready output for UART 1. (LPC2138 only) Match output for Timer 1, channel 1. A/D converter 1, input 4. This analog input is always connected to its pin. (LPC2138 only)
	41	I I I/O	**P0.14**	**DCD1** **EINT1** **SDA1** **Important:**	Data Carrier Detect input for UART 1. (LPC2138 only) External interrupt 1 input. I^2C1 data input/output. Open drain output (for I^2C compliance). LOW on P0.14 while $\overline{\text{RESET}}$ is LOW forces on-chip boot-loader to take control of th epart after reset.
	45	I I I	**P0.15**	**RI1** **EINT2** **AD1.5**	Ring Indicator input for UART 1. (LPC2138 only) External interrupt 2 input. A/D converter 1, input 5. This analog input is always connected to its pin. (LPC2138 only)
	46	I O I	**P0.16**	**EINT0** **MAT0.2** **CAP0.2**	External interrupt 0 input. Match output for Timer 0, channel 2. Capture input for Timer 0, channel 2.
	47	I I/O O	**P0.17**	**CAP1.2** **SCK1** **MAT1.2**	Capture input for Timer 1, channel 2. Serial Clock for SPI1. SPI clock output from master or input to slave. Match output for Timer 1, channel 2.

Table 48: Pin description for LPC2131/2132/2138

Pin Name	LQFP64 Pin #	Type	Description		
	53	I	**P0.18**	**CAP1.3**	Capture input for Timer 1, channel 3.
		I/O		**MISO1**	Master In Slave Out for SPI1. Data input to SPI master or data output from SPI slave.
		O		**MAT1.3**	Match output for Timer 1, channel 3.
	54	O	**P0.19**	**MAT1.2**	Match output for Timer 1, channel 2.
		I/O		**MOSI1**	Master Out Slave In for SPI1. Data output from SPI master or data input to SPI slave.
		I		**CAP1.2**	Capture input for Timer 1, channel 2.
	55	O	**P0.20**	**MAT1.3**	Match output for Timer 1, channel 3.
		I		**SSEL1**	Slave Select for SPI1. Selects the SPI interface as a slave.
		I		**EINT3**	External interrupt 3 input.
	1	O	**P0.21**	**PWM5**	Pulse Width Modulator output 5.
		I		**AD1.6**	A/D converter 1, input 6 (LPC2138 only).
		I		**CAP1.3**	Capture input for Timer 1, channel 3.
	2	I	**P0.22**	**AD1.7**	A/D converter 1, input 7. This analog input is always connected to its pin. (LPC2138 only)
		I		**CAP0.0**	Capture input for Timer 0, channel 0.
		O		**MAT0.0**	Match output for Timer 0, channel 0.
	58	I/0	**P0.23**		General purpose digital input-output pin.
	9	I	**P0.25**	**AD0.4**	A/D converter 0, input 4. This analog input is always connected to its pin.
		O		**Aout**	D/A converter output. (LPC2132/2138 only)
	10	I	**P0.26**	**AD0.5**	AD converter 0, input 5. This analog input is always connected to its pin.
	11	I	**P0.27**	**AD0.0**	A/D converter 0, input 0. This analog input is always connected to its pin.
		I		**CAP0.1**	Capture input for Timer 0, channel 1.
		O		**MAT0.1**	Match output for Timer 0, channel 1.
	13	I	**P0.28**	**AD0.1**	A/D converter 0, input 1. This analog input is always connected to its pin.
		I		**CAP0.2**	Capture input for Timer 0, channel 2.
		O		**MAT0.2**	Match output for Timer 0, channel 2.
	14	I	**P0.29**	**AD0.2**	A/D converter 0, input 2. This analog input is always connected to its pin.
		I		**CAP0.3**	Capture input for Timer 0, channel 3.
		O		**MAT0.3**	Match output for Timer 0, channel 3.
	15	I	**P0.30**	**AD0.3**	A/D converter 0, input 3. This analog input is always connected to its pin.
		I		**EINT3**	External interrupt 3 input.
		I		**CAP0.0**	Capture input for Timer 0, channel 0.

Table 48: Pin description for LPC2131/2132/2138

Pin Name	LQFP64 Pin #	Type	Description
	17	O	**P0.31** General purpose digital output only pin.
P1.0 to P1.31		I/O	**Port 1:** Port 1 is a 32-bit bi-directional I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the Pin Connect Block. **Note:** All Port 1 pins are 5V tolerant with built-in pull-up resistor that sets input level to high when corresponding pin is used as input. Pins 0 through 15 of port 1 are not available.
	16	O	**P1.16** **TRACEPKT0** Trace Packet, bit 0. Standard I/O port with internal pull-up.
	12	O	**P1.17** **TRACEPKT1** Trace Packet, bit 1. Standard I/O port with internal pull-up.
	8	O	**P1.18** **TRACEPKT2** Trace Packet, bit 2. Standard I/O port with internal pull-up.
	4	O	**P1.19** **TRACEPKT3** Trace Packet, bit 3. Standard I/O port with internal pull-up.
	48	O	**P1.20** **TRACESYNC** Trace Synchronization. Standard I/O port with internal pull-up. **Important:** LOW on pin P1.20 while $\overline{\text{RESET}}$ is LOW enables pins P1.25:16 to operate as a Trace port after reset.
	44	O	**P1.21** **PIPESTAT0** Pipeline Status, bit 0. Standard I/O port with internal pull-up.
	40	O	**P1.22** **PIPESTAT1** Pipeline Status, bit 1. Standard I/O port with internal pull-up.
	36	O	**P1.23** **PIPESTAT2** Pipeline Status, bit 2. Standard I/O port with internal pull-up.
	32	O	**P1.24** **TRACECLK** Trace Clock. Standard I/O port with internal pull-up.
	28	I	**P1.25** **EXTIN0** External Trigger Input. Standard I/O with internal pull-up.
	24	I/O	**P1.26** **RTCK** Returned Test Clock output. Extra signal added to the JTAG port. Assists debugger synchronization when processor frequency varies. Bi-directional pin with internal pullup. **Important:** LOW on pin P1.26 while $\overline{\text{RESET}}$ is LOW enables pins P1.31:26 to operate as a Debug port after reset.
	64	O	**P1.27** **TDO** Test Data out for JTAG interface.
	60	I	**P1.28** **TDI** Test Data in for JTAG interface.
	56	I	**P1.29** **TCK** Test Clock for JTAG interface.

Table 48: Pin description for LPC2131/2132/2138

Pin Name	LQFP64 Pin #	Type	Description
	52	I	**P1.30** **TMS** Test Mode Select for JTAG interface.
	20	I	**P1.31** **$\overline{\text{TRST}}$** Test Reset for JTAG interface.
$\overline{\text{RESET}}$	57	I	**External Reset input**: A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. TTL with hysteresis, 5V tolerant.
XTAL1	62	I	Input to the oscillator circuit and internal clock generator circuits.
XTAL2	61	O	Output from the oscillator amplifier.
RTCX1	3	I	Input to the RTC oscillator circuit.
RTCX2	5	O	Output from the RTC oscillator amplifier.
V_{SS}	6, 18, 25, 42, 50	I	**Ground:** 0V reference.
V_{SSA}	59	I	**Analog Ground:** 0V reference. This should nominally be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
V_{DD}	23, 43, 51	I	**3.3V Power Supply:** This is the power supply voltage for the core and I/O ports.
V_{DDA}	7	I	**Analog 3.3V Pad Power Supply:** This should be nominally the same voltage as V_3 but should be isolated to minimize noise and error. This voltage powers up the on-chip PLL.
V_{ref}	63	I	**A/D Converter Reference:** This should be nominally the same voltage as V_3 but should be isolated to minimize noise and error. Level on this pin is used as a reference for A/D and D/A convertors.
V_{bat}	49	I	**RTC Power Supply Pin.** 3.3V level on this pin supplies the power to the RTC.

7. PIN CONNECT BLOCK

FEATURES

- Allows individual pin configuration

APPLICATIONS

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

DESCRIPTION

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

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

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

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

REGISTER DESCRIPTION

The Pin Control Module contains 2 registers as shown in Table 49. below.

Table 49: Pin Connect Block Register Map

Name	Description	Access	Reset Value	Address
PINSEL0	Pin function select register 0	Read/Write	0x0000 0000	0xE002C000
PINSEL1	Pin function select register 1	Read/Write	0x0000 0000	0xE002C004
PINSEL2	Pin function select register 2	Read/Write	See Table 52	0xE002C014

Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

The PINSEL0 register controls the functions of the pins as per the settings listed in Table 53. The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 50: Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

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	TxD (UART0)	PWM1	Reserved	00
3:2	P0.1	GPIO Port 0.1	RxD (UART0)	PWM3	EINT0	00
5:4	P0.2	GPIO Port 0.2	SCL0 (I^2C)	Capture 0.0 (TIMER0)	Reserved	00
7:6	P0.3	GPIO Port 0.3	SDA0 (I^2C)	Match 0.0 (TIMER0)	EINT1	00
9:8	P0.4	GPIO Port 0.4	SCK (SPI0)	Capture 0.1 (TIMER0)	AD0.6	00
11:10	P0.5	GPIO Port 0.5	MISO (SPI0)	Match 0.1 (TIMER0)	AD0.7	00
13:12	P0.6	GPIO Port 0.6	MOSI (SPI0)	Capture 0.2 (TIMER0)	AD1.0 (LPC2138)	00
15:14	P0.7	GPIO Port 0.7	SSEL (SPI0)	PWM2	EINT2	00
17:16	P0.8	GPIO Port 0.8	TxD UART1	PWM4	AD1.1 (LPC2138)	00
19:18	P0.9	GPIO Port 0.9	RxD (UART1)	PWM6	EINT3	00
21:20	P0.10	GPIO Port 0.10	RTS (UART1) (LPC2138)	Capture 1.0 (TIMER1)	AD1.2 (LPC2138)	00
23:22	P0.11	GPIO Port 0.11	CTS (UART1) (LPC2138)	Capture 1.1 (TIMER1)	SCL1 (I^2C1)	00
25:24	P0.12	GPIO Port 0.12	DSR (UART1) (LPC2138)	Match 1.0 (TIMER1)	AD1.3 (LPC2138)	00
27:26	P0.13	GPIO Port 0.13	DTR (UART1) (LPC2138)	Match 1.1 (TIMER1)	AD1.4 (LPC2138)	00
29:28	P0.14	GPIO Port 0.14	CD (UART1) (LPC2138)	EINT1	SDA1 (I^2C1)	00
31:30	P0.15	GPIO Port 0.15	RI (UART1 (LPC2138)	EINT2	AD1.5 (LPC2138)	00

Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

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

Table 51: Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

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	EINT0	Match 0.2 (TIMER0)	Capture 0.2 (TIMER0)	00
3:2	P0.17	GPIO Port 0.17	Capture 1.2 (TIMER1)	SCK (SSP)	Match 1.2 (TIMER1)	00
5:4	P0.18	GPIO Port 0.18	Capture 1.3 (TIMER1)	MISO (SSP)	Match 1.3 (TIMER1)	00

Table 51: Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

PINSEL1	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
7:6	P0.19	GPIO Port 0.19	Match 1.2 (TIMER1)	MOSI (SSP)	Match 1.3 (TIMER1)	00
9:8	P0.20	GPIO Port 0.20	Match 1.3 (TIMER1)	SSEL (SSP)	EINT3	00
11:10	P0.21	GPIO Port 0.21	PWM5	AD1.6 (LPC2138)	Capture 1.3 (TIMER1)	00
13:12	P0.22	GPIO Port 0.22	AD1.7 (LPC2138)	Capture 0.0 (TIMER0)	Match 0.0 (TIMER0)	00
15:14	P0.23	GPIO Port 0.23	Reserved	Reserved	Reserved	00
17:16	P0.24	Reserved	Reserved	Reserved	Reserved	00
19:18	P0.25	GPIO Port 0.25	AD0.4	Aout (DAC) (LPC2132/2138)	Reserved	00
21:20	P0.26	Reserved				00
23:22	P0.27	GPIO Port 0.27	AD0.0	Capture 0.1 (TIMER0)	Match 0.1 (TIMER0)	00
25:24	P0.28	GPIO Port 0.28	AD0.1	Capture 0.2 (TIMER0)	Match 0.2 (TIMER0)	00
27:26	P0.29	GPIO Port 0.29	AD0.2	Capture 0.3 (TIMER0)	Match 0.3 (TIMER0)	00
29:28	P0.30	GPIO Port 0.30	AD0.3	EINT3	Capture 0.0 (TIMER0)	00
31:30	P0.31	GPI Port 0.31	Reserved	Reserved	Reserved	00

Pin Function Select Register 2 (PINSEL2 - 0xE002C014)

The PINSEL2 register controls the functions of the pins as per the settings listed in Table 52. The direction control bit in the IO1DIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

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

Table 52: Pin Function Select Register 2 (PINSEL2 - 0xE002C014)

PINSEL2	Description	Reset Value
1:0	Reserved. User software should should not write ones to reserved bits.	NA
2	When 0, pins P1.36:26 are used as GPIO pins. When 1, P1.31:26 are used as a Debug port.	$\overline{\text{P1.26/RTCK}}$
3	When 0, pins P1.25:16 are used as GPIO pins. When 1, P1.25:16 are used as a Trace port.	$\overline{\text{P1.20/}}$ $\overline{\text{TRACESYNC}}$
31:4	Reserved. User software should should not write ones to reserved bits.	NA

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 53: Pin Function Select Register Bits

Pinsel0 and Pinsel1 Values		Function	Value after Reset
0	0	Primary (default) function, typically GPIO Port	00
0	1	First alternate function	
1	0	Second alternate function	
1	1	Reserved	

The direction control bit in the IO0DIR/IO1DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically. Each derivative typically has a different pinout and therefore a different set of functions possible for each pin. Details for a specific derivative may be found in the appropriate data sheet.

8. GPIO

FEATURES

- Direction control of individual bits
- Separate control of output set and clear
- All I/O default to inputs after reset

APPLICATIONS

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

PIN DESCRIPTION

Table 54: GPIO Pin Description

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

REGISTER DESCRIPTION

LPC2131/2132/2138 has two 32-bit General Purpose I/O ports. Total of 30 input/output and a single output only pin out of 32 pins are available on PORT0. PORT1 has up to 16 pins available for GPIO functions. PORT0 and PORT1 are controlled via two groups of 4 registers as shown in Table 55.

Table 55: GPIO Register Map

Generic Name	Description	Access	Reset Value	PORT0 Address & Name	PORT1 Address & Name
IOPIN	GPIO Port Pin value register. The current state of the port pins can always be read from this register, regardless of pin direction and mode.	Read Only	NA	0xE0028000 IO0PIN	0xE0028010 IO1PIN
IOSET	GPIO Port Output set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.	Read/ Write	0x0000 0000	0xE0028004 IO0SET	0xE0028014 IO1SET
IODIR	GPIO Port Direction control register. This register individually controls the direction of each port pin.	Read/ Write	0x0000 0000	0xE0028008 IO0DIR	0xE0028018 IO1DIR
IOCLR	GPIO Port Output clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	Write Only	0x0000 0000	0xE002800C IO0CLR	0xE002801C IO1CLR

GPIO Pin Value Register (IO0PIN - 0xE0028000, IO1PIN - 0xE0028010)

This register provides the value of the GPIO pins. Register's value reflects any outside world influence on the GPIO configured pins only. Monitoring of non-GPIO configured port pins using IOPIN register will not be valid, since activities on non-GPIO configured pins are not indicated in the IOPIN register.

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

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

Table 56: GPIO Pin Value Register (IO0PIN - 0xE0028000, IO1PIN - 0xE0028010)

IOPIN	Description	Value after Reset
31:0	GPIO pin value bits. Bit 0 in IO0PIN corresponds to P0.0 ... Bit 31 in IO0PIN corresponds to P0.31	Undefined

GPIO Output Set Register (IO0SET - 0xE0028004, IO1SET - 0xE0028014)

This register is used to produce a HIGH level output at the port pins if they are 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 to IOSET has no effect.

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

Table 57: GPIO Output Set Register (IO0SET - 0xE0028004, IO1SET - 0xE0028014)

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit 0 in IO0SET corresponds to P0.0 ... Bit 31 in IO0SET corresponds to P0.31	0

GPIO Output Clear Register (IO0CLR - 0xE002800C, IO1CLR - 0xE002801C)

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

Table 58: GPIO Output Clear Register (IO0CLR - 0xE002800C, IO1CLR - 0xE002801C)

IOCLR	Description	Value after Reset
31:0	Output value CLEAR bits. Bit 0 in IO0CLR corresponds to P0.0 ... Bit 31 in IO0CLR corresponds to P0.31	0

GPIO Direction Register

(IO0DIR - 0xE0028008, IO1DIR - 0xE0028018)

This 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.

Table 59: GPIO Direction Register (IO0DIR - 0xE0028008, IO1DIR - 0xE0028018)

IODIR	Description	Value after Reset
31:0	Direction control bits (0 = INPUT, 1 = OUTPUT). Bit 0 in IO0DIR controls P0.0 ... Bit 31 in IO0DIR controls P0.31	0

GPIO USAGE NOTES

Example 1: sequential accesses to IOSET and IOCLR affecting the same GPIO pin/bit

State of the output configured GPIO pin is determined by writes into the pin's port IOSET and IOCLR registers. Last of these accesses to the IOSET/IOCLR register will determine the final output of a pin.

In case of a code:

```
IO0DIR = 0x0000 0080          ;pin P0.7 configured as output
IO0CLR = 0x0000 0080          ;P0.7 goes LOW
IO0SET = 0x0000 0080          ;P0.7 goes HIGH
IO0CLR = 0x0000 0080          ;P0.7 goes LOW
```

pin P0.7 is configured as an output (write to IO0DIR register). After this, P0.7 output is set to low (first write to IO0CLR register). Short high pulse follows on P0.7 (write access to IO0SET), and the final write to IO0CLR register sets pin P0.7 back to low level.

Example 2: immediate output of 0s and 1s on a GPIO port

Write access to port's IOSET followed by write to the IOCLR register results with pins outputting 0s being slightly later then pins outputting 1s. There are systems that can tolerate this delay of a valid output, but for some applications simultaneous output of a binary content (mixed 0s and 1s) within a group of pins on a single GPIO port is required. This can be accomplished by writing to the port's IOPIN register.

Following code will preserve existing output on PORT0 pins P0.[31:16] and P0.[7:0] and at the same time set P0.[15:8] to 0xA5, regardless of the previous value of pins P0.[15:8]:

IO0PIN = (IO0PIN && #0xFFFF00FF) || #0x0000A500

Writing to IOSET/IOCLR .vs. IOPIN

Write to IOSET/IOCLR register allows easy change of port's selected output pin(s) to high/low level at a time. Only pin/bit(s) in IOSET/IOCLR written with 1 will be set to high/low level, while those written as 0 will remain unaffected. However, by just writing to either IOSET or IOCLR register it is not possible to instantaneously output arbitrary binary data containing mixture of 0s and 1s on a GPIO port.

Write to IOPIN register enables instantaneous output of a desired content on a parallel GPIO. Binary data written into the IOPIN register will affect all output configured pins of that parallel port: 0s in the IOPIN will produce low level pin outputs and 1s in IOPIN will produce high level pin outputs. In order to change output of only a group of port's pins, application must logically AND readout from the IOPIN with mask containing 0s in bits corresponding to pins that will be changed, and 1s for all others. Finally, this result has to be logically ORred with the desired content and stored back into the IOPIN register. Example 2 from above illustrates output of 0xA5 on PORT0 pins 15 to 8 while preserving all other PORT0 output pins as they were before.

9. UART0

FEATURES

- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- LPC2131/2132/2138 contains mechanism that enables software flow control implementation.

PIN DESCRIPTION

Table 60: UART0 Pin Description

Pin Name	Type	Description
RxD0	Input	**Serial Input**. Serial receive data.
TxD0	Output	**Serial Output**. Serial transmit data.

REGISTER DESCRIPTION

Table 61: UART0 Register Map

Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value*	Address
U0RBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	un-defined	0xE000C000 DLAB = 0
U0THR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	NA	0xE000C000 DLAB = 0
U0DLL	Divisor Latch LSB	MSB							LSB	R/W	0x01	0xE000C000 DLAB = 1
U0DLM	Divisor Latch MSB	MSB							LSB	R/W	0	0xE000C004 DLAB = 1
U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0	0xE000C004 DLAB = 0
U0IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	0xE000C008
U0FCR	FIFO Control Register	Rx Trigger		Reserved			Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0	0xE000C008
U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0	0xE000C00C
U0LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000C014
U0SCR	Scratch Pad Register	MSB							LSB	R/W	0	0xE000C01C
U0TER	Transmit Enable	TxEn	Reserved [6:0]							R/W	0x80	0xE000C030

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

UART0 contains ten 8-bit registers as shown in Table 61. The Divisor Latch Access Bit (DLAB) is contained in U0LCR7 and enables access to the Divisor Latches.

UART0 Receiver Buffer Register (U0RBR - 0xE000C000 when DLAB = 0, Read Only)

The U0RBR is the top byte of the UART0 Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access the U0RBR. The U0RBR is always Read Only.

Since PE, FE and BI bits correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the U0LSR register, and then to read a byte from the U0RBR.

Table 62: UART0 Receiver Buffer Register (U0RBR - 0xE000C000 when DLAB = 0, Read Only)

U0RBR	Function	Description	Reset Value
7:0	Receiver Buffer Register	The UART0 Receiver Buffer Register contains the oldest received byte in the UART0 Rx FIFO.	un-defined

UART0 Transmitter Holding Register (U0THR - 0xE000C000 when DLAB = 0, Write Only)

The U0THR is the top byte of the UART0 Tx FIFO. The top byte is the newest character in the Tx FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access the U0THR. The U0THR is always Write Only.

Table 63: UART0 Transmitter Holding Register (U0THR - 0xE000C000 when DLAB = 0, Write Only)

U0THR	Function	Description	Reset Value
7:0	Transmit Holding Register	Writing to the UART0 Transmit Holding Register causes the data to be stored in the UART0 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	N/A

UART0 Divisor Latch LSB Register (U0DLL - 0xE000C000 when DLAB = 1)

UART0 Divisor Latch MSB Register (U0DLM - 0xE000C004 when DLAB = 1)

The UART0 Divisor Latch is part of the UART0 Baud Rate Generator and holds the value used to divide the VPB clock (pclk) in order to produce the baud rate clock, which must be 16x the desired baud rate. The U0DLL and U0DLM registers together form a 16 bit divisor where U0DLL contains the lower 8 bits of the divisor and U0DLM contains the higher 8 bits of the divisor. A ‘h0000 value is treated like a ‘h0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U0LCR must be one in order to access the UART0 Divisor Latches.

Table 64: UART0 Divisor Latch LSB Register (U0DLL - 0xE000C000 when DLAB = 1)

U0DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The UART0 Divisor Latch LSB Register, along with the U0DLM register, determines the baud rate of the UART0.	0x01

Table 65: UART0 Divisor Latch MSB Register (U0DLM - 0xE000C004 when DLAB = 1)

U0DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The UART0 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART0.	0

UART0 Interrupt Enable Register (U0IER - 0xE000C004 when DLAB = 0)

The U0IER is used to enable the four UART0 interrupt sources.

Table 66: UART0 Interrupt Enable Register (U0IER - 0xE000C004 when DLAB = 0)

U0IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. U0IER0 enables the Receive Data Available interrupt for UART0. It also controls the Character Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. U0IER1 enables the THRE interrupt for UART0. The status of this interrupt can be read from U0LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. U0IER2 enables the UART0 Rx line status interrupts. The status of this interrupt can be read from U0LSR[4:1].	0
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UART0 Interrupt Identification Register (U0IIR - 0xE000C008, Read Only)

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

Table 67: UART0 Interrupt Identification Register (U0IIR - 0xE000C008, Read Only)

U0IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that U0IIR0 is active low. The pending interrupt can be determined by evaluating U0IER3:1.	1
3:1	Interrupt Identification	011: 1. Receive Line Status (RLS) 010: 2a.Receive Data Available (RDA) 110: 2b.Character Time-out Indicator (CTI) 001: 3. THRE Interrupt. U0IER3 identifies an interrupt corresponding to the UART0 Rx FIFO. All other combinations of U0IER3:1 not listed above are reserved (000,100,101,111).	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable	These bits are equivalent to U0FCR0.	0

Interrupts are handled as described in Table 68. Given the status of U0IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. Interrupts are handled as described in Table 68. The U0IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART0 RLS interrupt (U0IIR3:1=011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART0 Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART0 Rx error condition that set the interrupt can be observed via U0LSR4:1. The interrupt is cleared upon an U0LSR read.

The UART0 RDA interrupt (U0IIR3:1=010) shares the second level priority with the CTI interrupt (U0IIR3:1=110). The RDA is activated when the UART0 Rx FIFO reaches the trigger level defined in U0FCR7:6 and is reset when the UART0 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (U0IIR3:1=110) is a second level interrupt and is set when the UART0 Rx FIFO contains at least one character and no UART0 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART0 Rx FIFO activity (read or write of UART0 RSR) will clear the interrupt. This interrupt is intended to flush the UART0 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was

10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 68: UART0 Interrupt Handling

U0IIR[3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	none	none	-
0110	Highest	Rx Line Status / Error	OE or PE or FE or BI	U0LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO (U0FCR0=1)	U0RBR Read or UART0 FIFO drops below trigger level
1100	Second	Character Time-out Indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) X 7 - 2] X 8 + {(trigger level - number of characters) X 8 + 1] RCLKs	U0 RBR Read
0010	Third	THRE	THRE	U0IIR Read (if source of interrupt) or THR write
note: values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.				

The UART0 THRE interrupt (U0IIR3:1=001) is a third level interrupt and is activated when the UART0 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART0 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE=1 and there have not been at least two characters in the U0THR at one time since the last THRE=1 event. This delay is provided to give the CPU time to write data to U0THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART0 THR FIFO has held two or more characters at one time and currently, the U0THR is empty. The THRE interrupt is reset when a U0THR write occurs or a read of the U0IIR occurs and the THRE is the highest interrupt (U0IIR3:1=001).

UART0 FIFO Control Register (U0FCR - 0xE000C008)

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

Table 69: UART0 FIFO Control Register (U0FCR - 0xE000C008)

U0FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both UART0 Rx and Tx FIFOs and U0FCR7:1 access. This bit must be set for proper UART0 operation. Any transition on this bit will automatically clear the UART0 FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to U0FCR1 will clear all bytes in UART0 Rx FIFO and reset the pointer logic. This bit is self-clearing.	0

Table 69: UART0 FIFO Control Register (U0FCR - 0xE000C008)

U0FCR	Function	Description	Reset Value
2	Tx FIFO Reset	Writing a logic 1 to U0FCR2 will clear all bytes in UART0 Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
5:3	Reserved	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 Select	00: trigger level 0 (1 character or 0x01h) 01: trigger level 1 (4 characters or 0x04h) 10: trigger level 2 (8 characters or 0x08h) 11: trigger level 3 (14 characters or 0x0eh) These two bits determine how many receiver UART0 FIFO characters must be written before an interrupt is activated.	0

UART0 Line Control Register (U0LCR - 0xE000C00C)

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

Table 70: UART0 Line Control Register (U0LCR - 0xE000C00C)

U0LCR	Function	Description	Reset Value
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0: 1 stop bit 1: 2 stop bits (1.5 if U0LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission. Output pin UART0 TxD is forced to logic 0 when U0LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

UART0 Line Status Register (U0LSR - 0xE000C014, Read Only)

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

Table 71: UART0 Line Status Register (U0LSR - 0xE000C014, Read Only)

U0LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: U0RBR is empty 1: U0RBR contains valid data U0LSR0 is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty.	0
1	Overrun Error (OE)	0: Overrun error status is inactive. 1: Overrun error status is active. The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.	0
2	Parity Error (PE)	0: Parity error status is inactive. 1: Parity error status is active. When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR2. Time of parity error detection is dependent on U0FCR0. A parity error is associated with the character at the top of the UART0 RBR FIFO.	0
3	Framing Error (FE)	0: Framing error status is inactive. 1: Framing error status is active. When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR3. The time of the framing error detection is dependent on U0FCR0. A framing error is associated with the character at the top of the UART0 RBR FIFO. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error.	0
4	Break Interrupt (BI)	0: Break interrupt status is inactive. 1: Break interrupt status is active. When RxD0 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RxD0 goes to marking state (all 1's). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCR0. The break interrupt is associated with the character at the top of the UART0 RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: U0THR contains valid data. 1: U0THR is empty. THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.	1
6	Transmitter Empty (TEMT)	0: U0THR and/or the U0TSR contains valid data. 1: U0THR and the U0TSR are empty. TEMT is set when both U0THR and U0TSR are empty; TEMT is cleared when either the U0TSR or the U0THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: U0RBR contains no UART0 Rx errors or U0FCR0=0. 1: UART0 RBR contains at least one UART0 Rx error. U0LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO.	0

UART0 Scratch Pad Register (U0SCR - 0xE000C01C)

The U0SCR has no effect on the UART0 operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the U0SCR has occurred.

Table 72: UART0 Scratch Pad Register (U0SCR - 0xE000C01C)

U0SCR	Function	Description	Reset Value
7:0	-	A readable, writable byte.	0

UART0 Baudrate Calculation

Example 1: Using $UART0_{baudrate}$ formula from above, it can be determined that system with pclk=20 MHz, U0DL=130 (U0DLM=0x00 and U0DLL=0x82), DivAddVal=0 and MulVal=1 will enable UART0 with $UART0_{baudrate}$=9615 bauds.

Example 2: Using $UART0_{baudrate}$ formula from above, it can be determined that system with pclk=20 MHz, U0DL=93 (U0DLM=0x00 and U0DLL=0x5D), DivAddVal=2 and MulVal=5 will enable UART0 with $UART0_{baudrate}$=9600 bauds.

Table 73: Baud-rates using 20 MHz peripheral clock (pclk)

Desired baud-rate	U0DLM:U0DLL[1)]	% error[2)]	Desired baud-rate	U0DLM:U0DLL[1)]	% error[2)]
50	25000	0.0000	4800	260	0.1600
75	16667	0.0020	7200	174	0.2240
110	11364	0.0032	9600	130	0.1600
134.5	9294	0.0034	19200	65	0.1600
150	8333	0.0040	38400	33	1.3760
300	4167	0.0080	56000	22	1.4400
600	2083	0.0160	57600	22	1.3760
1200	1042	0.0320	112000	11	1.4400
1800	694	0.0640	115200	11	1.3760
2000	625	0.0000	224000	6	7.5200
2400	521	0.0320	448000	3	7.5200
3600	347	0.0640			

1) values in the row represent decimal equivalent of a 16-bit long content {DLM:DLL}
2) refers to the percent error between desired and actual baud-rate

UART0 Transmit Enable Register (U0TER - 0xE0010030)

LPC2132/2138's U0TER enables implementation of software flow control. When TxEn=1, UART0 transmitter will keep sending data as long as they are available. As soon as TxEn becomes 0, UART0 transmittion will stop.

Table 74 describes how to use TxEn bit in order to achieve software flow control.

Table 74: UART0 Transmit Enable Register (U0TER - 0xE0010030)

U0TER	Function	Description	Reset Value
6:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
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.	0x01

ARCHITECTURE

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

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

The UART0 receiver block, U0Rx, monitors the serial input line, RxD0, for valid input. The UART0 Rx Shift Register (U0RSR) accepts valid characters via RxD0. After a valid character is assembled in the U0RSR, it is passed to the UART0 Rx Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART0 transmitter block, U0Tx, accepts data written by the CPU or host and buffers the data in the UART0 Tx Holding Register FIFO (U0THR). The UART0 Tx Shift Register (U0TSR) reads the data stored in the U0THR and assembles the data to transmit via the serial output pin, TxD0.

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

The interrupt interface contains registers U0IER and U0IIR. The interrupt interface receives several one clock wide enables from the U0Tx and U0Rx blocks.

Status information from the U0Tx and U0Rx is stored in the U0LSR. Control information for the U0Tx and U0Rx is stored in the U0LCR.

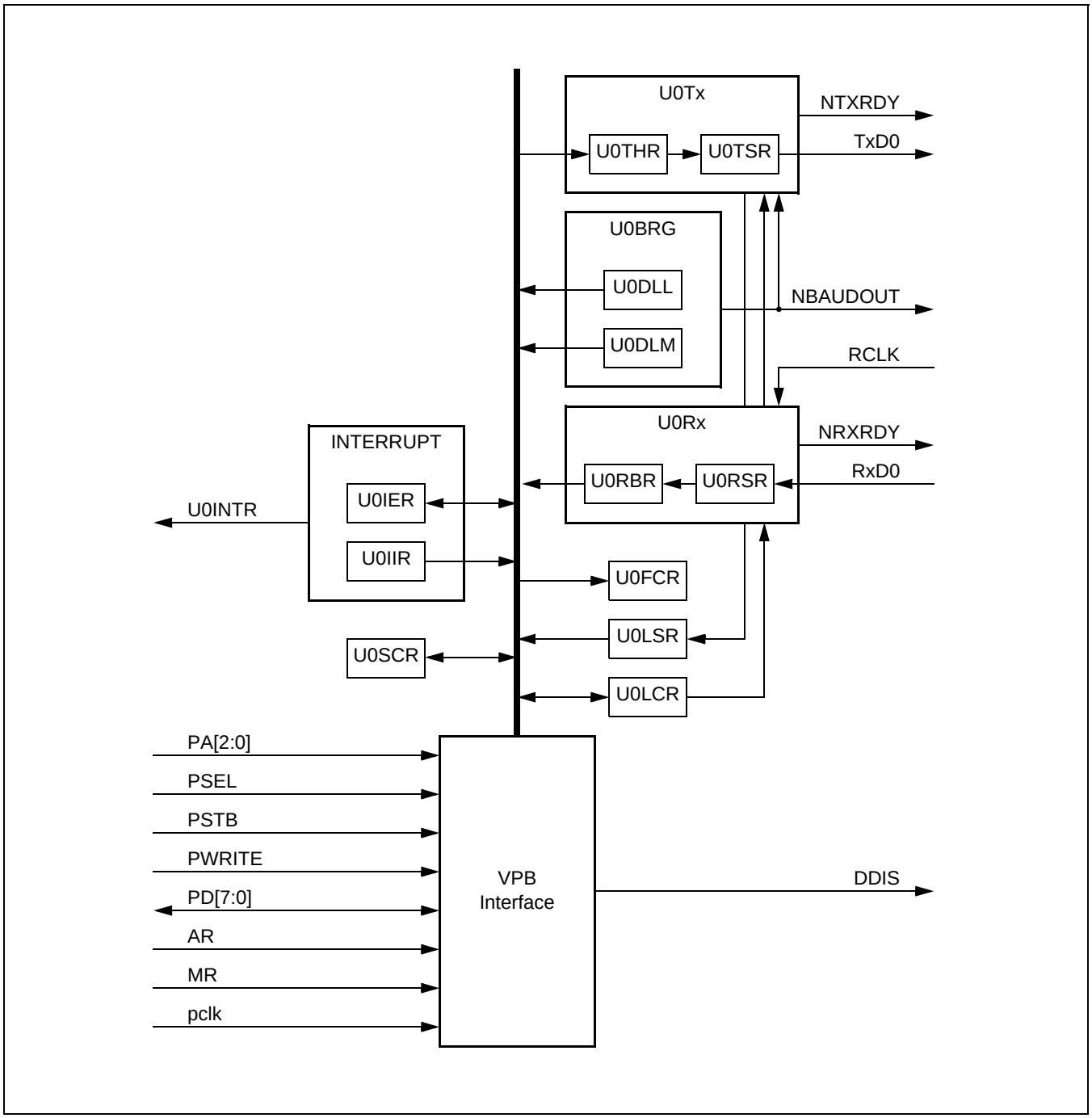


Figure 16: LPC2131/2132/2138 UART0 Block Diagram

10. UART1

FEATURES

- UART1 is identical to UART0, with the addition of a modem interface.
- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- Standard modem interface signals included (LPC2138 only).
- LPC2131/2132/2138 UART1 contains mechanism that enables implementation of either software or hardware flow control.

PIN DESCRIPTION

Table 75: UART1 Pin Description

Pin Name	Type	Description
RxD1	Input	**Serial Input**. Serial receive data.
TxD1	Output	**Serial Output**. Serial transmit data.
CTS1[(1)]	Input	**Clear To Send**. Active low signal indicates if the external modem is ready to accept transmitted data via TxD1 from the UART1. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR4. State change information is stored in U1MSR0 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
DCD1[(1)]	Input	**Data Carrier Detect**. Active low signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR7. State change information is stored in U1MSR3 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
DSR1[(1)]	Input	**Data Set Ready**. Active low signal indicates if the external modem is ready to establish a communications link with the UART1. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR5. State change information is stored in U1MSR1 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
DTR1[(1)]	Output	**Data Terminal Ready**. Active low signal indicates that the UART1 is ready to establish connection with external modem. The complement value of this signal is stored in U1MCR0.
RI1[(1)]	Input	**Ring Indicator**. Active low signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR6. State change information is stored in U1MSR2 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
RTS1[(1)]	Output	**Request To Send**. Active low signal indicates that the UART1 would like to transmit data to the external modem. The complement value of this signal is stored in U1MCR1.

[(1)]LPC2138 only.

REGISTER DESCRIPTION

Table 76: UART1 Register Map

Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value*	Address
U1RBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	un-defined	0xE0010000 DLAB = 0
U1THR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	NA	0xE0010000 DLAB = 0
U1DLL	Divisor Latch LSB	MSB							LSB	R/W	0	0xE0010000 DLAB = 1
U1DLM	Divisor Latch MSB	MSB							LSB	R/W	0	0xE0010004 DLAB = 1
U1IER	Interrupt Enable Register	0	0	0	0	Modem Status Interrupt Enable(1)	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0	0xE0010004 DLAB = 0
U1IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	0xE0010008
U1FCR	FIFO Control Register	Rx Trigger		Reserved		-	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0	0xE0010008
U1LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0	0xE001000C
U1MCR (1)	Modem Control Register	0	0	0	Loop Back	0	0	RTS	DTR	R/W	0	0xE0010010
U1LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE0010014
U1MSR (1)	Modem Status Register	DCD	RI	DSR	CTS	Delta DCD	Trailing Edge RI	Delta DSR	Delta CTS	RO	0	0xE0010018
U1SCR	Scratch Pad Register	MSB							LSB	R/W	0	0xE001001C
U1TER	Transmit Enable	TxEn	Reserved [6:0]							R/W	0x80	0xE0010030

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.
(1)LPC2138 only.

UART1 contains fourteen 8 and 32-bit organized registers as shown in Table 76. The Divisor Latch Access Bit (DLAB) is contained in U1LCR7 and enables access to the Divisor Latches.

UART1 Receiver Buffer Register (U1RBR - 0xE0010000 when DLAB = 0, Read Only)

The U1RBR is the top byte of the UART1 Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1RBR. The U1RBR is always Read Only.

Since PE, FE and BI bits correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the U1LSR register, and then to read a byte from the U1RBR.

Table 77: UART1 Receiver Buffer Register (U1RBR - 0xE0010000 when DLAB = 0, Read Only)

U1RBR	Function	Description	Reset Value
7:0	Receiver Buffer Register	The UART1 Receiver Buffer Register contains the oldest received byte in the UART1 Rx FIFO.	un-defined

UART1 Transmitter Holding Register (U1THR - 0xE0010000 when DLAB = 0, Write Only)

The U1THR is the top byte of the UART1 Tx FIFO. The top byte is the newest character in the Tx FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1THR. The U1THR is always Write Only.

