

LPC2106/2105/2104 User Manual

Preliminary Release

May 15, 2003

Table of Contents

List of Figures	7
List of Tables	9
Document Revision History	12
Introduction	14
Features	14
Applications	14
Architectural Overview	15
ARM7TDMI-S Processor	15
On-Chip Flash Memory System	15
On-Chip Static RAM	16
Block Diagram	17
LPC2106/2105/2104 Memory Addressing	18
Memory Maps	18
LPC2106/2105/2104 Memory Re-mapping and Boot Block	22
Prefetch Abort and Data Abort Exceptions	25
System Control Block	26
Summary of System Control Block Functions	26
Pin Description	26
Register Description	27
Crystal Oscillator	28
External Interrupt Inputs	28
Memory Mapping Control	31
PLL (Phase Locked Loop)	32
Power Control	39
Reset	41
VPB Divider	43
Wakeup Timer	45
Vectored Interrupt Controller (VIC)	46
Features	46
Description	46
Register Description	47
VIC Registers	49
GPIO	54
Features	54
Applications	54
Pin Description	54
Register Description	54
Pin Connect Block	56
Features	56
Applications	56
Description	56
Register Description	56

UART 0	62
Features	62
Pin Description	62
Register Description	63
Architecture	71
UART1	74
Features	74
Pin Description	74
Register Description	75
Architecture	86
I2C Interface	88
Features	88
Applications	88
Description	88
Pin Description	92
Register Description	93
Architecture	99
SPI Interface	100
Features	100
Description	100
Pin Description	104
Register Description	105
Architecture	109
Timer 0 and Timer 1	110
Features	110
Applications	110
Description	111
Pin Description	111
Register Description	112
Example Timer Operation	117
Architecture	118
Pulse Width Modulator (PWM)	120
Features	120
Description	120
Pin Description	125
Register Description	126
Real Time Clock	134
Features	134
Description	134
Architecture	135
Register Description	135
RTC Interrupts	137
Miscellaneous Register Group	138
Consolidated Time Registers	141
Time Counter Group	143
Alarm Register Group	144
Reference Clock Divider (Prescaler)	145

Watchdog	148
Features	148
Applications	148
Description	148
Register Description	149
Block Diagram	152
EmbeddedICE Logic	154
Features	154
Applications	154
Description	154
Pin Description	155
Register Description	156
Block Diagram	157
Embedded Trace Macrocell	158
Features	158
Applications	158
Description	158
Pin Description	159
Register Description	160
Block Diagram	161
RealMonitor	162
Features	162
Applications	162
Description	162
How to Enable RealMonitor	166
RealMonitor build options	172

List of Figures

Figure 1:	LPC2106/2105/2104 Block Diagram	17
Figure 2:	System Memory Map	18
Figure 3:	Peripheral Memory Map	19
Figure 4:	AHB Peripheral Map	20
Figure 5:	VPB Peripheral Map	21
Figure 6:	Map of lower memory are showing re-mapped and re-mappable areas.	24
Figure 7:	External Interrupt Logic	30
Figure 8:	PLL Block Diagram	33
Figure 9:	Reset Block Diagram including Wakeup Timer	42
Figure 10:	VPB Divider Connections	44
Figure 11:	Block Diagram of the Vectored Interrupt Controller	53
Figure 12:	UART Block Diagram	72
Figure 13:	UART Block Diagram	87
Figure 14:	I2C Bus Configuration	89
Figure 15:	Slave Mode Configuration	89
Figure 16:	Format in the master transmitter mode	90
Figure 17:	Format of master receiver mode	90
Figure 18:	A master receiver switch to master transmitter after sending repeated START	91
Figure 19:	Slave Mode Configuration	91
Figure 20:	Format of slave receiver mode	92
Figure 21:	Format of slave transmitter mode	92
Figure 22:	I2C Architecture	99
Figure 23:	SPI Data Transfer Format (CPHA = 0 and CPHA = 1)	101
Figure 24:	SPI Block Diagram	109
Figure 25:	A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled	117
Figure 26:	A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled	117
Figure 27:	Timer block diagram	118
Figure 28:	PWM block diagram	122
Figure 29:	Sample PWM waveforms	123
Figure 30:	RTC block diagram	135
Figure 31:	RTC Prescaler block diagram	146
Figure 32:	Watchdog Block Diagram	152
Figure 33:	EmbeddedICE Debug Environment Block Diagram	157
Figure 34:	ETM Debug Environment Block Diagram	161
Figure 35:	RealMonitor components	163
Figure 36:	RealMonitor as a state machine	164
Figure 37:	Exception Handlers	167

List of Tables

Table 1:	ARM Exception Vector Locations	22
Table 2:	LPC2106/2105/2104 Memory Mapping Modes	22
Table 3:	Pin summary	26
Table 4:	Summary of System Control Registers	27
Table 5:	External Interrupt Registers	28
Table 6:	External Interrupt Flag Register (EXTINT - 0xE01FC140)	29
Table 7:	External Interrupt Wakeup Register (EXTWAKE - 0xE01FC144)	29
Table 8:	MEMMAP Register	31
Table 9:	Memory Mapping Control Register (MEMMAP - 0xE01FC040)	31
Table 10:	PLL Registers	32
Table 11:	PLL Control Register (PLLCON - 0xE01FC080)	34
Table 12:	PLL Configuration Register (PLLCFG - 0xE01FC084)	34
Table 13:	PLL Status Register (PLLSTAT - 0xE01FC088)	35
Table 14:	PLL Control Bit Combinations	35
Table 15:	PLL Feed Register (PLLFEED - 0xE01FC08C)	36
Table 16:	PLL Divider Values	38
Table 17:	PLL Multiplier Values	38
Table 18:	Power Control Registers	39
Table 19:	Power Control Register (PCON - 0xE01FC0C0)	39
Table 20:	Power Control for Peripherals Register (PCONP - 0xE01FC0C4)	40
Table 21:	VPBDIV Register Map	43
Table 22:	VPB Divider Register (VPBDIV - 0xE01FC100)	43
Table 23:	VIC Register Map	47
Table 24:	Software Interrupt Register (VICSoftInt - 0xFFFFF018, Read/Write)	49
Table 25:	Software Interrupt Clear Register (VICSoftIntClear - 0xFFFFF01C, Write Only)	49
Table 26:	Raw Interrupt Status Register (VICRawIntr - 0xFFFFF008, Read-Only)	49
Table 27:	Interrupt Enable Register (VICIntEnable - 0xFFFFF010, Read/Write)	50
Table 28:	Software Interrupt Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)	50
Table 29:	Interrupt Select Register (VICIntSelect - 0xFFFFF00C, Read/Write)	50
Table 30:	IRQ Status Register (VICIRQStatus - 0xFFFFF000, Read-Only)	50
Table 31:	IRQ Status Register (VICFIQStatus - 0xFFFFF004, Read-Only)	51
Table 32:	Vector Control Registers (VICVectCntl0-15 - 0xFFFFF200-23C, Read/Write)	51
Table 33:	Vector Address Registers (VICVectAddr0-15 - 0xFFFFF100-13C, Read/Write)	51
Table 34:	Default Vector Address Register (VICDefVectAddr - 0xFFFFF034, Read/Write)	51
Table 35:	Vector Address Register (VICVectAddr - 0xFFFFF030, Read/Write)	52
Table 36:	Protection Enable Register (VICProtection - 0xFFFFF020, Read/Write)	52
Table 37:	GPIO Pin Description	54
Table 38:	GPIO Register Map	54
Table 39:	GPIO Pin Value Register (IOPIN - 0xE0028000)	55
Table 40:	GPIO Output Set Register (IOSET - 0xE0028004)	55
Table 41:	GPIO Output Clear Register (IOCLR - 0xE002800C)	55
Table 42:	GPIO Direction Register (IODIR - 0xE0028008)	55
Table 43:	Pin Connect Block Register Map	56
Table 44:	Pin Function Select Register 0 (PINSEL0 - 0xE002C000)	57
Table 45:	Pin Function Select Register 1 (PINSEL1 - 0xE002C004)	59
Table 46:	Pin Function Select Register Bits	61
Table 47:	UART 0 Pin Description	62
Table 48:	UART 0 Register Map	63
Table 49:	Receiver Buffer Register (RBR - 0xE000C000 when DLAB = 0, Read Only)	64
Table 50:	Transmit Holding Register (THR - 0xE000C000 when DLAB = 0, Write Only)	64
Table 51:	Divisor Latch LSB Register (DLL - 0xE000C000 when DLAB = 1)	64
Table 52:	Divisor Latch MSB Register (DLM - 0xE000C004 when DLAB = 1)	64
Table 53:	Interrupt Enable Register Bit Descriptions (IER - 0xE000C004 when DLAB = 0)	65
Table 54:	Interrupt Identification Register Bit Descriptions (IIR - 0xE000C008, Read Only)	65
Table 55:	Interrupt Handling	66