Table 78: UART1 Transmitter Holding Register (U1THR - 0xE0010000 when DLAB = 0, Write Only)

U1THR	Function	Description	Reset Value
7:0	Transmit Holding Register	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.	N/A

UART1 Divisor Latch LSB Register (U1DLL - 0xE0010000 when DLAB = 1)

UART1 Divisor Latch MSB Register (U1DLM - 0xE0010004 when DLAB = 1)

The UART1 Divisor Latch is part of the UART1 Baud Rate Generator and holds the value used to divide the VPB 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 ‘h0000 value is treated like a ‘h0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U1LCR must be one in order to access the UART1 Divisor Latches. Details on how to select the right value for U1DLL and U1DLM can be found later on in this chapter.

Table 79: UART1 Divisor Latch LSB Register (U1DLL - 0xE0010000 when DLAB = 1)

U1DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The UART1 Divisor Latch LSB Register, along with the U1DLM register, determines the baud rate of the UART1.	0x01

Table 80: UART1 Divisor Latch MSB Register (U1DLM - 0xE0010004 when DLAB = 1)

U1DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The UART1 Divisor Latch MSB Register, along with the U1DLL register, determines the baud rate of the UART1.	0

UART1 Interrupt Enable Register (U1IER - 0xE0010004 when DLAB = 0)

The U1IER is used to enable the four interrupt sources.

Table 81: UART1 Interrupt Enable Register (U1IER - 0xE0010004 when DLAB = 0)

U1IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. U1IER0 enables the Receive Data Available interrupt for UART1. It also controls the Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. U1IER1 enables the THRE interrupt for UART1. The status of this interrupt can be read from U1LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. U1IER2 enables the UART1 Rx line status interrupts. The status of this interrupt can be read from U1LSR[4:1].	0
3	Modem Status Interrupt Enable(1)	0: Disable the modem interrupt. 1: Enable the modem interrupt. U1IER3 enables the modem interrupt. The status of this interrupt can be read from U1MSR[3:0].	0

(1)LPC2138 only.

UART1 Interrupt Identification Register (U1IIR - 0xE0010008, Read Only)

The U1IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U1IIR access. If an interrupt occurs during an U1IIR access, the interrupt is recorded for the next U1IIR access.

Table 82: UART1 Interrupt Identification Register (U1IIR - 0xE0010008, Read Only)

U1IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that U1IIR0 is active low. The pending interrupt can be determined by evaluating U1IIR3:1.	1
3:1	Interrupt Identification	011: 1. Receive Line Status (RLS) 010: 2a.Receive Data Available (RDA) 110: 2b.Character Time-out Indicator (CTI) 001: 3. THRE Interrupt. 000: 4. Modem Interrupt.(1) U1IER3 identifies an interrupt corresponding to the UART1 Rx FIFO and modem signals. All other combinations of U1IER3:1 not listed above are reserved (100,101,111).	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
7:6	FIFO Enable	These bits are equivalent to U1FCR0.	0

(1)LPC2138 only.

Interrupts are handled as described in Table 83. Given the status of U1IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U1IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART1 RLS interrupt (U1IIR3:1=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 U1LSR4:1. The interrupt is cleared upon an U1LSR read.

The UART1 RDA interrupt (U1IIR3:1=010) shares the second level priority with the CTI interrupt (U1IIR3:1=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 (U1IIR3:1=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 83: UART1 Interrupt Handling

U1IIR[3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	none	none	-
0110	Highest	Rx Line Status / Error	OE or PE or FE or BI	U1LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO mode (FCR0=1)	U1RBR Read 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) X 7 - 2] X 8 + {(trigger level - number of characters) X 8 + 1] RCLKs	U1RBR Read
0010	Third	THRE	THRE	U1IIR Read (if source of interrupt) or THR write
0000[(1)]	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read

[(1)]LPC2138 only.
Values “0011”. “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

The UART1 THRE interrupt (U1IIR3:1=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 (U1IIR3:1=001).

The modem interrupt (U1IIR3:1=000) is available in LPC2138 only. It is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a low to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining U1MSR3:0. A U1MSR read will clear the modem interrupt.

UART1 FIFO Control Register (U1FCR - 0xE0010008)

The U1FCR controls the operation of the UART1 Rx and Tx FIFOs.

Table 84: UART1 FIFO Control Register (U1FCR - 0xE0010008)

U1FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both UART1 Rx and Tx FIFOs and U1FCR7:1 access. This bit must be set for proper UART1 operation. Any transition on this bit will automatically clear the UART1 FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to U1FCR1 will clear all bytes in UART1 Rx FIFO and reset the pointer logic. This bit is self-clearing.	0
2	Tx FIFO Reset	Writing a logic 1 to U1FCR2 will clear all bytes in UART1 Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
5:3	Reserved	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 Select	00: trigger level 0 (1 character or 0x01h) 01: trigger level 1 (4 characters or 0x04h) 10: trigger level 2 (8 characters or 0x08h) 11: trigger level 3 (14 characters or 0x0eh) These two bits determine how many receiver UART1 FIFO characters must be written before an interrupt is activated.	0

UART1 Line Control Register (U1LCR - 0xE001000C)

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

Table 85: UART1 Line Control Register (U1LCR - 0xE001000C)

U1LCR	Function	Description	Reset Value
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0: 1 stop bit 1: 2 stop bits (1.5 if U1LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission. Output pin UART1 TxD is forced to logic 0 when U1LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

UART1 Modem Control Register (U1MCR - 0xE0010010) (LPC2138 only)

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

Table 86: UART1 Modem Control Register (U1MCR - 0xE0010010) (LPC2138 only)

U1MCR	Function	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
2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	Loopback Mode Select	0: Disable modem loopback mode 1: Enable modem loopback mode The modem loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RxD1, has no effect on loopback and output pin, TxD1 is held in marking state. The four modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the four modem outputs are connected to the four modem inputs. As a result of these connections, the upper four bits of the U1MSR will be driven by the lower four bits of the U1MCR rather than the four modem inputs in normal mode. This permits modem status interrupts to be generated in loopback mode by writing the lower four bits of U1MCR.	0

UART1 Line Status Register (U1LSR - 0xE0010014, Read Only)

The U1LSR is a read-only register that provides status information on the UART1 Tx and Rx blocks.

UART1 Modem Status Register (U1MSR - 0x0xE0010018) (LPC2138 only)

Table 87: UART1 Line Status Register (U1LSR - 0xE0010014, Read Only)

U1LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: U1RBR is empty 1: U1RBR contains valid data U1LSR0 is set when the U1RBR holds an unread character and is cleared when the UART1 RBR FIFO is empty.	0
1	Overrun Error (OE)	0: Overrun error status is inactive. 1: Overrun error status is active. The overrun error condition is set as soon as it occurs. An U1LSR read clears U1LSR1. U1LSR1 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

Table 87: UART1 Line Status Register (U1LSR - 0xE0010014, Read Only)

U1LSR	Function	Description	Reset Value
2	Parity Error (PE)	0: Parity error status is inactive. 1: Parity error status is active. When the parity bit of a received character is in the wrong state, a parity error occurs. An U1LSR read clears U1LSR2. Time of parity error detection is dependent on U1FCR0. A parity error is associated with the character at the top of the UART1 RBR FIFO.	0
3	Framing Error (FE)	0: Framing error status is inactive. 1: Framing error status is active. When the stop bit of a received character is a logic 0, a framing error occurs. An U1LSR read clears this bit. The time of the framing error detection is dependent on U1FCR0. A framing error is associated with the character at the top of the UART1 RBR FIFO. 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.	0
4	Break Interrupt (BI)	0: Break interrupt status is inactive. 1: Break interrupt status is active. When RxD1 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RxD1 goes to marking state (all 1's). An U1LSR read clears this status bit. The time of break detection is dependent on U1FCR0. The break interrupt is associated with the character at the top of the UART1 RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: U1THR contains valid data. 1: U1THR is empty. THRE is set immediately upon detection of an empty U1THR and is cleared on a U1THR write.	1
6	Transmitter Empty (TEMT)	0: U1THR and/or the U1TSR contains valid data. 1: U1THR and the U1TSR are empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the U1TSR or the U1THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: U1RBR contains no UART1 Rx errors or U1FCR0=0. 1: U1RBR contains at least one UART1 Rx error. U1LSR7 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

UART1 Modem Status Register Bit Descriptions (U1MSR - 0x0xE0010018) (LPC2138 only)

The U1MSR is a read-only register that provides status information on the modem input signals. U1MSR3:0 is cleared on U1MSR read. Note that modem signals have no direct affect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 88: UART1 Modem Status Register Bit Descriptions (U1MSR - 0x0xE0010018) (LPC2138 only)

U1MSR	Function	Description	Reset Value
0	Delta CTS	0: No change detected on modem input, CTS. 1: State change detected on modem input, CTS. Set upon state change of input CTS. Cleared on an U1MSR read.	0
1	Delta DSR	0: No change detected on modem input, DSR. 1: State change detected on modem input, DSR. Set upon state change of input DSR. Cleared on an U1MSR read.	0
2	Trailing Edge RI	0: No change detected on modem input, RI. 1: Low-to-high transition detected on RI. Set upon low to high transition of input RI. Cleared on an U1MSR read.	0
3	Delta DCD	0: No change detected on modem input, DCD. 1: State change detected on modem input, DCD. Set upon state change of input DCD. Cleared on an U1MSR read.	0
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

UART1 Scratch Pad Register (U1SCR - 0xE001001C)

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 89: UART1 Scratch Pad Register (U1SCR - 0xE001001C)

U1SCR	Function	Description	Reset Value
7:0	-	A readable, writable byte.	0

UART1 Baudrate Calculation

Example 1: Using $UART1_{baudrate}$ formula from above, it can be determined that system with pclk=20 MHz, U1DL=130 (U1DLM=0x00 and U1DLL=0x82), DivAddVal=0 and MulVal=1 will enable UART1 with $UART1_{baudrate}$=9615 bauds.

Example 2: Using $UART1_{baudrate}$ formula from above, it can be determined that system with pclk=20 MHz, U1DL=93 (U1DLM=0x00 and U1DLL=0x5D), DivAddVal=2 and MulVal=5 will enable UART1 with $UART1_{baudrate}$=9600 bauds.

Table 90: Baud-rates using 20 MHz peripheral clock (pclk)

Desired baud-rate	U1DLM:U1DLL[1)]	% error[2)]	Desired baud-rate	U1DLM:U1DLL[1)]	% error[2)]
50	25000	0.0000	4800	260	0.1600
75	16667	0.0020	7200	174	0.2240
110	11364	0.0032	9600	130	0.1600
134.5	9294	0.0034	19200	65	0.1600
150	8333	0.0040	38400	33	1.3760
300	4167	0.0080	56000	22	1.4400
600	2083	0.0160	57600	22	1.3760
1200	1042	0.0320	112000	11	1.4400
1800	694	0.0640	115200	11	1.3760
2000	625	0.0000	224000	6	7.5200
2400	521	0.0320	448000	3	7.5200
3600	347	0.0640			

1) values in the row represent decimal equivalent of a 16-bit long content {DLM:DLL}
2) refers to the percent error between desired and actual baud-rate

UART1 Transmit Enable Register (U1TER - 0xE0010030)

LPC2131/2132/2138's U1TER enables implementation of software and hardware flow control. When TxEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TxEn becomes 0, UART1 transmittion will stop.

Table 91: UART1 Transmit Enable Register (U1TER - 0xE0010030)

U1TER	Function	Description	Reset Value
6:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
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 (LPC2138: CTS - otherwise any GPIO/external interrupt line) has gone false, or with software handshaking, when it receives an XOFF character (DC3). Software can set this bit again when it detects that the Tx-permit signal has gone true, or when it receives an XON (DC1) character.	0x01

ARCHITECTURE

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

The VPB interface provides a communications link between the CPU or host and the UART1.

The UART1 receiver block, U1Rx, monitors the serial input line, RxD1, for valid input. The UART1 Rx Shift Register (U1RSR) accepts valid characters via RxD1. After a valid character is assembled in the U1RSR, it is passed to the UART1 Rx Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART1 transmitter block, U1Tx, accepts data written by the CPU or host and buffers the data in the UART1 Tx Holding Register FIFO (U1THR). The UART1 Tx Shift Register (U1TSR) reads the data stored in the U1THR and assembles the data to transmit via the serial output pin, TxD1.

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 Tx block. The U1BRG clock input source is the VPB clock (pclk). The main clock is divided down per the divisor specified in the U1DLL and u1DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The modem interface contains registers U1MCR and U1MSR. This interface is responsible for handshaking between a modem peripheral and the UART1.

The interrupt interface contains registers U1IER and U1IIR. The interrupt interface receives several one clock wide enables from the U1Tx, U1Rx and modem 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.

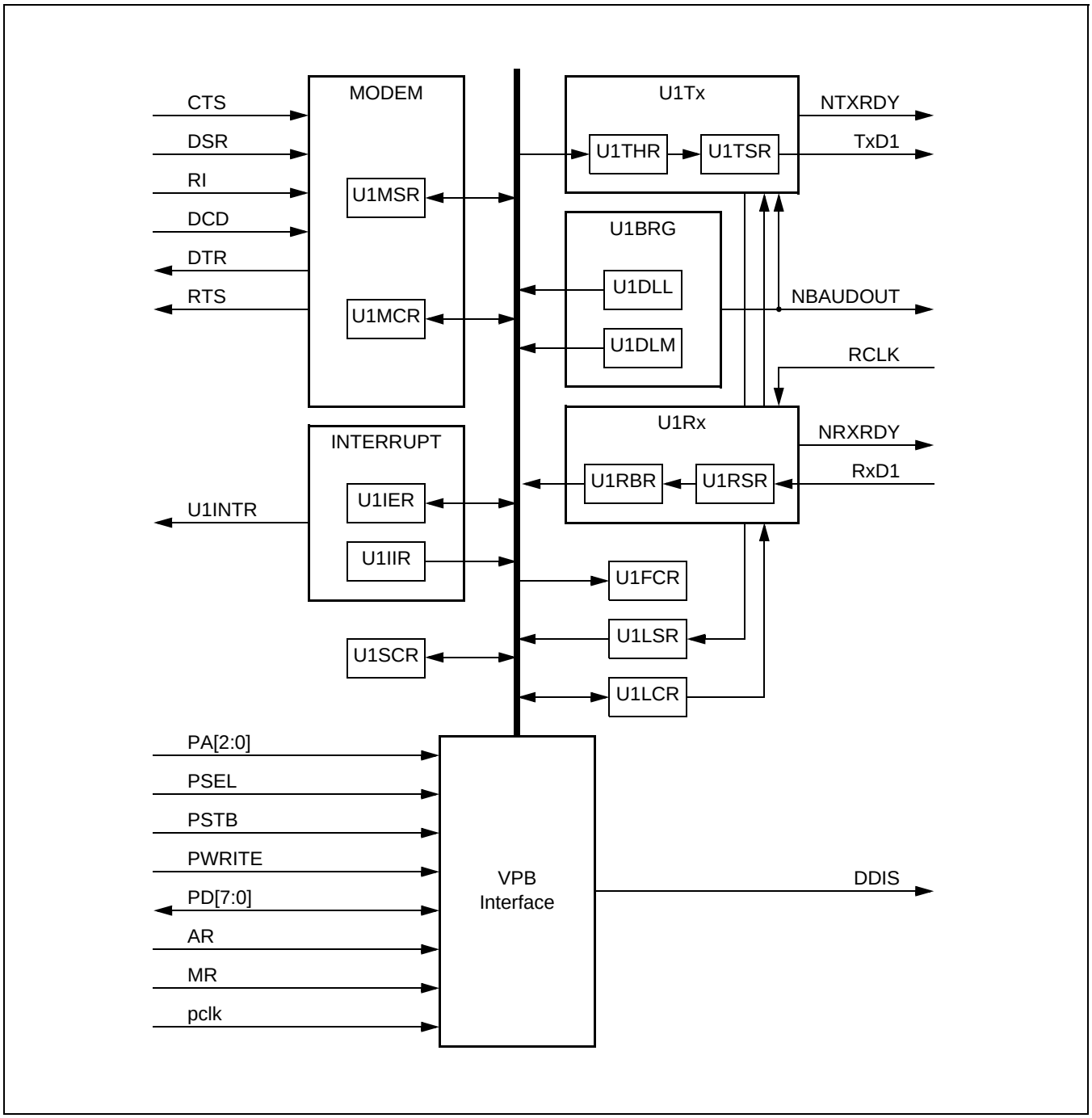


Figure 17: UART1 Block Diagram

11. I²C INTERFACES I2C0 AND I2C1

FEATURES

- Standard I²C compliant bus interfaces that may be configured as Master, Slave, or Master/Slave.
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock to allow adjustment of I²C transfer rates.
- Bidirectional data transfer between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.
- The I²C bus may be used for test and diagnostic purposes.

APPLICATIONS

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

DESCRIPTION

A typical I²C bus configuration is shown in Figure 18. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I²C bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I²C bus will not be released.

The LPC2131/2132/2138 I²C interfaces are byte oriented, and have four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The I²C interfaces complie with entire I²C specification, supporting the ability to turn power off to the LPC2131/2132/2138 without causing a problem with other devices on the same I²C bus (see "The I²C-bus specification" description under the heading "Fast-Mode", and notes for the table titled "Characteristics of the SDA and SCL I/O stages for F/S-mode I²C-bus devices" in the microcontrollers datasheet). This is sometimes a useful capability, but intrinsically limits alternate uses for the same pins if the I²C interface is not used. Seldom is this capability needed on multiple I²C interfaces within the same microcontroller.

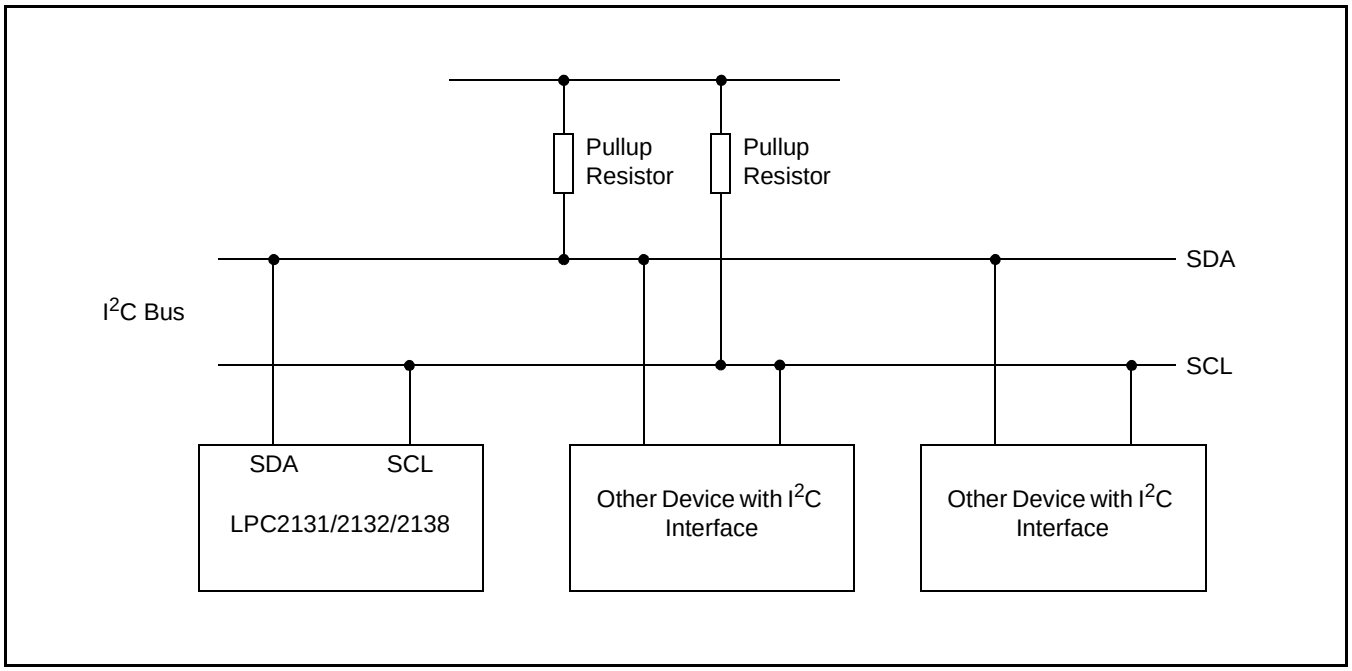


Figure 18: I²C Bus Configuration

PIN DESCRIPTION

Table 92: I²C Pin Description

Pin Name	Type	Description
SDA0,1	Input/Output	**I²C Serial Data.**
SCL0,1	Input/Output	**I²C Serial Clock.**

I²C OPERATING MODES

In a given application, the I²C block may operate as a master, a slave, or both. In the slave mode, the I²C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I²C block switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

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 Figure 19. I2EN must be set to 1 to enable the I²C function. If the AA bit is 0, the I²C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI Bit is cleared by writing 1 to the SIC bit in the I2CONCLR register.

	7	6	5	4	3	2	1	0
I2CONSET	-	I2EN	STA	STO	SI	AA	-	-
	-	1	0	0	0	0	-	-

Figure 19: Master Mode Configuration

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 08h. This status code is used to vector to a state service routine which will load the slave address and Write bit to the I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 18h, 20h, or 38h for the master mode, or 68h, 78h, or 0B0h if the slave mode was enabled (by setting AA 1). The appropriate actions to be taken for each of these status codes are shown in Table 102 to Table 105.

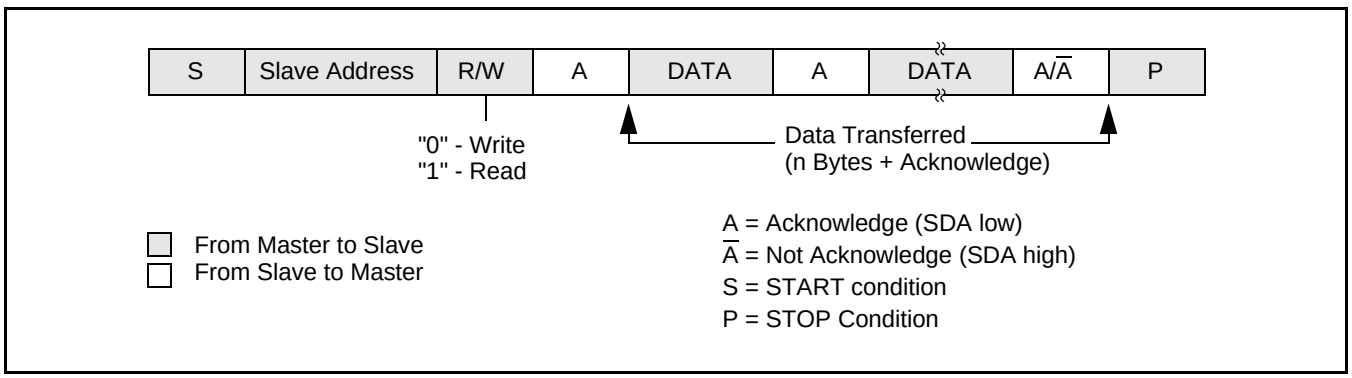


Figure 20: Format in the master transmitter mode

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 40H, 48H, or 38H. For slave mode, the possible status codes are 68H, 78H, or B0H. For details, refer to Table 103.

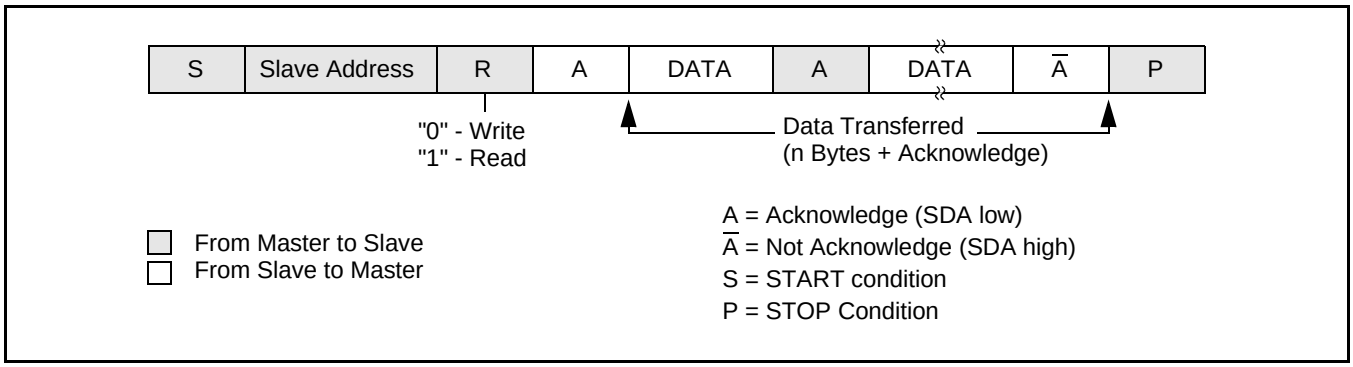


Figure 21: Format of master receiver mode

After a repeated START condition, I^2C may switch to the master transmitter mode.

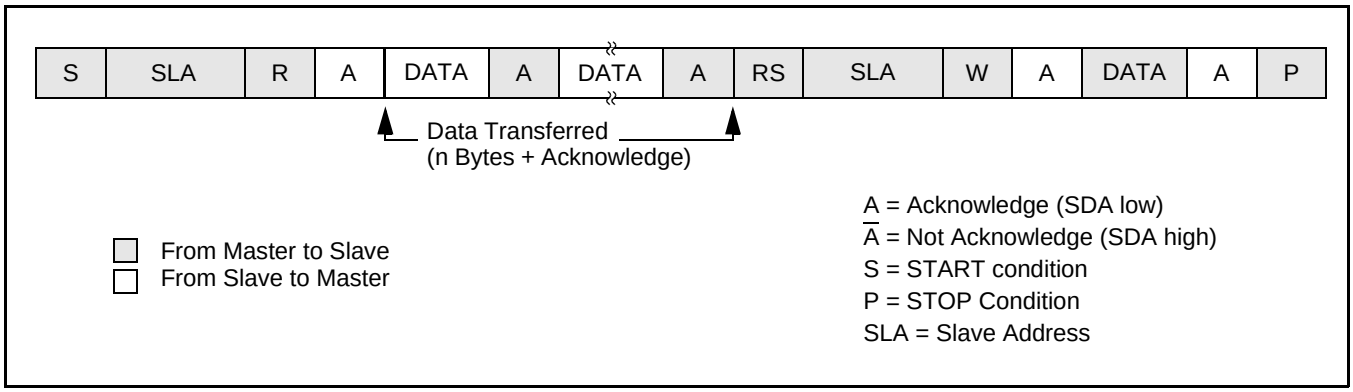


Figure 22: A master receiver switch to master transmitter after sending repeated START

Slave Receiver Mode:

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, user should write the Slave Address Register (I2ADR) and write the I^2C Control Set Register (I2CONSET) as shown in Figure 23.

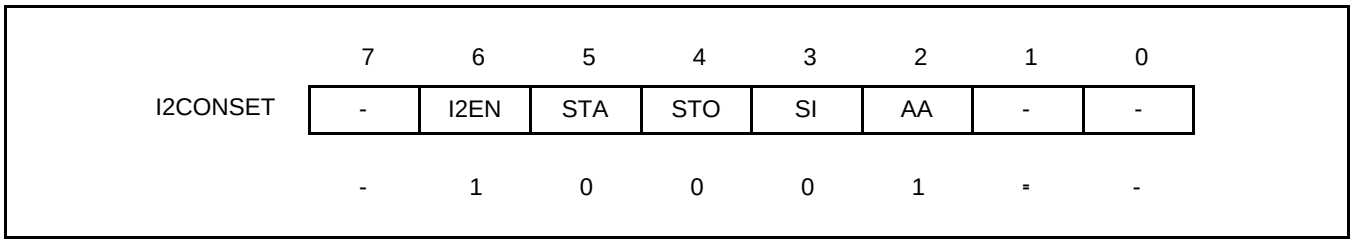


Figure 23: Slave Mode Configuration

I2EN must be set to 1 to enable the I^2C function. AA bit must be set to 1 to acknowledge its own slave address or the general call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I^2C interface waits until it is addressed by its own address or general address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status Register (I2STAT). Refer to Table 104 for the status codes and actions.

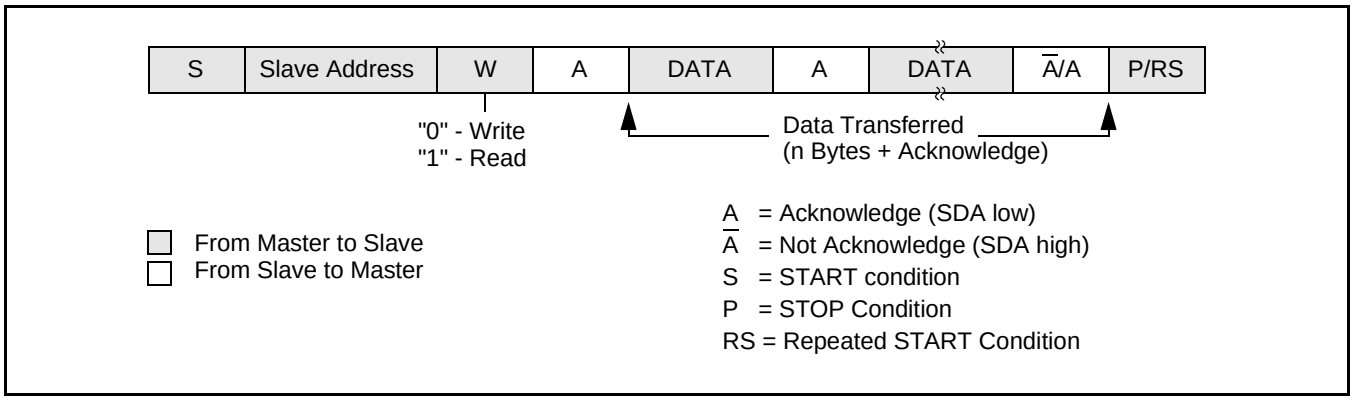


Figure 24: Format of slave receiver mode

Slave Transmitter Mode:

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^2C interface switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

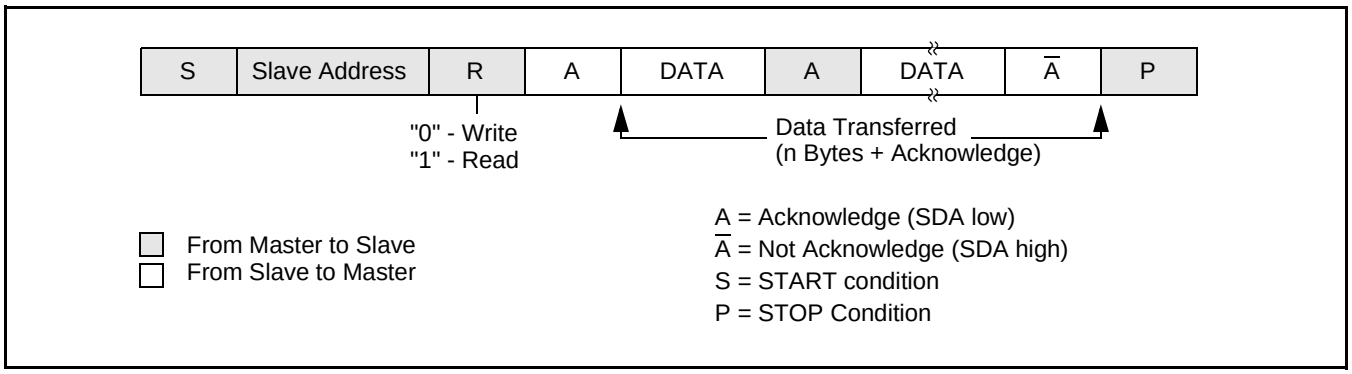


Figure 25: Format of slave transmitter mode

I^2C IMPLEMENTATION AND OPERATION

Figure 26 shows how the on-chip I^2C bus interface is implemented, and the following text describes the individual blocks.

Input Filters and Output Stages

Input signals are synchronized with the internal clock , and spikes shorter than three clocks are filtered out.

The output for I2C is a special pad designed to conform to the I^2C specification.

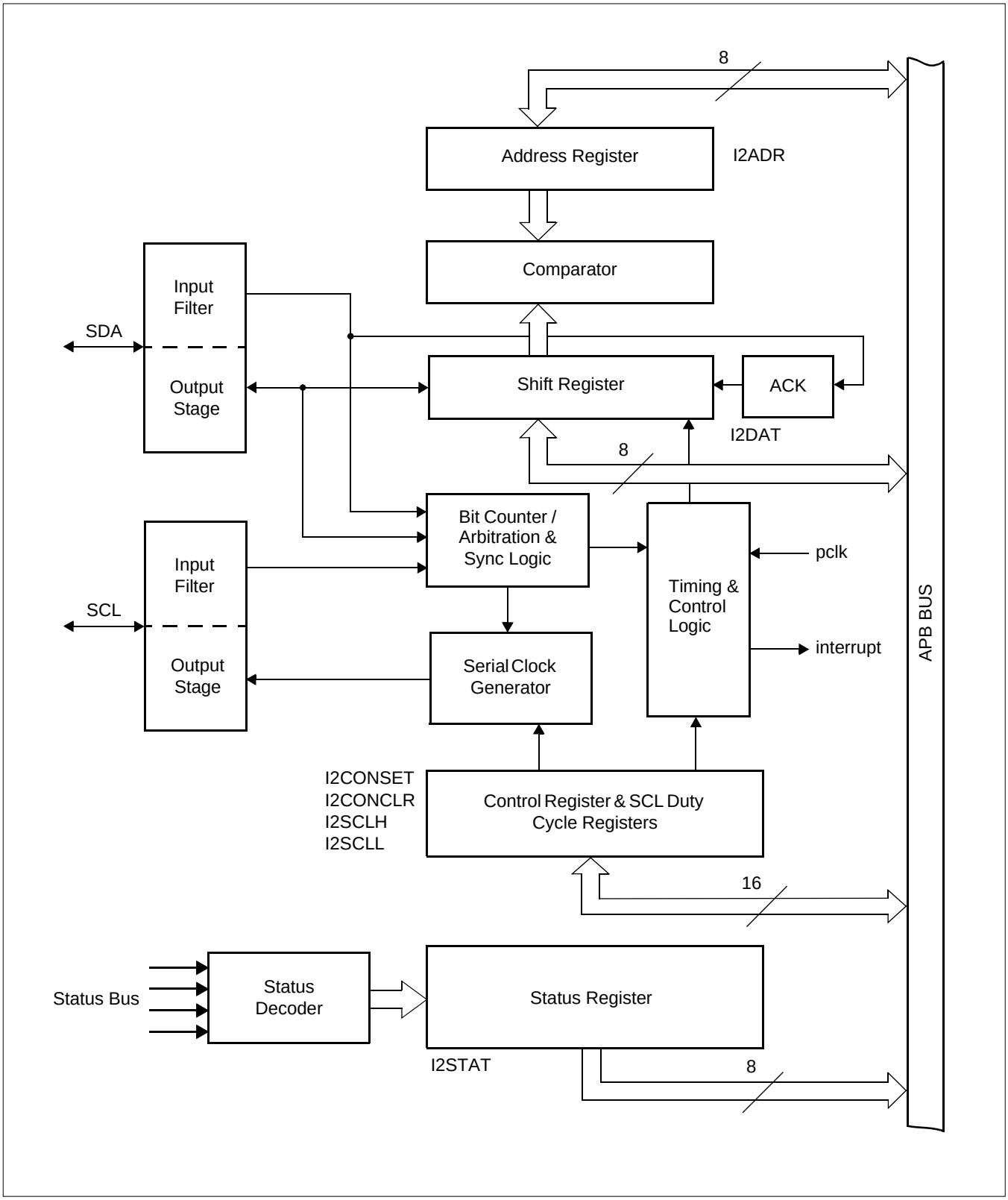


Figure 26: I²C Bus Serial Interface Block Diagram

Address Register, I2ADR

This register 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 (00H) recognition.

Comparator

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in I2ADR). It also compares the first received 8-bit byte with the general call address (00H). If an equality is found, the appropriate status bits are set and an interrupt is requested.

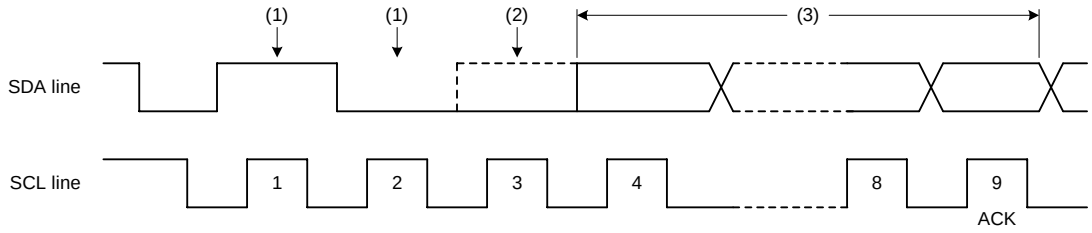
Shift Register, I2DAT

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.

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.

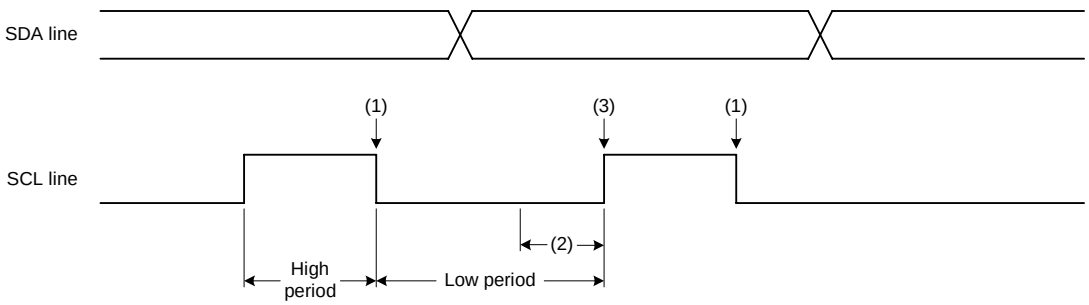
Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^2C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, the I^2C block generates no further clock pulses. Figure 27 shows the arbitration procedure.



1. Another device transmits identical serial data.
2. Another device overrules a logic 1 (dotted line) transmitted this I^2C master, by pulling the SDA line low. Arbitration is lost and this I^2C enters Slave receiver mode.
3. This I^2C is in the slave receiver mode, but still generates clock pulses until the current byte has been transmitted. This I^2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Figure 27: 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 28 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.

Figure 28: Serial Clock Synchronization (Figure 14)

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 I^2C 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.

Serial Clock Generator

This programmable clock pulse generator provides the SCL clock pulses when the I^2C block is in the master transmitter or master receiver mode. It is switched off when the I^2C block is in a slave mode. The I^2C output clock frequency and duty cycle is programmable via the I^2C 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.

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 I^2C bus status.

Control Register, I2CONSET and I2CONCLR

The I^2C control register contains bits used to control the following I^2C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I^2C control register may be read as I2CONSET. Writing to I2CONSET will set bits in the I^2C control register that correspond to ones in the value written. Conversely, writing to I2CONCLR will clear bits in the I^2C control register that correspond to ones in the value written.

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 I^2C 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 I^2C 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).

REGISTER DESCRIPTION

Each I^2C interface contains 7 registers as shown in Table 93 below.