ARM-based Microcontroller

LPC2106/2105/2104

Table 56: FIFO Control Register Bit Descriptions (FCR - 0xE000C008)	67
Table 57: Line Control Register Bit Descriptions (LCR - 0xE000C00C)	68
Table 58: Line Status Register Bit Descriptions (LSR - 0xE000C014, Read Only)	69
Table 59: Scratchpad Register (SCR - 0xE000C01C)	70
Table 60: UART1 Pin Description	74
Table 61: UART 1 Register Map	75
Table 62: Receiver Buffer Register (RBR - 0xE0010000 when DLAB = 0, Read Only)	76
Table 63: Transmit Holding Register (THR - 0xE0010000 when DLAB = 0, Write Only)	76
Table 64: Divisor Latch LSB Register (DLL - 0xE0010000 when DLAB = 1)	76
Table 65: Divisor Latch MSB Register (DLM - 0xE0010004 when DLAB = 1)	76
Table 66: IER Bit Descriptions (IER - 0xE0010004 when DLAB = 0)	77
Table 67: IIR Bit Descriptions (IIR - 0xE0010008, Read Only)	78
Table 68: Interrupt Handling	79
Table 69: FCR Bit Descriptions (FCR - 0xE0010008)	80
Table 70: LCR Bit Descriptions (LCR - 0xE001000C)	81
Table 71: MCR Bit Descriptions (MCR - 0xE0010010)	82
Table 72: LSR Bit Descriptions (LSR - 0xE0010014, Read Only)	83
Table 73: MSR Bit Descriptions (MSR - 0x0xE0010018)	84
Table 74: Scratchpad Register (SCR - 0xE001001C)	85
Table 75: I2C Pin Description	92
Table 76: I2C Register Map	93
Table 77: I2C Control Set Register (I2CONSET - 0xE001C000)	95
Table 78: I2C Control Clear Register (I2CONCLR - 0xE001C018)	95
Table 79: I2C Status Register (I2STAT - 0xE001C004)	96
Table 80: I2C Data Register (I2DAT - 0xE001C008)	96
Table 81: I2C Slave Address Register (I2ADR - 0xE001C00C)	96
Table 82: I2C SCL High Duty Cycle Register (I2SCLH - 0xE001C010)	97
Table 83: I2C SCL Low Duty Cycle Register (I2SCLL - 0xE001C014)	97
Table 84: I2C Clock Rate Selections for VPB Clock Divider = 1	97
Table 85: I2C Clock Rate Selections for VPB Clock Divider = 2	98
Table 86: I2C Clock Rate Selections for VPB Clock Divider = 4	98
Table 87: SPI Data To Clock Phase Relationship	101
Table 88: SPI Pin Description	104
Table 89: SPI Register Map	105
Table 90: SPI Control Register (SPCR - 0xE0020000)	105
Table 91: SPI Status Register (SPSR - 0xE0020004)	106
Table 92: SPI Data Register (SPDR - 0xE0020008)	106
Table 93: SPI Clock Counter Register (SPCCR - 0xE002000C)	106
Table 94: SPI Interrupt Register (SPINT - 0xE002001C)	107
Table 95: SPI Test Control Register (SPTCR - 0xE0020010)	107
Table 96: SPI Test Status Register (SPTSR - 0xE0020014)	107
Table 97: SPI Test Observe Register (SPTOR - 0xE0020018)	108
Table 98: Pin summary	111
Table 99: Timer Register Map	112
Table 100: Interrupt Register (IR - Timer 0: 0xE0004000; Timer 1: 0xE0008000)	113
Table 101: Timer Control Register (TCR - Timer 0: 0xE0004004; Timer 1: 0xE0008004)	113
Table 102: Match Control Register (MCR - Timer 0: 0xE0004014; Timer 1: 0xE0008014)	114
Table 103: Capture Control Register (CCR - Timer 0: 0xE0004028; Timer 1: 0xE0008028)	115
Table 104: External Match Register (EMR - Timer 0: 0xE000403C; Timer 1: 0xE000803C)	116
Table 105: External Match Control	116
Table 106: Set and Reset inputs for PWM Flip-Flops	123
Table 107: Pin summary	125
Table 108: Pulse Width Modulator Register Map	126
Table 109: Interrupt Register (IR - 0xE0014000)	128
Table 110: Timer Control Register (TCR - 0xE0014004)	129
Table 111: Match Control Register (MCR - 0xE0014014)	130
Table 112: PWM Control Register (PCR - 0xE001404C)	131

ARM-based Microcontroller

LPC2106/2105/2104

Table 113: Latch Enable Register (LER - 0xE0014050)	132
Table 114: Real Time Clock Register Map	136
Table 115: Miscellaneous Registers	138
Table 116: Interrupt Location Register Bits (ILR - 0xE0024000).	138
Table 117: Clock Tick Counter Bits (CTC - 0xE0024004).	138
Table 118: Clock Control Register Bits (CCR - 0xE0024008).	139
Table 119: Counter Increment Interrupt Register Bits (CIIR - 0xE002400C)	139
Table 120: Alarm Mask Register Bits (AMR - 0xE0024010)	140
Table 121: Consolidated Time Register 0 Bits (CTIME0 - 0xE0024014)	141
Table 122: Consolidated Time Register 1 Bits (CTIME1 - 0xE0024018)	141
Table 123: Consolidated Time Register 2 Bits (CTIME2 - 0xE002401C)	142
Table 124: Time Counter Relationships and Values	143
Table 125: Time Counter registers	143
Table 126: Alarm Registers	144
Table 127: Reference Clock Divider registers	145
Table 128: Prescaler Integer Register (PREINT - 0xE0024080).	145
Table 129: Prescaler Fraction Register (PREFRAC - 0xE0024084).	145
Table 130: Watchdog Register Map	149
Table 131: Watchdog Mode Register (WDMOD - 0xE0000000).	150
Table 132: Watchdog Feed Register (WDFEED - 0xE0000008)	151
Table 133: Watchdog Timer Value Register (WDTV - 0xE000000C)	151
Table 134: EmbeddedICE Pin Description	155
Table 135: EmbeddedICE Logic Registers	156
Table 136: ETM Configuration	158
Table 137: ETM Pin Description	159
Table 138: ETM Registers	160
Table 139: RealMonitor stack requirement	166

DOCUMENT REVISION HISTORY

May, 2003:

- Prototype LPC2106/2105/2104 User Manual created from the design specification.

1. INTRODUCTION

FEATURES

- ARM7TDMI-S processor.
- 128 kilobyte on-chip Flash Program Memory with In-System Programming (ISP) and In-Application Programming (IAP) capability. Flash programming time is 1 ms for up to a 512 byte line. Sector erase or chip erase is done in 400 ms.
- 64 kilobyte Static RAM.
- Vectored Interrupt Controller.
- Emulation Trace Module supports real-time trace.
- RealMonitor module enables real time debugging.
- Standard ARM Test/Debug interface for compatibility with existing tools.
- Very small package TQFP48 (7x7mm²).
- Two UARTs, one with full modem interface.
- I²C serial interface.
- SPI serial interface.
- Two timers, each with 4 capture/compare channels.
- PWM unit with up to 6 PWM outputs.
- Real Time Clock.
- Watchdog Timer.
- General purpose I/O pins.
- CPU operating range up to 60 MHz.
- Dual power supply.
 - CPU operating voltage range of 1.65V to 1.95V (1.8V +/- 8.3%).
 - I/O power supply range of 3.0V to 3.6V (3.3V +/- 10%).
- Two low power modes, Idle and Power Down.
- Processor wakeup from Power Down mode via external interrupt.
- Individual enable/disable of peripheral functions for power optimization.
- On-chip crystal oscillator with an operating range of 10 MHz to 25 MHz.
- On-chip PLL allows CPU operation up to the maximum CPU rate. May be used over the entire crystal operating range.

APPLICATIONS

- Internet gateway.
- Serial communications protocol converter.
- Access control.
- Industrial Control.
- Medical equipment.

ARCHITECTURAL OVERVIEW

The LPC2106/2105/2104 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 LPC2106/2105/2104 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. LPC2106/05/04 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 Connection 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 set.
- A 16-bit THUMB 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 LPC2106/2105/2104 incorporates a 128K byte Flash memory system. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways: 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.

ON-CHIP STATIC RAM

The LPC2106, LPC2105 and LPC 2104 provide a 64K byte, 32K byte and 16K byte static RAM memory respectively that may be used for code and/or data storage. The SRAM supports 8-bit, 16-bit, and 32-bit accesses.

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. If a chip reset occurs, actual SRAM contents will not reflect the most recent write request. Any software that checks SRAM contents after reset must take this into account. A dummy write to an unused location may be appended to any operation in order to guarantee that all data has really been written into the SRAM.