Table 93: I^2C Register Map

Generic Name	Description	Access	Reset Value	I2C0 Address & Name	I2C1 Address & Name
I2CONSET	I^2C Control Set Register. When a one is written to a bit of this register, the corresponding bit in the I^2C control register is set. Writing a zero has no effect on the corresponding bit in the I^2C control register.	Read/ Set	0x00	0xE001C000 I2C0CONSET	0xE005C000 I2C1CONSET
I2STAT,	I^2C Status Register. During I2C operation, this register provides detailed status codes that allow software to determine the next action needed.	Read Only	0xF8	0xE001C004 I2C0STAT	0xE005C004 I2C1STAT
I2DAT	I^2C 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.	Read/ Write	0x00	0xE001C008 I2C0DAT	0xE005C008 I2C1DAT
I2ADR	I^2C Slave Address Register. Contains the 7-bit slave address for operation of the I^2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the general call address.	Read/ Write	0x00	0xE001C00C I2C0ADR	0xE005C00C I2C1ADR
I2SCLH	SCH Duty Cycle Register High Half Word. Determines the high time of the I^2C clock.	Read/ Write	0x04	0xE001C010 I2C0SCLH	0xE005C010 I2C1SCLH
I2SCLL	SCL Duty Cycle Register Low Half Word. Determines the low time of the I^2C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I^2C master and certain times used in slave mode.	Read/ Write	0x04	0xE001C014 I2C0SCLL	0xE005C014 I2C1SCLL
I2CONCLR	I^2C Control Clear Register. When a one is written to a bit of this register, the corresponding bit in the I^2C control register is cleared. Writing a zero has no effect on the corresponding bit in the I^2C control register.	Clear Only	NA	0xE001C018 I2C0CONCLR	0xE005C018 I2C1CONCLR

I^2C Control Set Register (I2CONSET: I2C0 - I2C0CONSET: 0xE001C000; I2C1 - I2C1CONSET: 0xE005C000)

The I2CONSET registers control setting 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 set. Writing a zero has no effect.

Table 94: I2C Control Set Register (I2CONSET: I2C0 - I2C0CONSET: 0xE001C000; I2C1 - I2C1CONSET: 0xE005C000)

I2CONSET	Name	Description	Reset Value
0:1	Reserved	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag. See text below.	0
3	SI	I^2C interrupt flag. See text below.	0
4	STO	STOP flag. See text below.	0
5	STA	START flag. See text below.	0
6	I2EN	I^2C interface enable. See text below.	0
7	Reserved	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I2EN I^2C Interface Enable. When I2EN is 1, the I^2C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I^2C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I^2C 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 I^2C bus since, when I2EN is reset, the I^2C bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I^2C 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 I^2C 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 I^2C interface is already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I^2C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the I2CONCLR register. When STA is 0, no START condition or repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I^2C bus if it the interface is in master mode, and transmits a START condition thereafter. If the I^2C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

STO is the STOP flag. Setting this bit causes the I^2C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I^2C bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I^2C Interrupt Flag. This bit is set when the I^2C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is high, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in I2CONCLR register.

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:

5. The address in the Slave Address Register has been received.
6. The general call address has been received while the general call bit (GC) in I2ADR is set.
7. A data byte has been received while the I^2C is in the master receiver mode.
8. 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:

9. A data byte has been received while the I^2C is in the master receiver mode.
10. A data byte has been received while the I^2C is in the addressed slave receiver mode.

I^2C Control Clear Register (I2CONCLR: I2C0 - I2C0CONCLR: 0xE001C018; I2C1 - I2C1CONCLR: 0xE005C018)

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.

Table 95: I2C Control Clear Register (I2CONCLR: I2C0 - I2C0CONCLR: 0xE001C018; I2C1 - I2C1CONCLR: 0xE005C018)

I2CONCLR	Name	Description	Reset Value
0:1	Reserved	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
2	AAC	Assert Acknowledge Clear bit	0
3	SIC	I^2C Interrupt Clear Bit	0
4	Reserved	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
5	STAC	Start flag clear bit	0
6	I2ENC	I^2C interface disable	0
7	Reserved	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

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 I^2C 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 I^2C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register.Writing 0 has no effect.

I^2C Status Register (I2STAT: I2C0 - I2C0STAT: 0xE001C004; I2C1 - I2C1STAT: 0xE005C004)

Each I^2C Status register reflects the condition of the corresponding I^2C interface. The I^2C Status register is Read-Only.

Table 96: I2C Status Register (I2STAT: I2C0 - I2C0STAT: 0xE001C004; I2C1 - I2C1STAT: 0xE005C004)

I2STAT	Description	Reset Value
2:0	These bits are always 0.	0x00
7:3	These bits give the actual status information about the I^2C interface.	0x1F

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is F8H, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I^2C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to Table 102 to Table 105.

I^2C Data Register (I2DAT: I2C0 - I2C0DAT: 0xE001C008; I2C1 - I2C1DAT: 0xE005C008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

Table 97: I2C Data Register (I2DAT: I2C0 - I2C0DAT: 0xE001C008; I2C1 - I2C1DAT: 0xE005C008)

I2DAT	Description	Reset Value
7:0	This register holds data values that have been received, or are to be transmitted.	0

I^2C Slave Address Register (I2ADR: I2C0 - I2C0DAT: 0xE001C00C; I2C1 - I2C1DAT: 0xE005C00C)

These registers are readable and writable, and is only used when an I^2C 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 (00h) is recognized.

Table 98: I2C Slave Address Register (I2ADR: I2C0 - I2C0DAT: 0xE001C00C; I2C1 - I2C1DAT: 0xE005C00C)

I20ADR	Name	Description	Reset Value
0	GC	General Call enable bit.	0
7:1	SLADR	The I^2C device address for slave mode.	0

I²C Clock Control Registers

I²C SCL High Duty Cycle Register (I2SCLH: I2C0 - I2C0SCLH: 0xE001C010; I2C1 - I2C1SCLH: 0xE005C010)

I²C SCL Low Duty Cycle Register (I2SCLL: I2C0 - I2C0SCLL: 0xE001C014; I2C1 - I2C1SCLL: 0xE005C014)

Software must set values for the registers I2SCLH and I2SCLL to select the appropriate data rate and duty cycle. I2SCLH defines the number of pclk cycles for the SCL high time, I2SCLL defines the number of pclk cycles for the SCL low time. The frequency is determined by the following formula:

Bit Frequency = f_{PCLK} / (I2SCLH + I2SCLL)

Where f_{PCLK} is the frequency of pclk.

Table 99: I2C SCL High Duty Cycle Register (I2SCLH: I2C0 - I2C0SCLH: 0xE001C010; I2C1 - I2C1SCLH: 0xE005C010)

I20SCLH	Description	Reset Value
15:0	Count for SCL HIGH time period selection	0x 0004

Table 100: I2C SCL Low Duty Cycle Register (I2SCLL: I2C0 - I2C0SCLL: 0xE001C014; I2C1 - I2C1SCLL: 0xE005C014)

I20SCLL	Description	Reset Value
15:0	Count for SCL LOW time period selection	0x 0004

The values for I2SCLL and I2SCLH should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I²C bus specification defines the SCL low time and high time at different values for a 400 kHz I²C rate. The value of the register must ensure that the data rate is in the I²C data rate range of 0 through 400KHz.Each register value must be greater than or equal to 4. Table 101 gives some examples of I²C bus rates based on pclk frequency and I2SCLL and I2SCLH values.

Table 101: Example I²C Clock Rates

I2SCLL+ I2SCLH	I²C Bit Frequency (kHz) At f_{PCLK} (MHz)							
	1	5	10	16	20	40	60	80
8	125.0	-	-	-	-	-	-	-
10	100.0	-	-	-	-	-	-	-
25	40.0	200.0	400.0	-	-	-	-	-
50	20.0	100.0	200.0	320.0	400.0	-	-	-
100	10.0	50.0	100.0	160.0	200.0	400.0	-	-
160	6.25	31.25	62.5	100.0	125.0	250.0	375.0	-
200	5.0	25.0	50.0	80.0	100.0	200.0	300.0	400.0
400	2.5	12.5	25.0	40.0	50.0	100.0	150.0	200.0
800	1.25	6.25	12.5	20.0	25.0	50.0	75.0	100.0

DETAILS OF I^{2}C OPERATING MODES

The four operating modes are:

- Master Transmitter.
- Master Receiver.
- Slave Receiver.
- Slave Transmitter.

Data transfers in each mode of operation are shown in Figures 29 – 33. These figures contain the following abbreviations:

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)
/A	Not acknowledge bit (high level at SDA)
Data	8-bit data byte
P	Stop condition

In Figures 29 – 33, 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 102-106.

Master Transmitter Mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 29). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

	7	6	5	4	3	2	1	0
I2CONSET	-	I2EN	STA	STO	SI	AA	-	-
	-	1	0	0	0	X	-	-

The I^{2}C rate must also be configured in the I2SCLL and I2SCLH registers. I2EN must be set to logic 1 to enable the I^{2}C block. If the AA bit is reset, the I^{2}C 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^{2}C 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^{2}C logic will now test the I^{2}C 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 08H. 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 18H, 20H, or 38H for the master mode and also 68H, 78H, or B0H 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 102. After a repeated start condition (state 10H). The I^2C block may switch to the master receiver mode by loading I2DAT with SLA+R).

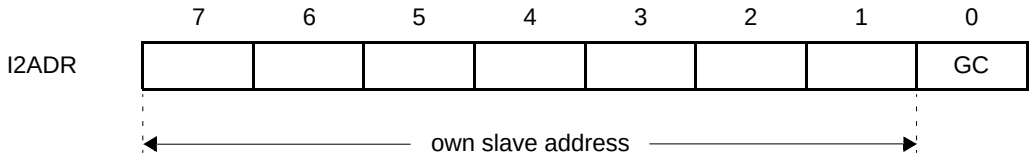
Master Receiver Mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 30). 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 40H, 48H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 103. After a repeated start condition (state 10H), the I^2C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

Slave Receiver Mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 31). To initiate the slave receiver mode, I2ADR and I2CON must be loaded as follows:



The upper 7 bits are the address to which the I^2C block will respond when addressed by a master. If the LSB (GC) is set, the I^2C block will respond to the general call address (00H); otherwise it ignores the general call address.

	7	6	5	4	3	2	1	0
I2CONSET	-	I2EN	STA	STO	SI	AA	-	-
	-	1	0	0	0	1	-	-

The I^2C bus rate settings do not affect the I^2C block in the slave mode. I2EN must be set to logic 1 to enable the I^2C block. The AA bit must be set to enable the I^2C block to acknowledge its own slave address or the general call address. STA, STO, and SI must be reset.

When I2ADR and I2CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^2C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status

code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 104. The slave receiver mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see status 68H and 78H).

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.

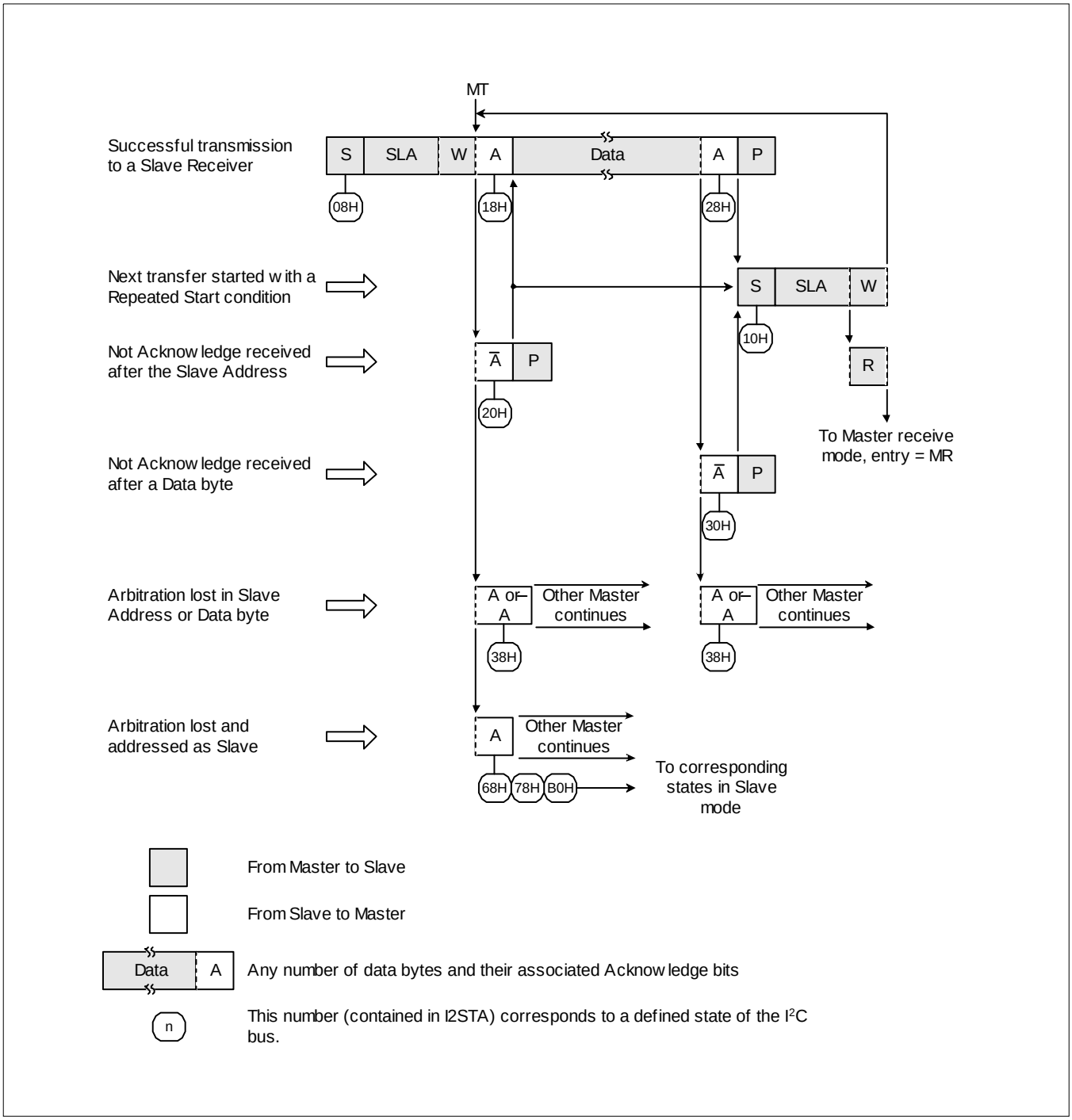


Figure 29: (Format and States in the Master Transmitter Mode

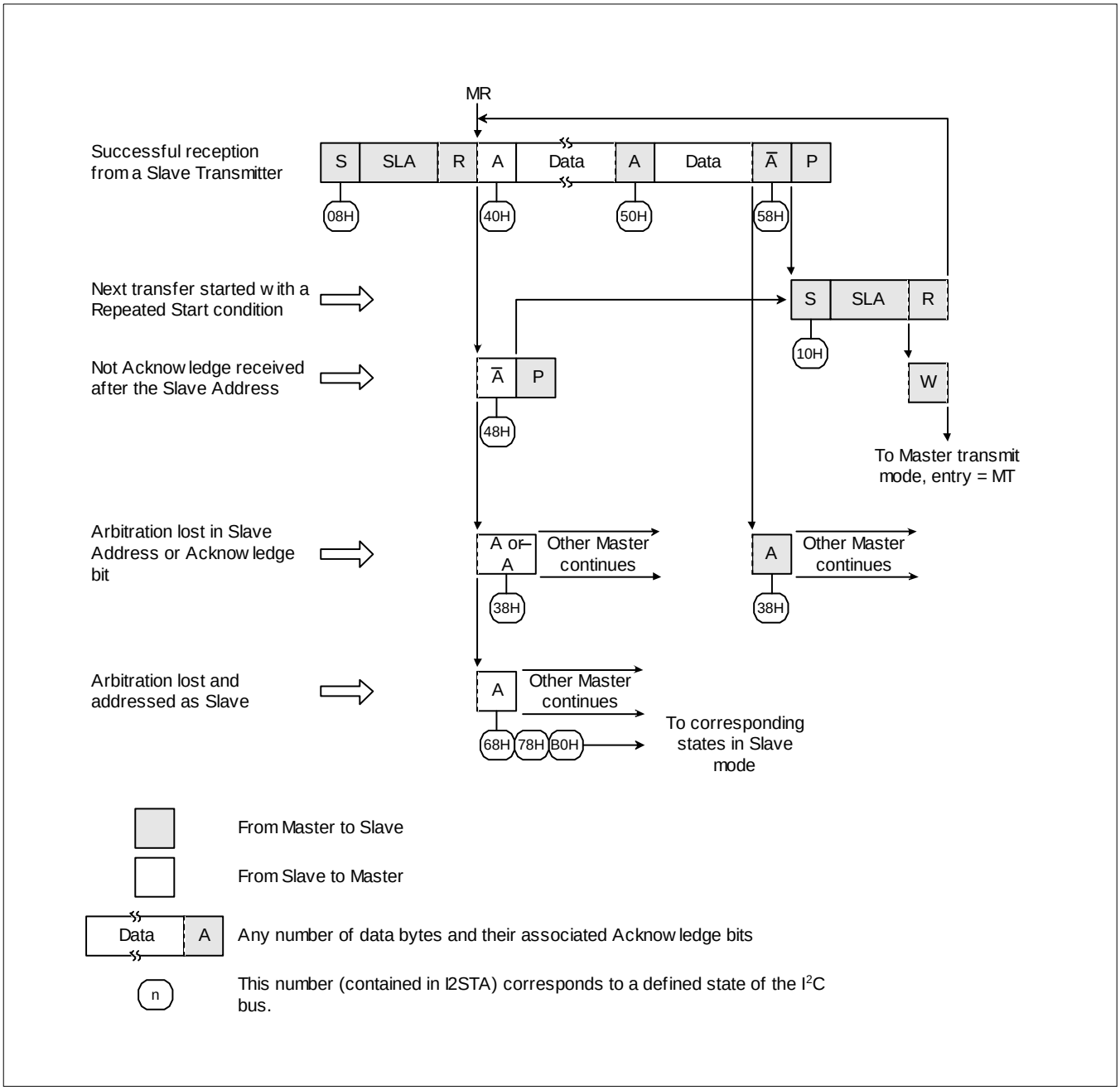


Figure 30: Format and States in the Master Receiver Mode

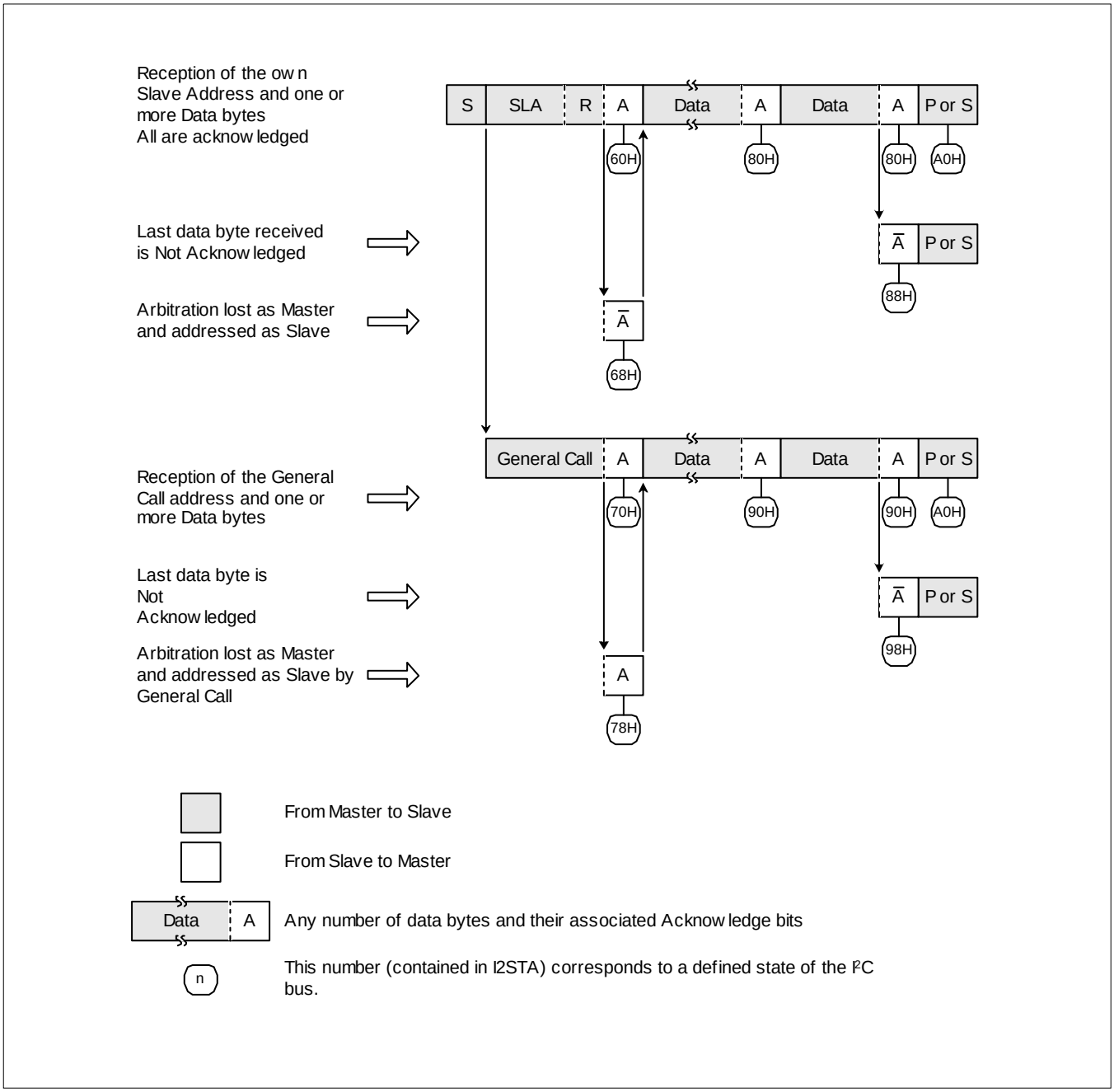


Figure 31: Format and States in the Slave Receiver Mode

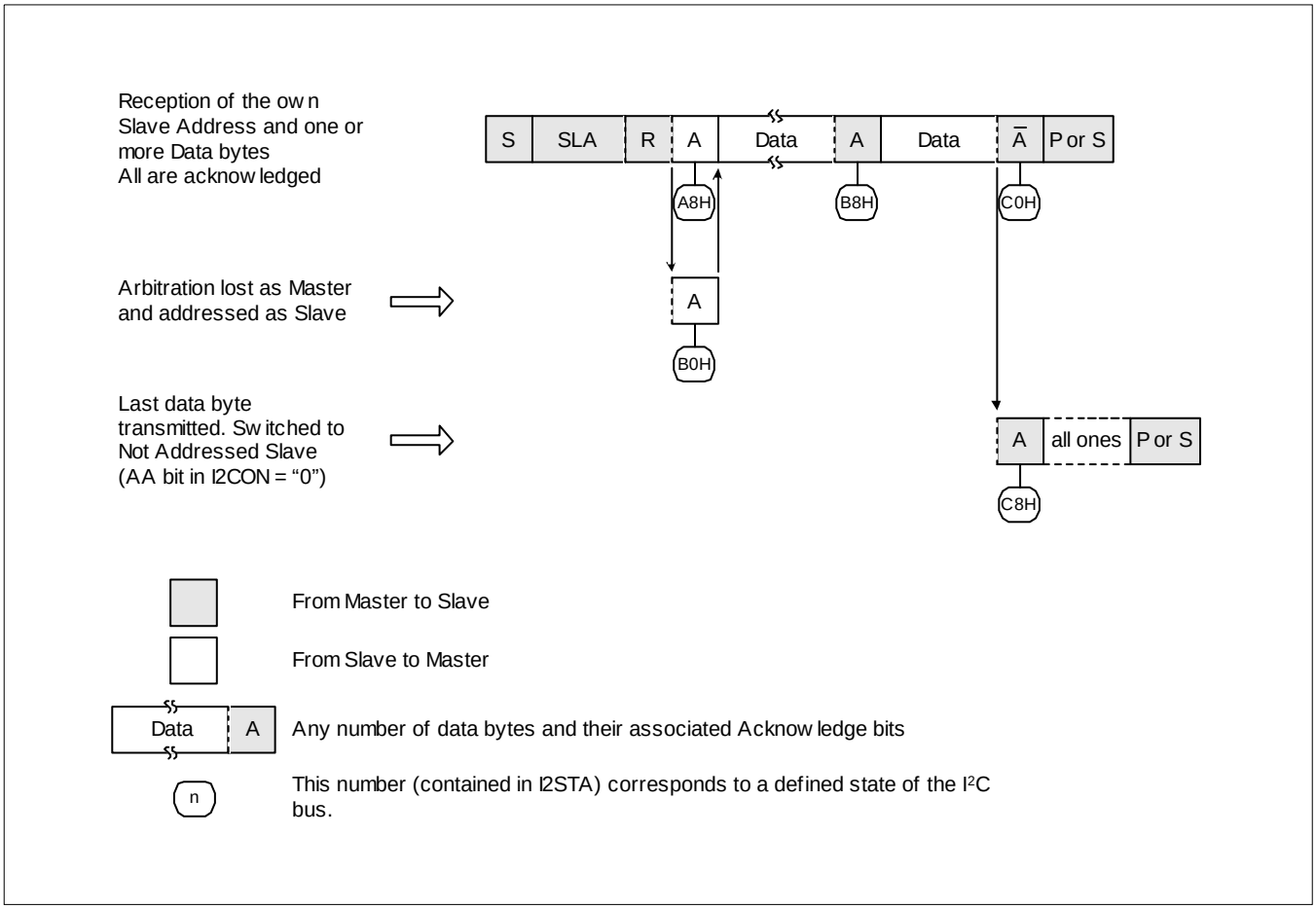


Figure 32: Format and States of the Slave Transmitter Mode

Slave Transmitter Mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 32). 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 105. The slave transmitter mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see state B0H).

If the AA bit is reset during a transfer, the I^2C block will transmit the last byte of the transfer and enter state C0H or C8H. 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.

Table 102: Master Transmitter Mode

STATUS CODE (I2STAT)	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	
08H	A START condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
10H	A repeated START condition has been transmitted	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/REC mode.
18H	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.
20H	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.
28H	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.
30H	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.
38H	Arbitration lost in SLA+R/W or Data bytes	No I2DAT action or	0	0	0	X	I^2C bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

Table 103: Master Receiver Mode

STATUS CODE (I2STAT)	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	
08H	A START condition has been transmitted	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
10H	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.
38H	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.
40H	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.
48H	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.
50H	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.
58H	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 104: Slave Receiver Mode

STATUS CODE (I2STAT)	STATUS OF THE I^2C BUS AND HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY I^2C HARDWARE
			TO I2CON				
		TO/FROM I2DAT	STA	STO	SI	AA	
60H	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.
68H	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.
70H	General call address (00H) 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.
78H	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.
80H	Previously addressed with own SLV address; DATA has been received; ACK has been returned	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
88H	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.
90H	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.
98H	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.

Table 104: Slave Receiver Mode

STATUS CODE (I2STAT)	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	
A0H	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR.0 = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Table 105: Slave Transmitter Mode

STATUS CODE (I2STAT)	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	
A8H	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.
B0H	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.
B8H	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.
C0H	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.
C8H	Last data byte in I2DAT has been transmitted (AA = 0); ACK has been received	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		no I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR.0 = logic 1.
		no I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		no I2DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Miscellaneous States:

There are two I2STAT codes that do not correspond to a defined I^2C hardware state (see Table 106). These are discussed below.

I2STAT = F8H:

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.

I2STAT = 00H:

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 106: Miscellaneous States

STATUS CODE (I2STAT)	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	
F8H	No relevant state information available; SI = 0	No I2DAT action	No I2CON action				Wait or proceed current transfer
00H	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 00H 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.

Some Special Cases:

The I^2C hardware has facilities to handle the following special cases that may occur during a serial transfer:

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 33). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I^2C hardware detects a repeated START condition on the I^2C 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 I^2C block will transmit a normal START condition (state 08H), and a retry of the total serial data transfer can commence.

Data Transfer After Loss of Arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 27). Loss of arbitration is indicated by the following states in I2STAT; 38H, 68H, 78H, and B0H (see Figures 29 and 30).

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 08H) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

Forced Access to the I^2C Bus

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

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I^2C bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I^2C bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I^2C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 34).

I^2C Bus Obstructed by a Low Level on SCL or SDA

An I^2C bus hang-up occurs if SDA or SCL is pulled LOW by an uncontrolled source. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the I^2C hardware cannot resolve this type of problem. When this occurs, the problem must be resolved by the device that is pulling the SCL bus line LOW.

If the SDA line is obstructed by another device on the bus (e.g., a slave device out of bit synchronization), the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 35). The I^2C hardware transmits additional clock pulses when the STA flag is set, but no START condition can be generated because the SDA line is pulled LOW while the I^2C bus is considered free. The I^2C hardware attempts to generate a START condition after every two additional clock pulses on the SCL line. When the SDA line is eventually released, a normal START condition is transmitted, state 08H is entered, and the serial transfer continues.

If a forced bus access occurs or a repeated START condition is transmitted while SDA is obstructed (pulled LOW), the I^2C hardware performs the same action as described above. In each case, state 08H is entered after a successful START condition is transmitted and normal serial transfer continues. Note that the CPU is not involved in solving these bus hang-up problems.

Bus Error

A bus error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I^2C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^2C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 00H. 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 106.

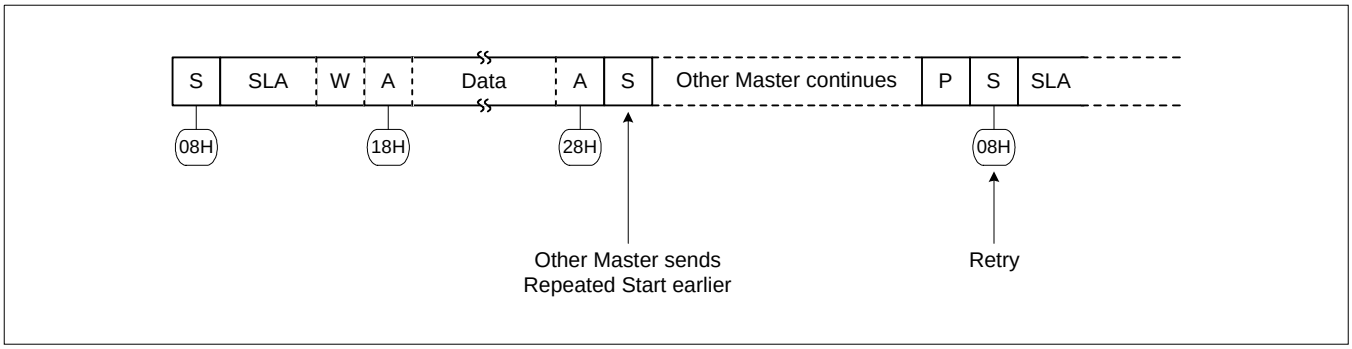


Figure 33: Simultaneous Repeated START Conditions from 2 Masters

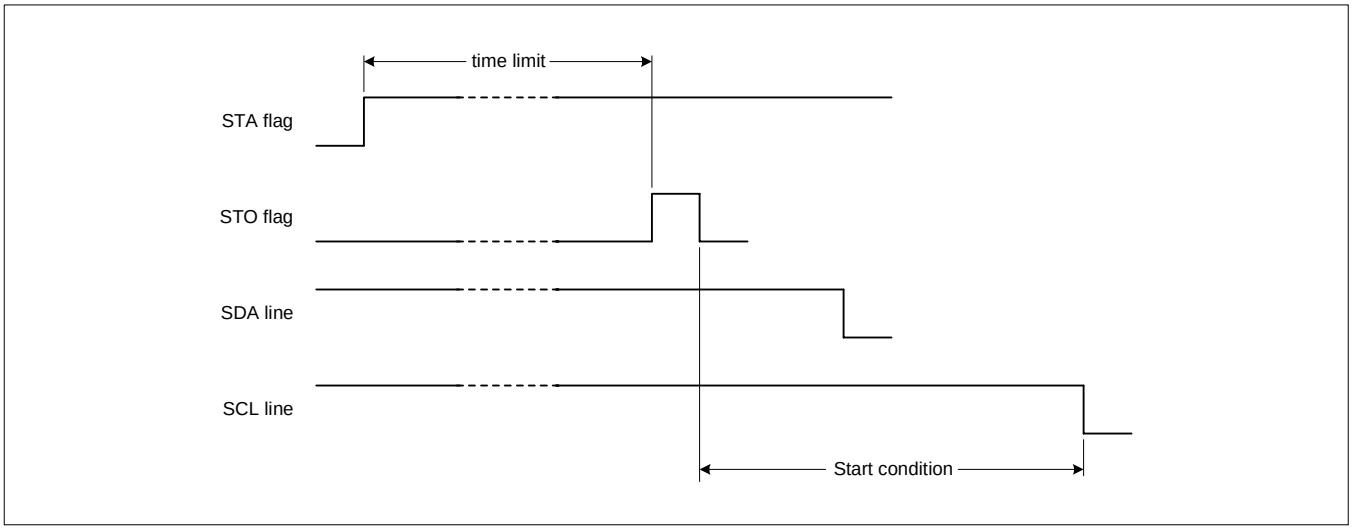
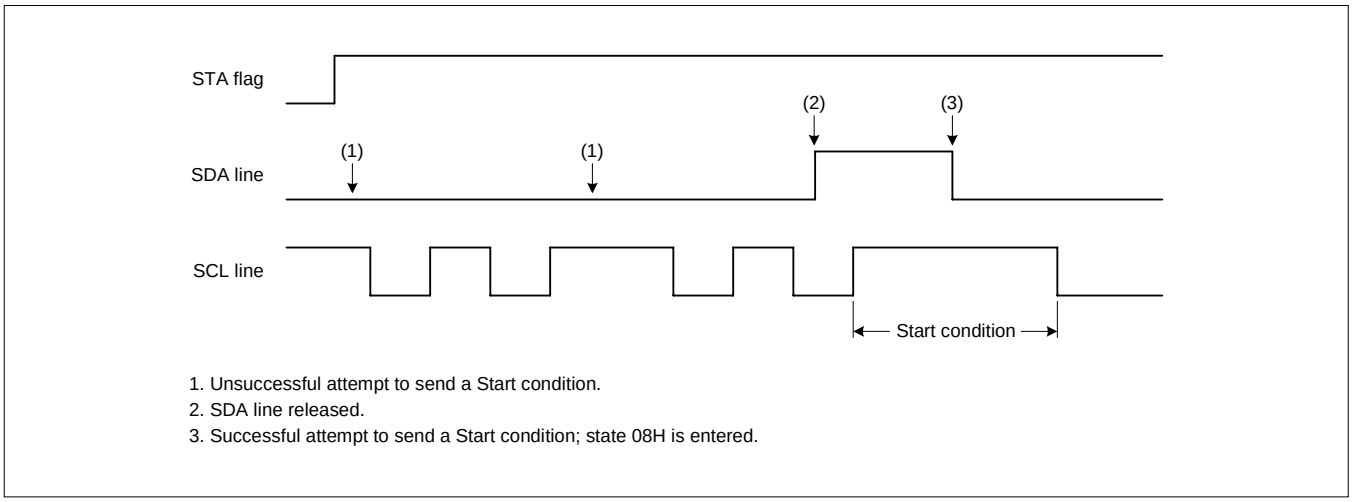


Figure 34: Forced Access to a Busy I²C Bus



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

Figure 35: Recovering from a Bus Obstruction Caused by a Low Level on SDA

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.

Initialization

In the initialization example, the I²C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- I2ADR is loaded with the part's own slave address and the general call bit (GC)
- The I²C interrupt enable and interrupt priority bits are set

- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by loading CR0 and CR1 in I2CON. The master routines must be started in the main program.

The I^2C hardware now begins checking the I^2C 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.

I^2C Interrupt Service

When the I^2C interrupt is entered, I2STAT contains a status code which identifies one of the 26 state services to be executed.

The State Service Routines

Each state routine is part of the I^2C interrupt routine and handles one of the 26 states.

Adapting State Services to an Application

The state service examples show the typical actions that must be performed in response to the 26 I^2C state codes. If one or more of the four I^2C 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 I^2C operations, in order to trap an inoperative bus or a lost service routine.

SOFTWARE EXAMPLE

Initialization Routine

Example to initialize I^2C Interface as a Slave and/or Master.

11. Load I2ADR with own Slave Address, enable general call recognition if needed.
12. Enable I^2C interrupt.
13. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only funcitons, write 0x40 to I2CONSET.

Start Master Transmit Function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a Start.

14. Initialize Master data counter.
15. Set up the Slave Address to which data will be transmitted, and add the Write bit.
16. Write 0x20 to I2CONSET to set the STA bit.
17. Set up data to be transmitted in Master Transmit buffer.
18. Initialize the Master data counter to match the length of the message being sent.
19. Exit

Start Master Receive Function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a Start.

20. Initialize Master data counter.
21. Set up the Slave Address to which data will be transmitted, and add the Read bit.
22. Write 0x20 to I2CONSET to set the STA bit.
23. Set up the Master Receive buffer.
24. Initialize the Master data counter to match the length of the message to be received.
25. Exit

I^2C Interrupt Routine

Determine the I^2C state and which state routine will be used to handle it.

26. Read the I^2C status from I2STA.
27. Use the status value to branch to one of 26 possible state routines.

Non Mode Specific States

State : 00

Bus Error. Enter not addressed Slave mode and release bus.

28. Write 0x14 to I2CONSET to set the STO and AA bits.
29. Write 0x08 to I2CONCLR to clear the SI flag.

30. Exit.

Master States

State 08 and State 10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

State : 08

A Start condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

31. Write Slave Address with R/W bit to I2DAT.
32. Write 0x04 to I2CONSET to set the AA bit.
33. Write 0x08 to I2CONCLR to clear the SI flag.
34. Set up Master Transmit mode data buffer.
35. Set up Master Receive mode data buffer.
36. Initialize Master data counter.
37. Exit.

State : 10

A repeated Start condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

38. Write Slave Address with R/W bit to I2DAT.
39. Write 0x04 to I2CONSET to set the AA bit.
40. Write 0x08 to I2CONCLR to clear the SI flag.
41. Set up Master Transmit mode data buffer.
42. Set up Master Receive mode data buffer.
43. Initialize Master data counter.
44. Exit.

Master Transmitter States

State : 18

Previous state was State 8 or State 10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted, an ACK bit will be received.

45. Load I2DAT with first data byte from Master Transmit buffer.
46. Write 0x04 to I2CONSET to set the AA bit.
47. Write 0x08 to I2CONCLR to clear the SI flag.
48. Increment Master Transmit buffer pointer.
49. Exit.

State : 20

Slave Address + Write has been transmitted, NOT ACK has been received. A Stop condition will be transmitted.

50. Write 0x14 to I2CONSET to set the STO and AA bits.
51. Write 0x08 to I2CONCLR to clear the SI flag.
52. Exit.

State : 28

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.

53. Decrement the Master data counter, skip to step 5 if not the last data byte.
54. Write 0x14 to I2CONSET to set the STO and AA bits.
55. Write 0x08 to I2CONCLR to clear the SI flag.
56. Exit.
57. Load I2DAT with next data byte from Master Transmit buffer.
58. Write 0x04 to I2CONSET to set the AA bit.
59. Write 0x08 to I2CONCLR to clear the SI flag.
60. Increment Master Transmit buffer pointer.
61. Exit.

State : 30

Data has been transmitted, NOT ACK received. A Stop condition will be transmitted.