BLOCK DIAGRAM

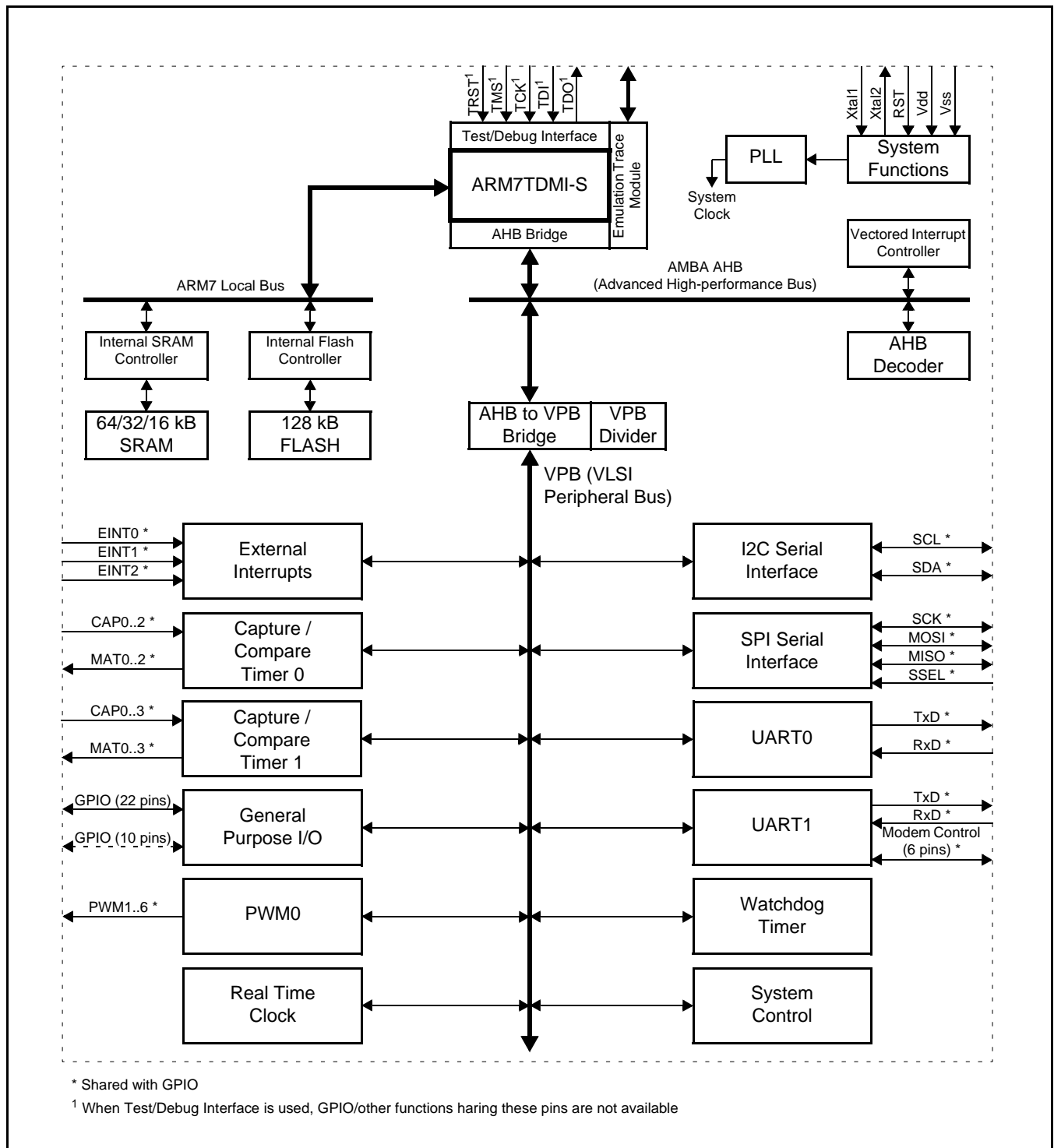


Figure 1: LPC2106/2105/2104 Block Diagram

2. LPC2106/2105/2104 MEMORY ADDRESSING

MEMORY MAPS

The LPC2106/2105/2104 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.

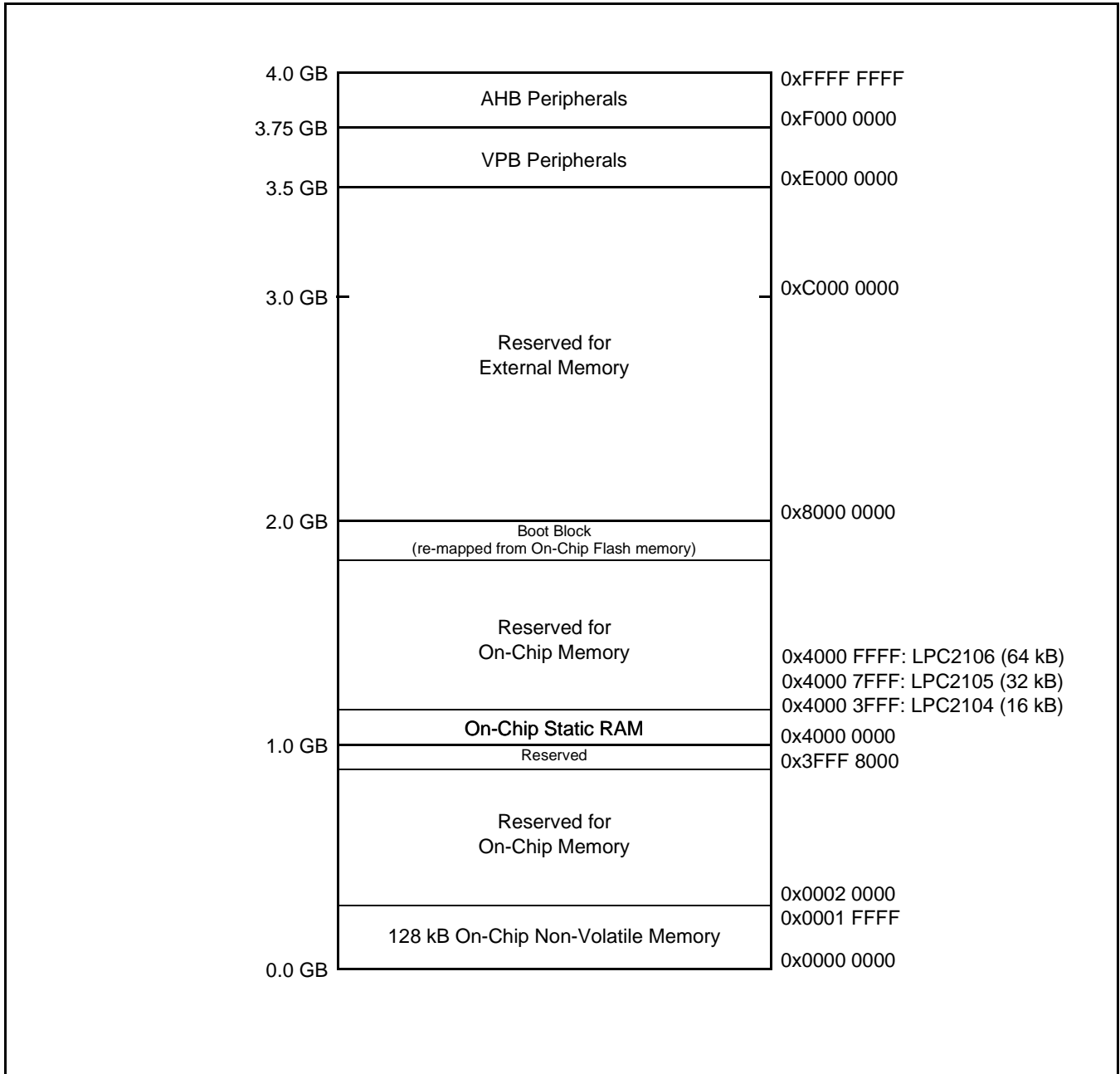


Figure 2: System Memory Map

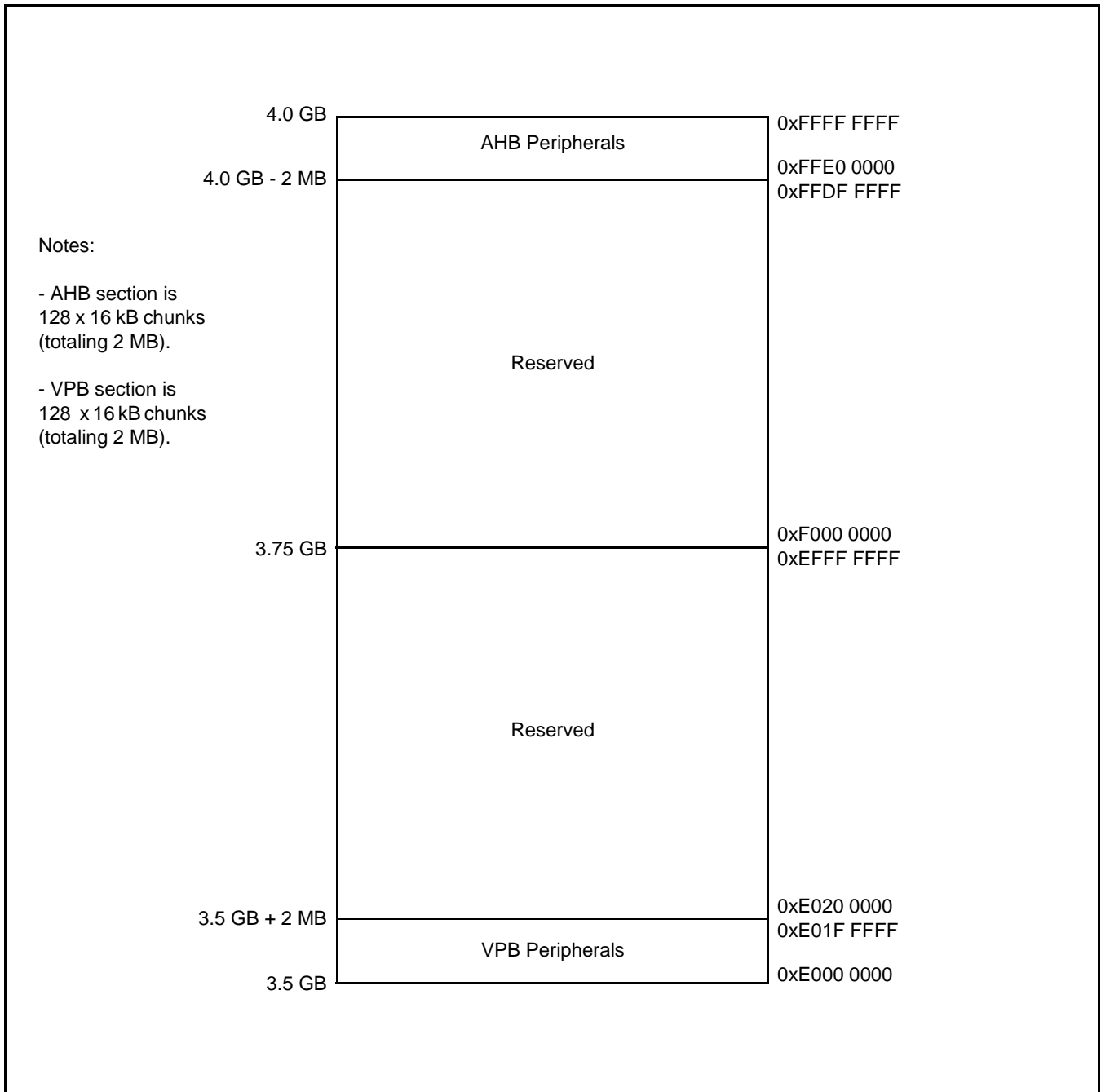


Figure 3: Peripheral Memory Map

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. A 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.