62. Write 0x14 to I2CONSET to set the STO and AA bits.
63. Write 0x08 to I2CONCLR to clear the SI flag.
64. Exit.

State : 38

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.

65. Write 0x24 to I2CONSET to set the STA and AA bits.
66. Write 0x08 to I2CONCLR to clear the SI flag.

67. Exit.

Master Receiver States

State : 40

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.

68. Write 0x04 to I2CONSET to set the AA bit.
69. Write 0x08 to I2CONCLR to clear the SI flag.
70. Exit.

State : 48

Slave Address + Read has been transmitted, NOT ACK has been received. A Stop condition will be transmitted.

71. Write 0x14 to I2CONSET to set the STO and AA bits.
72. Write 0x08 to I2CONCLR to clear the SI flag.
73. Exit.

State : 50

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.

74. Read data byte from I2DAT into Master Receive buffer.
75. Decrement the Master data counter, skip to step 5 if not the last data byte.
76. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
77. Exit.
78. Write 0x04 to I2CONSET to set the AA bit.
79. Write 0x08 to I2CONCLR to clear the SI flag.
80. Increment Master Receive buffer pointer.
81. Exit.

State : 58

Data has been received, NOT ACK has been returned. Data will be read from I2DAT. A Stop condition will be transmitted.

82. Read data byte from I2DAT into Master Receive buffer.
83. Write 0x14 to I2CONSET to set the STO and AA bits.
84. Write 0x08 to I2CONCLR to clear the SI flag.
85. Exit.

Slave Receiver States

State : 60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

86. Write 0x04 to I2CONSET to set the AA bit.
87. Write 0x08 to I2CONCLR to clear the SI flag.
88. Set up Slave Receive mode data buffer.
89. Initialize Slave data counter.
90. Exit.

State : 68

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.

91. Write 0x24 to I2CONSET to set the STA and AA bits.
92. Write 0x08 to I2CONCLR to clear the SI flag.
93. Set up Slave Receive mode data buffer.
94. Initialize Slave data counter.
95. Exit.

State : 70

General call has been received, ACK has been returned. Data will be received and ACK returned.

96. Write 0x04 to I2CONSET to set the AA bit.
97. Write 0x08 to I2CONCLR to clear the SI flag.
98. Set up Slave Receive mode data buffer.
99. Initialize Slave data counter.
100. Exit.

State : 78

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.

101. Write 0x24 to I2CONSET to set the STA and AA bits.
102. Write 0x08 to I2CONCLR to clear the SI flag.
103. Set up Slave Receive mode data buffer.
104. Initialize Slave data counter.
105. Exit.

State : 80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

106. Read data byte from I2DAT into the Slave Receive buffer.
107. Decrement the Slave data counter, skip to step 5 if not the last data byte.
108. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.

109. Exit
110. Write 0x04 to I2CONSET to set the AA bit.

111. Write 0x08 to I2CONCLR to clear the SI flag.
112. Increment Slave Receive buffer pointer.
113. Exit.

State : 88

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.

114. Write 0x04 to I2CONSET to set the AA bit.
115. Write 0x08 to I2CONCLR to clear the SI flag.
116. Exit.

State : 90

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.

117. Read data byte from I2DAT into the Slave Receive buffer.
118. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
119. Exit

State : 98

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.

120. Write 0x04 to I2CONSET to set the AA bit.
121. Write 0x08 to I2CONCLR to clear the SI flag.
122. Exit

State : A0

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.

123. Write 0x04 to I2CONSET to set the AA bit.
124. Write 0x08 to I2CONCLR to clear the SI flag.
125. Exit

Slave Transmitter States

State : A8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

126. Load I2DAT from Slave Transmit buffer with first data byte.
127. Write 0x04 to I2CONSET to set the AA bit.
128. Write 0x08 to I2CONCLR to clear the SI flag.
129. Set up Slave Transmit mode data buffer.
130. Increment Slave Transmit buffer pointer.
131. Exit.

State : B0

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.

132. Load I2DAT from Slave Transmit buffer with first data byte.
133. Write 0x24 to I2CONSET to set the STA and AA bits.
134. Write 0x08 to I2CONCLR to clear the SI flag.
135. Set up Slave Transmit mode data buffer.
136. Increment Slave Transmit buffer pointer.
137. Exit.

State : B8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

138. Load I2DAT from Slave Transmit buffer with data byte.
139. Write 0x04 to I2CONSET to set the AA bit.
140. Write 0x08 to I2CONCLR to clear the SI flag.
141. Increment Slave Transmit buffer pointer.
142. Exit.

State : C0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

143. Write 0x04 to I2CONSET to set the AA bit.
144. Write 0x08 to I2CONCLR to clear the SI flag.
145. Exit

State : C8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

146. Write 0x04 to I2CONSET to set the AA bit.
147. Write 0x08 to I2CONCLR to clear the SI flag.
148. Exit

12. SPI INTERFACE (SPI0)

FEATURES

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

DESCRIPTION

SPI Overview

SPI is a full duplex serial interfaces. It can handle multiple masters and slaves being connected to a given bus. Only a single master and a single slave can communicate on the interface during a given data transfer. During a data transfer the master always sends a byte of data to the slave, and the slave always sends a byte of data to the master.

SPI Data Transfers

Figure 36 is a timing diagram that illustrates the four different data transfer formats that are available with the SPI. This timing diagram illustrates a single 8 bit data transfer. The first thing one should notice in this timing diagram is that it is divided into three horizontal parts. The first part describes the SCK and SSEL signals. The second part describes the MOSI and MISO signals when the CPHA variable is 0. The third part describes the MOSI and MISO signals when the CPHA variable is 1.

In the first part of the timing diagram, note two points. First, the SPI is illustrated wit CPOL set to both 0 and 1. The second point to note is the activation and de-activation of the SSEL signal. When CPHA = 1, the SSEL signal will always go inactive between data transfers. This is not guaranteed when CPHA = 0 (the signal can remain active).

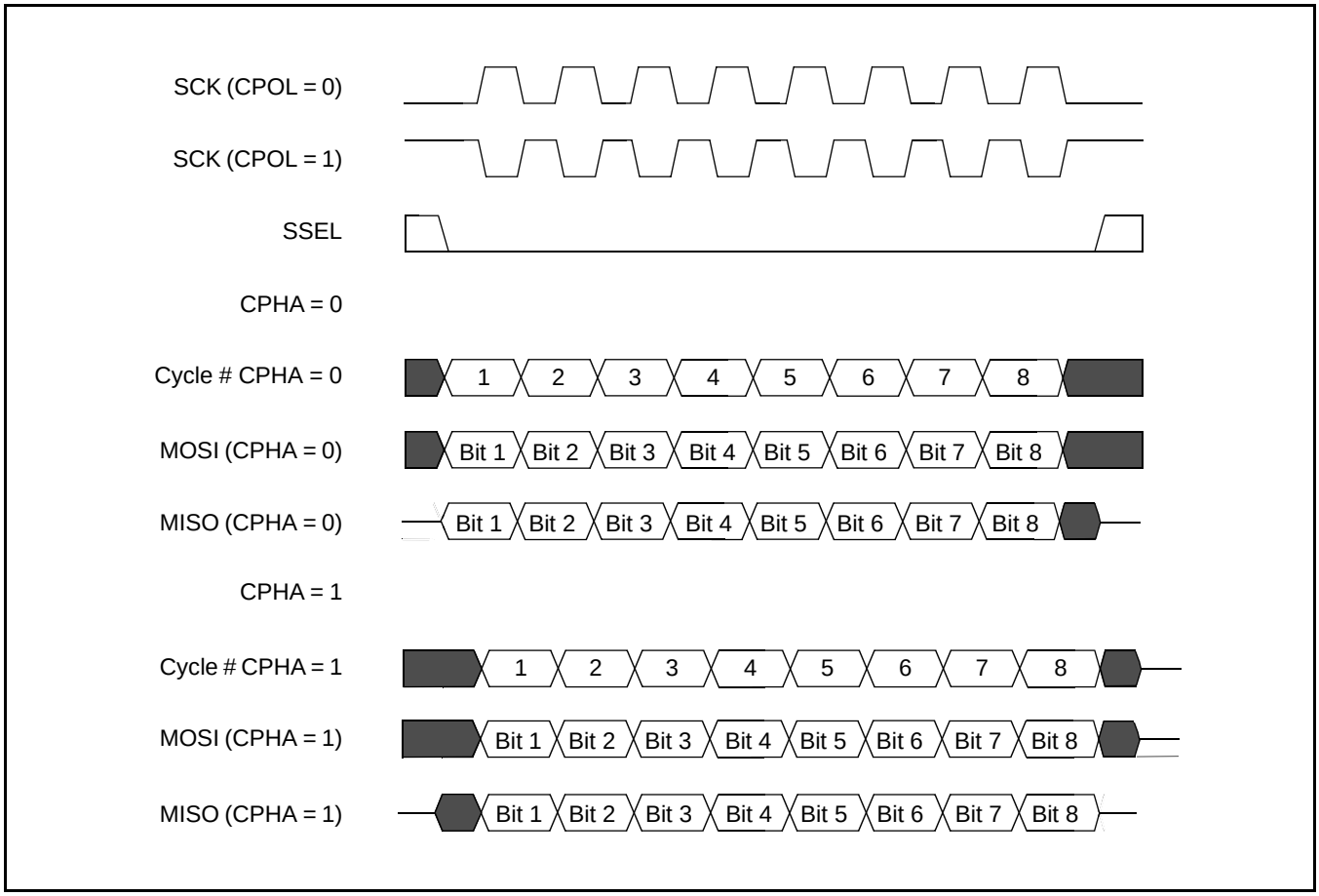


Figure 36: SPI Data Transfer Format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in Table 107. This table summarizes the following for each setting of CPOL and CPHA.

- When the first data bit is driven.
- When all other data bits are driven.
- When data is sampled.

Table 107: SPI Data To Clock Phase Relationship

CPOL And CPHA Settings	First Data Driven	Other Data Driven	Data Sampled
CPOL = 0, CPHA = 0	Prior to first SCK rising edge	SCK falling edge	SCK rising edge
CPOL = 0, CPHA = 1	First SCK rising edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 0	Prior to first SCK falling edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 1	First SCK falling edge	SCK falling edge	SCK rising edge

The definition of when an 8 bit transfer starts and stops is dependent on whether a device is a master or a slave, and the setting of the CPHA variable.

When a device is a master, the start of a transfer is indicated by the master having a byte of data that is ready to be transmitted. At this point, the master can activate the clock, and begin the transfer. The transfer ends when the last clock cycle of the transfer is complete.

When a device is a slave, and CPHA is set to 0, the transfer starts when the SSEL signal goes active, and ends when SSEL goes inactive. When a device is a slave, and CPHA is set to 1, the transfer starts on the first clock edge when the slave is selected, and ends on the last clock edge where data is sampled.

SPI Peripheral Details

General Information

There are four registers that control the SPI peripheral. They are described in detail in “Register Description” section.

The SPI control register contains a number of programmable bits used to control the function of the SPI block. The settings for this register must be set up prior to a given data transfer taking place.

The SPI status register contains read only bits that are used to monitor the status of the SPI interface, including normal functions, and exception conditions. The primary purpose of this register is to detect completion of a data transfer. This is indicated by the SPIF bit. The remaining bits in the register are exception condition indicators. These exceptions will be described later in this section.

The SPI data register is used to provide the transmit and receive data bytes. An internal shift register in the SPI block logic is used for the actual transmission and reception of the serial data. Data is written to the SPI data register for the transmit case. There is no buffer between the data register and the internal shift register. A write to the data register goes directly into the internal shift register. Therefore, data should only be written to this register when a transmit is not currently in progress. Read data is buffered. When a transfer is complete, the receive data is transferred to a single byte data buffer, where it is later read. A read of the SPI data register returns the value of the read data buffer.

The SPI clock counter register controls the clock rate when the SPI block is in master mode. This needs to be set prior to a transfer taking place, when the SPI block is a master. This register has no function when the SPI block is a slave.

The I/Os for this implementation of SPI are standard CMOS I/Os. The open drain SPI option is not implemented in this design. When a device is set up to be a slave, its I/Os are only active when it is selected by the SSEL signal being active.

Master Operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be the master. This process assumes that any prior data transfer has already completed.

1. Set the SPI clock counter register to the desired clock rate.
2. Set the SPI control register to the desired settings.
3. Write the data to transmitted to the SPI data register. This write starts the SPI data transfer.
4. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
5. Read the SPI status register.
6. Read the received data from the SPI data register (optional).
7. Go to step 3 if more data is required to transmit.

Note that a read or write of the SPI data register is required in order to clear the SPIF status bit. Therefore, if the optional read of the SPI data register does not take place, a write to this register is required in order to clear the SPIF status bit.

Slave Operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be a slave. This process assumes that any prior data transfer has already completed. It is required that the system clock driving the SPI logic be at least 8X faster than the SPI.

1. Set the SPI control register to the desired settings.
2. Write the data to transmitted to the SPI data register (optional). Note that this can only be done when a slave SPI transfer is not in progress.
3. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last sampling clock edge of the SPI data transfer.
4. Read the SPI status register.
5. Read the received data from the SPI data register (optional).
6. Go to step 2 if more data is required to transmit.

Note that a read or write of the SPI data register is required in order to clear the SPIF status bit. Therefore, at least one of the optional reads or writes of the SPI data register must take place, in order to clear the SPIF status bit.

Exception Conditions

Read Overrun - A read overrun occurs when the SPI block internal read buffer contains data that has not been read by the processor, and a new transfer has completed. The read buffer containing valid data is indicated by the SPIF bit in the status register being active. When a transfer completes, the SPI block needs to move the received data to the read buffer. If the SPIF bit is active (the read buffer is full), the new receive data will be lost, and the read overrun (ROVR) bit in the status register will be activated.

Write Collision - As stated previously, there is no write buffer between the SPI block bus interface, and the internal shift register. As a result, data must not be written to the SPI data register when a SPI data transfer is currently in progress. The time frame where data cannot be written to the SPI data register is from when the transfer starts, until after the status register has been read when the SPIF status is active. If the SPI data register is written in this time frame, the write data will be lost, and the write collision (WCOL) bit in the status register will be activated.

Mode Fault - The SSEL signal must always be inactive when the SPI block is a master. If the SSEL signal goes active, when the SPI block is a master, this indicates another master has selected the device to be a slave. This condition is known as a mode fault. When a mode fault is detected, the mode fault (MODF) bit in the status register will be activated, the SPI signal drivers will be de-activated, and the SPI mode will be changed to be a slave.

Slave Abort - A slave transfer is considered to be aborted, if the SSEL signal goes inactive before the transfer is complete. In the event of a slave abort, the transmit and receive data for the transfer that was in progress are lost, and the slave abort (ABRT) bit in the status register will be activated.

PIN DESCRIPTION

Table 108: SPI Pin Description

Pin Name	Type	Pin Description
SCK0	Input/ Output	**Serial Clock.** The SPI is a clock signal used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL0	Input	**Slave Select.** The SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. Each slave has its own unique slave select signal input. The SSEL must be low before data transactions begin and normally stays low for the duration of the transaction. If the SSEL signal goes high any time during a data transfer, the transfer is considered to be aborted. In this event, the slave returns to idle, and any data that was received is thrown away. There are no other indications of this exception. This signal is not directly driven by the master. It could be driven by a simple general purpose I/O under software control. **On the LPC2131/2132/2138 (unlike earlier Philips ARM devices) the SSEL0 pin can be used for a different function when the SPI0 interface is only used in Master mode. For example, pin hosting the SSEL0 function can be configured as an output digital GPIO pin and used to select one of the SPI0 slaves.**
MISO0	Input/ Output	**Master In Slave Out.** The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. When a device is a slave, serial data is output on this signal. When a device is a master, serial data is input on this signal. When a slave device is not selected, the slave drives the signal high impedance.
MOSI0	Input/ Output	**Master Out Slave In.** The MOSI signal is a unidirectional signal used to transfer serial data from the master to the slave. When a device is a master, serial data is output on this signal. When a device is a slave, serial data is input on this signal.

REGISTER DESCRIPTION

The SPI contains 5 registers as shown in Table 109. All registers are byte, half word and word accessible.

Table 109: SPI Register Map

Generic Name	Description	Access	Reset Value*	SPI0 Address & Name
SPCR	SPI Control Register. This register controls the operation of the SPI.	Read/ Write	0	0xE0020000 S0SPCR
SPSR	SPI Status Register. This register shows the status of the SPI.	Read Only	0	0xE0020004 S0SPSR
SPDR	SPI Data Register. This bi-directional 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.	Read/ Write	0	0xE0020008 S0SPDR
SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK.	Read/ Write	0	0xE002000C S0SPCCR
SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	Read/ Write	0	0xE002001C S0SPINT

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

SPI Control Register (S0SPCR - 0xE0020000)

The SPCR register controls the operation of the SPI as per the configuration bits setting.

Table 110: SPI Control Register (S0SPCR - 0xE0020000)

SPCR	Function	Description	Reset Value
2:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
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. When 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. When 0, data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	0
4	CPOL	Clock polarity control. When 1, SCK is active low. When 0, SCK is active high.	0
5	MSTR	Master mode select. When 1, the SPI operates in Master mode. When 0, the SPI operates in Slave mode.	0
6	LSBF	LSB First controls which direction each byte is shifted when transferred. When 1, SPI data is transferred LSB (bit 0) first. When 0, SPI data is transferred MSB (bit 7) first.	0
7	SPIE	Serial peripheral interrupt enable. When 1, a hardware interrupt is generated each time the SPIF or MODF bits are activated. When 0, SPI interrupts are inhibited.	0

SPI Status Register (S0SPSR - 0xE0020004)

The SPSR register controls the operation of the SPI as per the configuration bits setting.

Table 111: SPI Status Register (S0SPSR - 0xE0020004)

SPSR	Function	Description	Reset Value
2:0	Reserved	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 SPI 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

SPI Data Register (S0SPDR - 0xE0020008)

This bi-directional data register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register. When a master, a write to this register will start a SPI data transfer. Writes to this register will be blocked from when a data transfer starts to when the SPIF status bit is set, and the status register has not been read.

Table 112: SPI Data Register (S0SPDR - 0xE0020008)

SPDR	Function	Description	Reset Value
7:0	Data	SPI Bi-directional data port	0

SPI Clock Counter Register (S0SPCCR - 0xE002000C)

This register controls the frequency of a master's SCK. The register indicates the number of pclk cycles that make up an SPI clock. The value of this register must always be an even number. As a result, bit 0 must always be 0. The value of the register must also always be greater than or equal to 8. Violations of this can result in unpredictable behavior.

Table 113: SPI Clock Counter Register (S0SPCCR - 0xE002000C)

SPCCR	Function	Description	Reset Value
7:0	Counter	SPI Clock counter setting	0

The SPI rate may be calculated as: PCLK rate / SPCCR value. The pclk rate is CCLK / VPB divider rate as determined by the VPBDIV register contents.

SPI Interrupt Register (S0SPINT - 0xE002001C)

This register contains the interrupt flag for the SPI interface.

Table 114: SPI Interrupt Register (S0SPINT - 0xE002001C)

SPINT	Function	Description	Reset Value
0	SPI Interrupt	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 SPI Interrupt bit is set and SPI Interrupt is enabled in the VIC, SPI based interrupt can be processed by interrupt handling software.	0
7:1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

ARCHITECTURE

The block diagram of the SPI solution implemented in SPI0 interface is shown in the Figure 37.

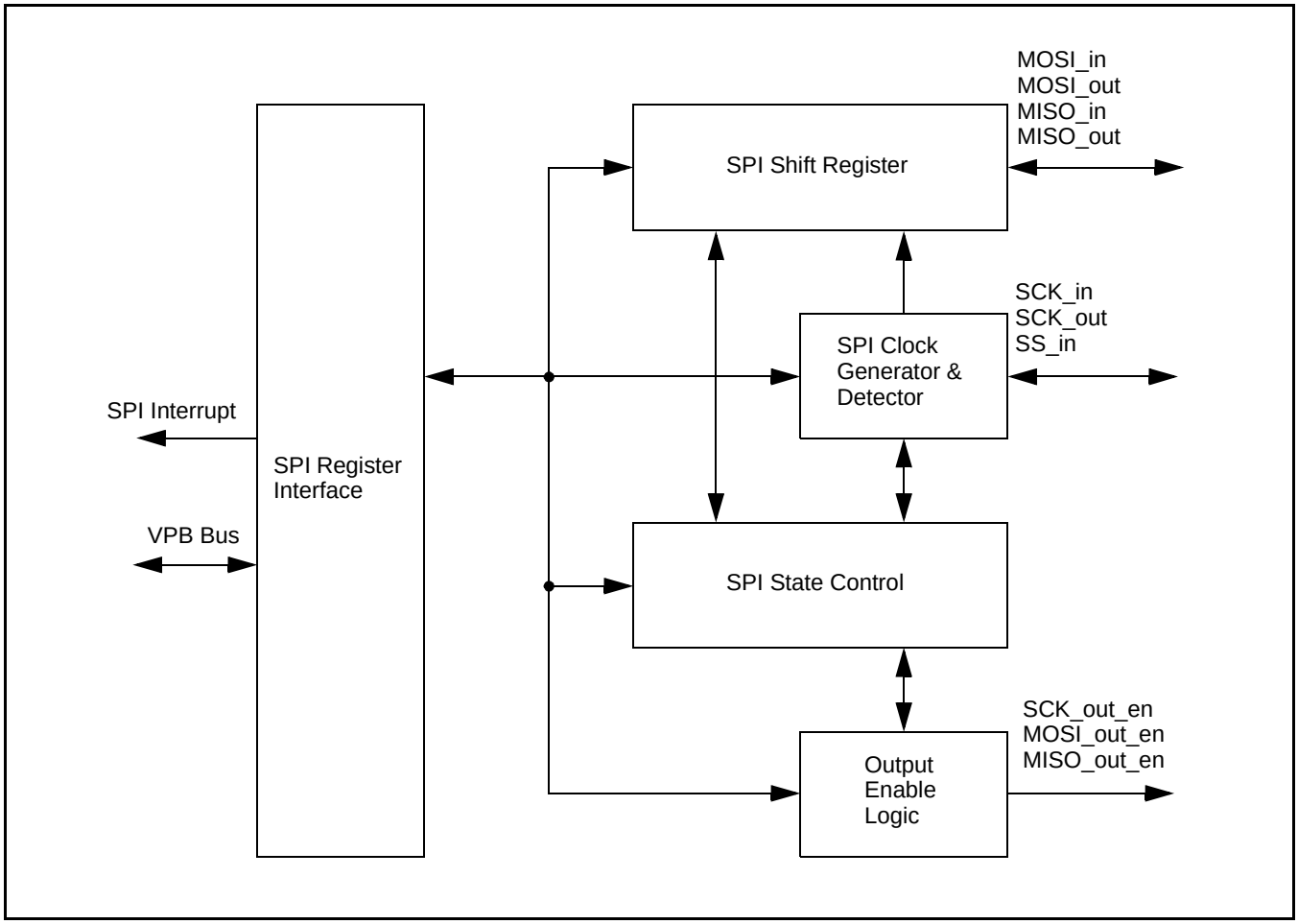


Figure 37: SPI Block Diagram

13. SSP CONTROLLER (SPI1)

FEATURES

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses.
- Synchronous Serial Communication.
- Master or slave operation.
- 8-frame FIFOs for both transmit and receive.
- 4 to 16 bits per frame

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.

PIN DESCRIPTIONS

Pin Name	Type	Interface pin name/function			Pin Description
		SPI	SSI	Microwire	
SCK1	I/O	SCK	CLK	SK	**Serial Clock.** SCK/CLK/SK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When SPI interface is used the clock is programmable to be active high or active low, otherwise it is always active high. SCK1 only switches during a data transfer. Any other time, the SSP either holds it in its inactive state, or does not drive it (leaves it in high impedance state).
SSEL1	I/O	SSEL	FS	CS	**Slave Select/Frame Sync/Chip Select.** When the SSP is a bus master, it drives this signal from shortly before the start of serial data, to shortly after the end of serial data, to signify a data transfer as appropriate for the selected bus and mode. When the SSP is a bus slave, this signal qualifies the presence of data from the Master, according to the protocol in use. When there is just one bus master and one bus slave, the Frame Sync or Slave Select signal from the Master can be connected directly to the slave's corresponding input. When there is more than one slave on the bus, further qualification of their Frame Select/Slave Select inputs will typically be necessary to prevent more than one slave from responding to a transfer.
MISO1	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**Master In Slave Out.** The MISO signal transfers serial data from the slave to the master. When the SSP is a slave, serial data is output on this signal. When the SSP is a master, it clocks in serial data from this signal. When the SSP is a slave and is not selected by SSEL, it does not drive this signal (leaves it in high impedance state).
MOSI1	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**Master Out Slave In.** The MOSI signal transfers serial data from the master to the slave. When the SSP is a master, it outputs serial data on this signal. When the SSP is a slave, it clocks in serial data from this signal.

Table 115: SSP Pin Descriptions

TEXAS INSTRUMENTS SYNCHRONOUS SERIAL FRAME FORMAT

Figure 38 shows the 4-wire Texas Instruments synchronous serial frame format supported by the SSP module.

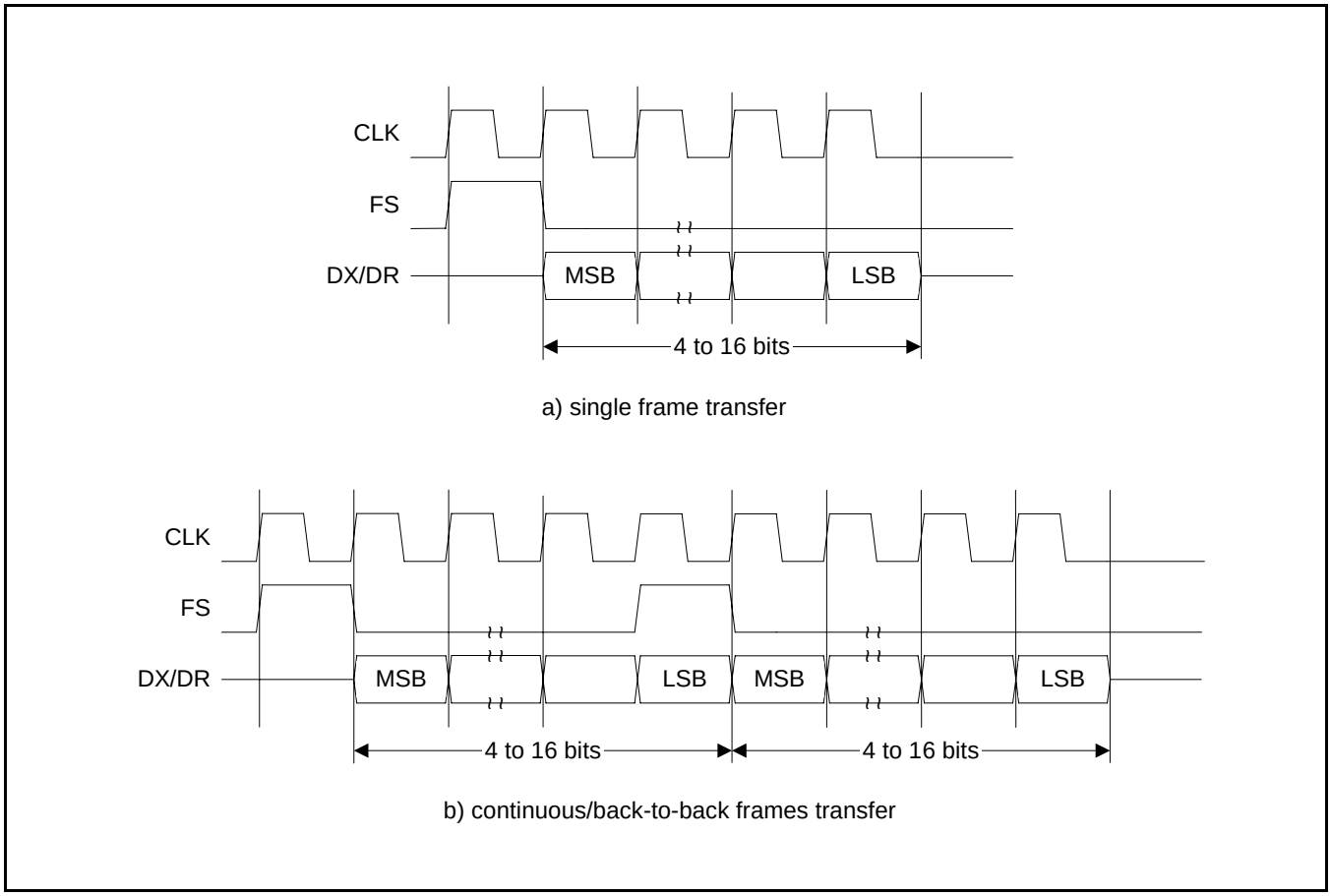


Figure 38: Texas Instruments synchronous serial frame format: a) single and b) continuous/back-to-back two frames transfer

For device configured as a master in this mode, CLK and FS are forced LOW, and the transmit data line DX is tristated whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, FS is pulsed HIGH for one CLK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of CLK, the MSB of the 4 to 16-bit data frame is shifted out on the DX pin. Likewise, the MSB of the received data is shifted onto the DR pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each CLK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of CLK after the LSB has been latched.

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.

Clock Polarity (CPOL)

When the CPOL clock polarity control bit is LOW, it produces a steady state low value on the SCK pin. If the CPOL clock polarity control bit is HIGH, a steady state high value is placed on the CLK pin when data is not being transferred.

Clock Phase (CPHA)

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

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 39.

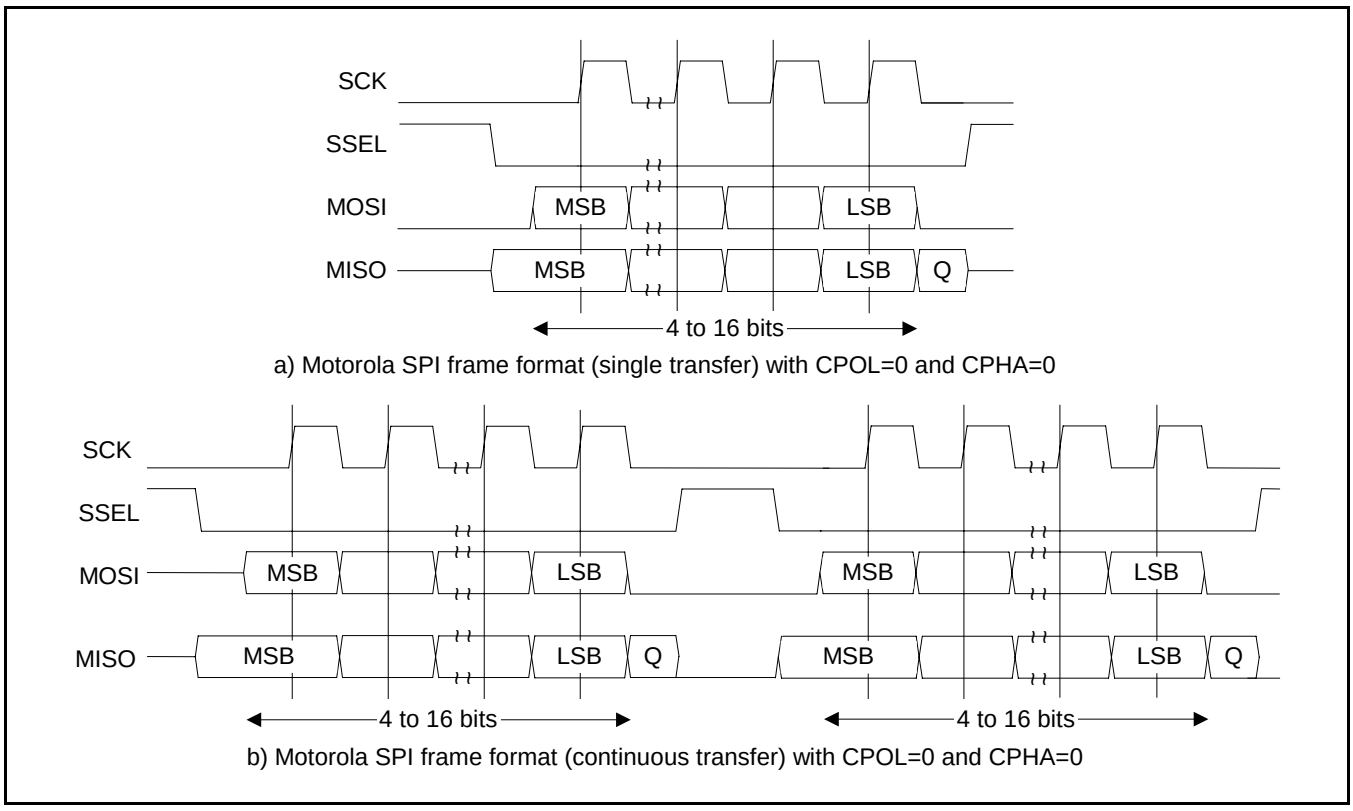


Figure 39: SPI frame format with CPOL=0 and CPHA=0 (a) single and b) continuous transfer)

In this configuration, during idle periods:

- the CLK signal is forced LOW
- SSEL is forced HIGH
- the transmit MOSI/MISO pad is in high impedance

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. This causes slave data to be enabled onto the MISO input line of the master. Master's MOSI is enabled.

One half SCK period later, valid master data is transferred to the MOSI pin. Now that both the master and slave data have been set, the SCK master clock pin goes HIGH after one further half SCK period.

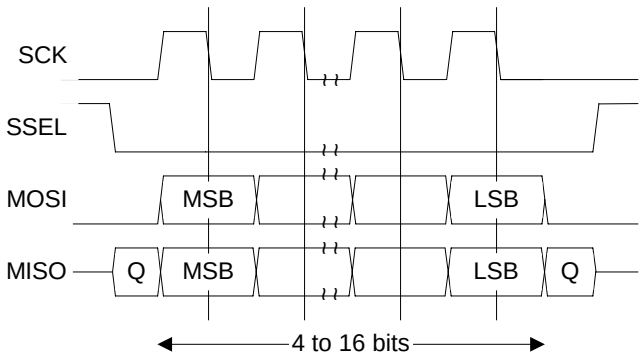
The data is now captured on the rising and propagated on the falling edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

SPI Format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL=0, CPHA=1 is shown in Figure 40, which covers both single and continuous transfers.



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

Figure 40: SPI frame format with CPOL=0 and CPHA=1

In this configuration, during idle periods:

- the SCK signal is forced LOW
- SSEL is forced HIGH
- the transmitting MOSI/MISO pin 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.

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 41.

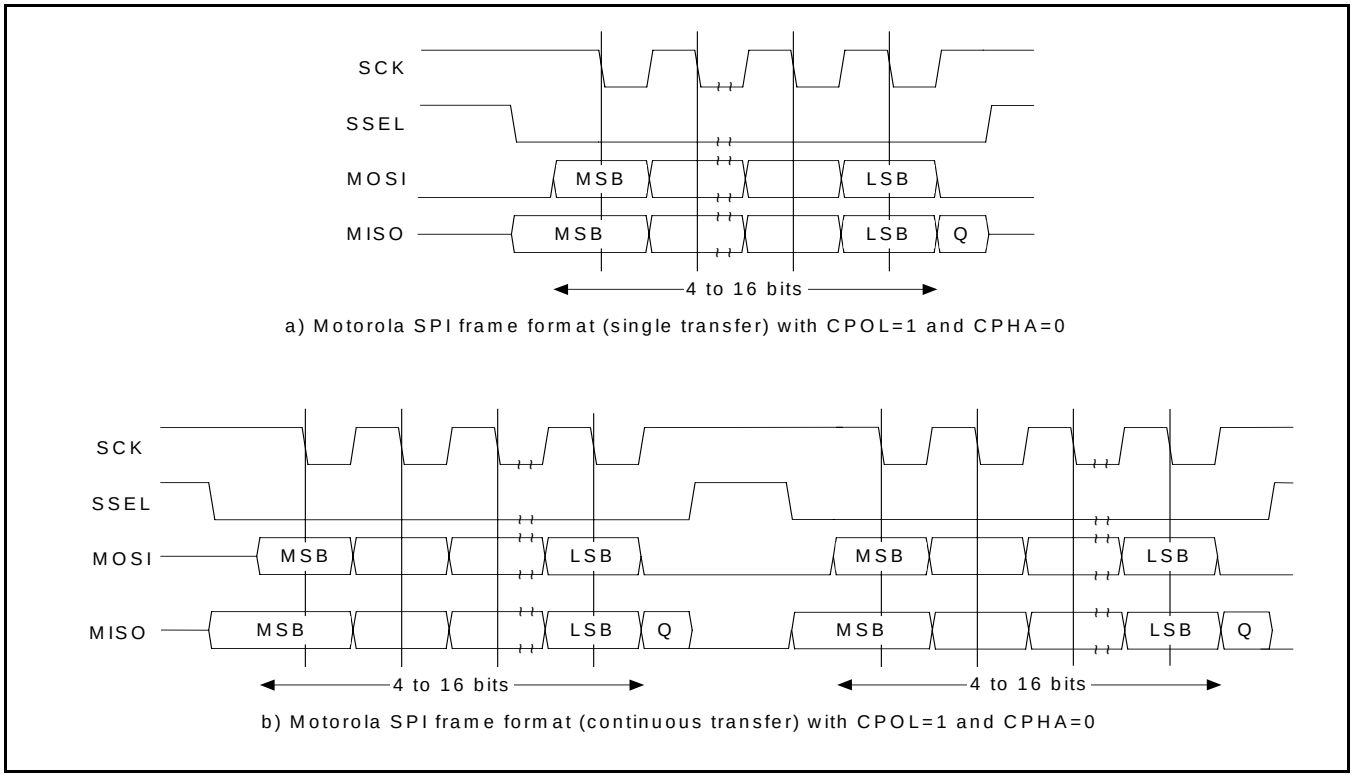


Figure 41: SPI frame format with CPOL=1 and CPHA=0 (a) single and b) continuous transfer)

In this configuration, during idle periods

- the SCK signal is forced HIGH
- SSEL is forced HIGH
- the transmitting pin MOSI/MISO 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.

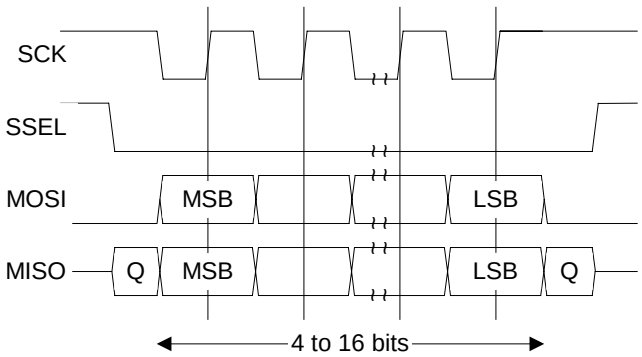
One half period later, valid master data is transferred to the MOSI line. Now that both the master and slave data have been set, the SCK master clock pin becomes LOW after one further half SCK period. This means that data is captured on the falling edges and be propagated on the rising edges of the SCK 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.

SPI Format with CPOL=1,CPHA=1

The transfer signal sequence for SPI format with CPOL=1, CPHA=1 is shown in Figure 42, which covers both single and continuous transfers.



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

Figure 42: SPI frame format with CPOL=1 and CPHA=1

In this configuration, during idle periods:

- the SCK signal is forced HIGH
- SSEL is forced HIGH
- the transmitting MOSI/MISO pins are 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.