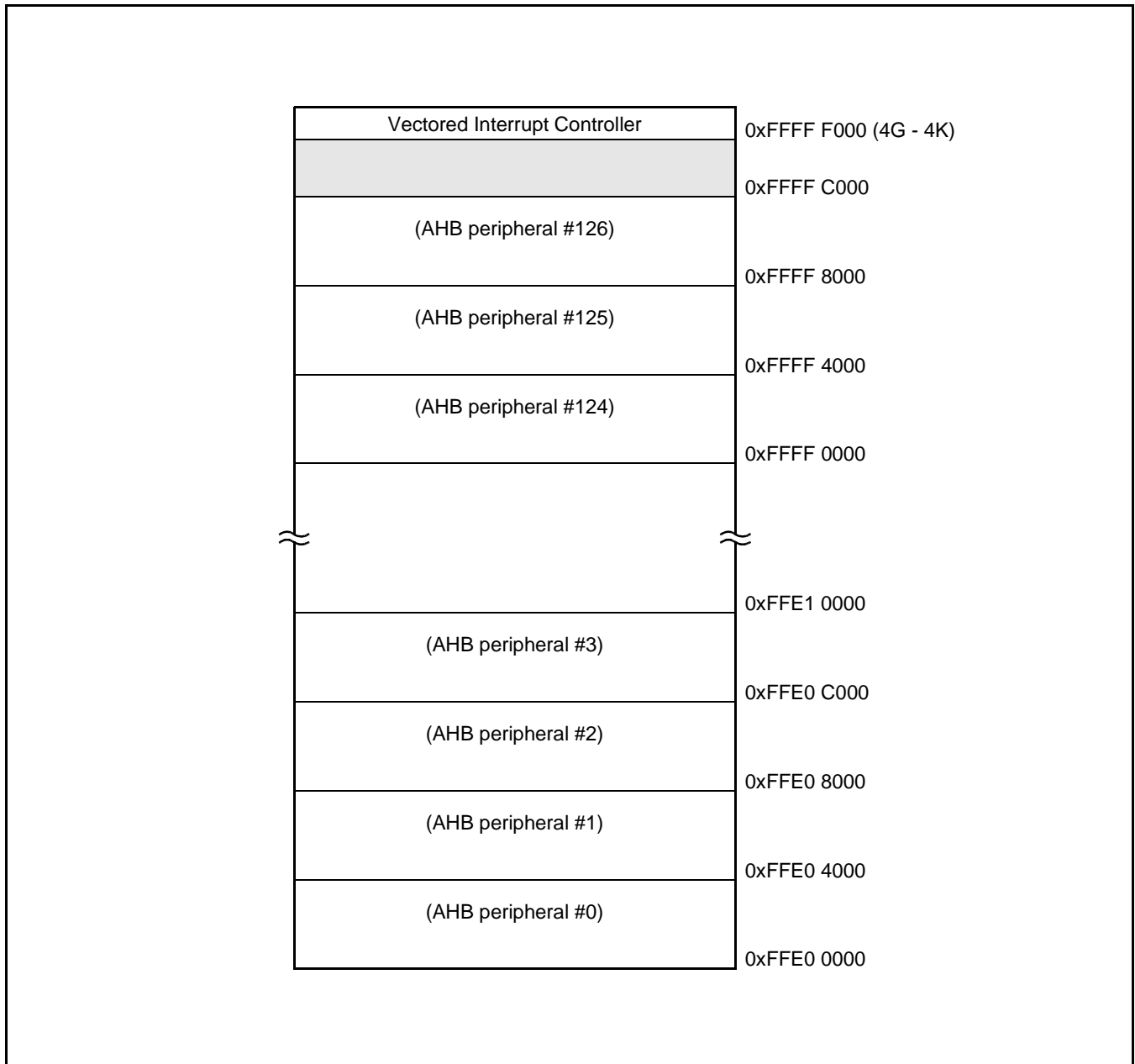


Figure 4: AHB Peripheral Map

System Control Block (VPB peripheral #127)	0xE01F FFFF
(VPB peripheral #126)	0xE01F C000
	0xE01F 8000
≈	≈
(VPB peripheral #16)	0xE004 4000
(VPB peripheral #15)	0xE004 0000
(VPB peripheral #14)	0xE003 C000
(VPB peripheral #13)	0xE003 8000
(VPB peripheral #12)	0xE003 4000
	0xE003 0000
Pin Connect Block (VPB peripheral #11)	0xE002 C000
GPIO (VPB peripheral #10)	0xE002 8000
RTC (VPB peripheral #9)	0xE002 4000
SPI (VPB peripheral #8)	0xE002 0000
I ² C (VPB peripheral #7)	0xE001 C000
(VPB peripheral #6)	0xE001 8000
PWM0 (VPB peripheral #5)	0xE001 4000
UART1 (VPB peripheral #4)	0xE001 0000
UART0 (VPB peripheral #3)	0xE000 C000
Timer1 (VPB peripheral #2)	0xE000 8000
Timer0 (VPB peripheral #1)	0xE000 4000
Watchdog Timer (VPB peripheral #0)	0xE000 0000

Figure 5: VPB Peripheral Map

LPC2106/2105/2104 MEMORY RE-MAPPING AND BOOT BLOCK

Memory Map Concepts and Operating Modes

The basic concept on the LPC2106/2105/2104 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 1 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 2. Re-mapping of the interrupts is accomplished via the Memory Mapping Control feature described in the System Control Block section.

Table 1: 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.

Table 2: LPC2106/2105/2104 Memory Mapping Modes

Mode	Activation	Usage
Boot Loader mode	Hardware activation by any Reset	The Boot Loader <u>always</u> 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 from the bottom of the Static RAM.
User External mode	Software activation by User program	Activated by a User Program as desired. Interrupt vectors are re-mapped from the bottom of the external memory map.

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.

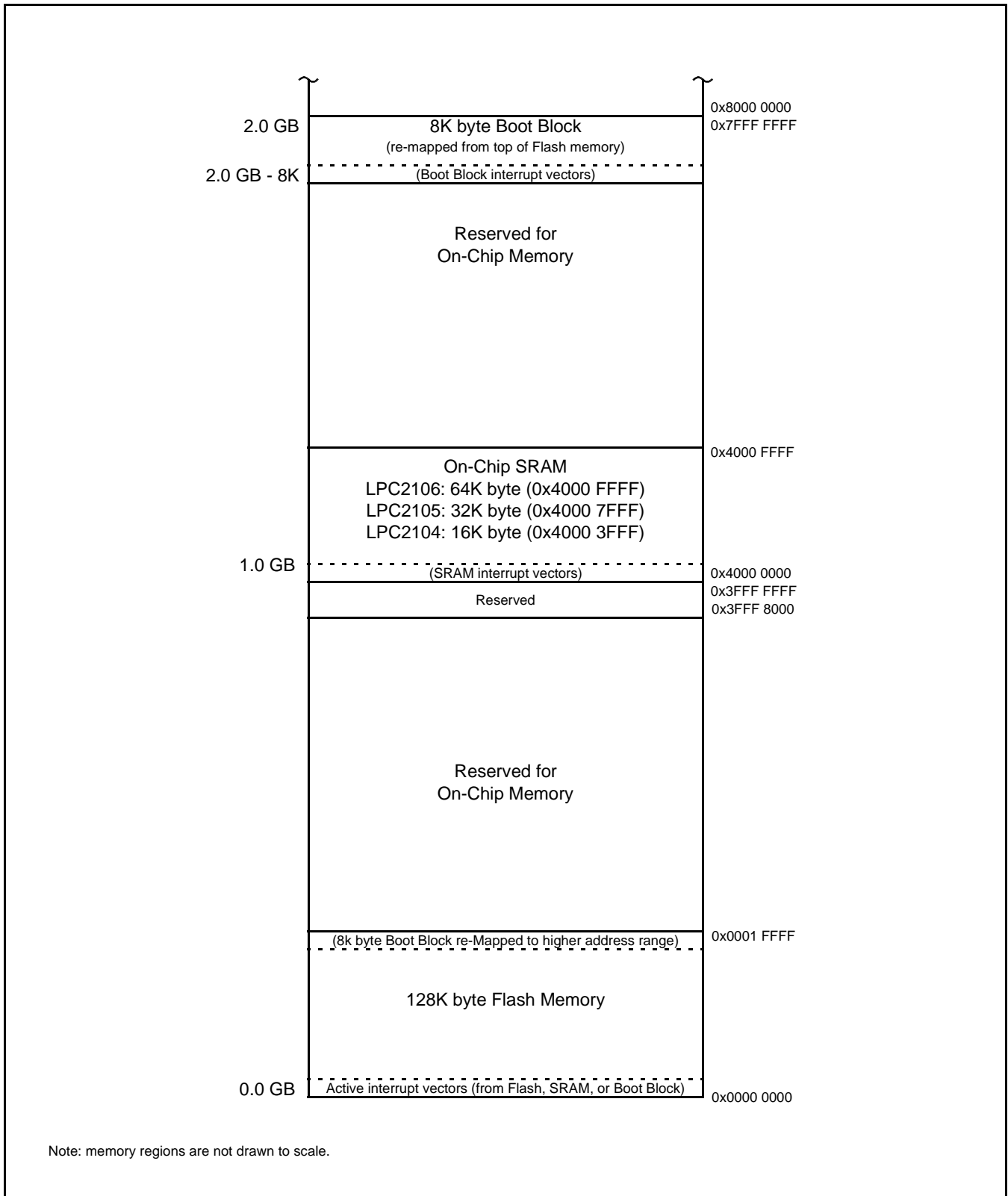


Figure 6: Map of lower memory are showing re-mapped and re-mappable areas.

PREFETCH ABORT AND DATA ABORT EXCEPTIONS

The LPC2106/2105/2104 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 LPC2106/2105/2104, this is:
 - Address space between On-Chip Non-Volatile Memory and the Special registers. Labelled "Reserved for On-Chip Memory" in Figure 2 and Figure 6.
 - Address space between On-Chip Static RAM and External Memory. Labelled "Reserved for On-Chip Memory" in Figure 2.
 - External Memory (since it is not implemented for the LPC2106/2105/2104).
 - 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, or to the Special Register space located just below the SRAM at addresses 0x3FFF8000 through 0x3FFFFFFF.