SEMICONDUCTOR MICROWIRE FRAME FORMAT

Figure 43 shows the Microwire frame format for a single frame. Figure 44 shows the same format when back-to-back frames are transmitted.

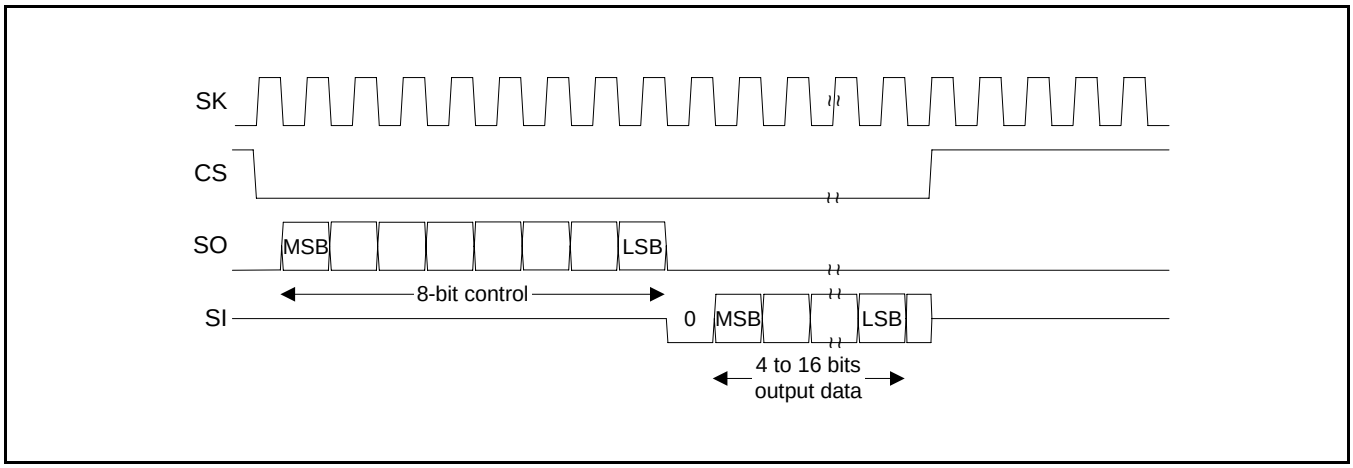


Figure 43: Microwire frame format (single transfer)

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.

.

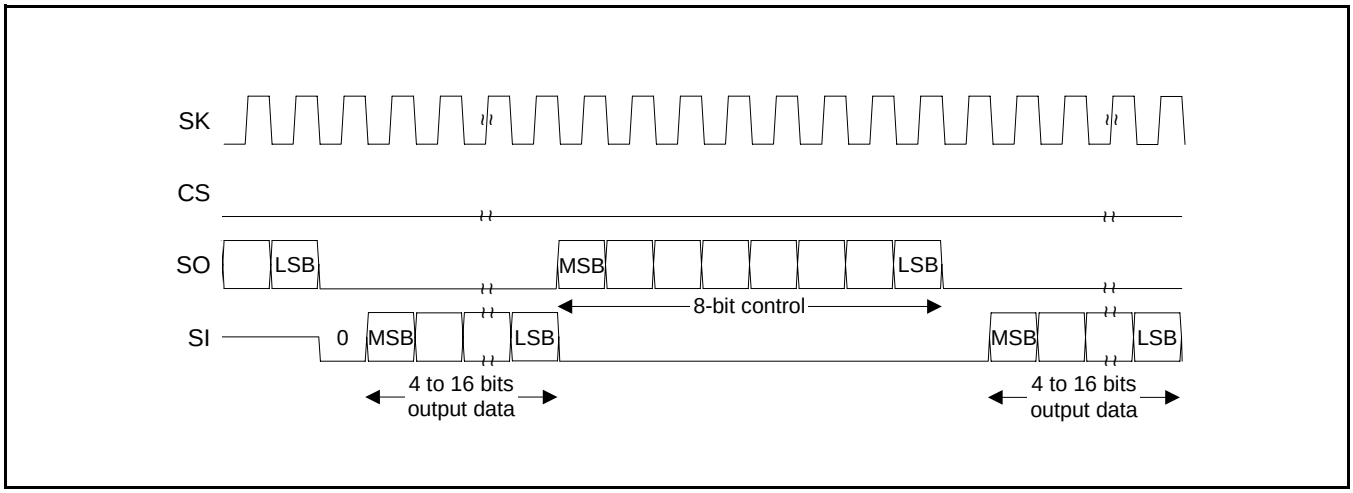


Figure 44: Microwire frame format (continuos transfers)

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 45 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.

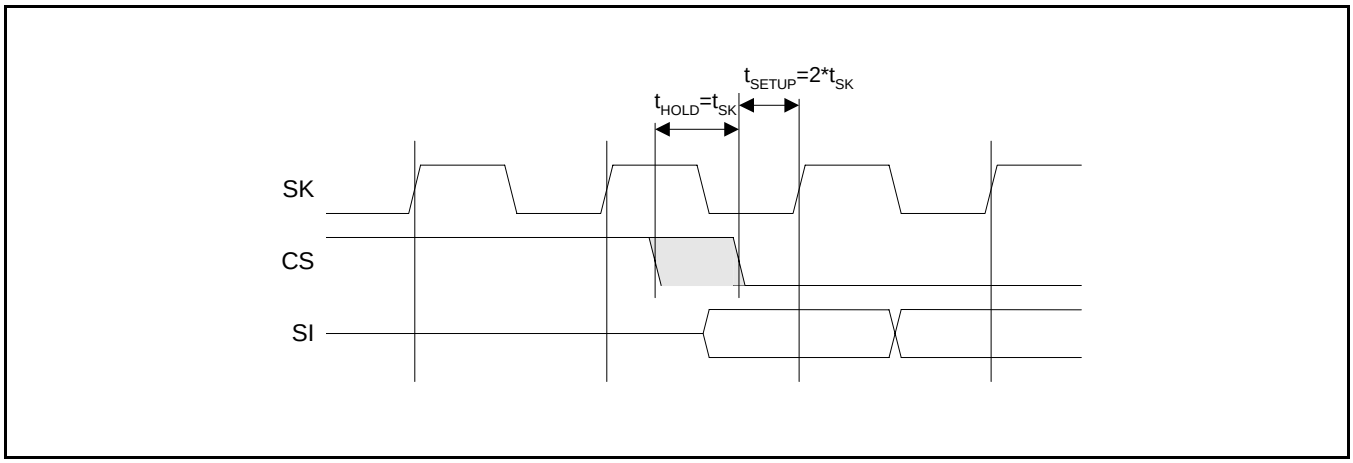


Figure 45: Microwire frame format (continuos transfers)

REGISTER DESCRIPTIONS

The base address of the SSP controller is 0xE006 8000. Its registers' offsets from this base are shown in the following table.

Table 116: SSP Registers

Name	Description	Access	Address
SSPCR0	Control Register 0. Selects the serial clock rate, bus type, and data size.	Read/Write	0xE0068000
SSPCR1	Control Register 1. Selects master/slave and other modes.	Read/Write	0xE0068004
SSPDR	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	Read/Write	0xE0068008
SSPSR	Status Register.	Read Only	0xE006800C
SSPCPSR	Clock Prescale Register.	Read/Write	0xE0068010
SSPIMSC	Interrupt Mask Set and Clear Register	Read/Write	0xE0068014
SSPRIS	Raw Interrupt Status Register.	Read/Write	0xE0068018
SSPMIS	Masked Interrupt Status Register	Read/Write	0xE006801C
SSPICR	Interrupt Clear Register	Read/Write	0xE0068020

SSP Control Register 0 (SSPCR0 - 0xE0068000)

This register controls the basic operation of the SSP controller.

Table 117: SSP Control Register 0 (SSPCR0 - 0xE0068000)

SSPCR0	Name	Description	Reset Value
15:8	SCR	Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVR is the prescale divider, and the VPB clock PCLK clocks the prescaler, the bit frequency is f(PCLK) / (CPSDVSR * (SCR+1)).	0
7	CPHA	Clock Out Phase. This bit is only used in SPI mode. If it is 0, the SSP controller maintains the bus clock low between frames, while if its 1, the SSP controller maintains the bus clock high between frames	0
6	CPOL	Clock Out Polarity. This bit is only used in SPI mode. If it is 0, the 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. If this bit is 1, the SSP controller captures serial data on the second clock transition of the frame, that is, the transition back to the inter-frame state of the clock line.	0
5:4	FRF	Frame Format. 00 = SPI, 01 = SSI, 10 = Microwire, 11 = reserved.	0
3:0	DSS	Data Size Select. This field controls the number of bits transferred in each frame: 0000-0010=reserved, 0011=4 bits, ..., 0111=8 bits, ..., 1111=16 bits.	0

SSP Control Register 1 (SSPCR1 - 0xE0068004)

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

Table 118: SSP Control Register 1 (SSPCR1 - 0xE0068004)

SSPCR1	Name	Description	Reset Value
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
2	MS	Master/Slave Mode. A 0 in this bit (as after Reset) makes the SSP controller act as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line. A 1 in this bit makes the SSP controller act as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines. This bit can only be written when the SSE bit is 0. (One can set MS and SSE to 1 simultaneously).	0
1	SSE	SSP Enable. When this bit is 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.	0
0	LBM	Loop Back Mode. This bit should be 0 for normal operation. When it is 1, serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively).	0

SSP Data Register (SSPDR - 0xE0068008)

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

Table 119: SSP Data Register (SSPDR - 0xE0068008)

SSPDR	Name	Description	Reset Value
15:0	DATA	**Write:** software can write data to be sent in a future frame to this register whenever the TNF bit in the Status register is 1, indicating that the Tx FIFO is not full. If the Tx FIFO was previously empty and the SSP controller is not busy on the bus, transmission of the data will begin immediately. Otherwise the data written to this register will be sent as soon as all previous data has been sent (and received). If the data length is less than 16 bits, software must right-justify the data written to this register. **Read:** software can read data from this register whenever the RNE bit in the Status register is 1, indicating that the Rx FIFO is not empty. When software reads this register, the SSP controller returns data from the least recent frame in the Rx FIFO. If the data length is less than 16 bits, the data is right-justified in this field with higher order bits filled with 0s.	0

SSP Status Register (SSPSR - 0xE006800C)

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

Table 120: SSP Status Register (SSPSR - 0xE006800C)

SSPSR	Name	Description	Reset Value
4	BSY	Busy. This bit is 0 if the SSP controller is idle, or 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1

SSP Clock Prescale Register (SSPCPSR - 0xE0068010)

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

Table 121: SSP Clock Prescale Register (SSPCPSR - 0xE0068010)

SSPCPSR	Name	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which PCLK is divided to yield the prescaler-output clock. Bit 0 always reads as 0.	0

SSP Interrupt Mask Set/Clear Register (SSPIMSC - 0xE0068014)

This register controls whether each of the four possible interrupt conditions in the SSP controller are enabled. Note that ARM uses the word “masked” in the opposite sense from classic computer terminology, in which “masked” meant “disabled”. ARM uses the word “masked” to mean “enabled”. To avoid confusion we will not use the word “masked”.

Table 122: SSP Interrupt Mask Set/Clear Register (SSPIMSC - 0xE0068014)

SSPIMSC	Name	Description	Reset Value
3	TXIM	Software should set this bit to enable interrupt when the Tx FIFO is at least half empty.	0
2	RXIM	Software should set this bit to enable interrupt when the Rx FIFO is at least half full.	0
1	RTIM	Software should set this bit to enable interrupt when a Receive Timeout condition occurs. A Receive Timeout occurs when the Rx FIFO is not empty, and no new data has been received, nor has data been read from the FIFO, for 32 bit times.	0
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

SSP Raw Interrupt Status Register (SSPRIS - 0xE0068018)

This read-only register contains a 1 for each interrupt condition that is asserted, regardless of whether or not the interrupt is enabled in the SSPIMSC.

Table 123: SSP Raw Interrupt Status Register (SSPRIS - 0xE0068018)

SSPRIS	Name	Description	Reset Value
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
1	RTRIS	This bit is 1 if when there is a Receive Timeout condition. Note that a Receive Timeout can be negated if further data is received.	0
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

SSP Masked Interrupt Status Register (SSPMIS - 0xE006801C)

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

Table 124: SSP Masked Interrupt Status Register (SSPMIS - 0xE006801C)

SSPMIS	Name	Description	Reset Value
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, 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
1	RTMIS	This bit is 1 when there is a Receive Timeout condition and this interrupt is enabled. Note that a Receive Timeout can be negated if further data is received.	0
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0

SSP Interrupt Clear Register (SSPICR - 0xE0068020)

Software can write one or more one(s) to this write-only register, to clear the corresponding interrupt condition(s) in the SSP controller. Note that the other two interrupt conditions can be cleared by writing or reading the appropriate FIFO, or disabled by clearing the corresponding bit in SSPIMSC.

Table 125: SSP Interrupt Clear Register (SSPICR - 0xE0068020)

SSPICR	Name	Description	Reset Value
1	RTIC	Writing a 1 to this bit clears the Receive Timeout interrupt.	NA
0	RORIC	Writing a 1 to this bit clears the “frame was received when RxFIFO was full” interrupt.	NA

14. TIMER/COUNTER0 AND TIMER/COUNTER1

Timer/Counter0 and Timer/Counter1 are functionally identical except for the peripheral base address.

FEATURES

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Counter or Timer operation.
- Up to four 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to four external outputs corresponding to match registers, with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.

APPLICATIONS

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

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.

PIN DESCRIPTION

Table 126 gives a brief summary of each of the Timer related pins.

Table 126: Pin summary

Pin name	Pin direction	Pin Description
CAP0.3..0 CAP1.3..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. If for example pins 30 (P0.6) and 46 (P0.16) are selected for CAP0.2, only pin 30 will be used by TIMER0 to perform CAP0.2 function. Here is the list of all CAPTURE signals, together with pins on where they can be selected: CAP0.0 (3 pins) : P0.2, P0.22 and P0.30 CAP0.1 (2 pins) : P0.4 and P0.27 CAP0.2 (3 pin) : P0.6, P0.16 and P0.28 CAP0.3 (1 pin): : P0.29 CAP1.0 (1 pin): : P0.10 CAP1.1 (1 pin): : P0.11 CAP1.2 (2 pins) : P0.17 and P0.19 CAP1.3 (2 pins) : P0.18 and P0.21 Timer/Counter block can select a capture signal as a clock source instead of the pclk derived clock. For more details see Count Control Register (CTCR: TIMER0 - T0CTCR: 0xE0004070; TIMER1 - T1TCR: 0xE0008070) on page 178
MAT0.3...0 MAT1.0...0	Output	**External Match Output 0/1-** When a match register 0/1 (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel. It is also possible for example, to have 2 pins selected at the same time so that they provide MAT1.3 function in parallel. Here is the list of all MATCH signals, together with pins on where they can be selected: MAT0.0 (2 pins) : P0.3 and P0.22 MAT0.1 (2 pins) : P0.5 and P0.27 MAT0.2 (2 pin) : P0.16 and P0.28 MAT0.3 (1 pin): : P0.29 MAT1.0 (1 pin): : P0.12 MAT1.1 (1 pin): : P0.13 MAT1.2 (2 pins) : P0.17 and P0.19 MAT1.3 (2 pins) : P0.18 and P0.20

REGISTER DESCRIPTION

Each Timer contains the registers shown in Table 127 ("Reset Value" refers to the data stored in used bits only; it does not include reserved bits content). More detailed descriptions follow.

Table 127: TIMER0 and TIMER1 Register Map

Generic Name	Description	Access	Reset Value*	TIMER0 Address & Name	TIMER1 Address & Name
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	0xE0004000 T0IR	0xE0008000 T1IR
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	0xE0004004 T0TCR	0xE0008004 T1TCR
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0	0xE0004008 T0TC	0xE0008008 T1TC
PR	Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0	0xE000400C T0PR	0xE000800C T1PR
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented.	R/W	0	0xE0004010 T0PC	0xE0008010 T1PC
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	0xE0004014 T0MCR	0xE0008014 T1MCR
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	0xE0004018 T0MR0	0xE0008018 T1MR0
MR1	Match Register 1. See MR0 description.	R/W	0	0xE000401C T0MR1	0xE000801C T1MR1
MR2	Match Register 2. See MR0 description.	R/W	0	0xE0004020 T0MR2	0xE0008020 T1MR2
MR3	Match Register 3. See MR0 description.	R/W	0	0xE0004024 T0MR3	0xE0008024 T1MR3
CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0	0xE0004028 T0CCR	0xE0008028 T1CCR
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAP0.0(CAP1.0) input.	RO	0	0xE000402C T0CR0	0xE000802C T1CR0
CR1	Capture Register 1. See CR0 description.	RO	0	0xE0004030 T0CR1	0xE0008030 T1CR1
CR2	Capture Register 2. See CR0 description.	RO	0	0xE0004034 T0CR2	0xE0008034 T1CR2
CR3	Capture Register 3. See CR0 description.	RO	0	0xE0004038 T0CR3	0xE0008038 T1CR3
EMR	External Match Register. The EMR controls the external match pins MAT0.0-3 (MAT1.0-3).	R/W	0	0xE000403C T0EMR	0xE000803C T1EMR
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	0xE0004070 T0CTCR	0xE0008070 T1CTCR

Interrupt Register (IR: TIMER0 - T0IR: 0xE0004000; TIMER1 - T1IR: 0xE0008000)

The Interrupt Register consists of four bits for the match interrupts and four bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 128: Interrupt Register (IR: TIMER0 - T0IR: 0xE0004000; TIMER1 - T1IR: 0xE0008000)

IR	Function	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
6	CR2 Interrupt	Interrupt flag for capture channel 2 event.	0
7	CR3 Interrupt	Interrupt flag for capture channel 3 event.	0

Timer Control Register (TCR: TIMER0 - T0TCR: 0xE0004004; TIMER1 - T1TCR: 0xE0008004)

The Timer Control Register (TCR) is used to control the operation of the Timer Counter.

Table 129: Timer Control Register (TCR: TIMER0 - T0TCR: 0xE0004004; TIMER1 - T1TCR: 0xE0008004)

TCR	Function	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0

Count Control Register (CTCR: TIMER0 - T0CTCR: 0xE0004070; TIMER1 - T1TCR: 0xE0008070)

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the pclk clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event corresponds to the one selected by bits 1:0 in the CTCR register, the Timer Counter register will be incremented.

Effective processing of the externaly supplied clock to the counter has some limitations. Since two successive rising edges of the pclk clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input can not exceed one half of the pclk clock. Consequently, duration of the high/low levels on the same CAP input in this case can not be shorter than $1/f_{pclk}$.

Table 130: Count Control Register (CTCR: TIMER0 - T0CTCR: 0xE0004070; TIMER1 - T1TCR: 0xE0008070)

CTCR	Function	Description	Reset Value
1:0	Counter/ Timer Mode	This field selects which rising PCLK edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC): 00: Timer Mode: 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.	00
3:2	Count Input Select	When bits 1:0 are not 00, these bits select which CAP pin is sampled for clocking: 00 = CAPn.0 01 = CAPn.1 10 = CAPn.2 11 = CAPn.3 **Note:** If Counter mode is selected for a particular CAPn input in the TnCTCR, the 3 bits for that input in the Capture Control Register (TnCCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	00

Timer Counter (TC: TIMER0 - T0TC: 0xE0004008; TIMER1 - T1TC: 0xE0008008)

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFFFFFF and then wrap back to the value 0x00000000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Prescale Register (PR: TIMER0 - T0PR: 0xE000400C; TIMER1 - T1PR: 0xE000800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

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.

Match Control Register (MCR: TIMER0 - T0MCR: 0xE0004014; TIMER1 - T1MCR: 0xE0008014)

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 131.

Table 131: Match Control Register (MCR: TIMER0 - T0MCR: 0xE0004014; TIMER1 - T1MCR: 0xE0008014)

MCR	Function	Description	Reset Value
0	Interrupt on MR0	When one, an interrupt is generated when MR0 matches the value in the TC. When zero this interrupt is disabled.	0
1	Reset on MR0	When one, the TC will be reset if MR0 matches it. When zero this feature is disabled.	0
2	Stop on MR0	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. When zero this feature is disabled.	0
3	Interrupt on MR1	When one, an interrupt is generated when MR1 matches the value in the TC. When zero this interrupt is disabled.	0
4	Reset on MR1	When one, the TC will be reset if MR1 matches it. When zero this feature is disabled.	0
5	Stop on MR1	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. When zero this feature is disabled.	0
6	Interrupt on MR2	When one, an interrupt is generated when MR2 matches the value in the TC. When zero this interrupt is disabled.	0
7	Reset on MR2	When one, the TC will be reset if MR2 matches it. When zero this feature is disabled.	0
8	Stop on MR2	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. When zero this feature is disabled.	0
9	Interrupt on MR3	When one, an interrupt is generated when MR3 matches the value in the TC. When zero this interrupt is disabled.	0
10	Reset on MR3	When one, the TC will be reset if MR3 matches it. When zero this feature is disabled.	0
11	Stop on MR3	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. When zero this feature is disabled.	0

Capture Registers (CR0 - CR3)

Each Capture register is associated with a device pin and may be loaded with the Timer Counter value when a specified event occurs on that pin. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

Capture Control Register (CCR: TIMER0 - T0CCR: 0xE0004028; TIMER1 - T1CCR: 0xE0008028)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the Timer number, 0 or 1.

Table 132: Capture Control Register (CCR: TIMER0 - T0CCR: 0xE0004028; TIMER1 - T1CCR: 0xE0008028)

CCR	Function	Description	Reset Value
0	Capture on CAPn.0 rising edge	When one, a sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of the TC. When zero, this feature is disabled.	0
1	Capture on CAPn.0 falling edge	When one, a sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC. When zero, this feature is disabled.	0

Table 132: Capture Control Register (CCR: TIMER0 - T0CCR: 0xE0004028; TIMER1 - T1CCR: 0xE0008028)

CCR	Function	Description	Reset Value
2	Interrupt on CAPn.0 event	When one, a CR0 load due to a CAPn.0 event will generate an interrupt. When zero, this feature is disabled.	0
3	Capture on CAPn.1 rising edge	When one, a sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of the TC. When zero, this feature is disabled.	0
4	Capture on CAPn.1 falling edge	When one, a sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC. When zero, this feature is disabled.	0
5	Interrupt on CAPn.1 event	When one, a CR1 load due to a CAPn.1 event will generate an interrupt. When zero, this feature is disabled.	0
6	Capture on CAPn.2 rising edge	When one, a sequence of 0 then 1 on CAPn.2 will cause CR2 to be loaded with the contents of the TC. When zero, this feature is disabled.	0
7	Capture on CAPn.2 falling edge	When one, a sequence of 1 then 0 on CAPn.2 will cause CR2 to be loaded with the contents of TC. When zero, this feature is disabled.	0
8	Interrupt on CAPn.2 event	When one, a CR2 load due to a CAPn.2 event will generate an interrupt. When zero, this feature is disabled.	0
9	Capture on CAPn.3 rising edge	When one, a sequence of 0 then 1 on CAPn.3 will cause CR3 to be loaded with the contents of TC. When zero, this feature is disabled.	0
10	Capture on CAPn.3 falling edge	When one, a sequence of 1 then 0 on CAPn.3 will cause CR3 to be loaded with the contents of TC. When zero, this feature is disabled.	0
11	Interrupt on CAPn.3 event	When one, a CR3 load due to a CAPn.3 event will generate an interrupt. When zero, this feature is disabled.	0

External Match Register (EMR: TIMER0 - T0EMR: 0xE000403C; TIMER1 - T1EMR: 0xE000803C)

The External Match Register provides both control and status of the external match pins M(0-3).

Table 133: External Match Register (EMR: TIMER0 - T0EMR: 0xE000403C; TIMER1 - T1EMR: 0xE000803C)

EMR	Function	Description	Reset Value
0	External Match 0	This bit reflects the state of output MAT0.0/MAT1.0, whether or not this output is connected to its pin. When a match occurs for MR0, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[4:5] control the functionality of this output.	0
1	External Match 1	This bit reflects the state of output MAT0.1/MAT1.1, whether or not this output is connected to its pin. When a match occurs for MR1, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[6:7] control the functionality of this output.	0
2	External Match 2	This bit reflects the state of output MAT0.2/MAT1.2, whether or not this output is connected to its pin. When a match occurs for MR2, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[8:9] control the functionality of this output.	0

Table 133: External Match Register (EMR: TIMER0 - T0EMR: 0xE000403C; TIMER1 - T1EMR: 0xE000803C)

EMR	Function	Description	Reset Value
3	External Match 3	This bit reflects the state of output MAT0.3/MAT1.3, whether or not this output is connected to its pin. When a match occurs for MR3, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[10:11] control the functionality of this output.	0
5:4	External Match Control 0	Determines the functionality of External Match 0. Table 134 shows the encoding of these bits.	0
7:6	External Match Control 1	Determines the functionality of External Match 1. Table 134 shows the encoding of these bits.	0
9:8	External Match Control 2	Determines the functionality of External Match 2. Table 134 shows the encoding of these bits.	0
11:10	External Match Control 3	Determines the functionality of External Match 3. Table 134 shows the encoding of these bits.	0

Table 134: External Match Control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
0 0	Do Nothing
0 1	Clear corresponding External Match output to 0 (LOW if pinned out)
1 0	Set corresponding External Match output to 1(HIGH if pinned out)
1 1	Toggle corresponding External Match output

EXAMPLE TIMER OPERATION

Figure 46 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 47 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.

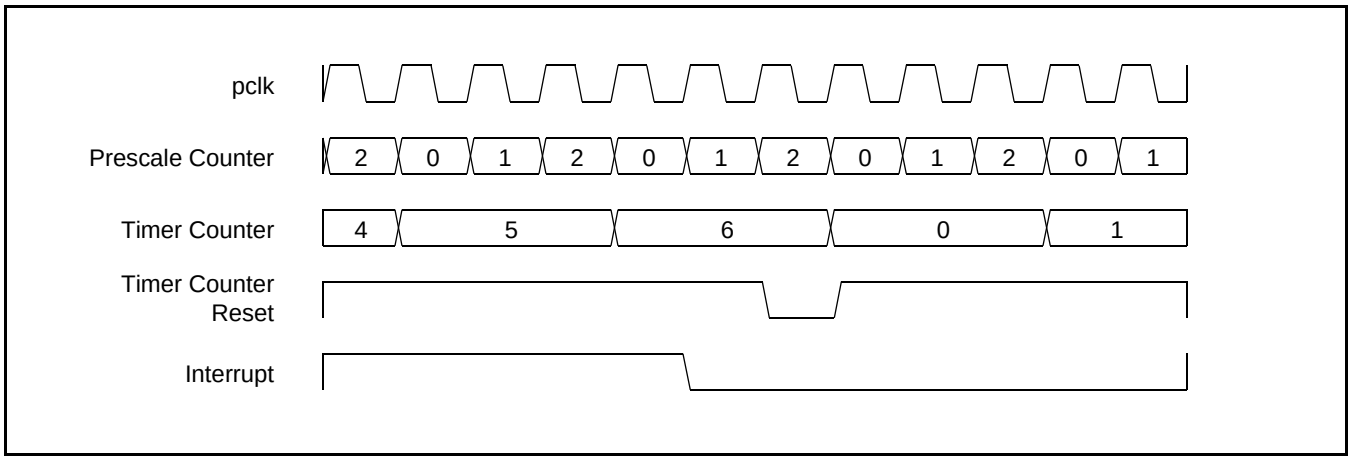


Figure 46: A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

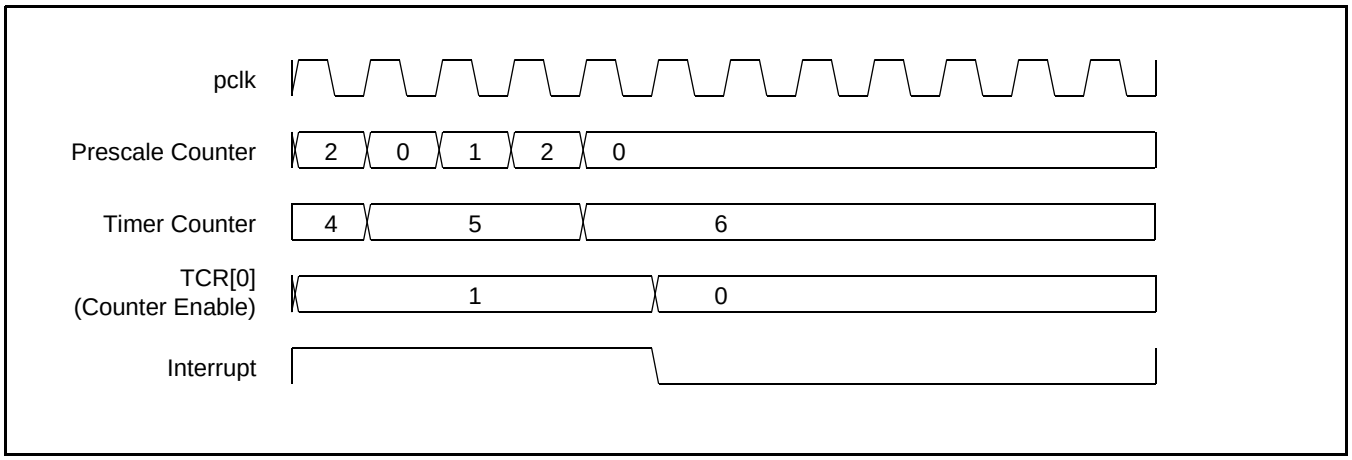


Figure 47: A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled.

ARCHITECTURE

The block diagram for TIMER0 and TIMER1 is shown in Figure 48.

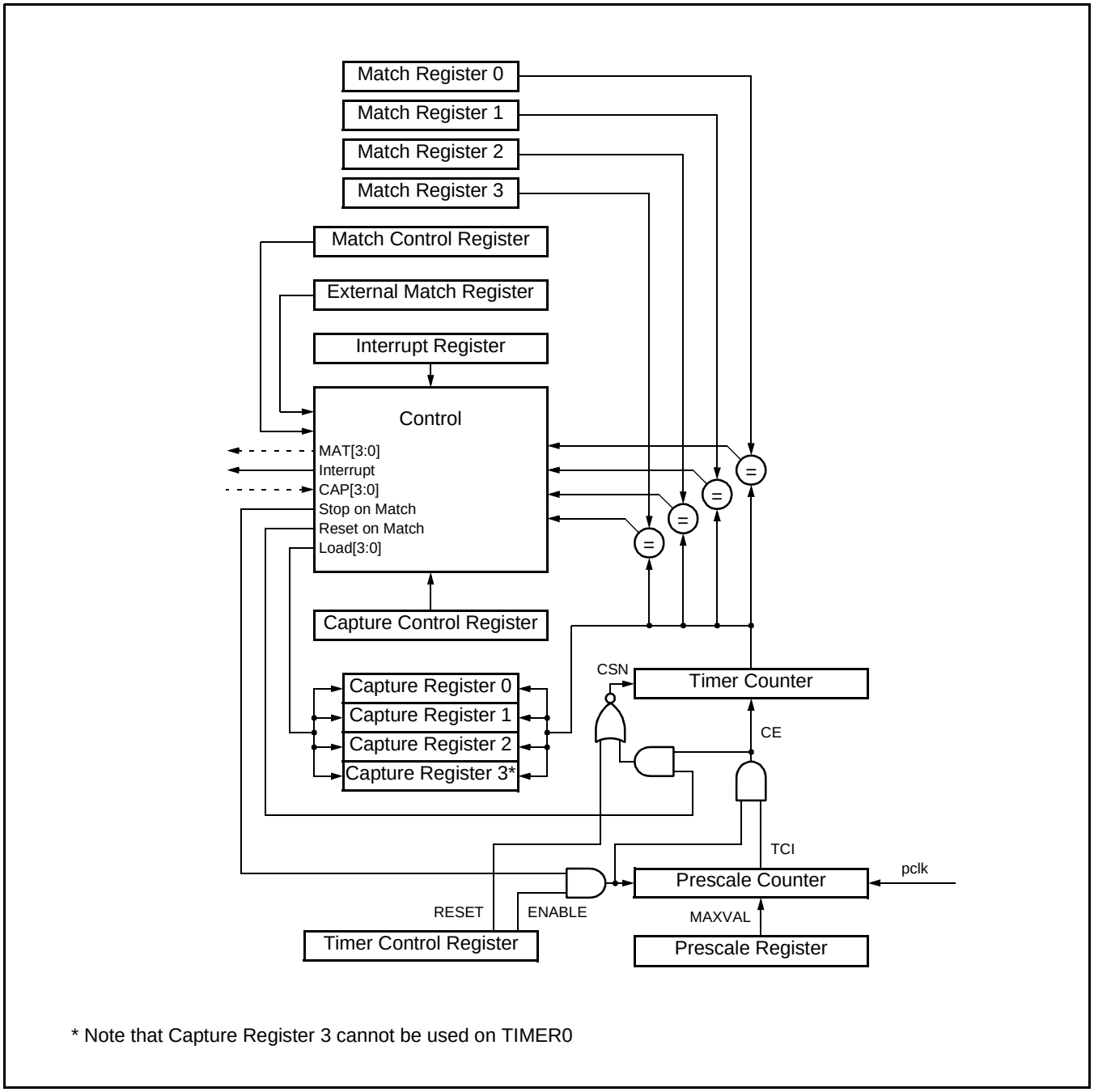


Figure 48: Timer block diagram

15. PULSE WIDTH MODULATOR (PWM)

LPC2131/2132/2138 Pulse Width Modulator is based on standard Timer 0/1 described in the previous chapter. Application can choose among PWM and match functions available.

FEATURES

- Seven match registers allow up to 6 single edge controlled or 3 double edge controlled PWM outputs, or a mix of both types. The match registers also allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- An external output for each match register with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.
- Supports single edge controlled and/or double edge controlled PWM outputs. Single edge controlled PWM outputs all go high at the beginning of each cycle unless the output is a constant low. Double edge controlled PWM outputs can have either edge occur at any position within a cycle. This allows for both positive going and negative going pulses.
- Pulse period and width can be any number of timer counts. This allows complete flexibility in the trade-off between resolution and repetition rate. All PWM outputs will occur at the same repetition rate.
- Double edge controlled PWM outputs can be programmed to be either positive going or negative going pulses.
- Match register updates are synchronized with pulse outputs to prevent generation of erroneous pulses. Software must "release" new match values before they can become effective.
- May be used as a standard timer if the PWM mode is not enabled.
- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Four 32-bit capture channels take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.

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 LPC2131/2132/2138. The Timer is designed to count cycles of the peripheral clock (pclk) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. It also includes four capture inputs to save the timer value when an input signal transitions, and optionally generate an interrupt when those events occur. The PWM function is in addition to these features, and is based on match register events.

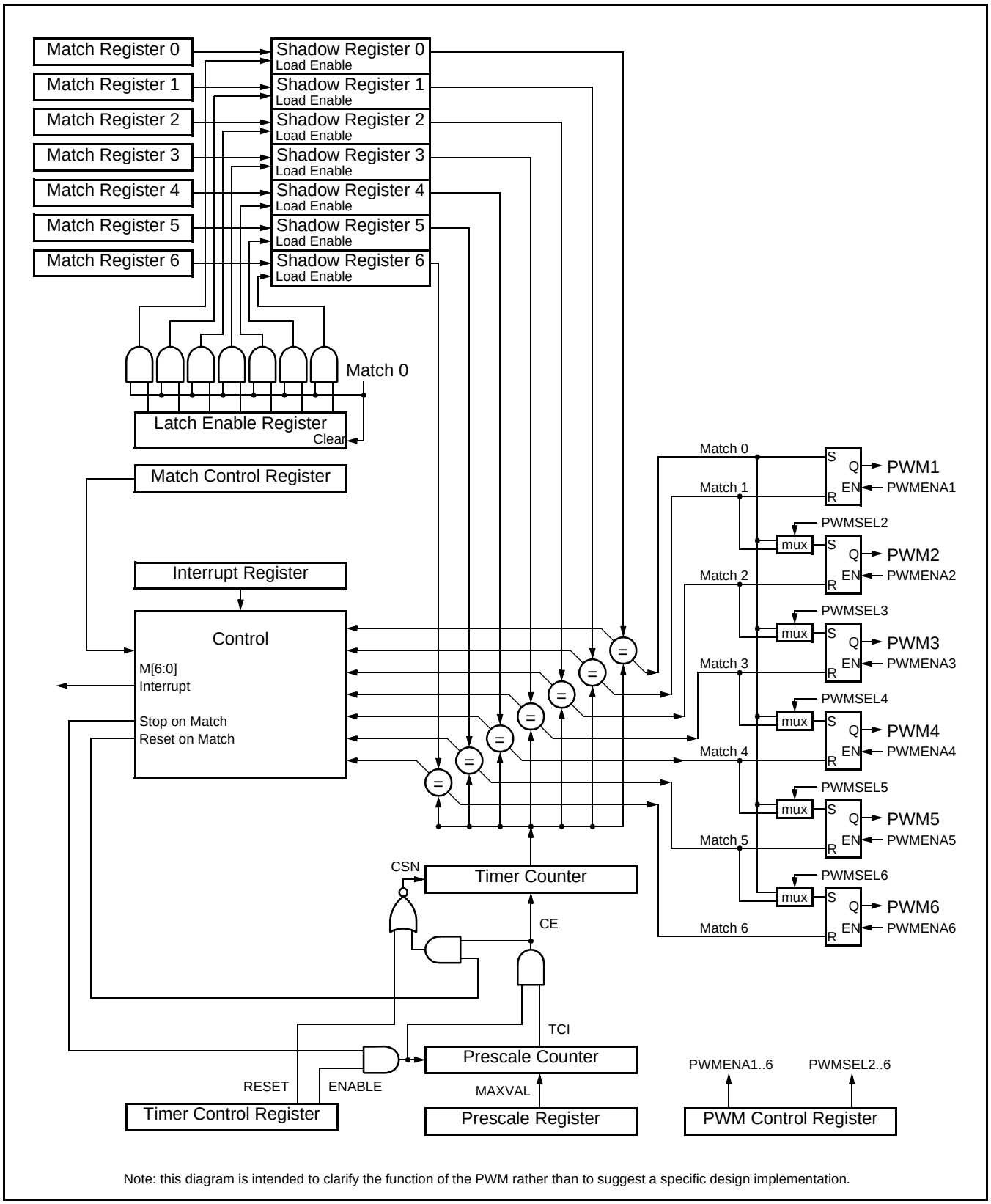
The ability to separately control rising and falling edge locations allows the PWM to be used for more applications. For instance, multi-phase motor control typically requires three non-overlapping PWM outputs with individual control of all three pulse widths and positions.

Two match registers can be used to provide a single edge controlled PWM output. One match register (PWMMR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position. Additional single edge controlled PWM outputs require only one match register each, since the repetition rate is the same for all PWM outputs. Multiple single edge controlled PWM outputs will all have a rising edge at the beginning of each PWM cycle, when an PWMMR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the PWMMR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions. Additional double edge controlled PWM outputs require only two match registers each, since the repetition rate is the same for all PWM outputs.

With double edge controlled PWM outputs, specific match registers control the rising and falling edge of the output. This allows both positive going PWM pulses (when the rising edge occurs prior to the falling edge), and negative going PWM pulses (when the falling edge occurs prior to the rising edge).

Figure 49 shows the block diagram of the PWM. The portions that have been added to the standard timer block are on the right hand side and at the top of the diagram.



Note: this diagram is intended to clarify the function of the PWM rather than to suggest a specific design implementation.

Figure 49: PWM block diagram

A sample of how PWM values relate to waveform outputs is shown in Figure 50. PWM output logic is shown in Figure 49 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 135. This implementation supports up to N-1 single edge PWM outputs or (N-1)/2 double edge PWM outputs, where N is the number of match registers that are implemented. PWM types can be mixed if desired.

The waveforms below show a single PWM cycle and demonstrate PWM outputs under the following conditions:

- The timer is configured for PWM mode.
- Match 0 is configured to reset the timer/counter when a match event occurs.
- Control bits PWMSEL2 and PWMSEL4 are set.
- The Match register values are as follows:
  MR0= 100(PWM rate)
  MR1= 41, MR2 = 78(PWM2 output)
  MR3= 53, MR4 = 27(PWM4 output)
  MR5= 65(PWM5 output)

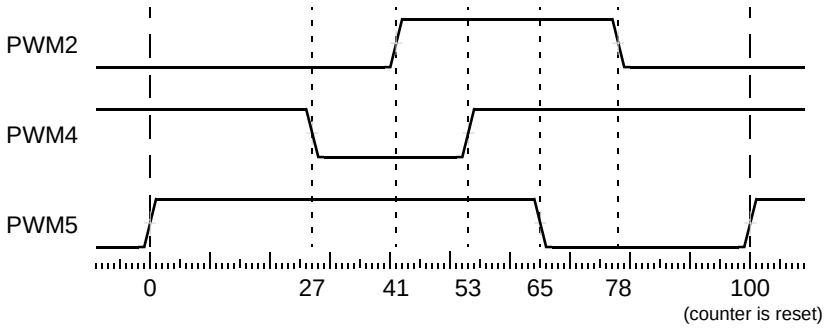


Figure 50: Sample PWM waveforms

Table 135: 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

Notes:
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.

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.

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.

PIN DESCRIPTION

Table 136 gives a brief summary of each of PWM related pins.

Table 136: Pin summary

Pin name	Pin direction	Pin Description
PWM1	Output	Output from PWM channel 1.
PWM2	Output	Output from PWM channel 2.
PWM3	Output	Output from PWM channel 3.
PWM4	Output	Output from PWM channel 4.
PWM5	Output	Output from PWM channel 5.
PWM6	Output	Output from PWM channel 6.

REGISTER DESCRIPTION

The PWM function adds new registers and registers bits as shown in Table 137 below.

Table 137: Pulse Width Modulator Register Map

Name	Description	Access	Reset Value*	Address
PWMIR	PWM Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of the possible interrupt sources are pending.	R/W	0	0xE0014000
PWMTCR	PWM 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	0xE0014004
PWMTC	PWM Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0	0xE0014008
PWMPR	PWM Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0	0xE001400C
PWMPC	PWM 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.	R/W	0	0xE0014010
PWMMCR	PWM 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	0xE0014014
PWMMR0	PWM Match Register 0. MR0 can be enabled through 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 MR0 and the TC sets all PWM outputs that are in single-edge mode, and sets PWM1 if it is in double-edge mode.	R/W	0	0xE0014018
PWMMR1	PWM Match Register 1. MR1 can be enabled through 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 MR1 and the TC clears PWM1 in either single-edge mode or double-edge mode, and sets PWM2 if it is in double-edge mode.	R/W	0	0xE001401C
PWMMR2	PWM Match Register 2. MR2 can be enabled through 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 MR2 and the TC clears PWM2 in either single-edge mode or double-edge mode, and sets PWM3 if it is in double-edge mode.	R/W	0	0xE0014020
PWMMR3	PWM Match Register 3. MR3 can be enabled through 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 MR3 and the TC clears PWM3 in either single-edge mode or double-edge mode, and sets PWM4 if it is in double-edge mode.	R/W	0	0xE0014024
PWMMR4	PWM Match Register 4. MR4 can be enabled through 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 MR4 and the TC clears PWM4 in either single-edge mode or double-edge mode, and sets PWM5 if it is in double-edge mode.	R/W	0	0xE0014040
PWMMR5	PWM Match Register 5. MR5 can be enabled through 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 MR5 and the TC clears PWM5 in either single-edge mode or double-edge mode, and sets PWM6 if it is in double-edge mode.	R/W	0	0xE0014044
PWMMR6	PWM Match Register 6. MR6 can be enabled through 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 MR6 and the TC clears PWM6 in either single-edge mode or double-edge mode.	R/W	0	0xE0014048

Table 137: Pulse Width Modulator Register Map

Name	Description	Access	Reset Value*	Address
PWMPCR	PWM Control Register. Enables PWM outputs and selects PWM channel types as either single edge or double edge controlled.	R/W	0	0xE001404C
PWMLER	PWM Latch Enable Register. Enables use of new PWM match values.	R/W	0	0xE0014050

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

PWM Interrupt Register (PWMIR - 0xE0014000)

The PWM Interrupt Register consists of eleven bits (Table 138), seven for the match interrupts and four 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 one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 138: PWM Interrupt Register (PWMIR - 0xE0014000)

PWMIR	Function	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	Reserved.	Application must not write 1 to this bit.	0
5	Reserved.	Application must not write 1 to this bit.	0
6	Reserved.	Application must not write 1 to this bit.	0
7	Reserved.	Application must not write 1 to this bit.	0
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

PWM Timer Control Register (PWMTCR - 0xE0014004)

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 139.

Table 139: PWM Timer Control Register (PWMTCR - 0xE0014004)

PWMTCR	Function	Description	Reset Value
0	Counter Enable	When one, the PWM Timer Counter and PWM Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0
2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PWM Enable	When one, PWM mode is enabled. PWM mode causes shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match 0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	0

PWM Timer Counter (PWMTC - 0xE0014008)

The 32-bit PWM Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the PWMTC will count up through the value 0xFFFFFFFF and then wrap back to the value 0x00000000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

PWM Prescale Register (PWMPR - 0xE001400C)

The 32-bit PWM Prescale Register specifies the maximum value for the PWM Prescale Counter.

PWM Prescale Counter Register (PWMPC - 0xE0014010)

The 32-bit PWM Prescale Counter controls division of pclk by some constant value before it is applied to the PWM Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The PWM Prescale Counter is incremented on every pclk. When it reaches the value stored in the PWM Prescale Register, the PWM Timer Counter is incremented and the PWM Prescale Counter is reset on the next pclk. This causes the PWM TC to increment on every pclk when PWMPR = 0, every 2 pclks when PWMPR = 1, etc.

PWM Match Registers (PWMMR0 - PWMMR6)

The 32-bit PWM Match register values are continuously compared to the PWM Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the PWM Timer Counter, or stop the timer. Actions are controlled by the settings in the PWMMCR register.

PWM Match Control Register (PWMMCR - 0xE0014014)

The PWM Match Control Register is used to control what operations are performed when one of the PWM Match Registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 140.

Table 140: PWM Match Control Register (PWMMCR - 0xE0014014)

PWMMCR	Function	Description	Reset Value
0	Interrupt on PWMMR0	When one, an interrupt is generated when PWMMR0 matches the value in the PWMTC. When zero this interrupt is disabled.	0
1	Reset on PWMMR0	When one, the PWMTC will be reset if PWMMR0 matches it. When zero this feature is disabled.	0
2	Stop on PWMMR0	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR0 matches the PWMTC. When zero this feature is disabled.	0
3	Interrupt on PWMMR1	When one, an interrupt is generated when PWMMR1 matches the value in the PWMTC. When zero this interrupt is disabled.	0
4	Reset on PWMMR1	When one, the PWMTC will be reset if PWMMR1 matches it. When zero this feature is disabled.	0
5	Stop on PWMMR1	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR1 matches the PWMTC. When zero this feature is disabled	0
6	Interrupt on PWMMR2	When one, an interrupt is generated when PWMMR2 matches the value in the PWMTC. When zero this interrupt is disabled.	0
7	Reset on PWMMR2	When one, the PWMTC will be reset if PWMMR2 matches it. When zero this feature is disabled.	0
8	Stop on PWMMR2	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR2 matches the PWMTC. When zero this feature is disabled	0
9	Interrupt on PWMMR3	When one, an interrupt is generated when PWMMR3 matches the value in the PWMTC. When zero this interrupt is disabled	0
10	Reset on PWMMR3	When one, the PWMTC will be reset if PWMMR3 matches it. When zero this feature is disabled.	0
11	Stop on PWMMR3	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR3 matches the PWMTC. When zero this feature is disabled	0
12	Interrupt on PWMMR4	When one, an interrupt is generated when PWMMR4 matches the value in the PWMTC. When zero this interrupt is disabled.	0
13	Reset on PWMMR4	When one, the PWMTC will be reset if PWMMR4 matches it. When zero this feature is disabled.	0
14	Stop on PWMMR4	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR4 matches the PWMTC. When zero this feature is disabled	0
15	Interrupt on PWMMR5	When one, an interrupt is generated when PWMMR5 matches the value in the PWMTC. When zero this interrupt is disabled.	0
16	Reset on PWMMR5	When one, the PWMTC will be reset if PWMMR5 matches it. When zero this feature is disabled.	0

Table 140: PWM Match Control Register (PWMMCR - 0xE0014014)

PWMMCR	Function	Description	Reset Value
17	Stop on PWMMR5	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR5 matches the PWMTC. When zero this feature is disabled	0
18	Interrupt on PWMMR6	When one, an interrupt is generated when PWMMR6 matches the value in the PWMTC. When zero this interrupt is disabled.	0
19	Reset on PWMMR6	When one, the PWMTC will be reset if PWMMR6 matches it. When zero this feature is disabled.	0
20	Stop on PWMMR6	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR6 matches the PWMTC. When zero this feature is disabled	0

PWM Control Register (PWMPCR - 0xE001404C)

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 141.

Table 141: PWM Control Register (PWMPCR - 0xE001404C)

PWMPCR	Function	Description	Reset Value
1:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	PWMSEL2	When zero, selects single edge controlled mode for PWM2. When one, selects double edge controlled mode for the PWM2 output.	0
3	PWMSEL3	When zero, selects single edge controlled mode for PWM3. When one, selects double edge controlled mode for the PWM3 output.	0
4	PWMSEL4	When zero, selects single edge controlled mode for PWM4. When one, selects double edge controlled mode for the PWM4 output.	0
5	PWMSEL5	When zero, selects single edge controlled mode for PWM5. When one, selects double edge controlled mode for the PWM5 output.	0
6	PWMSEL6	When zero, selects single edge controlled mode for PWM6. When one, selects double edge controlled mode for the PWM6 output.	0
8:7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9	PWMENA1	When one, enables the PWM1 output. When zero, disables the PWM1 output.	0
10	PWMENA2	When one, enables the PWM2 output. When zero, disables the PWM2 output.	0
11	PWMENA3	When one, enables the PWM3 output. When zero, disables the PWM3 output.	0
12	PWMENA4	When one, enables the PWM4 output. When zero, disables the PWM4 output.	0
13	PWMENA5	When one, enables the PWM5 output. When zero, disables the PWM5 output.	0
14	PWMENA6	When one, enables the PWM6 output. When zero, disables the PWM6 output.	0
15	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PWM Latch Enable Register (PWMLER - 0xE0014050)

The PWM Latch Enable Register is used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to the location of a PWM Match register while the Timer is in PWM mode, the value is held in a shadow register. When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the actual Match registers if the corresponding bit in the Latch Enable Register has been set. At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared. Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.

For example, if PWM2 is configured for double edge operation and is currently running, a typical sequence of events for changing the timing would be:

- Write a new value to the PWM Match1 register.
- Write a new value to the PWM Match2 register.
- Write to the PWMLER, setting bits 1 and 2 at the same time.
- The altered values will become effective at the next reset of the timer (when a PWM Match 0 event occurs).

The order of writing the two PWM Match registers is not important, since neither value will be used until after the write to PWMLER. This insures that both values go into effect at the same time, if that is required. A single value may be altered in the same way if needed.

The function of each of the bits in the PWMLER is shown in Table 140.

Table 142: PWM Latch Enable Register (PWMLER - 0xE0014050)

PWMLER	Function	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 the description of the PWM Match Control Register (PWMMCR).	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 the description of the PWM Match Control Register (PWMMCR).	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 the description of the PWM Match Control Register (PWMMCR).	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 the description of the PWM Match Control Register (PWMMCR).	0
4	Enable PWM Match 4 Latch	Writing a one to this bit allows the last value written to the PWM Match 4 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	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 the description of the PWM Match Control Register (PWMMCR).	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 the description of the PWM Match Control Register (PWMMCR).	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16. A/D CONVERTER

FEATURES

- 10 bit successive approximation analog to digital converter (one in LPC2131/2132 and two in LPC2138).
- Input multiplexing among 8 pins
- Power down mode
- Measurement range 0 to 3 V
- 10 bit conversion time >= 2.44 uS
- Burst conversion mode for single or multiple inputs
- Optional conversion on transition on input pin or Timer Match signal
- Global Start command for both converters (LPC2138 only).

DESCRIPTION

Basic clocking for the A/D converters is provided by the VPB clock. A programmable divider is included in each converter, to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks.

PIN DESCRIPTIONS

Table 143: A/D Pin Description

Pin Name	Type	Pin Description
AD0.7:0 & AD1.7:0 (LPC2138)	Input	**Analog Inputs.** The A/D converter cell can measure the voltage on any of these input signals. Note that these analog inputs are always connected to their pins, even if the Pin Multiplexing Register assigns them to port pins. A simple self-test of the A/D Converter can be done by driving these pins as port outputs. **Note:** if the A/D converter is used, signal levels on analog input pins must not be above the level of V_{3A} at any time. Otherwise, A/D converter readings will be invalid. If the A/D converter is not used in an application then the pins associated with A/D inputs can be used as 5V tolerant digital IO pins.
V_{ref}	Reference	**Voltage Reference.** This pin is connected to the VrefP signals of both A/D converters.
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should be nominally the same voltages as V_3 and V_{SSD}, but should be isolated to minimize noise and error.

REGISTER DESCRIPTION

The base address of the A/D Converter is 0xE003 4000 (page 24). The A/D Converter includes 2 registers as shown in Table 144.

Table 144: A/D Registers

Generic Name	Description	Access	Reset Value	AD0 Address & Name	AD1 Address & Name
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	Read/Write	0x0000 0001	0xE003 4000 AD0CR	0xE006 0000 AD1CR
ADDR	A/D Data Register. This register contains the ADC's DONE bit and (when DONE is 1) the 10-bit result of the conversion.	Read/Write	NA	0xE003 4004 AD0DR	0xE006 0004 AD1DR
ADGSR	A/D Global Start Register. This address can be written (in the AD0 address range) to start conversions in both A/D converters simultaneously.	Write Only	0x00	ADGSR 0xE003 4008	

A/D Control Register (AD0CR - 0xE0034000, AD1CR - 0xE0060000)

Table 145: A/D Control Register (AD0CR - 0xE0034000, AD1CR - 0xE0060000)

ADCR	Name	Description	Reset Value
7:0	SEL	Selects which of the AD0.7:0/AD1.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV	The VPB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST	If this bit is 0, conversions are software controlled and require 11 clocks. If this bit is 1, the AD converter does repeated conversions at the rate selected by the CLKS field, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed.	0
19:17	CLKS	This field selects the number of clocks used for each conversion in Burst mode, and the number of bits of accuracy of the result in the LS bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits): 000=11 clocks/10 bits, 001=10 clocks/9 bits,..., 111=4 clocks/3 bits	000
21	PDN	1: the A/D converter is operational 0: the A/D converter is in power down mode	0
23:22	TEST1:0	These bits are used in device testing. 00=normal operation, 01=digital test mode, 10=DAC test mode, and 11=simple conversion test mode.	0
26:24	START	When the BURST bit is 0, these bits control whether and when an A/D conversion is started: 000: no start (this value should be used when clearing PDN to 0) 001: start conversion now 010: start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/ CAP0.2 011: start conversion when the edge selected by bit 27 occurs on P0.22/TD3/CAP0.0/ MAT0.0 *Note: for choices 100-111 the MAT signal need not be pinned out:* 100: start conversion when the edge selected by bit 27 occurs on MAT0.1 101: start conversion when the edge selected by bit 27 occurs on MAT0.3 110: start conversion when the edge selected by bit 27 occurs on MAT1.0 111: start conversion when the edge selected by bit 27 occurs on MAT1.1	000
27	EDGE	This bit is significant only when the START field contains 010-111. In these cases: 0: start conversion on a falling edge on the selected CAP/MAT signal 1: start conversion on a rising edge on the selected CAP/MAT signal	0
31:28	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

A/D Data Register (AD0DR - 0xE0034004, AD1DR - 0xE0060004)

Table 146: A/D Data Register (AD0DR - 0xE0034004, AD1DR - 0xE0060004)

ADDR	Name	Description	Reset Value
5:0	Reserved	Reserved. These bits always read as zeroes. User should not write ones to reserved bits.	0
15:6	V/V_{3A}	When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin selected by the SEL field, divided by the voltage on the VddA pin. Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on V_{SSA}, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on V_{3A}. For testing, data written to this field is captured in a shift register that is clocked by the A/D converter clock. The MS bit of this register sources the DINSERI input of the A/D converter, which is used only when TEST1:0 are 10.	X
23:16	Reserved	Reserved. These bits always read as zeroes. User should not write ones to reserved bits.	0
26:24	CHN	These bits contain the channel from which the LS bits were converted.	X
29:27	Reserved	Reserved. These bits always read as zeroes. User should not write ones to reserved bits.	0
30	OVERUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the LS bits. In non-FIFO operation, this bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

A/D Global Start Register (ADGSR - 0xE0034008)

Table 147: A/D Global Start Register (ADGSR - 0xE0034008)

ADCR	Name	Description	Reset Value
15:0	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
16	BURST	If this bit is 0, conversions are software controlled and require 11 clocks. If this bit is 1, the AD converters do repeated conversions at the rate selected by their CLKS fields, scanning (if necessary) through the pins selected by 1s in the SEL fields. 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.	0

Table 147: A/D Global Start Register (ADGSR - 0xE0034008)

ADCR	Name	Description	Reset Value
23:17	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
26:24	START	When the BURST bit is 0, these bits control whether and when an A/D conversion is started: 000: no start (this value should be used when clearing PDN to 0) 001: start conversion now 010: start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/ CAP0.2 011: start conversion when the edge selected by bit 27 occurs on P0.22/TD3/CAP0.0/ MAT0.0 *Note: for choices 100-111 the MAT signal need not be pinned out:* 100: start conversion when the edge selected by bit 27 occurs on MAT0.1 101: start conversion when the edge selected by bit 27 occurs on MAT0.3 110: start conversion when the edge selected by bit 27 occurs on MAT1.0 111: start conversion when the edge selected by bit 27 occurs on MAT1.1	000
27	EDGE	This bit is significant only when the START field contains 010-111. In these cases: 0: start conversion on a falling edge on the selected CAP/MAT signal 1: start conversion on a rising edge on the selected CAP/MAT signal	0
31:28	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

OPERATION

Hardware-Triggered Conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the A/D converter will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

Clock Generation

It is highly desirable that the clock divider for the 4.5 MHz conversion clock be held in a Reset state when the A/D converter is idle, so that the sampling clock can begin immediately when 01 is written to the START field of the ADCR, or the selected edge occurs on the selected signal. This feature also saves power, particularly if the A/D converter is used infrequently.

Interrupts

An interrupt request is asserted to the Vectored Interrupt Controller (VIC) when the DONE bit is 1. Software can use the Interrupt Enable bit for the A/D Converter in the VIC to control whether this assertion results in an interrupt. DONE is negated when the ADDR is read.

Accuracy vs. Digital Receiver

While the A/D converter can be used to measure the voltage on any AIN pin, regardless of the pin's setting in the Pin Select register (Pin Connect Block on page 81), selecting the AIN function improves the conversion accuracy by disabling the pin's digital receiver.

17. D/A CONVERTER (LPC2132/2138 ONLY)

FEATURES

- 10 bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power down mode
- Selectable speed vs. power

PIN DESCRIPTIONS

Table 148: D/A Pin Description

Pin Name	Type	Pin Description
A_{OUT}	Input	**Analog Output.** After the selected settling time after the DACR is written with a new value, the voltage on this pin (with respect to V_{SSA}) is VALUE/1024 * V_{ref}.
V_{ref}	Reference	**Voltage Reference.** This pin is connected to the VrefP signals of both A/D converters.
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should be nominally the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error.

REGISTER DESCRIPTION

D/A Converter Register (DACR - 0xE006C000)

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 149: D/A Converter Register (DACR - 0xE006C000)

ADCR	Name	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.	0
15:6	VALUE	After the selected settling time after this field is written with a new VALUE, the voltage on the A_{OUT} pin (with respect to V_{SSA}) is VALUE/1024 * V_{ref}.	0
16	BIAS	If this bit is 0, the settling time of the DAC is 1 uS max, and the maximum current is 700 uA. If this bit is 1, the settling time of the DAC is 2.5 uS and the maximum current is 350 uA.	0
31:17	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

OPERATION

Bits 19:18 of the PINSEL1 register (page 82) control whether the DAC is enabled and controlling the state of pin P0.25/AD0.4/ A_{OUT}. When these bits are 10, the DAC is powered on and active. The settling times noted in the description of the BIAS bit are valid for a load impedance on the A_{OUT} pin, that's greater than or equal to a certain value (TBD). A load impedance value less 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.

18. REAL TIME CLOCK

FEATURES

- Measures the passage of time to maintain a calendar and clock.
- Ultra Low Power design to support battery powered systems.
- Provides Seconds, Minutes, Hours, Day of Month, Month, Year, Day of Week, and Day of Year.
- Dedicated 32 KHz oscillator or programmable prescaler from VPB clock.
- Dedicated power supply pin can be connected to a battery or to the main 3.3V.

DESCRIPTION

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off. It uses little power in power down mode. On the LPC2131/2132/2138, the RTC can be clocked by a separate 32.768 KHz oscillator, or by a programmable prescale divider based on the VPB clock. Also, the RTC is powered by its own power supply pin, V_{bat}, which can be connected to a battery or to the same 3.3 volt supply used by the rest of the device.

ARCHITECTURE

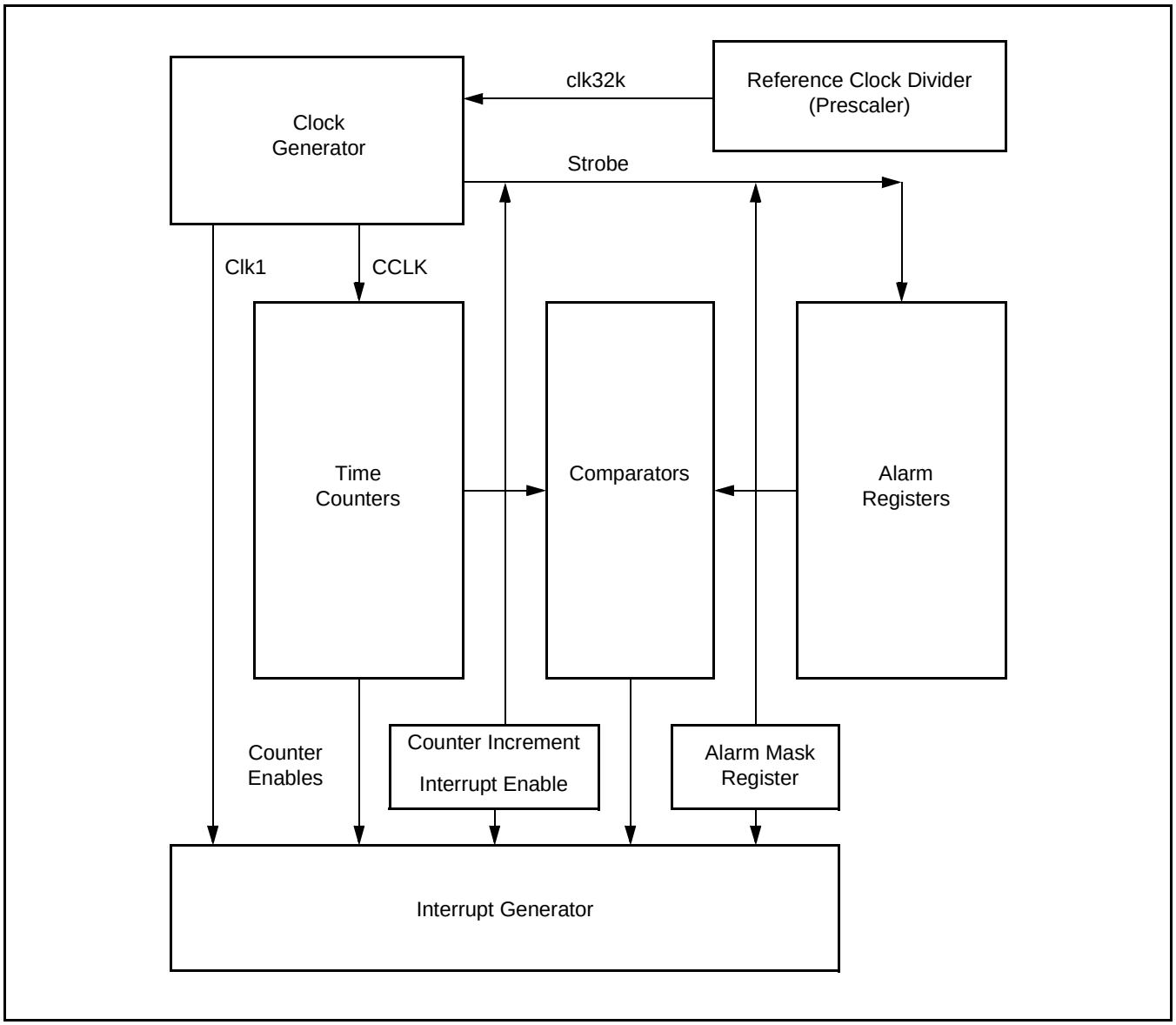


Figure 51: RTC block diagram

REGISTER DESCRIPTION

The RTC includes a number of registers. The address space is split into four sections by functionality. The first eight addresses are the Miscellaneous Register Group. The second set of eight locations are the Time Counter Group. The third set of eight locations contain the Alarm Register Group. The remaining registers control the Reference Clock Divider.

The Real Time Clock includes the register shown in Table 150. Detailed descriptions of the registers follow.

Table 150: Real Time Clock Register Map

Name	Size	Description	Access	Reset Value	Address
ILR	2	Interrupt Location Register	R/W	*	0xE0024000
CTC	15	Clock Tick Counter.	RO	*	0xE0024004
CCR	4	Clock Control Register	R/W	*	0xE0024008
CIIR	8	Counter Increment Interrupt Register	R/W	*	0xE002400C
AMR	8	Alarm Mask Register	R/W	*	0xE0024010
CTIME0	(32)	Consolidated Time Register 0	RO	*	0xE0024014
CTIME1	(32)	Consolidated Time Register 1	RO	*	0xE0024018
CTIME2	(32)	Consolidated Time Register 2	RO	*	0xE002401C
SEC	6	Seconds Register	R/W	*	0xE0024020
MIN	6	Minutes Register	R/W	*	0xE0024024
HOUR	5	Hours Register	R/W	*	0xE0024028
DOM	5	Day of Month Register	R/W	*	0xE002402C
DOW	3	Day of Week Register	R/W	*	0xE0024030
DOY	9	Day of Year Register	R/W	*	0xE0024034
MONTH	4	Months Register	R/W	*	0xE0024038
YEAR	12	Years Register	R/W	*	0xE002403C
ALSEC	6	Alarm value for Seconds	R/W	*	0xE0024060
ALMIN	6	Alarm value for Minutes	R/W	*	0xE0024064
ALHOUR	5	Alarm value for Hours	R/W	*	0xE0024068
ALDOM	5	Alarm value for Day of Month	R/W	*	0xE002406C
ALDOW	3	Alarm value for Day of Week	R/W	*	0xE0024070
ALDOY	9	Alarm value for Day of Year	R/W	*	0xE0024074
ALMON	4	Alarm value for Months	R/W	*	0xE0024078
ALYEAR	12	Alarm value for Year	R/W	*	0xE002407C
PREINT	13	Prescale value, integer portion	R/W	0	0xE0024080
PREFRAC	15	Prescale value, fractional portion	R/W	0	0xE0024084

* Registers in the RTC other than those that are part of the Prescaler are not affected by chip Reset. These registers must be initialized by software if the RTC is enabled.

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 LPC2131/2132/2138 out of power-down mode if the RTC is operating from its own oscillator on the RTCX1-2 pins. When the RTC interrupt is enabled for wakeup and its selected event occurs, XTAL1/2 pins associated oscillator wakeup cycle is started. For details on the RTC based wakeup process see Interrupt Wakeup Register (INTWAKE - 0xE01FC144) on page 34 and Wakeup Timer on page 51.

MISCELLANEOUS REGISTER GROUP

Table 151 summarizes the registers located from 0 to 7 of A[6:2]. More detailed descriptions follow.

Table 151: Miscellaneous Registers

Address	Name	Size	Description	Access
0xE0024000	ILR	2	Interrupt Location. Reading this location indicates the source of an interrupt. Writing a one to the appropriate bit at this location clears the associated interrupt.	RW
0xE0024004	CTC	15	Clock Tick Counter. Value from the clock divider.	RO
0xE0024008	CCR	4	Clock Control Register. Controls the function of the clock divider.	RW
0xE002400C	CIIR	8	Counter Increment Interrupt. Selects which counters will generate an interrupt when they are incremented.	RW
0xE0024010	AMR	8	Alarm Mask Register. Controls which of the alarm registers are masked.	RW
0xE0024014	CTIME0	32	Consolidated Time Register 0	RO
0xE0024018	CTIME1	32	Consolidated Time Register 1	RO
0xE002401C	CTIME2	32	Consolidated Time Register 2	RO

Interrupt Location (ILR - 0xE0024000)

The Interrupt Location Register is a 2-bit register that specifies which blocks are generating an interrupt (see Table 152). 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 152: Interrupt Location (ILR - 0xE0024000)

ILR	Function	Description
0	RTCCIF	When one, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.

Clock Tick Counter (CTC - 0xE0024004)

The Clock Tick Counter is read only. It can be reset to zero through the Clock Control Register (CCR). The CTC consists of the bits of the clock divider counter.

Table 153: Clock Tick Counter (CTC - 0xE0024004)

CTC	Function	Description
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
15:1	Clock Tick Counter	Prior to the Seconds counter, the CTC counts 32,768 clocks per second. Due to the RTC Prescaler, these 32,768 time increments may not all be of the same duration. Refer to the Reference Clock Divider (Prescaler) description for details.

Clock Control Register (CCR - 0xE0024008)

The clock register is a 5-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 154.

Table 154: Clock Control Register (CCR - 0xE0024008)

CCR	Function	Description
0	CLKEN	Clock Enable. When this bit is a one the time counters are enabled. When it is a zero, they are disabled so that they may be initialized.
1	CTCRST	CTC Reset. When one, the elements in the Clock Tick Counter are reset. The elements remain reset until CCR[1] is changed to zero.
3:2	CTTEST	Test Enable. These bits should always be zero during normal operation.
4	CLKSRC	If this bit is 0, the Clock Tick Counter takes its clock from the Prescaler, as on earlier devices in the Philips Embedded ARM family. If this bit is 1, the CTC takes its clock from the 32 KHz oscillator that's connected to the RTCX1 and RTCX2 pins.

Counter Increment Interrupt Register (CIIR - 0xE002400C)

The Counter Increment Interrupt Register (CIIR) gives the ability to generate an interrupt every time a counter is incremented. This interrupt remains valid until cleared by writing a one to bit zero of the Interrupt Location Register (ILR[0]).

Table 155: Counter Increment Interrupt Register (CIIR - 0xE002400C)

CIIR	Function	Description
0	IMSEC	When one, an increment of the Second value generates an interrupt.
1	IMMIN	When one, an increment of the Minute value generates an interrupt.
2	IMHOUR	When one, an increment of the Hour value generates an interrupt.
3	IMDOM	When one, an increment of the Day of Month value generates an interrupt.
4	IMDOW	When one, an increment of the Day of Week value generates an interrupt.
5	IMDOY	When one, an increment of the Day of Year value generates an interrupt.
6	IMMON	When one, an increment of the Month value generates an interrupt.
7	IMYEAR	When one, an increment of the Year value generates an interrupt.

Alarm Mask Register (AMR - 0xE0024010)

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 156 shows the relationship between the bits in the AMR and the alarms. For the alarm function, every non-masked alarm register must match the corresponding time counter for an interrupt to be generated. The interrupt is generated only when the counter comparison first changes from no match to match. The interrupt is removed when a one is written to the appropriate bit of the Interrupt Location Register (ILR). If all mask bits are set, then the alarm is disabled.

Table 156: Alarm Mask Register (AMR - 0xE0024010)

AMR	Function	Description
0	AMRSEC	When one, the Second value is not compared for the alarm.
1	AMRMIN	When one, the Minutes value is not compared for the alarm.
2	AMRHOUR	When one, the Hour value is not compared for the alarm.
3	AMRDOM	When one, the Day of Month value is not compared for the alarm.
4	AMRDOW	When one, the Day of Week value is not compared for the alarm.
5	AMRDOY	When one, the Day of Year value is not compared for the alarm.
6	AMRMON	When one, the Month value is not compared for the alarm.
7	AMRYEAR	When one, the Year value is not compared for the alarm.

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 Tables 157, 158, and 159. 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.

Consolidated Time Register 0 (CTIME0 - 0xE0024014)

The Consolidated Time Register 0 contains the low order time values: Seconds, Minutes, Hours, and Day of Week.

Table 157: Consolidated Time Register 0 (CTIME0 - 0xE0024014)

CTIME0	Function	Description
31:27	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
26:24	Day of Week	Day of week value in the range of 0 to 6.
23:21	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
20:16	Hours	Hours value in the range of 0 to 23.
15:14	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
13:8	Minutes	Minutes value in the range of 0 to 59.
7:6	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
5:0	Seconds	Seconds value in the range of 0 to 59.

Consolidated Time Register 1 (CTIME1 - 0xE0024018)

The Consolidate Time Register 1 contains the Day of Month, Month, and Year values.

Table 158: Consolidated Time Register 1 (CTIME1 - 0xE0024018)

CTIME1	Function	Description
31:28	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
27:16	Year	Year value in the range of 0 to 4095.
15:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
11:8	Month	Month value in the range of 1 to 12.
7:5	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
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).

Consolidated Time Register 2 (CTIME2 - 0xE002401C)

The Consolidate Time Register 2 contains just the Day of Year value.

Table 159: Consolidated Time Register 2 (CTIME2 - 0xE002401C)

CTIME2	Function	Description
11:0	Day of Year	Day of year value in the range of 1 to 365 (366 for leap years).
31:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

TIME COUNTER GROUP

The time value consists of the eight counters shown in Tables 160 and 161. These counters can be read or written at the locations shown in Table 161.

Table 160: Time Counter Relationships and Values

Counter	Size	Enabled by	Min value	Maximum value
Second	6	Clk1 (see Figure 51)	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 161: Time Counter registers

Address	Name	Size	Description	Access
0xE0024020	SEC	6	Seconds value in the range of 0 to 59.	R/W
0xE0024024	MIN	6	Minutes value in the range of 0 to 59.	R/W
0xE0024028	HOUR	5	Hours value in the range of 0 to 23.	R/W
0xE002402C	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
0xE0024030	DOW	3	Day of week value in the range of 0 to 6. [1]	R/W
0xE0024034	DOY	9	Day of year value in the range of 1 to 365 (366 for leap years). [1]	R/W
0xE0024038	MONTH	4	Month value in the range of 1 to 12.	R/W
0xE002403C	YEAR	12	Year value in the range of 0 to 4095.	R/W

Notes:
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.

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.

ALARM REGISTER GROUP

The alarm registers are shown in Table 162. The values in these registers are compared with the time counters. If all the unmasked (See “Alarm Mask Register (AMR - 0xE0024010)” on page 210.) alarm registers match their corresponding time counters then an interrupt is generated. The interrupt is cleared when a one is written to bit one of the Interrupt Location Register (ILR[1]).

Table 162: Alarm Registers

Address	Name	Size	Description	Access
0xE0024060	ALSEC	6	Alarm value for Seconds	R/W
0xE0024064	ALMIN	6	Alarm value for Minutes	R/W
0xE0024068	ALHOUR	5	Alarm value for Hours	R/W
0xE002406C	ALDOM	5	Alarm value for Day of Month	R/W
0xE0024070	ALDOW	3	Alarm value for Day of Week	R/W
0xE0024074	ALDOY	9	Alarm value for Day of Year	R/W
0xE0024078	ALMON	4	Alarm value for Months	R/W
0xE002407C	ALYEAR	12	Alarm value for Years	R/W

RTC USAGE NOTES

If the RTC is used, V_{bat} must be connected to either pin V_3 or an independent power supply (external battery). Otherwise, V_{bat} should be tied to the ground (V_{ss}). No provision is made in the LPC2131/2132/2138 to retain RTC status upon the V_{bat} power loss, or to maintain time incrementation if the clock source is lost, interrupted, or altered.

Since the RTC operates using one of two available clocks (the VPB clock (pclk) or the 32 kHz signal coming from the RTCX1-2 pins), any interruption of the selected clock will cause the time to drift away from the time value it would have provided otherwise. The variance could be to actual clock time if the RTC was initialized to that, or simply an error in elapsed time since the RTC was activated.

While the signal from RTCX1-2 pins can be used to supply the RTC clock at anytime, selecting the pclk as the RTC clock and entering the Power Down mode will cause a lapse in the time update. Also, feeding the RTC with the pclk and altering this timebase during system operation (by reconfiguring the PLL, the VPB divider, or the RTC prescaler) will result in some form of accumulated time error. Accumulated time errors may occur in case RTC clock source is switched between the pclk to the RTCX pins, too.

Once the 32 kHz signal from RTCX1-2 pins is selected as a clock source, the RTC can operate completely without the presence of the VPB clock (pclk). Therefore, power sensitive applications (i.e. battery powered application) utilizing the RTC will reduce the power consumption by using the signal from RTCX1-2 pins, and writing a 0 into the PCRTC bit in the PCONP power control register (see "Power Control" section in the "System Control Block" chapter).

REFERENCE CLOCK DIVIDER (PRESCALER)

The reference clock divider (hereafter referred to as the Prescaler) allows generation of a 32.768 kHz reference clock from any peripheral clock frequency greater than or equal to 65.536 kHz (2 x 32.768 kHz). This permits the RTC to always run at the proper rate regardless of the peripheral clock rate. Basically, the Prescaler divides the peripheral clock (pclk) by a value which contains both an integer portion and a fractional portion. The result is not a continuous output at a constant frequency, some clock periods will be one pclk longer than others. However, the overall result can always be 32,768 counts per second.

The reference clock divider consists of a 13-bit integer counter and a 15-bit fractional counter. The reasons for these counter sizes are as follows:

1. For frequencies that are expected to be supported by the LPC2131/2132/2138, a 13-bit integer counter is required. This can be calculated as 160 MHz divided by 32,768 minus 1 = 4881 with a remainder of 26,624. Thirteen bits are needed to hold the value 4881, but actually supports frequencies up to 268.4 MHz (32,768 x 8192).
2. The remainder value could be as large as 32,767, which requires 15 bits.

Table 163: Reference Clock Divider registers

Address	Name	Size	Description	Access
0xE0024080	PREINT	13	Prescale Value, integer portion	R/W
0xE0024084	PREFRAC	15	Prescale Value, fractional portion	R/W

Prescaler Integer Register (PREINT - 0xE0024080)

This is the integer portion of the prescale value, calculated as:

PREINT = int (pclk / 32768) - 1. The value of PREINT must be greater than or equal to 1.