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 LPC2106/2105/2104 documentation and are not a supported feature.

Note that the ARM 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 3 shows pins that are associated with System Control block functions.

Table 3: 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.
EINT1	Input	External Interrupt Input 1- See the EINT0 description above.
EINT2	Input	External Interrupt Input 2- See the EINT0 description above.
RST	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 4: Summary of System Control Registers

Address	Name	Description	Access	Reset Value
External Interrupts				
0xE01FC140	EXTINT	External interrupt flag register.	R/W	0
0xE01FC144	EXTWAKE	External interrupt wakeup register.	R/W	0
Memory Mapping Control				
0xE01FC040	MEMMAP	Memory mapping control.	R/W	0
Phase Locked Loop				
0xE01FC080	PLLCON	PLL control register.	R/W	0
0xE01FC084	PLLCFG	PLL configuration register.	R/W	0
0xE01FC088	PLLSTAT	PLL status register.	RO	0
0xE01FC08C	PLLFEED	PLL feed register.	WO	NA
Power Control				
0xE01FC0C0	PCON	Power control register.	R/W	0
0xE01FC0C4	PCONP	Power control for peripherals.	R/W	0x3BE
VPB Divider				
0xE01FC100	VPBDIV	VPB divider control.	R/W	0

CRYSTAL OSCILLATOR

The oscillator supports crystals in the 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.

EXTERNAL INTERRUPT INPUTS

The LPC2106/2105/2104 includes three External Interrupt Inputs as selectable pin functions. The External Interrupt Inputs can optionally be used to wake up the processor from Power Down mode.

Register Description

The external interrupt function has two 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 LPC2106/2105/2104 from Power Down mode.

Table 5: External Interrupt Registers

Address	Name	Description	Access
0xE01FC140	EXTINT	The External Interrupt Flag register contains interrupt flags for EINT0, EINT1, and EINT2. See Table 6.	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 7.	R/W

EXTINT Register (EXTINT - 0xE01FC140)

When an external interrupt is mapped to its related pin, the presence of a logic zero on that pin will set the corresponding interrupt flag in the EXTINT register. This will cause the VIC to respond appropriately if that interrupt is enabled. Software may clear the flags by writing a 1 to the corresponding bit in EXTINT.

Table 6: External Interrupt Flag Register (EXTINT - 0xE01FC140)

EXTINT	Function	Description	Reset Value
0	EINT0	Set when external the EINT0 pin goes low and EINT0 is mapped to its related pin. Cleared by writing a one to this bit.	0
1	EINT1	Set when external the EINT1 pin goes low and EINT1 is mapped to its related pin. Cleared by writing a one to this bit.	0
2	EINT2	Set when external the EINT2 pin goes low and EINT2 is mapped to its related pin. Cleared by writing a one to this bit.	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

EXTWAKE Register (EXTWAKE - 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 7: External Interrupt Wakeup Register (EXTWAKE - 0xE01FC144)

EXTWAKE	Function	Description	Reset Value
0	EXTWAKE0	When one, assertion of EINT0 will wake up the processor from Power Down mode.	0
1	EXTWAKE1	When one, assertion of EINT1 will wake up the processor from Power Down mode.	0
2	EXTWAKE2	When one, assertion of EINT2 will wake up the processor from Power Down mode.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

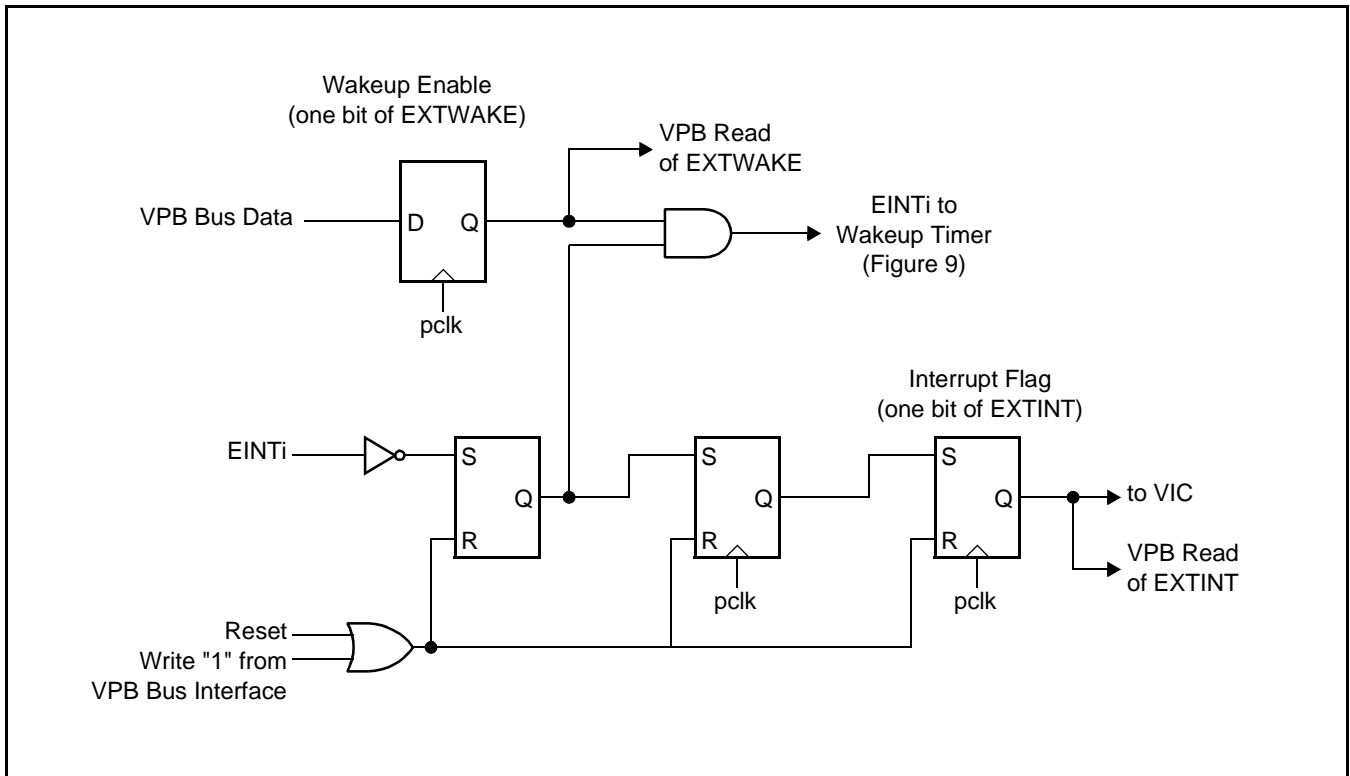


Figure 7: External Interrupt Logic

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 8: 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, RAM, or (for devices with an external bus) external memory.	R/W

Table 9: Memory Mapping Control Register (MEMMAP - 0xE01FC040)

MEMMAP	Function	Description	Reset Value*
1:0	MAP1:0	00: Boot Loader Mode. Interrupt vectors are re-mapped from 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 from Static RAM. 11: User External Memory Mode. Interrupt vectors are re-mapped from external memory. (This mode is reserved for future use on devices with an external bus interface.) 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 LPC2106/2105/2104 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.

PLL (PHASE LOCKED LOOP)

The PLL accepts an input clock frequency in the range of 10 MHz to 25 MHz. The input frequency is multiplied up into the range of 10 MHz to 60 MHz with 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 LPC2106/2105/2104 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 8.

PLL activation is controlled via the PLLCON register. The PLL multiplier and divider values are controlled by the PLLCFG register. These two registers are protected in order to prevent accidental alteration of PLL parameters or deactivation of the PLL. Since all chip operations, including the Watchdog Timer, are dependent on the PLL when it is providing the chip clock, accidental changes to the PLL setup could be fatal. 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 may be enabled by software. 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 10. More detailed descriptions follow.

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

Table 10: PLL Registers

Address	Name	Description	Access
0xE01FC080	PLLCON	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	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	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	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