Table 164: Prescaler Integer Register (PREINT - 0xE0024080)

PREINT	Function	Description	Reset Value
15:13	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
12:0	Prescaler Integer	Contains the integer portion of the RTC prescaler value.	0

Prescaler Fraction Register (PREFRAC - 0xE0024084)

This is the fractional portion of the prescale value, and may be calculated as:

PREFRAC = pclk - ((PREINT +1) x 32768).

Table 165: Prescaler Fraction Register (PREFRAC - 0xE0024084)

PREFRAC	Function	Description	Reset Value
15	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
14:0	Prescaler Fraction	Contains the fractional portion of the RTC prescaler value.	0

Example of Prescaler Usage

In a simplistic case, the pclk frequency is 65.537 kHz. So:

PREINT = int (pclk / 32768) - 1 = 1 and PREFRAC = pclk - ((PREINT +1) x 32768) = 1

With this prescaler setting, exactly 32,768 clocks per second will be provided to the RTC by counting 2 pclks 32,767 times, and 3 pclks once.

In a more realistic case, the pclk frequency is 10 MHz. Then,

PREINT = int (pclk / 32768) - 1 = 304 and PREFRAC = pclk - ((PREINT +1) x 32768) = 5,760.

In this case, 5,760 of the prescaler output clocks will be 306 (305+1) pclks long, the rest will be 305 pclks long.

In a similar manner, any pclk rate greater than 65.536 kHz (as long as it is an even number of cycles per second) may be turned into a 32 kHz reference clock for the RTC. The only caveat is that if PREFRAC does not contain a zero, then not all of the 32,768 per second clocks are of the same length. Some of the clocks are one pclk longer than others. While the longer pulses are distributed as evenly as possible among the remaining pulses, this "jitter" could possibly be of concern in an application that wishes to observe the contents of the Clock Tick Counter (CTC) directly.

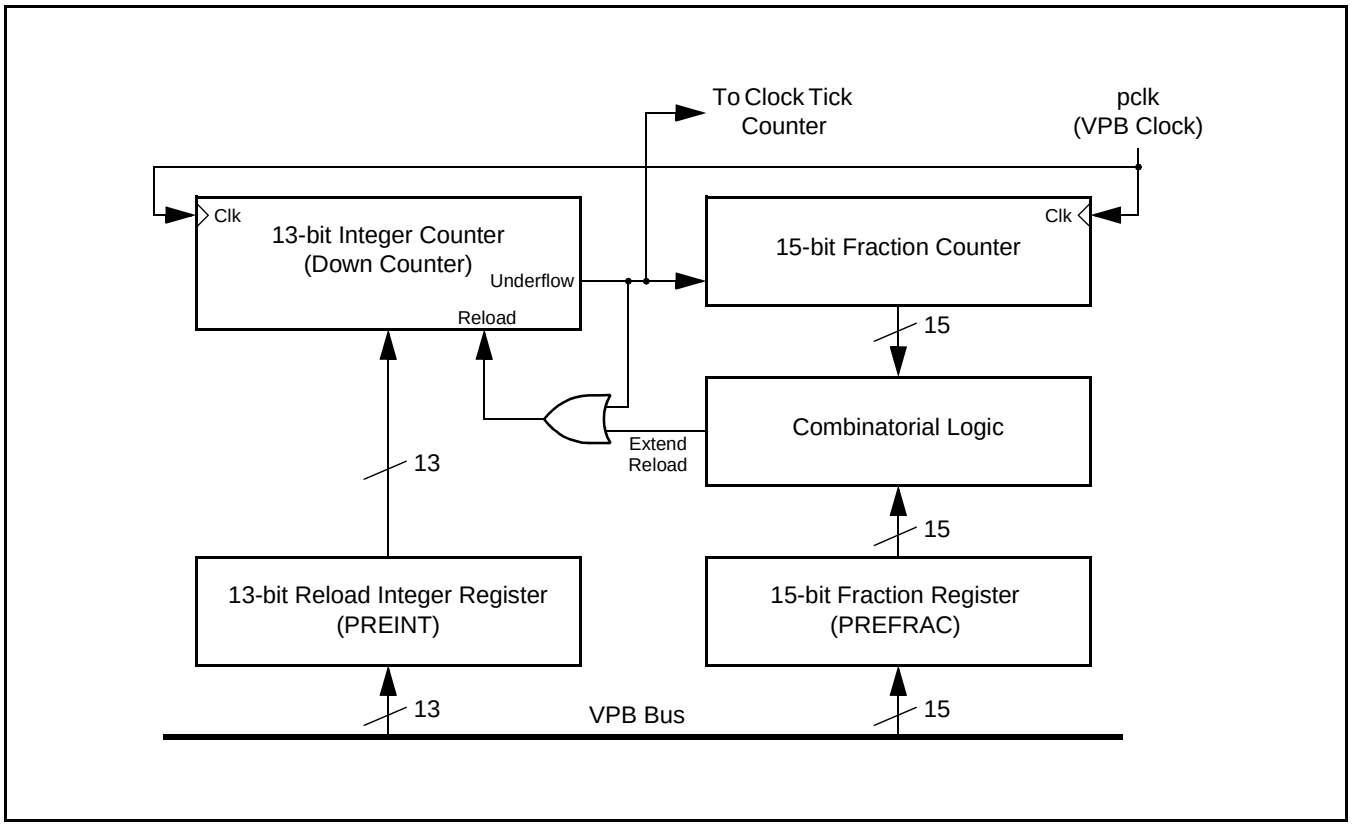


Figure 52: RTC Prescaler block diagram

Prescaler Operation

The Prescaler block labelled "Combination Logic" in Figure 52 determines when the decrement of the 13-bit PREINT counter is extended by one pclk. In order to both insert the correct number of longer cycles, and to distribute them evenly, the Combinatorial Logic associates each bit in PREFRAC with a combination in the 15-bit Fraction Counter. These associations are shown in the following Table 166.

For example, if PREFRAC bit 14 is a one (representing the fraction 1/2), then half of the cycles counted by the 13-bit counter need to be longer. When there is a 1 in the LSB of the Fraction Counter, the logic causes every alternate count (whenever the LSB of the Fraction Counter=1) to be extended by one pclk, evenly distributing the pulse widths. Similarly, a one in PREFRAC bit 13 (representing the fraction 1/4) will cause every fourth cycle (whenever the two LSBs of the Fraction Counter=10) counted by the 13-bit counter to be longer.

Table 166: Prescaler cases where the Integer Counter reload value is incremented

Fraction Counter	PREFRAC Bit														
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
--- ---- ---- ---1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- --10	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- -100	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- 1000	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---1 0000	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-
--- ---- --10 0000	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
--- ---- -100 0000	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-
--- ---- 1000 0000	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-
--- ---1 0000 0000	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-
--- --10 0000 0000	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
--- -100 0000 0000	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
--- 1000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-
--1 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-
-10 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-
100 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

19. WATCHDOG

FEATURES

- Internally resets chip if not periodically reloaded
- Debug mode
- Enabled by software but requires a hardware reset or a Watchdog reset/interrupt to be disabled
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled
- Flag to indicate Watchdog reset
- Programmable 32-bit timer with internal pre-scaler
- Selectable time period from (t_{pclk} x 256 x 4) to (t_{pclk} x 2^{32} x 4) in multiples of t_{pclk} x 4

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 "Reset" section of this document.

DESCRIPTION

The Watchdog consists of a divide by 4 fixed pre-scaler and a 32-bit counter. The clock is fed to the timer via a pre-scaler. The timer decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is (t_{pclk} x 256 x 4) and the maximum Watchdog interval is (t_{pclk} x 2^{32} x 4) in multiples of (t_{pclk} x 4). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup mode in WDMOD register.
- Start the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register.
- Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.

When the Watchdog counter underflows, the program counter will start from 0x00000000 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.

REGISTER DESCRIPTION

The Watchdog contains 4 registers as shown in Table 167 below.

Table 167: Watchdog Register Map

Name	Description	Access	Reset Value*	Address
WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	Read/Set	0	0xE0000000
WDTC	Watchdog timer constant register. This register determines the time-out value.	Read/Write	0xFF	0xE0000004
WDFEED	Watchdog feed sequence register. Writing AAh followed by 55h to this register reloads the Watchdog timer to its preset value.	Write Only	NA	0xE0000008
WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	Read Only	0xFF	0xE000000C

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

Watchdog Mode Register (WDMOD - 0xE0000000)

The WDMOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits.

WDEN	WDRESET	
0	X	Debug/Operate without the Watchdog running
1	0	Debug with the Watchdog interrupt but no WDRESET
1	1	Operate with the Watchdog interrupt and WDRESET

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.

Table 168: Watchdog Mode Register (WDMOD - 0xE0000000)

WDMOD	Function	Description	Reset Value
0	WDEN	Watchdog interrupt enable bit (Set only)	0
1	WDRESET	Watchdog reset enable bit (Set Only)	0
2	WDTOF	Watchdog time-out flag	0 (Only after external reset)
3	WDINT	Watchdog interrupt flag (Read Only)	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Watchdog Timer Constant Register (WDTC - 0xE0000004)

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. It's a 32-bit register with 8 LSB set to 1 on reset. Writing values below 0xFF will cause 0xFF to be loaded to the WDTC. Thus the minimum time-out interval is t_{pclk} x 256 x 4.

Table 169: Watchdog Constatnt Register (WDTC - 0xE0000004)

WDTC	Function	Description	Reset Value
31:0	Count	Watchdog time-out interval	0xFF

Watchdog Feed Register (WDFEED - 0xE0000008)

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer to the WDTC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must first be completed before the Watchdog is capable of generating an interrupt/reset. Until then, the Watchdog will ignore feed errors. Once 0xAA is written to the WDFEED register the next operation in the Watchdog register space should be a **WRITE** (0x55) to the WDFFED register otherwise the Watchdog is triggered. The interrupt/reset will be generated during the second **pclk** following an incorrect access to a watchdog timer register during a feed sequence.

Table 170: Watchdog Feed Register (WDFEED - 0xE0000008)

WDFEED	Function	Description	Reset Value
7:0	Feed	Feed value should be 0xAA followed by 0x55	undefined

Watchdog Timer Value Register (WDTV - 0xE000000C)

The WDTV register is used to read the current value of Watchdog timer.

Table 171: Watchdog Timer Value Register (WDTV - 0xE000000C)

WDTV	Function	Description	Reset Value
31:0	Count	Current timer value	0xFF

BLOCK DIAGRAM

The block diagram of the Watchdog is shown below in the Figure 53.

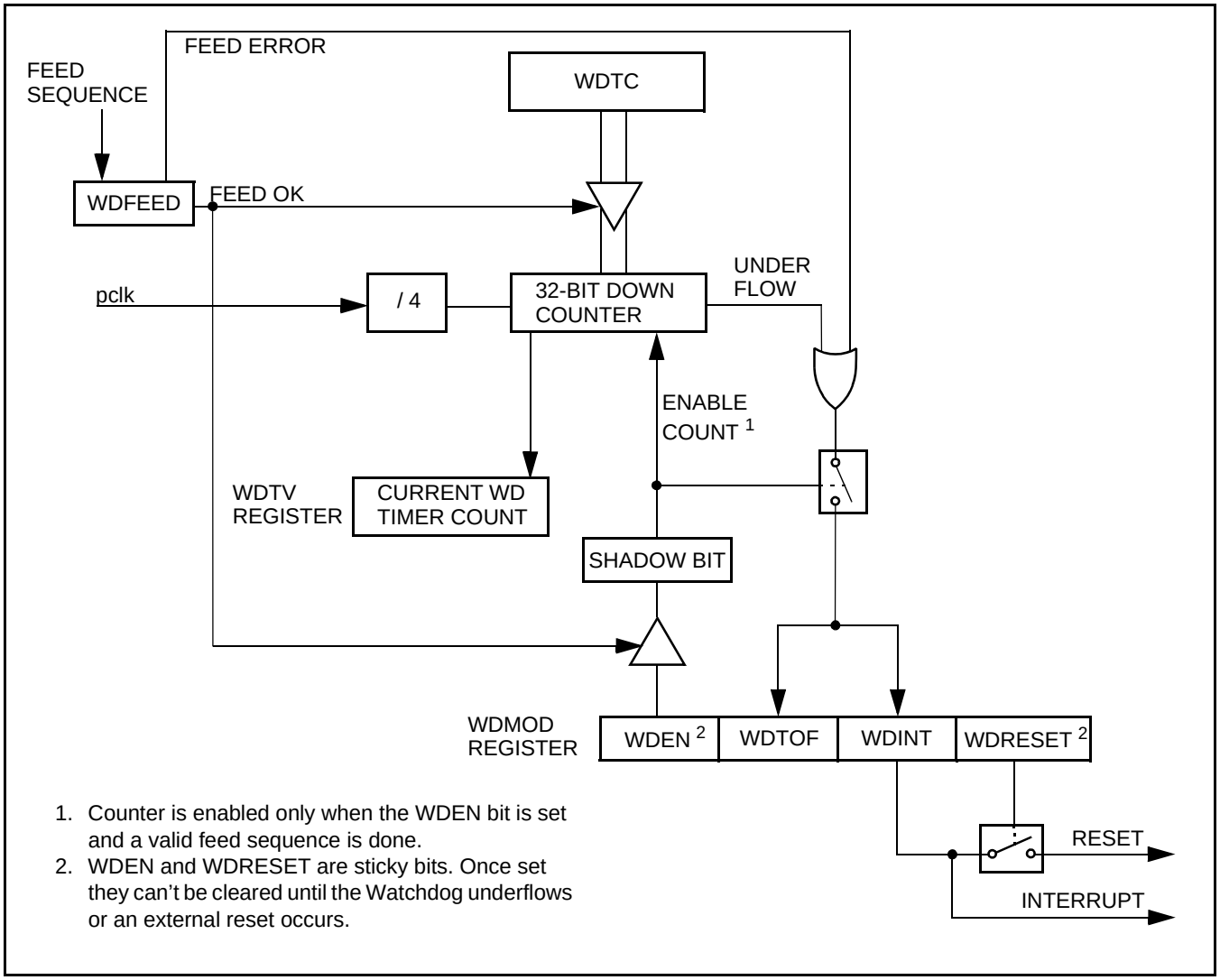


Figure 53: Watchdog Block Diagram

20. FLASH MEMORY SYSTEM AND PROGRAMMING

This chapter describes the Flash Boot Loader firmware, which includes In-System Programming (ISP) and In-Application Programming (IAP) interfaces.

The next chapter describes the Flash Programming Logic, which is the physical interface on the APB that actually accomplishes Flash programming, and is used by the Boot Loader to implement the ISP and IAP interfaces.

FLASH BOOT LOADER

The Boot Loader controls initial operation after reset, and also provides the means to accomplish programming of the Flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device, or programming of the Flash memory by the application program in a running system.

FEATURES

- In-System Programming: In-System programming (**ISP**) is programming or reprogramming the on-chip flash memory, using the boot loader software and a serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (**IAP**) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.

APPLICATIONS

The flash boot loader provides both In-System and In-Application programming interfaces for programming the on-chip flash memory.

DESCRIPTION

The flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A a LOW level after reset at the P0.14 pin is considered as an external hardware request to start the ISP command handler. Assuming that proper signal is present on X1 pin when the rising edge on RST pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on whether to continue with user code or ISP handler is made. If P0.14 is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P0.14 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

Pin P0.14 that is used as hardware request for ISP requires special attention. Since P0.14 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

Memory map after any reset:

The boot block is 12 kB in size and resides in the top portion (starting from 0x0007 D000) of the on-chip flash memory. After any reset the entire boot block is also mapped to the top of the on-chip memory space i.e. the boot block is also visible in the memory region starting from the address 0x7FFF D000. The flash boot loader is designed to run from this memory area but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in this chapter. The interrupt vectors residing in the boot block of the on-chip flash memory also become active after reset, i.e., the bottom 64 bytes of the boot block are also visible in the memory region starting from the address 0x0000 0000. The reset vector contains a jump instruction to the entry point of the flash boot loader software.

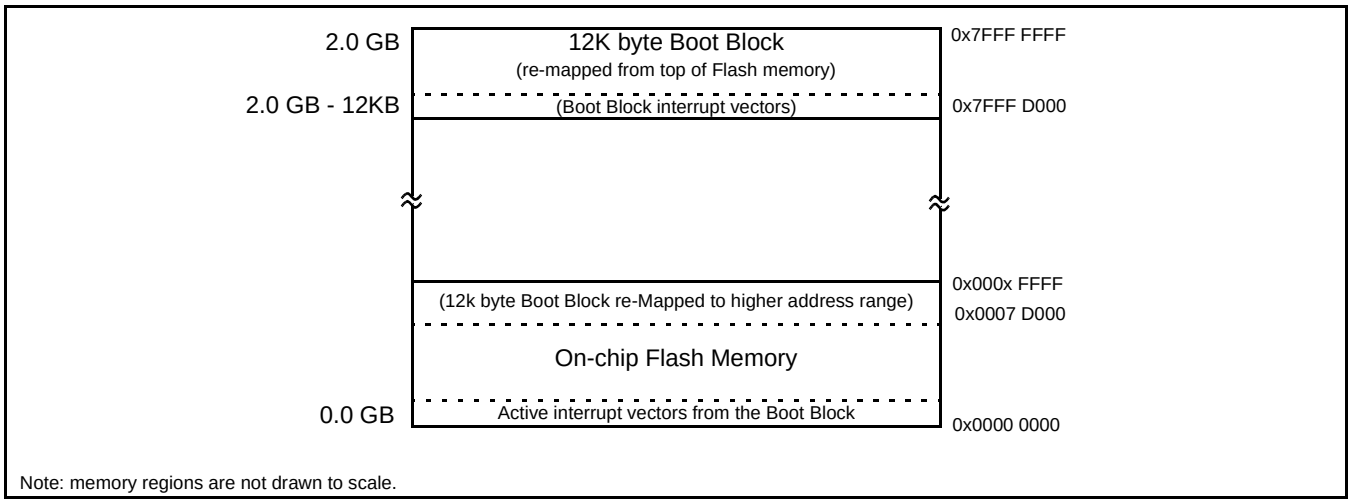


Figure 54: Map of lower memory after reset

Criterion for valid user code: The reserved ARM interrupt vector location (0x0000 0014) should contain the 2's complement of the check-sum of the remaining interrupt vectors. This causes the checksum of all of the vectors together to be 0. The boot loader code disables the overlaying of the interrupt vectors from the boot block, then checksums the interrupt vectors in sector 0 of the flash. If the signatures match then the execution control is transferred to the user code by loading the program counter with 0x 0000 0000. Hence the user flash reset vector should contain a jump instruction to the entry point of the user application code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. The host should send a '?' (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the Host. In response to this host should send the same string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. Host should respond by sending the crystal frequency (in KHz) at which the part is running. For example, if the part is running at 10 MHz , the response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly, the crystal frequency should be greater than or equal to 10 MHz. The on-chip PLL is not used by the boot code.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in "ISP Commands" section.

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.

ISP Command Format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands)

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)

ISP Data Format

The data stream is in UU-encode format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

A description of UU-encode is available at http://www.wotsit.org.

ISP Flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

ISP Command Abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

Interrupts during ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

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.

RAM used by ISP command handler

ISP commands use on-chip RAM from 0x4000 0120 to 0x4000 01FF. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 256 bytes and it grows downwards.

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.

RAM used by RealMonitor

The RealMonitor uses on-chip RAM from 0x4000 0040 to 0x4000 011F. he user could use this area if RealMonitor based debug is not required. The Flash boot loader does not initialize the stack for RealMonitor.

BOOT PROCESS FLOWCHART

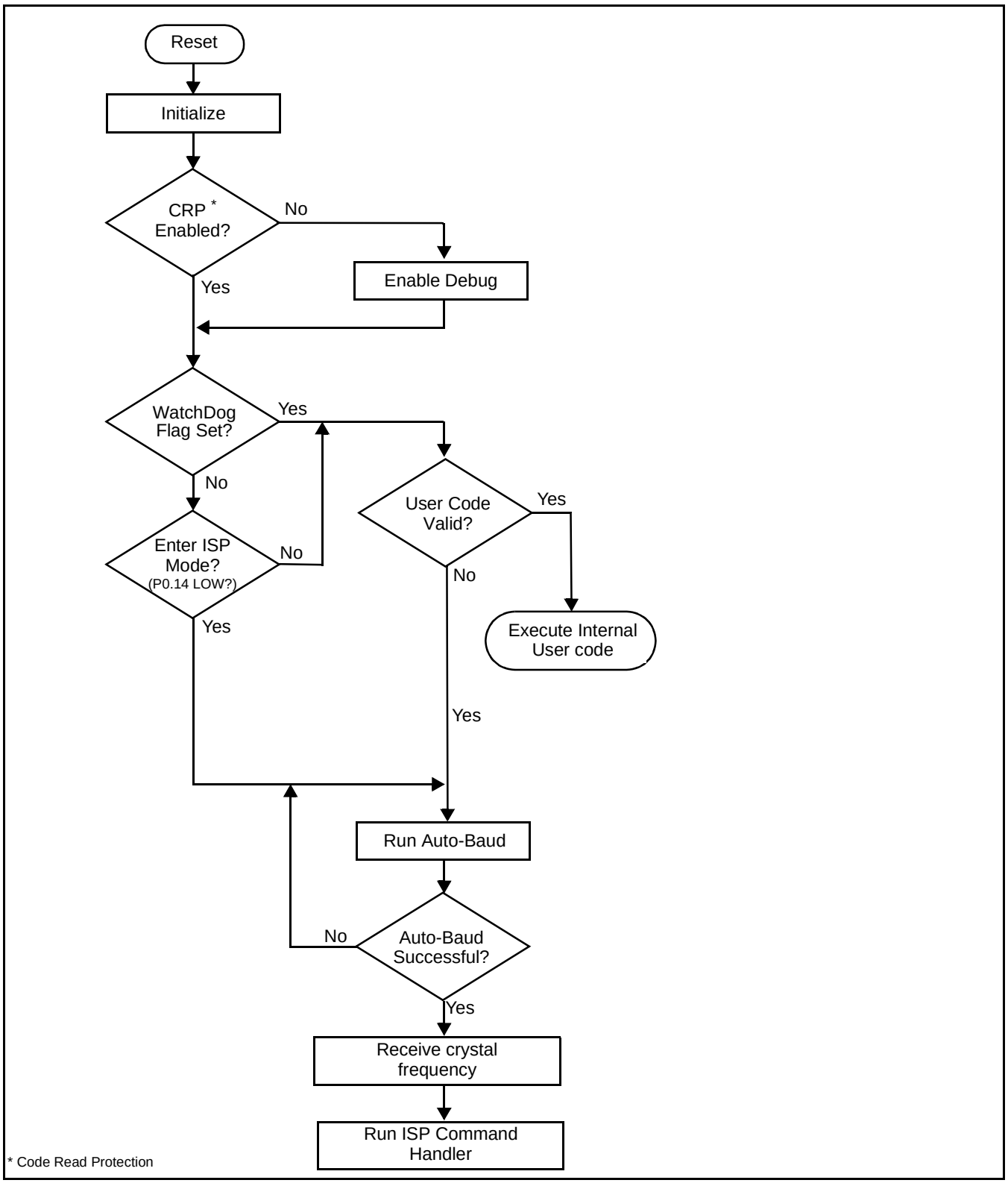


Figure 55: Boot Process flowchart

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 LPC2131/2132/2138 devices containing 32K, 64K and 512K bytes of Flash respectively. IAP, ISP, and RealMonitor routines are located in the boot block. The boot block is present at addresses 0007D000-0007FFFF in all devices. ISP and IAP commands do not allow write/erase/go operation on the boot block. Because of the boot block, the amount of Flash available for user code and data is 500K bytes in "512K" devices. On the other hand, in case of the LPC2131/2132 microcontroller all 32/64K of Flash are available for user's application.

Table 172: Flash sectors in LPC2131, LPC2132 and LPC2138

Sector Number	Sector size [kB]	Memory Addresses		
		LPC2131 (32 kB Flash)	LPC2132 (64 kB Flash)	LPC2138 (512 kB Flash)
0	4	0x0000 0000 - 0FFF	0x0000 0000 - 0FFF	0x0000 0000 - 0FFF
1	4	0x0000 1000 - 1FFF	0x0000 1000 - 1FFF	0x0000 1000 - 1FFF
2	4	0x0000 2000 - 2FFF	0x0000 2000 - 2FFF	0x0000 2000 - 2FFF
3	4	0x0000 3000 - 3FFF	0x0000 3000 - 3FFF	0x0000 3000 - 3FFF
4	4	0x0000 4000 - 4FFF	0x0000 4000 - 4FFF	0x0000 4000 - 4FFF
5	4	0x0000 5000 - 5FFF	0x0000 5000 - 5FFF	0x0000 5000 - 5FFF
6	4	0x0000 6000 - 6FFF	0x0000 6000 - 6FFF	0x0000 6000 - 6FFF
7	4	0x0000 7000 - 7FFF	0x0000 7000 - 7FFF	0x0000 7000 - 7FFF
8	32	-	0x0000 8000 - FFFF	0x0000 8000 - FFFF
9	32		-	0x0001 0000 - 7FFF
10 (0x0A)	32			0x0001 8000 - FFFF
11 (0x0B)	32			0x0002 0000 - 7FFF
12 (0x0C)	32			0x0002 8000 - FFFF
13 (0x0D)	32			0x0003 0000 - 7FFF
14 (0x0E)	32			0x0003 8000 - FFFF
15 (0x0F)	32			0x0004 0000 - 7FFF
16 (0x10)	32			0x0004 8000 - FFFF
17 (0x11)	32			0x0005 0000 - 7FFF
18 (0x12)	32			0x0005 8000 - FFFF
19 (0x13)	32			0x0006 0000 - 7FFF
20 (0x14)	32			0x0006 8000 -FFFF
21 (0x15)	32			0x0007 0000 - 7FFF
22 (0x16)	4			0x0007 8000 - 8FFF
23 (0x17)	4			0x0007 9000 - 9FFF
24 (0x18)	4			0x0007 A000 - AFFF
25 (0x19)	4			0x0007 B000 - BFFF
26 (0x1A)	4			0x0007 C000 - CFFF

FLASH CONTENT PROTECTION MECHANISM

The LPC2131/2132/2138 is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user's code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 0003 are protected by the first ECC byte, Flash bytes from 0x0000 0004 to 0x0000 0007 are protected by the second ECC byte, etc.

Whenever the CPU requests a read from user's Flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user's Flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of user's Flash memory is erased, corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the Flash memory in groups of 4 bytes (or multiples of 4), aligned as described above.

CODE READ PROTECTION

Code read protection is enabled by programming the flash address location 0x1FC (User flash sector 0) with value 0x87654321 (2271560481 Decimal). Address 0x1FC is used to allow some room for the fiq exception handler. When the code read protection is enabled the JTAG debug port, external memory boot and the following ISP commands are disabled:

- Read Memory
- Write to RAM
- Go
- Copy RAM to Flash

The ISP commands mentioned above terminate with return code CODE_READ_PROTECTION_ENABLED.

The ISP erase command only allows erasure of all user sectors when the code read protection is enabled. This limitation does not exist if the code read protection is not enabled. IAP commands are not affected by the code read protection.

Important: code read protection is active/inactive once the device has gone through a power cycle.

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 173: ISP Command Summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 174
Set Baud Rate	B <Baud Rate> <stop bit>	Table 175
Echo	A <setting>	Table 177
Write to RAM	W <start address> <number of bytes>	Table 178
Read Memory	R <address> <number of bytes>	Table 179
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 180
Copy RAM to Flash	C <Flash address> <RAM address> <number of bytes>	Table 181
Go	G <address> <Mode>	Table 182
Erase sector(s)	E <start sector number> <end sector number>	Table 183
Blank check sector(s)	I <start sector number> <end sector number>	Table 184
Read Part ID	J	Table 185
Read Boot code version	K	Table 186
Compare	M <address1> <address2> <number of bytes>	Table 187

Unlock <Unlock code>

Table 174: ISP Unlock command

Command	U
Input	Unlock code: 23130
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.

Set Baud Rate <Baud Rate> <stop bit>

Table 175: 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 176: Correlation between possible ISP baudrates and external crystal frequency (in MHz)

ISP Baudrate .vs. External Crystal Frequency	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

Echo <setting>

Table 177: 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.

Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after

transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 178: ISP Write to RAM command

Command	W
Input	Start Address: RAM address where data bytes are to be written. This address should be a word boundary. Number of Bytes: Number of bytes to be written. Count should be a multiple of 4.
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. Data should be in UU-encoded format. This command is blocked when code read protection is enabled.
Example	"W 1073742336 4<CR><LF>" writes 4 bytes of data to address 0x4000 0200.

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 check-sum 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 179: ISP Read Memory command

Command	R
Input	Start Address: Address from where data bytes are to be read. This address should be a word boundary. Number of Bytes: Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (UU-encoded)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or Flash memory. This command is blocked when code read protection is enabled.
Example	"R 1073741824 4<CR><LF>" reads 4 bytes of data from address 0x4000 0000.

Prepare sector(s) for write operation <start sector number> <end sector number>

This command makes flash write/erase operation a two step process.

Table 180: ISP Prepare sector(s) for write operation command

Command	P
Input	Start Sector Number End Sector Number: Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot block can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the flash sector 0.

Copy RAM to Flash <Flash address> <RAM address> <no of bytes>

Table 181: ISP Copy command

Command	C
Input	Flash Address(DST): Destination Flash address where data bytes are to be written. The destination address should be a 256 byte boundary. RAM Address(SRC): Source RAM address from where data bytes are to be read. Number of Bytes: Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot block cannot be written by this command. This command is blocked when code read protection is enabled.
Example	"C 0 1073774592 512<CR><LF>" copies 512 bytes from the RAM address 0x4000 8000 to the flash address 0.

Go <address> <mode>

Table 182: 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. Instead of address if string "tEsT" is entered the program residing in reserved test area will be executed. Mode: T (Execute program in Thumb Mode) | A (Execute program in ARM mode)
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or Flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled.
Example	"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.

Erase sector(s) <start sector number> <end sector number>

Table 183: ISP Erase sector command

Command	E
Input	Start Sector Number End Sector Number: Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip Flash memory. The boot block can not be erased using this command. This command only allows erasure of all user sectors when the code read protection is enabled.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

Blank check sector(s) <sector number> <end sector number>

Table 184: ISP Blank check sector command

Command	I
Input	Start Sector Number End Sector Number: Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check one or more sectors of on-chip Flash memory. **Blank check on sector 0 always fails as first 64 bytes are re-mapped to flash boot block.**
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

Read Part Identification number

Table 185: ISP Read Part Identification command

Command	J
Input	None
Return Code	CMD_SUCCESS followed by part identification number in ASCII
Description	This command is used to read the part identification number.

Read Boot code version number

Table 186: 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.

Compare <address1> <address2> <no of bytes>

Table 187: ISP Compare command

Command	M
Input	Address1 (DST): Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. Address2 (SRC): Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. Number of Bytes: Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR
Description	This command is used to compare the memory contents at two locations. **Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. First 64 bytes are re-mapped to flash boot sector.**
Example	"M 8192 1073741824 4<CR><LF>" compares 4 bytes from the RAM address 0x4000 0000 to the 4 bytes from the flash address 0x2000.

Table 188: ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy
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.

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 56. The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 2, returned by the "Blank

check sector(s)" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 0x7FFFFFF0 location and it is thumb code.

The IAP function could be called in the following way using C.

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x7ffffff1
```

Define data structure or pointers to pass IAP command table and result table to the IAP function

```
unsigned long command[5];

unsigned long result[2];
```

or

```
unsigned long * command;

unsigned long * result;

command=(unsigned long *) 0x……

result= (unsigned long *) 0x……
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);

IAP iap_entry;
```

Setting function pointer

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, result);
```

The IAP call could be simplified further by using the symbol definition file feature supported by ARM Linker in ADS (ARM Developer Suite). You could also call the IAP routine using assembly code.

The following symbol definitions can be used to link IAP routine and user application:

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
0x7fffffb0 A rm_prefetchabort_handler
0x7fffffc0 A rm_dataabort_handler
0x7fffffd0 A rm_irqhandler
0x7fffffe0 A rm_irqhandler2
0x7ffffff0 T iap_entry
```

As per the ARM specification (The ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05) up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not be use this space if IAP flash programming is permitted in the application.

Table 189: IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50	Table 190
Copy RAM to Flash	51	Table 191
Erase sector(s)	52	Table 192
Blank check sector(s)	53	Table 193
Read Part ID	54	Table 194
Read Boot code version	55	Table 195
Compare	56	Table 197
Reinvoke ISP	57	

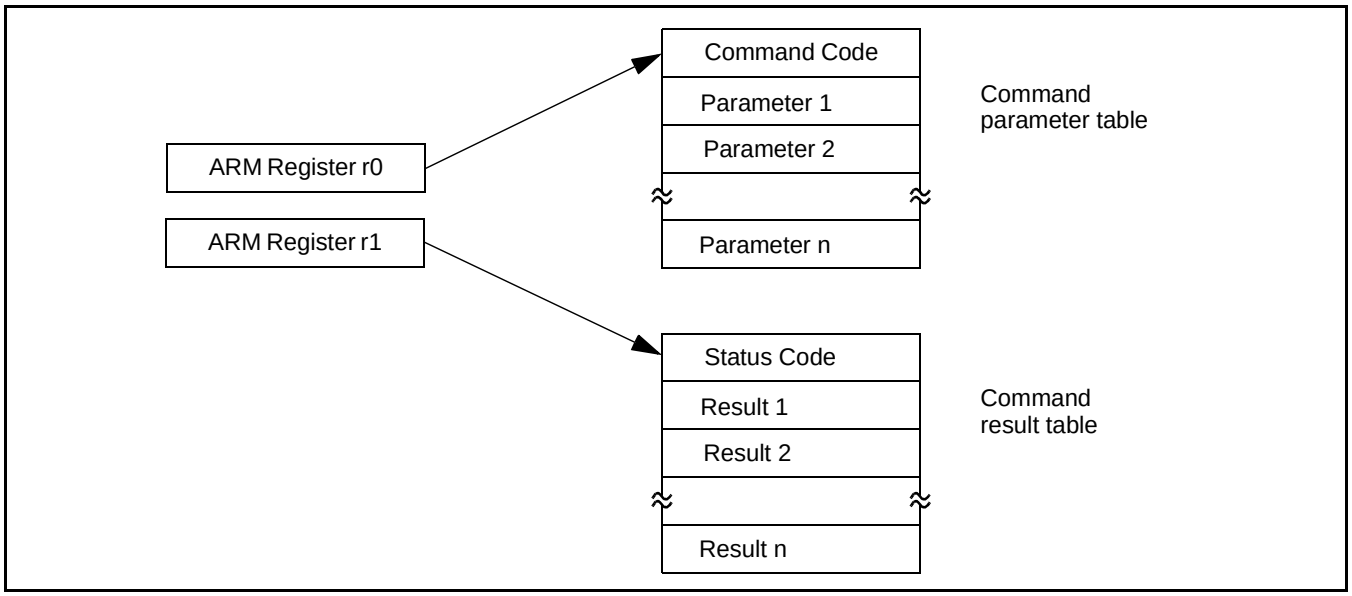


Figure 56: IAP Parameter passing

Prepare sector(s) for write operation

This command makes flash write/erase operation a two step process.

Table 190: IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Input	Command code: 50 Param0: Start Sector Number Param1: End Sector Number: Should be greater than or equal to start sector number.
Status Code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.

Copy RAM to Flash

Table 191: IAP Copy RAM to Flash command

Command	Copy RAM to Flash
Input	Command code: 51_{10} Param0(DST): Destination Flash address where data bytes are to be written. This address should be a 256 byte boundary. Param1(SRC): Source RAM address from which data bytes are to be read. This address should be a word boundary. Param2: Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. Param3: System Clock Frequency (CCLK) in KHz.
Status Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command.

Erase Sector(s)

Table 192: IAP Erase Sector(s) command

Command	Erase Sector(s)
Input	Command code: 52_{10} Param0: Start Sector Number Param1: End Sector Number: Should be greater than or equal to start sector number. Param2: System Clock Frequency (CCLK) in KHz.
Status Code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip Flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers.

Blank check sector(s)

Table 193: 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 the starting sector)
Status 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.

Read Part Identification number

Table 194: IAP Read Part Identification command

Command	Read part identification number
Input	Command Code: 54_{10} parameters: None
Status Code	CMD_SUCCESS
Result	Result0: Part Identification Number
Description	This command is used to read the part identification number.

Read Boot code version number

Table 195: IAP Read Boot Code version number command

Command	Read boot code version number
Input	Command code: 55_{10} Parameters: None
Status 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.

Compare <address1> <address2> <no of bytes>

Table 196: 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.
Status 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.**

Reinvoke ISP

Table 197: Reinvoke ISP

Command	Reinvoke ISP
Input	Command Code: 57_{10}
Status Code	None.
Result	None.
Description	This command is used to invoke the bootloader in ISP mode. This command maps boot vectors, configures P0.1 as an input and sets the vpb divider register to 0 before entering the ISP mode. This command may be used when a valid user program is present in the internal flash memory and the P0.14 pin is not accessible to force the ISP mode. This command does not disable the PLL hence it is possible to invoke the bootloader when the part is running off the PLL. In such case the ISP utility should pass the PLL frequency after autobaud handshake. Another option is to disable the PLL before making this IAP call.

Table 198: 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.

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.

21. EMBEDDEDICE LOGIC

FEATURES

- No target resources are required by the software debugger in order to start the debugging session
- Allows the software debugger to talk via a JTAG (Joint Test Action Group) port directly to the core
- Inserts instructions directly in to the ARM7TDMI-S core
- The ARM7TDMI-S core or the System state can be examined, saved or changed depending on the type of instruction inserted
- Allows instructions to execute at a slow debug speed or at a fast system speed

APPLICATIONS

The EmbeddedICE logic provides on-chip debug support. The debugging of the target system requires a host computer running the debugger software and an EmbeddedICE protocol convertor. EmbeddedICE protocol convertor converts the Remote Debug Protocol commands to the JTAG data needed to access the ARM7TDMI-S core present on the target system.

DESCRIPTION

The ARM7TDMI-S Debug Architecture uses the existing JTAG* port as a method of accessing the core. The scan chains that are around the core for production test are reused in the debug state to capture information from the databus and to insert new information into the core or the memory. There are two JTAG-style scan chains within the ARM7TDMI-S. A JTAG-style Test Access Port Controller controls the scan chains. In addition to the scan chains, the debug architecture uses EmbeddedICE logic which resides on chip with the ARM7TDMI-S core. The EmbeddedICE has its own scan chain that is used to insert watchpoints and breakpoints for the ARM7TDMI-S core. The EmbeddedICE logic consists of two real time watchpoint registers, together with a control and status register. One or both of the watchpoint registers can be programmed to halt the ARM7TDMI-S core. Execution is halted when a match occurs between the values programmed into the EmbeddedICE logic and the values currently appearing on the address bus, databus and some control signals. Any bit can be masked so that its value does not affect the comparison. Either watchpoint register can be configured as a watchpoint (i.e. on a data access) or a break point (i.e. on an instruction fetch). The watchpoints and breakpoints can be combined such that:

- The conditions on both watchpoints must be satisfied before the ARM7TDMI core is stopped. The CHAIN functionality requires two consecutive conditions to be satisfied before the core is halted. An example of this would be to set the first breakpoint to trigger on an access to a peripheral and the second to trigger on the code segment that performs the task switching. Therefore when the breakpoints trigger the information regarding which task has switched out will be ready for examination.
- The watchpoints can be configured such that a range of addresses are enabled for the watchpoints to be active. The RANGE function allows the breakpoints to be combined such that a breakpoint is to occur if an access occurs in the bottom 256 bytes of memory but not in the bottom 32 bytes.

The ARM7TDMI-S core has a Debug Communication Channel function in-built. The debug communication channel allows a program running on the target to communicate with the host debugger or another separate host without stopping the program flow or even entering the debug state. The debug communication channel is accessed as a co-processor 14 by the program running on the ARM7TDMI-S core. The debug communication channel allows the JTAG port to be used for sending and receiving data without affecting the normal program flow. The debug communication channel data and control registers are mapped in to addresses in the EmbeddedICE logic.

* For more details refer to IEEE Standard 1149.1 - 1990 Standard Test Access Port and Boundary Scan Architecture.

PIN DESCRIPTION

Table 199: EmbeddedICE Pin Description

Pin Name	Type	Description
TMS	Input	**Test Mode Select.** The TMS pin selects the next state in the TAP state machine.
TCK	Input	**Test Clock.** This allows shifting of the data in, on the TMS and TDI pins. It is a positive edge-triggered clock with the TMS and TCK signals that define the internal state of the device.
TDI	Input	**Test Data In.** This is the serial data input for the shift register.
TDO	Output	**Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal
nTRST	Input	**Test Reset.** The nTRST pin can be used to reset the test logic within the EmbeddedICE logic.
RTCK	Output	**Returned Test Clock.** Extra signal added to the JTAG port. Required for designs based on ARM7TDMI-S processor core. Multi-ICE (Development system from ARM) uses this signal to maintain synchronization with targets having slow or widely varying clock frequency. For details refer to "Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)".

RESET STATE OF MULTIPLEXED PINS

On the LPC2131/2132/2138, the pins above are multiplexed with P1.31-26. To have them come up as a Debug port, connect a weak bias resistor (4.7-10 kΩ depending on the external JTAG circuitry) between VSS and the P1.26/RTCK pin. To have them come up as GPIO pins, do not connect a bias resistor, and ensure that any external driver connected to P1.26/RTCK is either driving high, or is in high-impedance state, during Reset.

REGISTER DESCRIPTION

The EmbeddedICE logic contains 16 registers as shown in Table 200. below. The ARM7TDMI-S debug architecture is described in detail in "ARM7TDMI-S (rev 4) Technical Reference Manual" (ARM DDI 0234A) published by ARM Limited and is available via Internet at http://www.arm.com.

Table 200: EmbeddedICE Logic Registers

Name	Width	Description	Address
Debug Control	6	Force debug state, disable interrupts	00000
Debug Status	5	Status of debug	00001
Debug Comms Control Register	32	Debug communication control register	00100
Debug Comms Data Register	32	Debug communication data register	00101
Watchpoint 0 Address Value	32	Holds watchpoint 0 address value	01000
Watchpoint 0 Address Mask	32	Holds watchpoint 0 address mask	01001
Watchpoint 0 Data Value	32	Holds watchpoint 0 data value	01010
Watchpoint 0 Data Mask	32	Holds watchpoint 0 data Mask	01011
Watchpoint 0 Control Value	9	Holds watchpoint 0 control value	01100
Watchpoint 0 Control Mask	8	Holds watchpoint 0 control mask	01101
Watchpoint 1 Address Value	32	Holds watchpoint 1 address value	10000
Watchpoint 1 Address Mask	32	Holds watchpoint 1 address mask	10001
Watchpoint 1 Data Value	32	Holds watchpoint 1 data value	10010
Watchpoint 1 Data Mask	32	Holds watchpoint 1 data Mask	10011
Watchpoint 1 Control Value	9	Holds watchpoint 1 control value	10100
Watchpoint 1 Control Mask	8	Holds watchpoint 1 control mask	10101

BLOCK DIAGRAM

The block diagram of the debug environment is shown below in Figure 57.

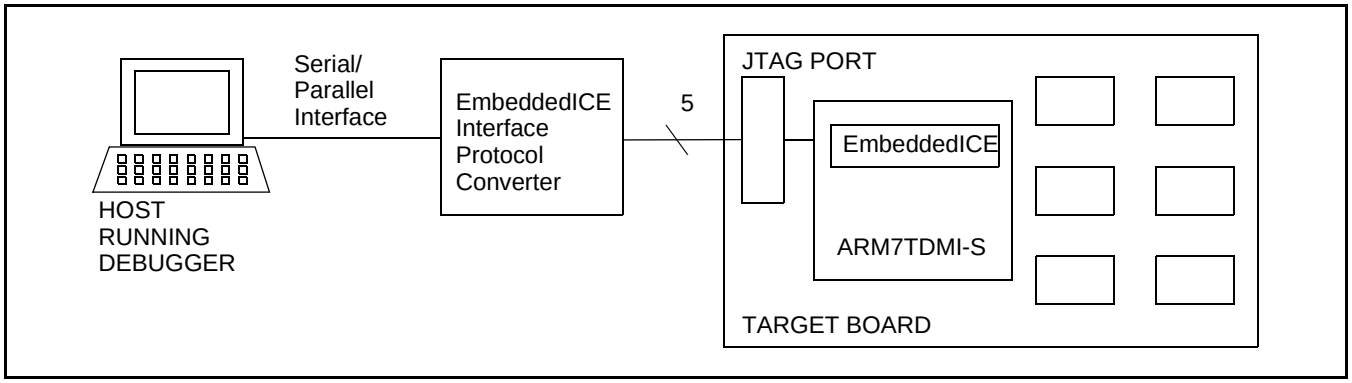


Figure 57: EmbeddedICE Debug Environment Block Diagram

22. EMBEDDED TRACE MACROCELL

FEATURES

- Closely track the instructions that the ARM core is executing
- 10 pin interface
- 1 External trigger input
- All registers are programmed through JTAG interface
- Does not consume power when trace is not being used
- THUMB instruction set support

APPLICATIONS

As the microcontroller has significant amounts of on-chip memories, it is not possible to determine how the processor core is operating simply by observing the external pins. The ETM provides real-time trace capability for deeply embedded processor cores. It outputs information about processor execution to a trace port. A software debugger allows configuration of the ETM using a JTAG interface and displays the trace information that has been captured, in a format that a user can easily understand.

DESCRIPTION

The ETM is connected directly to the ARM core and not to the main AMBA system bus. It compresses the trace information and exports it through a narrow trace port. An external Trace Port Analyzer captures the trace information under software debugger control. Trace port can broadcast the Instruction trace information. Instruction trace (or PC trace) shows the flow of execution of the processor and provides a list of all the instructions that were executed. Instruction trace is significantly compressed by only broadcasting branch addresses as well as a set of status signals that indicate the pipeline status on a cycle by cycle basis. Trace information generation can be controlled by selecting the trigger resource. Trigger resources include address comparators, counters and sequencers. Since trace information is compressed the software debugger requires a static image of the code being executed. Self-modifying code can not be traced because of this restriction.

ETM Configuration

The following standard configuration is selected for the ETM macrocell.

Table 201: ETM Configuration

Resource number/type	Small [1]
Pairs of address comparators	1
Data Comparators	0 (Data tracing is not supported)
Memory Map Decoders	4
Counters	1
Sequencer Present	No
External Inputs	2

Table 201: ETM Configuration

Resource number/type	Small [1]
External Outputs	0
FIFOFULL Present	Yes (Not wired)
FIFO depth	10 bytes
Trace Packet Width	4/8

1. For details refer to ARM documentation "Embedded Trace Macrocell Specification (ARM IHI 0014E)".

PIN DESCRIPTION

Table 202: ETM Pin Description

Pin Name	Type	Description
TRACECLK	Output	**Trace Clock.** The trace clock signal provides the clock for the trace port. PIPESTAT[2:0], TRACESYNC, and TRACEPKT[3:0] signals are referenced to the rising edge of the trace clock. This clock is not generated by the ETM block. It is to be derived from the system clock. The clock should be balanced to provide sufficient hold time for the trace data signals. Half rate clocking mode is supported. Trace data signals should be shifted by a clock phase from TRACECLK. Refer to Figure 3.14 page 3.26 and figure 3.15 page 3.27 in "ETM7 Technical Reference Manual" (ARM DDI 0158B), for example circuits that implements both half-rate-clocking and shifting of the trace data with respect to the clock. For TRACECLK timings refer to section 5.2 on page 5-13 in "Embedded Trace Macrocell Specification" (ARM IHI 0014E).
PIPESTAT[2:0]	Output	**Pipe Line status.** The pipeline status signals provide a cycle-by-cycle indication of what is happening in the execution stage of the processor pipeline.
TRACESYNC	Output	**Trace synchronization.** The trace sync signal is used to indicate the first packet of a group of trace packets and is asserted HIGH only for the first packet of any branch address.
TRACEPKT[3:0]	Output	**Trace Packet.** The trace packet signals are used to output packaged address and data information related to the pipeline status. All packets are eight bits in length. A packet is output over two cycles. In the first cycle, Packet[3:0] is output and in the second cycle, Packet[7:4] is output.
EXTIN[0]	Input	External Trigger Input.

RESET STATE OF MULTIPLEXED PINS

On the LPC2131/2132/2138, the ETM pin functions are multiplexed with P1.25-16. To have these pins come as a Trace port, connect a weak bias resistor (4.7 kΩ) between the P1.20/TRACESYNC pin and V_{SS}. To have them come up as port pins, do not connect a bias resistor to P1.20/TRACESYNC, and ensure that any external driver connected to P1.20/TRACESYNC is either driving high, or is in high-impedance state, during Reset.

REGISTER DESCRIPTION

The ETM contains 29 registers as shown in Table 203. below. They are described in detail in the ARM IHI 0014E document published by ARM Limited, which is available via the Internet at http://www.arm.com.

Table 203: ETM Registers

Name	Description	Access	Register encoding
ETM Control	Controls the general operation of the ETM	Read/Write	000 0000
ETM Configuration Code	Allows a debugger to read the number of each type of resource	Read Only	000 0001
Trigger Event	Holds the controlling event	Write Only	000 0010
Memory Map Decode Control	Eight-bit register, used to statically configure the memory map decoder	Write Only	000 0011
ETM Status	Holds the pending overflow status bit	Read Only	000 0100
System Configuration	Holds the configuration information using the SYSOPT bus	Read Only	000 0101
Trace Enable Control 3	Holds the trace on/off addresses	Write Only	000 0110
Trace Enable Control 2	Holds the address of the comparison	Write Only	000 0111
Trace Enable Event	Holds the enabling event	Write Only	000 1000
Trace Enable Control 1	Holds the include and exclude regions	Write Only	000 1001
FIFOFULL Region	Holds the include and exclude regions	Write Only	000 1010
FIFOFULL Level	Holds the level below which the FIFO is considered full	Write Only	000 1011
ViewData event	Holds the enabling event	Write Only	000 1100
ViewData Control 1	Holds the include/exclude regions	Write Only	000 1101
ViewData Control 2	Holds the include/exclude regions	Write Only	000 1110
ViewData Control 3	Holds the include/exclude regions	Write Only	000 1111
Address Comparator 1 to 16	Holds the address of the comparison	Write Only	001 xxxx
Address Access Type 1 to 16	Holds the type of access and the size	Write Only	010 xxxx
reserved	-	-	000 xxxx
reserved	-	-	100 xxxx
Initial Counter Value 1 to 4	Holds the initial value of the counter	Write Only	101 00xx
Counter Enable 1 to 4	Holds the counter clock enable control and event	Write Only	101 01xx
Counter reload 1 to 4	Holds the counter reload event	Write Only	101 10xx
Counter Value 1 to 4	Holds the current counter value	Read Only	101 11xx
Sequencer State and Control	Holds the next state triggering events.	-	110 00xx
External Output 1 to 4	Holds the controlling events for each output	Write Only	110 10xx
Reserved	-	-	110 11xx
Reserved	-	-	111 0xxx
Reserved	-	-	111 1xxx

BLOCK DIAGRAM

The block diagram of the ETM debug environment is shown below in Figure 58.

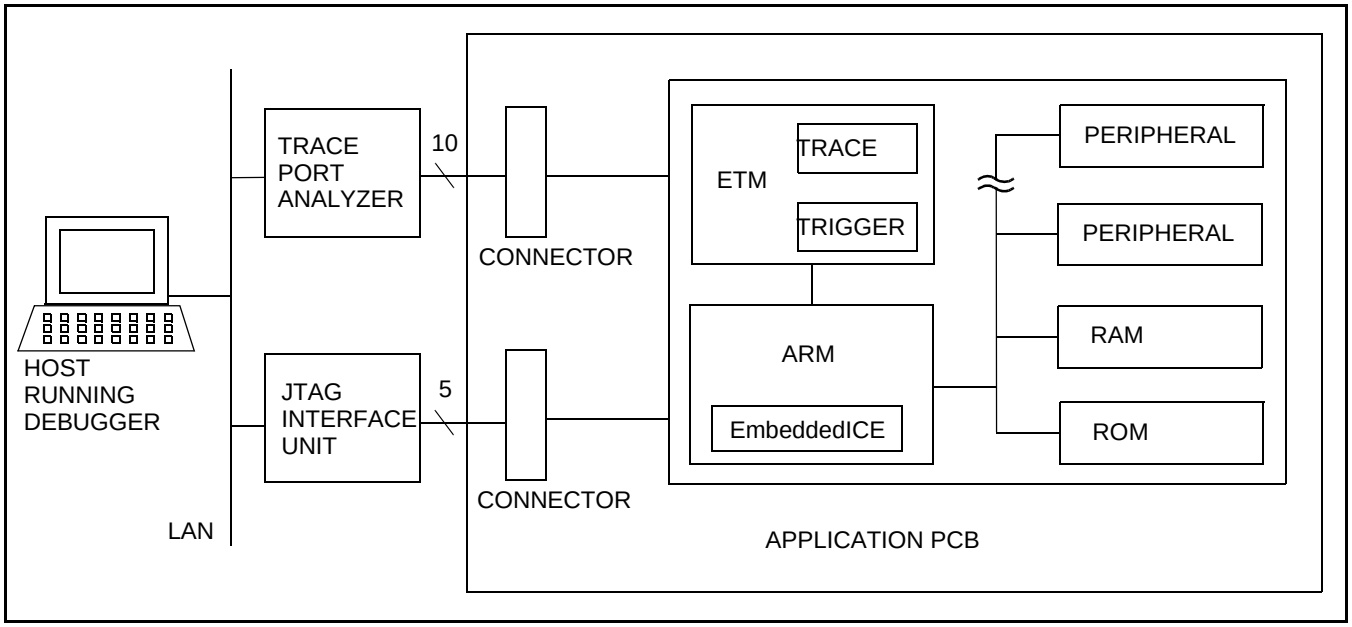


Figure 58: ETM Debug Environment Block Diagram

23. REALMONITOR

RealMonitor is a configurable software module which enables real time debug. RealMonitor is developed by ARM Inc. Information presented in this chapter is taken from the ARM document *RealMonitor Target Integration Guide (ARM DUI 0142A).* It applies to a specific configuration of RealMonitor software programmed in the on-chip ROM boot memory of this device.

Refer to the white paper *"Real Time Debug for System-on-Chip"* available at *http://www.arm.com/support/White_Papers?OpenDocument* for background information.

FEATURES

- Allows user to establish a debug session to a currently running system without halting or resetting the system.
- Allows user time-critical interrupt code to continue executing while other user application code is being debugged.

APPLICATIONS

Real time debugging.

DESCRIPTION

RealMonitor is a lightweight debug monitor that allows interrupts to be serviced while user debug their foreground application. It communicates with the host using the DCC (Debug Communications Channel), which is present in the EmbeddedICE logic. RealMonitor provides advantages over the traditional methods for debugging applications in ARM systems. The traditional methods include:

- Angel (a target-based debug monitor).
- Multi-ICE or other JTAG unit and EmbeddedICE logic (a hardware-based debug solution).

Although both of these methods provide robust debugging environments, neither is suitable as a lightweight real-time monitor.

Angel is designed to load and debug independent applications that can run in a variety of modes, and communicate with the debug host using a variety of connections (such as a serial port or ethernet). Angel is required to save and restore full processor context, and the occurrence of interrupts can be delayed as a result. Angel, as a fully functional target-based debugger, is therefore too heavyweight to perform as a real-time monitor.

Multi-ICE is a hardware debug solution that operates using the EmbeddedICE unit that is built into most ARM processors. To perform debug tasks such as accessing memory or the processor registers, Multi-ICE must place the core into a debug state. While the processor is in this state, which can be millions of cycles, normal program execution is suspended, and interrupts cannot be serviced.

RealMonitor combines features and mechanisms from both Angel and Multi-ICE to provide the services and functions that are required. In particular, it contains both the Multi-ICE communication mechanisms (the DCC using JTAG), and Angel-like support for processor context saving and restoring. RealMonitor is pre-programmed in the on-chip ROM memory (boot sector). When enabled It allows user to observe and debug while parts of application continue to run. Refer to section How to Enable RealMonitor for details.

RealMonitor Components

As shown in Figure 59, RealMonitor is split in to two functional components:

RMHost

This is located between a debugger and a JTAG unit. The RMHost controller, RealMonitor.dll, converts generic *Remote Debug Interface* (RDI) requests from the debugger into DCC-only RDI messages for the JTAG unit. For complete details on debugging a RealMonitor-integrated application from the host, see the *ARM RMHost User Guide (ARM DUI 0137A).*

RMTarget

This is pre-programmed in the on-chip ROM memory (boot sector), and runs on the target hardware. It uses the EmbeddedICE logic, and communicates with the host using the DCC. For more details on RMTarget functionality, see the *RealMonitor Target Integration Guide (ARM DUI 0142A).*

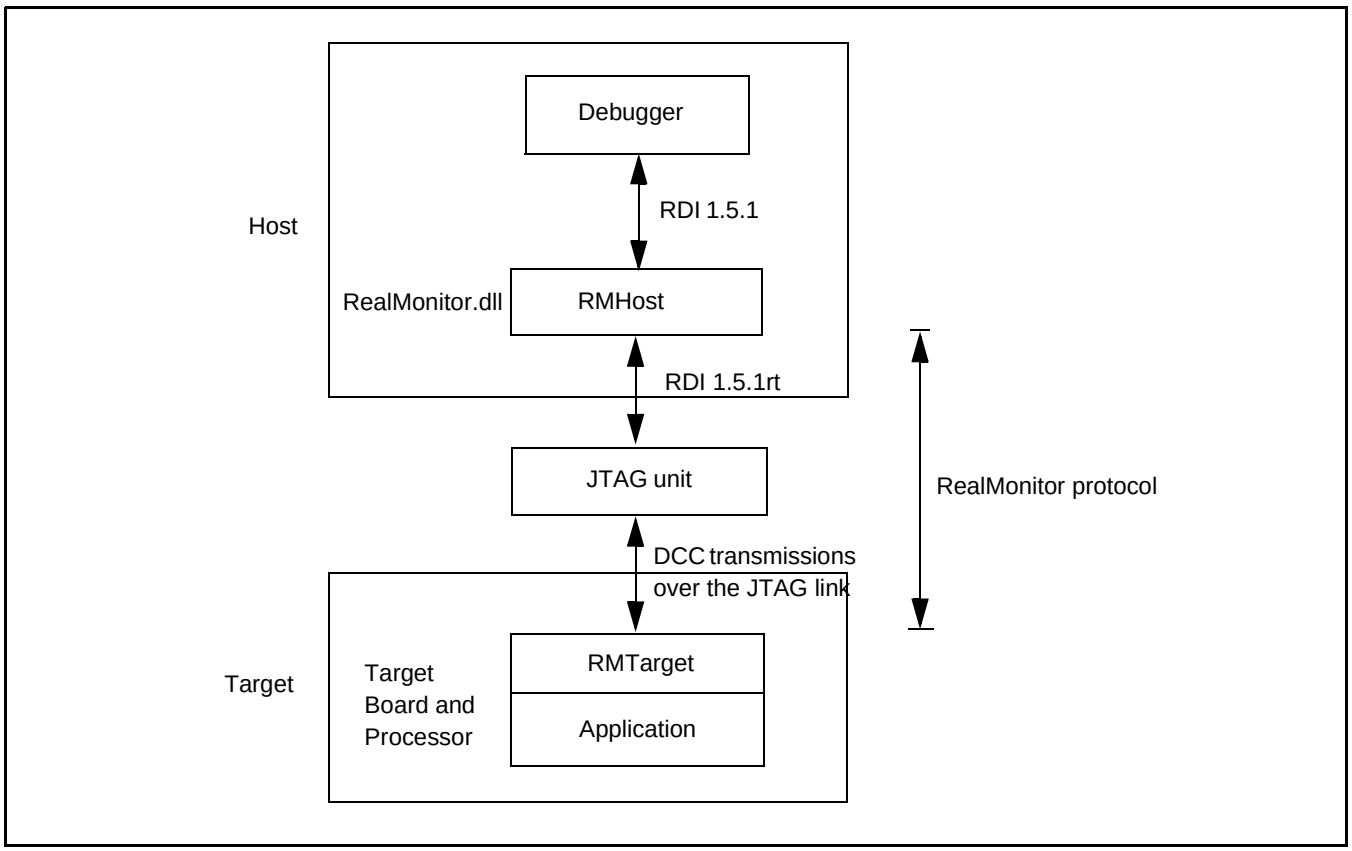


Figure 59: RealMonitor components

How RealMonitor works

In general terms, the RealMonitor operates as a state machine, as shown in Figure 60. RealMonitor switches between running and stopped states, in response to packets received by the host, or due to asynchronous events on the target. RMTarget supports the triggering of only one breakpoint, watchpoint, stop, or semihosting SWI at a time. There is no provision to allow nested events to be saved and restored. So, for example, if user application has stopped at one breakpoint, and another breakpoint occurs in an IRQ handler, RealMonitor enters a panic state. No debugging can be performed after RealMonitor enters this state.

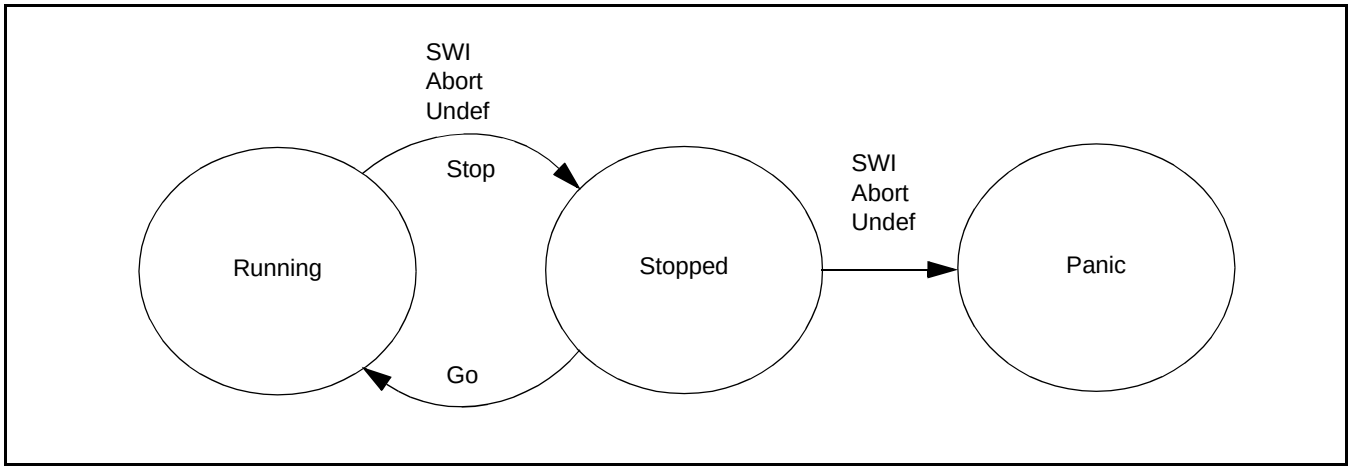


Figure 60: RealMonitor as a state machine

A debugger such as the ARM eXtended Debugger (AXD) or other RealMonitor aware debugger, that runs on a host computer, can connect to the target to send commands and receive data. This communication between host and target is illustrated in Figure 59.

The target component of RealMonitor, RMTarget, communicates with the host component, RMHost, using the Debug Communications Channel (DCC), which is a reliable link whose data is carried over the JTAG connection.

While user application is running, RMTarget typically uses IRQs generated by the DCC. This means that if user application also wants to use IRQs, it must pass any DCC-generated interrupts to RealMonitor.

To allow nonstop debugging, the EmbeddedICE-RT logic in the processor generates a Prefetch Abort exception when a breakpoint is reached, or a Data Abort exception when a watchpoint is hit. These exceptions are handled by the RealMonitor exception handlers that inform the user, by way of the debugger, of the event. This allows user application to continue running without stopping the processor. RealMonitor considers user application to consist of two parts:

- a foreground application running continuously, typically in User, System, or SVC mode
- a background application containing interrupt and exception handlers that are triggered by certain events in user system, including:
  - IRQs or FIQs
  - Data and Prefetch aborts caused by user foreground application. This indicates an error in the application being debugged. In both cases the host is notified and the user application is stopped.
  - Undef exception caused by the undefined instructions in user foreground application. This indicates an error in the application being debugged. RealMonitor stops the user application until a "Go" packet is received from the host.

When one of these exceptions occur that is not handled by user application, the following happens:

- RealMonitor enters a loop, polling the DCC. If the DCC read buffer is full, control is passed to rm_ReceiveData() (RealMonitor internal function). If the DCC write buffer is free, control is passed to rm_TransmitData() (RealMonitor internal function). If there is nothing else to do, the function returns to the caller. The ordering of the above comparisons gives reads from the DCC a higher priority than writes to the communications link.
- RealMonitor stops the foreground application. Both IRQs and FIQs continue to be serviced if they were enabled by the application at the time the foreground application was stopped.

HOW TO ENABLE REALMONITOR

The following steps must be performed to enable RealMonitor. A code example which implements all the steps can be found at the end of this section.

Adding stacks

User must ensure that stacks are set up within application for each of the processor modes used by RealMonitor. For each mode, RealMonitor requires a fixed number of words of stack space. User must therefore allow sufficient stack space for both RealMonitor and application.

RealMonitor has the following stack requirements:

Table 204: RealMonitor stack requirement

Processor Mode	RealMonitor Stack Usage (Bytes)
Undef	48
Prefetch Abort	16
Data Abort	16
IRQ	8

IRQ mode

A stack for this mode is always required. RealMonitor uses two words on entry to its interrupt handler. These are freed before nested interrupts are enabled.

Undef mode

A stack for this mode is always required. RealMonitor uses 12 words while processing an undefined instruction exception.

SVC mode

RealMonitor makes no use of this stack.

Prefetch Abort mode

RealMonitor uses four words on entry to its Prefetch abort interrupt handler.

Data Abort mode

RealMonitor uses four words on entry to its data abort interrupt handler.

User/System mode

RealMonitor makes no use of this stack.

FIQ mode

RealMonitor makes no use of this stack.

Handling exceptions

This section describes the importance of sharing exception handlers between RealMonitor and user application.

RealMonitor exception handling

To function properly, RealMonitor must be able to intercept certain interrupts and exceptions. Figure 61 illustrates how exceptions can be claimed by RealMonitor itself, or shared between RealMonitor and application. If user application requires the exception sharing, they must provide function (such as *app_IRQDispatch* ()). Depending on the nature of the exception, this handler can either:

- pass control to the RealMonitor processing routine, such as rm_irqhandler2()
- claim the exception for the application itself, such as *app_IRQHandler* ().

In a simple case where an application has no exception handlers of its own, the application can install the RealMonitor low-level exception handlers directly into the vector table of the processor. Although the irq handler must get the address of the Vectored Interrupt Controller. The easiest way to do this is to write a branch instruction (<*address* >) into the vector table, where the target of the branch is the start address of the relevant RealMonitor exception handler.

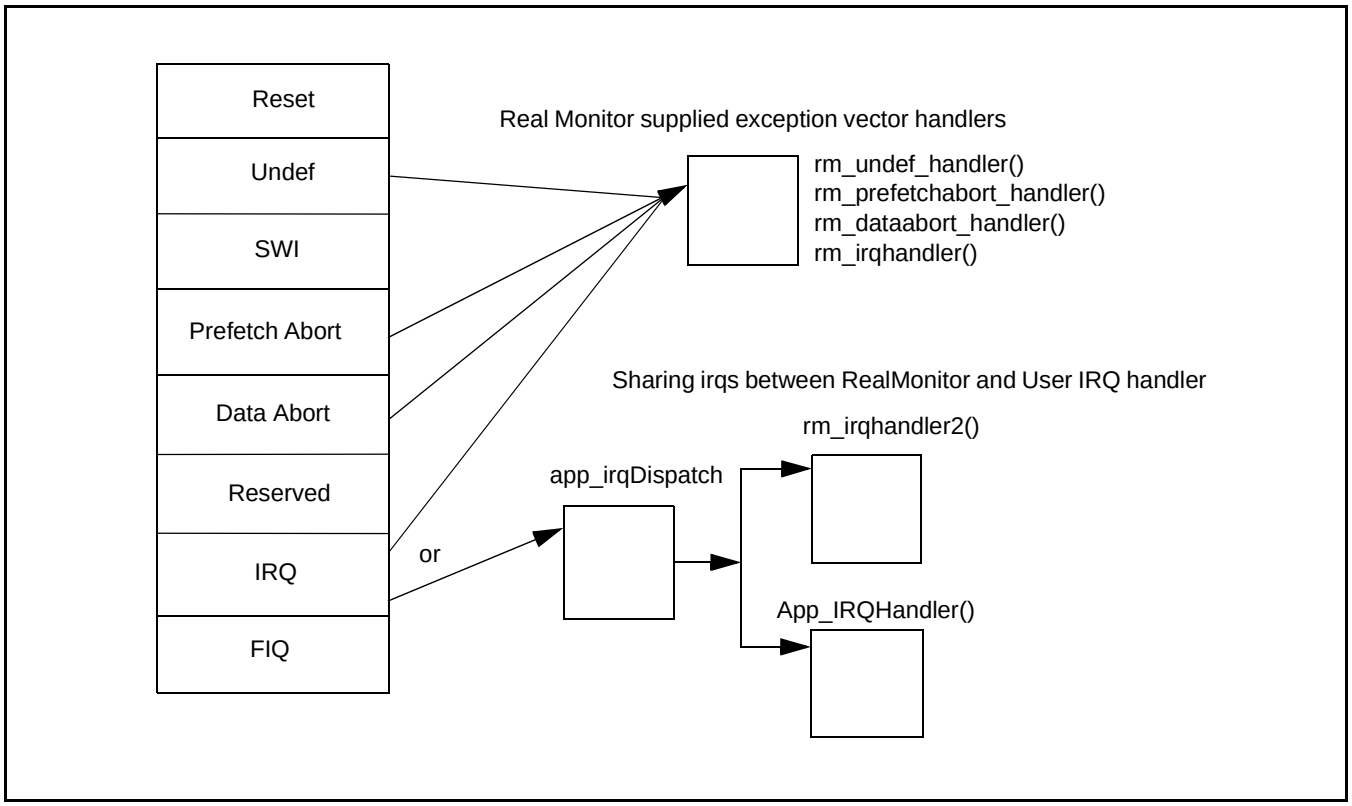


Figure 61: Exception Handlers

RMTarget initialization

While the processor is in a privileged mode, and IRQs are disabled, user must include a line of code within the start-up sequence of application to call rm_init_entry().

Code Example

The following example shows how to setup stack, VIC, initialize RealMonitor and share non vectored interrupts:

```
    IMPORT rm_init_entry
    IMPORT rm_prefetchabort_handler
    IMPORT rm_dataabort_handler
    IMPORT rm_irqhandler2
    IMPORT rm_undef_handler
    IMPORT User_Entry ;Entry point of user application.
    CODE32
    ENTRY
   ;Define exception table. Instruct linker to place code at address 0x0000 0000

    AREA exception_table, CODE

    LDR pc, Reset_Address
    LDR pc, Undefined_Address
    LDR pc, SWI_Address
    LDR pc, Prefetch_Address
    LDR pc, Abort_Address
    NOP ; Insert User code valid signature here.
    LDR pc, [pc, #-0xFF0] ;Load IRQ vector from VIC
    LDR PC, FIQ_Address

Reset_Address      DCD     __init                    ;Reset Entry point
Undefined_Address  DCD     rm_undef_handler          ;Provided by RealMonitor
SWI_Address        DCD     0                         ;User can put address of SWI handler here
Prefetch_Address   DCD     rm_prefetchabort_handler ;Provided by RealMonitor
Abort_Address      DCD     rm_dataabort_handler      ;Provided by RealMonitor
FIQ_Address        DCD     0                         ;User can put address of FIQ handler here

AREA init_code, CODE

ram_end EQU 0x4000xxxx ; Top of on-chip RAM.
__init
; /*********************************************************************
; * Set up the stack pointers for various processor modes. Stack grows
; * downwards.
; *********************************************************************/
    LDR r2, =ram_end    ;Get top of RAM
    MRS r0, CPSR        ;Save current processor mode
    ; Initialize the Undef mode stack for RealMonitor use
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x1b
    MSR     CPSR_c, r1
    ;Keep top 32 bytes for flash programming routines.
    ;Refer to Flash Memory System and Programming chapter
    SUB sp,r2,#0x1F

    ; Initialize the Abort mode stack for RealMonitor
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x17
    MSR     CPSR_c, r1
    ;Keep 64 bytes for Undef mode stack
    SUB sp,r2,#0x5F
```

```
    ; Initialize the IRQ mode stack for RealMonitor and User
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x12
    MSR     CPSR_c, r1
    ;Keep 32 bytes for Abort mode stack
    SUB sp,r2,#0x7F

    ; Return to the original mode.
    MSR     CPSR_c, r0

    ; Initialize the stack for user application
    ; Keep 256 bytes for IRQ mode stack
    SUB sp,r2,#0x17F

; /*********************************************************************
; * Setup Vectored Interrupt controller. DCC Rx and Tx interrupts
; * generate Non Vectored IRQ request. rm_init_entry is aware
; * of the VIC and it enables the DBGCommRX and DBGCommTx interrupts.
; * Default vector address register is programmed with the address of
; * Non vectored app_irqDispatch mentioned in this example. User can setup
; * Vectored IRQs or FIQs here.
; *********************************************************************/

    VICBaseAddr          EQU 0xFFFFF000    ; VIC Base address
    VICDefVectAddrOffset EQU 0x34

    LDR r0, =VICBaseAddr
    LDR r1, =app_irqDispatch
    STR r1, [r0,#VICDefVectAddrOffset]

    BL rm_init_entry ;Initialize RealMonitor
    ;enable FIQ and IRQ in ARM Processor
    MRS     r1, CPSR        ; get the CPSR
    BIC     r1, r1, #0xC0   ; enable IRQs and FIQs
    MSR     CPSR_c, r1      ; update the CPSR
; /*********************************************************************
; * Get the address of the User entry point.
; *********************************************************************/
    LDR lr, =User_Entry
    MOV pc, lr
; /*********************************************************************
; * Non vectored irq handler (app_irqDispatch)
; *********************************************************************/

AREA app_irqDispatch, CODE
VICVectAddrOffset EQU 0x30
app_irqDispatch

    ;enable interrupt nesting
    STMFD sp!, {r12,r14}
    MRS r12, spsr             ;Save SPSR in to r12
    MSR cpsr_c,0x1F           ;Re-enable IRQ, go to system mode

    ;User should insert code here if non vectored Interrupt sharing is
    ;required. Each non vectored shared irq handler must return to
    ;the interrupted instruction by using the following code.
    ;MSR cpsr_c, #0x52              ;Disable irq, move to IRQ mode
```

```
;MSR spsr, r12                   ;Restore SPSR from r12
;STMFD sp!, {r0}
;LDR r0, =VICBaseAddr
;STR r1, [r0,#VICVectAddrOffset] ;Acknowledge Non Vectored irq has finished
;LDMFD sp!, {r12,r14,r0}         ;Restore registers
;SUBS pc, r14, #4                ;Return to the interrupted instruction

;user interrupt did not happen so call rm_irqhandler2. This handler
;is not aware of the VIC interrupt priority hardware so trick
;rm_irqhandler2 to return here

STMFD sp!, {ip,pc}
LDR pc, rm_irqhandler2
;rm_irqhandler2 returns here
MSR cpsr_c, #0x52               ;Disable irq, move to IRQ mode
MSR spsr, r12                   ;Restore SPSR from r12
STMFD sp!, {r0}
LDR r0, =VICBaseAddr
STR r1, [r0,#VICVectAddrOffset] ;Acknowledge Non Vectored irq has finished
LDMFD sp!, {r12,r14,r0}         ;Restore registers
SUBS pc, r14, #4                ;Return to the interrupted instruction

END
```

REALMONITOR BUILD OPTIONS

RealMonitor was built with the following options:

RM_OPT_DATALOGGING=FALSE

This option enables or disables support for any target-to-host packets sent on a non RealMonitor (third-party) channel.

RM_OPT_STOPSTART=TRUE

This option enables or disables support for all stop and start debugging features.

RM_OPT_SOFTBREAKPOINT=TRUE

This option enables or disables support for software breakpoints.

RM_OPT_HARDBREAKPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_HARDWATCHPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_SEMIHOSTING=FALSE

This option enables or disables support for SWI semi-hosting. Semi-hosting provides code running on an ARM target use of facilities on a host computer that is running an ARM debugger. Examples of such facilities include the keyboard input, screen output, and disk I/O.

RM_OPT_SAVE_FIQ_REGISTERS=TRUE

This option determines whether the FIQ-mode registers are saved into the registers block when RealMonitor stops.

RM_OPT_READBYTES=TRUE

RM_OPT_WRITEBYTES=TRUE

RM_OPT_READHALFWORDS=TRUE

RM_OPT_WRITEHALFWORDS=TRUE

RM_OPT_READWORDS=TRUE

RM_OPT_WRITEWORDS=TRUE

Enables/Disables support for 8/16/32 bit read/write.

RM_OPT_EXECUTECODE=FALSE

Enables/Disables support for executing code from "execute code" buffer. The code must be downloaded first.

RM_OPT_GETPC=TRUE

This option enables or disables support for the RealMonitor GetPC packet. Useful in code profiling when real monitor is used in interrupt mode.

RM_EXECUTECODE_SIZE=NA

"execute code" buffer size. Also refer to RM_OPT_EXECUTECODE option.

RM_OPT_GATHER_STATISTICS=FALSE

This option enables or disables the code for gathering statistics about the internal operation of RealMonitor.

RM_DEBUG=FALSE

This option enables or disables additional debugging and error-checking code in RealMonitor.

RM_OPT_BUILDIDENTIFIER=FALSE

This option determines whether a build identifier is built into the capabilities table of RMTarget. Capabilities table is stored in ROM.

RM_OPT_SDM_INFO=FALSE

SDM gives additional information about application board and processor to debug tools.

RM_OPT_MEMORYMAP=FALSE

This option determines whether a memory map of the board is built into the target and made available through the capabilities table

RM_OPT_USE_INTERRUPTS=TRUE

This option specifies whether RMTarget is built for interrupt-driven mode or polled mode.

RM_FIFOSIZE=NA

This option specifies the size, in words, of the data logging FIFO buffer.

CHAIN_VECTORS=FALSE

This option allows RMTarget to support vector chaining through µHAL (ARM HW abstraction API).