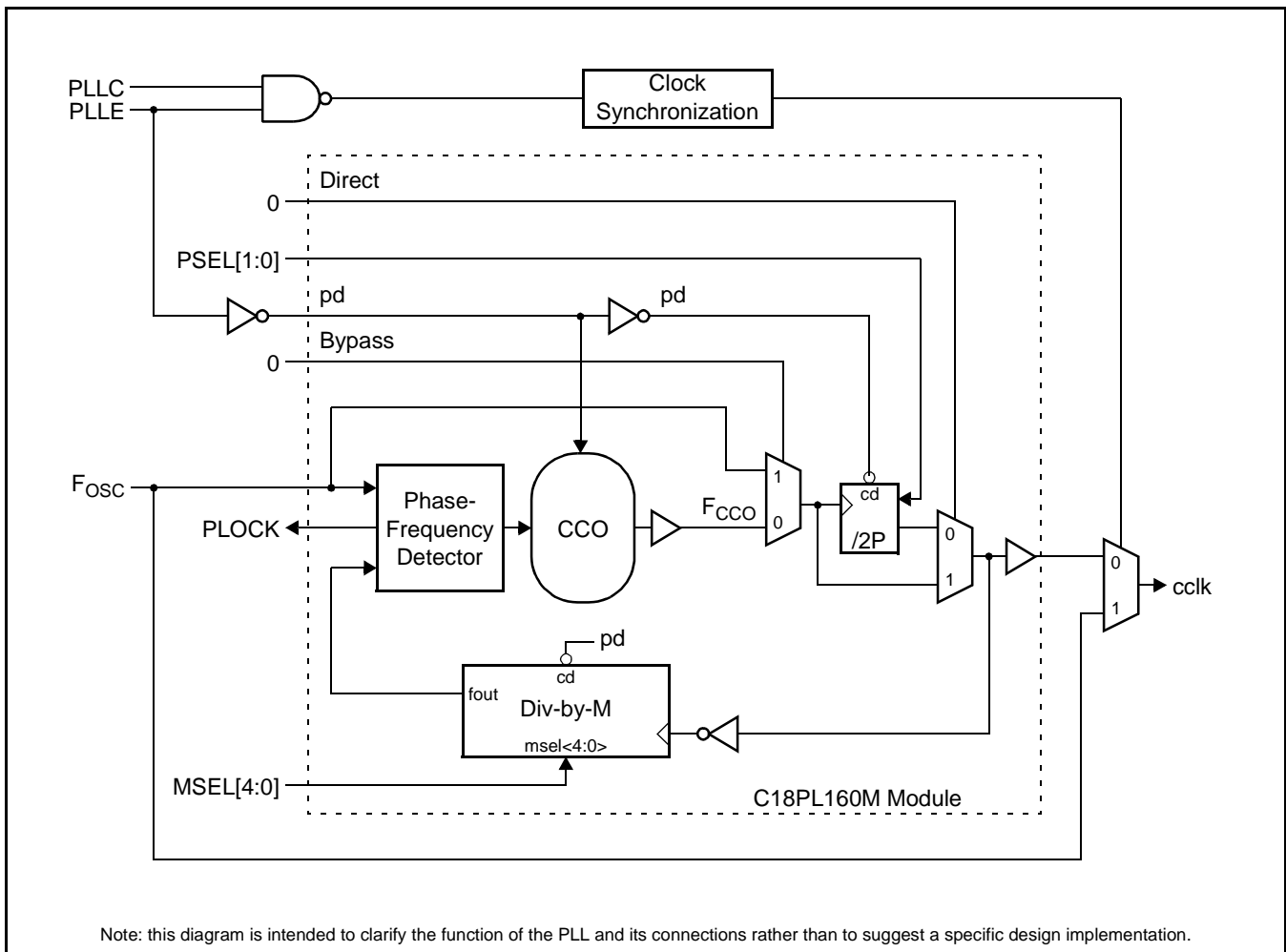


Figure 8: PLL Block Diagram

PLLCON 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 PLLFEED Register (PLLFEED - 0xE01FC08C) description).

Table 11: 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 13.	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 LPC2106/2105/2104. Otherwise, the oscillator clock is used directly by the LPC2106/2105/2104. See PLLSTAT register, Table 13.	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 insure 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.

PLLCFG 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 PLLFEED Register (PLLFEED - 0xE01FC08C) description). Calculations for the PLL frequency, and multiplier and divider values are found in the PLL Frequency Calculation section.

Table 12: 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.	0
6:5	PSEL1:0	PLL Divider value. Supplies the value "P" in the PLL frequency calculations.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLLSTAT 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 PLLFEED Register (PLLFEED - 0xE01FC08C) description).

Table 13: 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 LPC2106/2105/2104. When zero, the PLL is bypassed and the oscillator clock is used directly by the LPC2106/2105/2104. 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, the PLL may be connected, and the interrupt disabled.

PLL Modes

The combinations of PLLE and PLLC are shown in Table 14.

Table 14: 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.

PLLFEED 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 15: 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 LPC2106/2105/2104).
- 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 an even 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 17).
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 16).

Table 16: PLL Divider Values

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

Table 17: PLL Multiplier Values

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

POWER CONTROL

The LPC2106/2105/2104 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.

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 18. More detailed descriptions follow.

Table 18: 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 LPC2106/2105/2104. See Table 19.	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. See Table 20.	R/W

PCON 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 19: Power Control Register (PCON - 0xE01FC0C0)

PCON	Function	Description	Reset Value
0	IDL	Idle mode - when set, 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 set, this bit causes the oscillator and all on-chip clocks to be stopped. A wakeup condition from an external interrupt can cause the oscillator to restart, the PD bit to be cleared, and the processor to resume execution.	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

Power Control for Peripherals Register (PCONP - 0xE01FC0C4)

The PCONP register allows turning off selected peripheral functions for the purpose of saving power. A few peripheral functions cannot be turned off (i.e. the Watchdog timer, GPIO, the Pin Connect block, and the System Control block). Each bit in PCONP controls one peripheral as shown in Table 20. The bit numbers correspond to the related peripheral number as shown in the VPB peripheral map in the LPC2106/2105/2104 Memory Addressing section.

Table 20: Power Control for Peripherals Register (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.	NA
1	PCTIM0	When 1, Timer 0 is enabled. When 0, Timer 0 is disabled to conserve power.	1
2	PCTIM1	When 1, Timer 1 is enabled. When 0, Timer 1 is disabled to conserve power.	1
3	PCURT0	When 1, UART 0 is enabled. When 0, UART 0 is disabled to conserve power.	1
4	PCURT1	When 1, UART 1 is enabled. When 0, UART 1 is disabled to conserve power.	1
5	PCPWM0	When 1, PWM 0 is enabled. When 0, PWM 0 is disabled to conserve power.	1
6	Reserved	Reserved for PWM1. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	PCI2C	When 1, the I ² C interface is enabled. When 0, the I ² C interface is disabled to conserve power.	1
8	PCSPI	When 1, the SPI interface is enabled. When 0, the SPI interface is disabled to conserve power.	1
9	PCRTC	When 1, the RTC is enabled. When 0, the RTC is disabled to conserve power.	1
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

RESET

Reset has two sources on the LPC2106/2105/2104: the \overline{RST} pin and Watchdog Reset. The \overline{RST} 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 Flash controller has completed its initialization. The relationship between Reset, the oscillator, and the Wakeup Timer are shown in Figure 9.

The Reset glitch filter allows the processor to ignore external reset pulses that are very short, and also determines the minimum duration of \overline{RST} that must be asserted in order to guarantee a chip reset. Details of the reset timing requirements will be found in the LPC2106/2105/2104 data sheet DC Specifications.

When the internal Reset is removed, the processor begins executing at address 0, which is the Reset vector. 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: DBGSEL, RTCK, and EINT1.

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.

In devices that may have additional on-chip reset sources, a reset source identification register should be implemented to aid the user application in determining the cause of a reset without having to check each possible flag in a different location. Future devices in the LPC2106/2105/2104 family may implement additional sources of chip reset such as Software Reset, Brownout Detect (or other power monitoring function), etc.

This requires first that each additional reset source implements a flag that persists through the particular hardware reset that it is associated with. However, each flag must be cleared by all of the other sources of reset. The watchdog timer already contains such a flag. The reset source identification register would then contain, upon reset, an identification of the most recent hardware reset that has occurred. For example, if the Watchdog Timer overflows and resets the chip, and an external reset occurs before the program can be executed, the reset source identification register will indicate only that an external reset has occurred. No attempt is made to accumulate multiple causes of reset.

It is not necessary (although it would be desirable) to allow clearing of the flags that are assembled for appearance in the reset source register by writing to that register. Once identified, any flags found may be cleared by a software write to the peripheral device in which they reside.

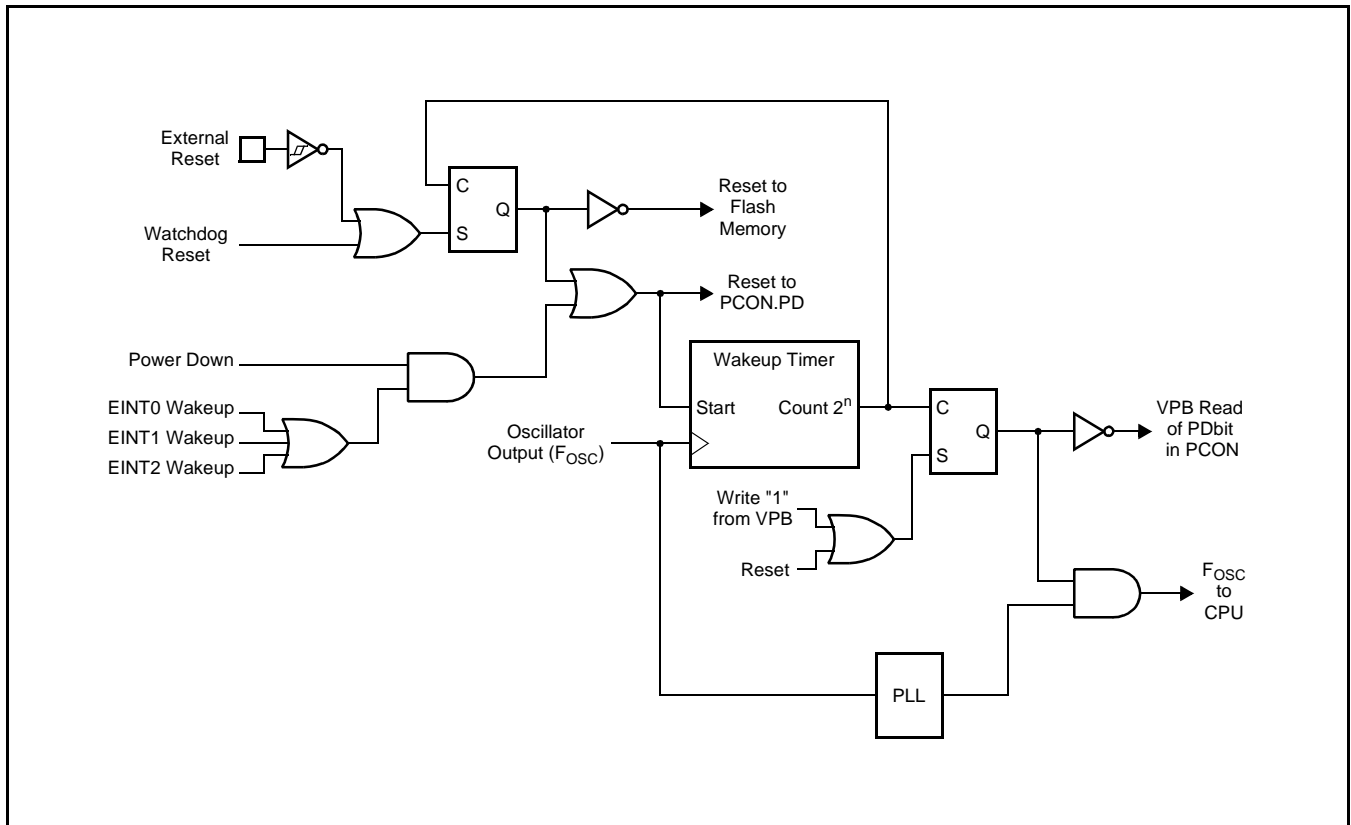


Figure 9: Reset Block Diagram including Wakeup Timer

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 that the VPB bus may not be able to operate correctly at the highest speeds of the ARM processor. In order to compensate for 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 10. 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 22.

Table 21: 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 22: VPB Divider Register (VPBDIV - 0xE01FC100)

VPBDIV	Function	Description	Reset Value
1:0	VPBDIV	<p>The rate of the VPB clock is as follows:</p> <p>0 0: VPB bus clock is one fourth of the processor clock. May be used at any processor clock frequency. Must be used at processor frequencies above TBD MHz.</p> <p>0 1: VPB bus clock is the same as the processor clock. May only be used at processor frequencies below TBD MHz.</p> <p>1 0: VPB bus clock is one half of the processor clock. May be used at processor frequencies below TBD MHz.</p> <p>1 1: Reserved. If this value is written to the VPBDIV register, it has no effect (the previous setting is retained).</p> <p>Warning: Improper setting of this value may result in incorrect operation of the device.</p>	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

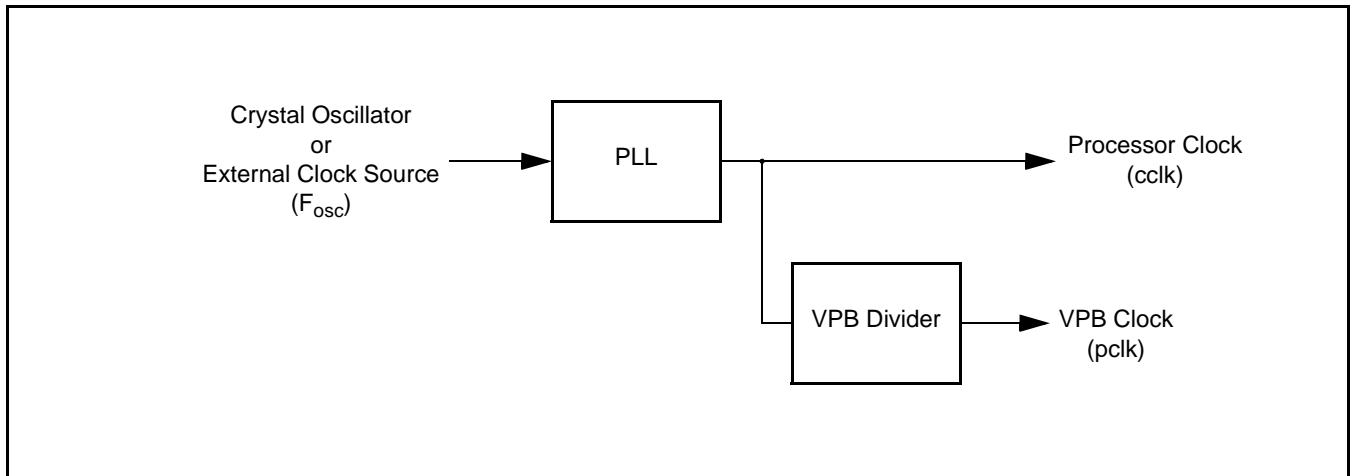


Figure 10: VPB Divider Connections

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 Flash memory to initialize. When the Flash memory 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.

The LPC2106/2105/2104 does not contain any analog function such as comparators that operate without clocks or any independent clock source such as a dedicated Watchdog oscillator. The only remaining functions that can operate in the absence of a clock source are the external interrupts, EINT0, EINT1, and EINT2. If the external interrupt is enabled to cause wakeup and becomes active (is driven low externally), an oscillator wakeup must be initiated. If the interrupt is also enabled in the Vectored Interrupt Controller, the completion of interrupt processing is postponed until the wakeup timer expires.

To summarize: on the LPC2106/2105/2104, 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.

4. 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. Sixteen 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 23. More detailed descriptions follow.

Table 23: VIC Register Map

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

ARM-based Microcontroller

LPC2106/2105/2104

Table 23: VIC Register Map

Address	Name	Description	Access	Reset Value
0xFFFF F128	VICVectAddr10	Vector address 10 register	R/W	0
0xFFFF F12C	VICVectAddr11	Vector address 11 register	R/W	0
0xFFFF F130	VICVectAddr12	Vector address 12 register	R/W	0
0xFFFF F134	VICVectAddr13	Vector address 13 register	R/W	0
0xFFFF F138	VICVectAddr14	Vector address 14 register	R/W	0
0xFFFF F13C	VICVectAddr15	Vector address 15 register	R/W	0
0xFFFF F200	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 F204	VICVectCntl1	Vector control 1 register	R/W	0
0xFFFF F208	VICVectCntl2	Vector control 2 register	R/W	0
0xFFFF F20C	VICVectCntl3	Vector control 3 register	R/W	0
0xFFFF F210	VICVectCntl4	Vector control 4 register	R/W	0
0xFFFF F214	VICVectCntl5	Vector control 5 register	R/W	0
0xFFFF F218	VICVectCntl6	Vector control 6 register	R/W	0
0xFFFF F21C	VICVectCntl7	Vector control 7 register	R/W	0
0xFFFF F220	VICVectCntl8	Vector control 8 register	R/W	0
0xFFFF F224	VICVectCntl9	Vector control 9 register	R/W	0
0xFFFF F228	VICVectCntl10	Vector control 10 register	R/W	0
0xFFFF F22C	VICVectCntl11	Vector control 11 register	R/W	0
0xFFFF F230	VICVectCntl12	Vector control 12 register	R/W	0
0xFFFF F234	VICVectCntl13	Vector control 13 register	R/W	0
0xFFFF F238	VICVectCntl14	Vector control 14 register	R/W	0
0xFFFF F23C	VICVectCntl15	Vector control 15 register	R/W	0

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 24: 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 25: 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 26: 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 27: 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 46 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 28: 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 29: 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 30: 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 31: 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 32: 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 33: 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 34: 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 35: Vector Address Register (VICVectAddr - 0xFFFFF030, Read/Write)

VICVectAddr	Function	Reset Value
31:0	<p>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. Otherwise it returns the address in the Default Vector Address Register.</p> <p>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.</p>	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 36: Protection Enable Register (VICProtection - 0xFFFFF020, Read/Write)

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

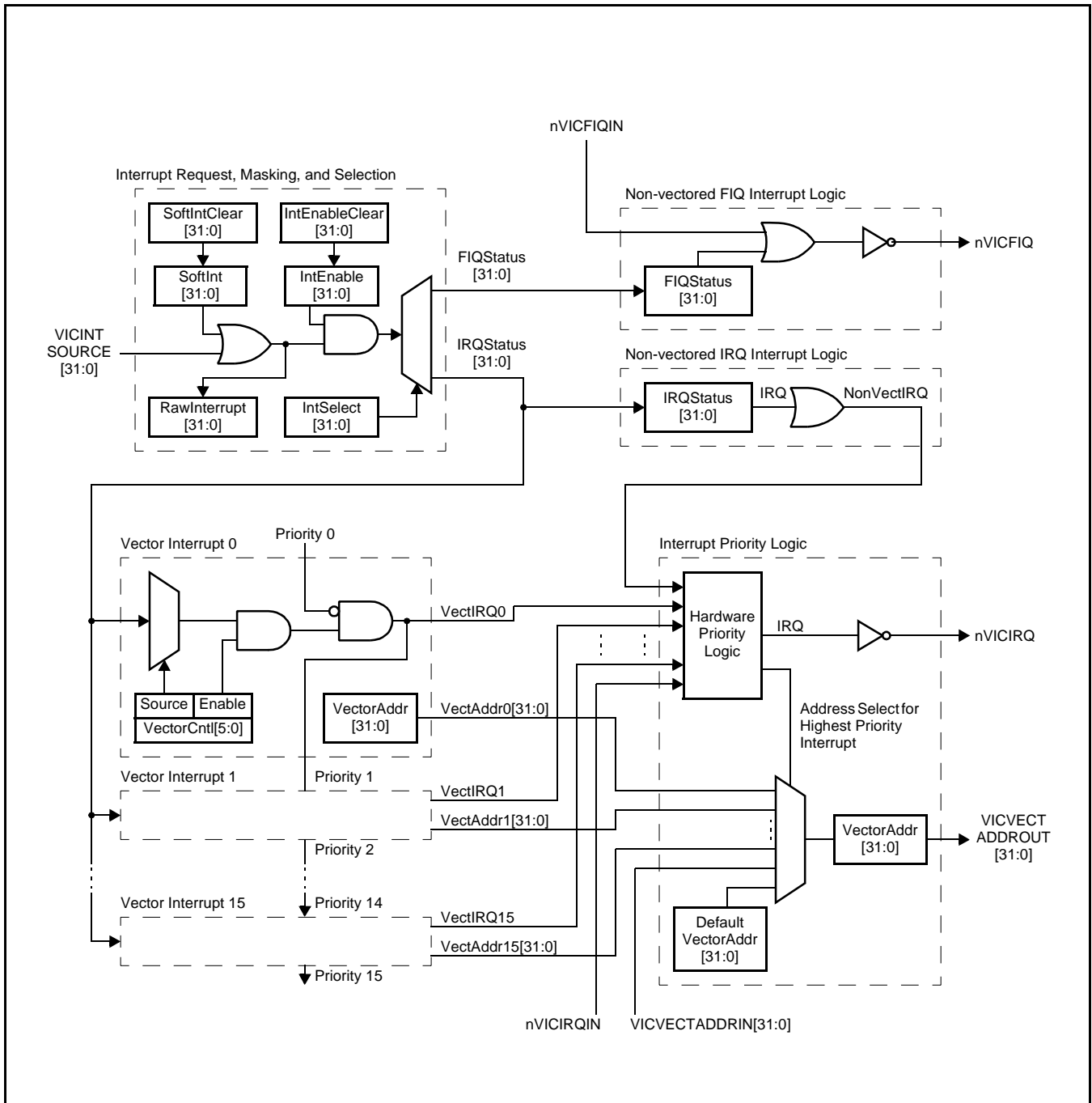


Figure 11: Block Diagram of the Vectored Interrupt Controller

5. 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 37: GPIO Pin Description

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

REGISTER DESCRIPTION

The GPIO contains 4 registers as shown in Table 38.

Table 38: GPIO Register Map

Address	Name	Description	Access
0xE0028000	IOPIN	GPIO 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
0xE0028004	IOSET	GPIO 0 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/Set
0xE0028008	IODIR	GPIO 0 Direction control register. This register individually controls the direction of each port pin.	Read/Write
0xE002800C	IOCLR	GPIO 0 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.	Clear Only

GPIO Pin Value Register (IOPIN - 0xE0028000)

This register provides the value of the GPIO pins. This value reflects any outside world influence on the pins.

Note: for test purposes, writing to this register stores the value in the output register, bypassing the need to use both the IOSET and IOCLR registers. This feature is of little or no use in an application because it is not possible to write to individual bytes in this register.

Table 39: GPIO Pin Value Register (IOPIN - 0xE0028000)

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

GPIO Output Set Register (IOSET - 0xE0028004)

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 in the GPIO output 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 40: GPIO Output Set Register (IOSET - 0xE0028004)

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

GPIO Output Clear Register (IOCLR - 0xE002800C)

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 41: GPIO Output Clear Register (IOCLR - 0xE002800C)

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

GPIO Direction Register (IODIR - 0xE0028008)

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 42: GPIO Direction Register (IODIR - 0xE0028008)

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

6. 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.

REGISTER DESCRIPTION

The Pin Control Module contains 2 registers as shown in Table 43. below.

Table 43: Pin Connect Block Register Map

Address	Name	Description	Access
0xE002C000	PINSEL0	Pin function select register 0	Read/Write
0xE002C004	PINSEL1	Pin function select register 1	Read/Write

Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

The PINSEL0 register controls the functions of the pins as per the settings listed in Table 46. The direction control bit in the IODIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 44: Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

PINSEL0	Pin Name	Value		Function	Value after Reset
1:0	P0.0	0	0	GPIO Port 0.0	0
		0	1	TxD (UART 0)	
		1	0	PWM1	
		1	1	Reserved	
3:2	P0.1	0	0	GPIO Port 0.1	0
		0	1	RxD (UART 0)	
		1	0	PWM3	
		1	1	Reserved	
5:4	P0.2	0	0	GPIO Port 0.2	0
		0	1	SCL (I ² C)	
		1	0	Capture 0.0 (Timer 0)	
		1	1	Reserved	
7:6	P0.3	0	0	GPIO Port 0.3	0
		0	1	SDA (I ² C)	
		1	0	Match 0.0 (Timer 0)	
		1	1	Reserved	
9:8	P0.4	0	0	GPIO Port 0.4	0
		0	1	SCK (SPI)	
		1	0	Capture 0.1 (Timer 0)	
		1	1	Reserved	
11:10	P0.5	0	0	GPIO Port 0.5	0
		0	1	MISO (SPI)	
		1	0	Match 0.1 (Timer 0)	
		1	1	Reserved	
13:12	P0.6	0	0	GPIO Port 0.6	0
		0	1	MOSI (SPI)	
		1	0	Capture 0.2 (Timer 0)	
		1	1	Reserved	
15:14	P0.7	0	0	GPIO Port 0.7	0
		0	1	SSEL (SPI)	
		1	0	PWM2	
		1	1	Reserved	

Table 44: Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

PINSEL0	Pin Name	Value		Function	Value after Reset
17:16	P0.8	0	0	GPIO Port 0.8	0
		0	1	TxD UART 1	
		1	0	PWM4	
		1	1	Reserved	
19:18	P0.9	0	0	GPIO Port 0.9	0
		0	1	RxD (UART 1)	
		1	0	PWM6	
		1	1	Reserved	
21:20	P0.10	0	0	GPIO Port 0.10	0
		0	1	RTS (UART1)	
		1	0	Capture 1.0 (Timer 1)	
		1	1	Reserved	
23:22	P0.11	0	0	GPIO Port 0.11	0
		0	1	CTS (UART1)	
		1	0	Capture 1.1 (Timer 1)	
		1	1	Reserved	
25:24	P0.12	0	0	GPIO Port 0.12	0
		0	1	DSR (UART1)	
		1	0	Match 1.0 (Timer 1)	
		1	1	Reserved	
27:26	P0.13	0	0	GPIO Port 0.13	0
		0	1	DTR (UART 1)	
		1	0	Match 1.1 (Timer 1)	
		1	1	Reserved	
29:28	P0.14	0	0	GPIO Port 0.14	0
		0	1	CD (UART 1)	
		1	0	EINT1	
		1	1	Reserved	
31:30	P0.15	0	0	GPIO Port 0.15	0
		0	1	RI (UART1)	
		1	0	EINT2	
		1	1	Reserved	

Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

The PINSEL1 register controls the functions of the pins as per the settings listed in Table 45. The direction control bit in the IODIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically. Function control for the pins P0.17 - P0.31 is effective only when the DBGSEL input is pulled LOW during RESET.

Table 45: Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

PINSEL1	Pin Name	Value		Function	Value after Reset
1:0	P0.16	0	0	GPIO Port 0.16	0
		0	1	EINT0	
		1	0	Match 0.2 (Timer 0)	
		1	1	Reserved	
3:2	P0.17	0	0	GPIO Port 0.17	0
		0	1	Capture 1.2 (Timer 1)	
		1	0	Reserved	
		1	1	Reserved	
5:4	P0.18	0	0	GPIO Port 0.18	0
		0	1	Capture 1.3 (Timer 1)	
		1	0	Reserved	
		1	1	Reserved	
7:6	P0.19	0	0	GPIO Port 0.19	0
		0	1	Match 1.2 (Timer 1)	
		1	0	Reserved	
		1	1	Reserved	
9:8	P0.20	0	0	GPIO Port 0.20	0
		0	1	Match 1.3 (Timer 1)	
		1	0	Reserved	
		1	1	Reserved	
11:10	P0.21	0	0	GPIO Port 0.21	0
		0	1	PWM5	
		1	0	Reserved	
		1	1	Reserved	
13:12	P0.22	0	0	GPIO Port 0.22	0
		0	1	Reserved	
		1	0	Reserved	
		1	1	Reserved	

Table 45: Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

PINSEL1	Pin Name	Value		Function	Value after Reset
15:14	P0.23	0	0	GPIO Port 0.23	0
		0	1	Reserved	
		1	0	Reserved	
		1	1	Reserved	
17:16	P0.24	0	0	GPIO Port 0.24	0
		0	1	Reserved	
		1	0	Reserved	
		1	1	Reserved	
19:18	P0.25	0	0	GPIO Port 0.25	0
		0	1	Reserved	
		1	0	Reserved	
		1	1	Reserved	
21:20	P0.26	0	0	GPIO Port 0.26	0
		0	1	Reserved	
		1	0	Reserved	
		1	1	Reserved	
23:22	P0.27	0	0	GPIO Port 0.27	0
		0	1	TRST	
		1	0	Reserved	
		1	1	Reserved	
25:24	P0.28	0	0	GPIO Port 0.28	0
		0	1	TMS	
		1	0	Reserved	
		1	1	Reserved	
27:26	P0.29	0	0	GPIO Port 0.29	0
		0	1	TCK	
		1	0	Reserved	
		1	1	Reserved	
29:28	P0.30	0	0	GPIO Port 0.30	0
		0	1	TDI	
		1	0	Reserved	
		1	1	Reserved	

Table 45: Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

PINSEL1	Pin Name	Value		Function	Value after Reset
31:30	P0.31	0	0	GPIO Port 0.31	0
		0	1	TDO	
		1	0	Reserved	
		1	1	Reserved	

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 46: Pin Function Select Register Bits

PinSel0 and PinSel1 Values		Function	Value after Reset
0	0	Primary (default) function, typically GPIO Port	0
0	1	First alternate function	
1	0	Second alternate function	
1	1	Reserved	

The direction control bit in the IODIR 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.

7. UART 0

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.

PIN DESCRIPTION

Table 47: UART 0 Pin Description

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

REGISTER DESCRIPTION

Table 48: UART 0 Register Map

Address Offset	Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value	
0xE000C000 DLAB = 0	RBR	Receiver Buffer Register	MSB READ DATA LSB								RO	un-defined	
0xE000C000 DLAB = 0	THR	Transmit Holding Register	MSB WRITE DATA LSB								WO	NA	
0xE000C004 DLAB = 0	IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0	
0xE000C008	IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	
0xE000C008	FCR	FIFO Control Register	Rx Trigger		Reserved		DMA Mode Select	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0	
0xE000C00C	LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0	
0xE000C014	LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	
0xE000C01C	SCR	Scratch Pad Register	MSB								LSB	R/W	0
0xE000C000 DLAB = 1	DLL	Divisor Latch LSB	MSB								LSB	R/W	0x01
0xE000C004 DLAB = 1	DLM	Divisor Latch MSB	MSB								LSB	R/W	0

UART 0 contains ten 8-bit registers as shown in Table 48. The Divisor Latch Access Bit (DLAB) is contained in LCR7 and enables access to the Divisor Latches.

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

The RBR is the top byte of the 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) must be zero in order to access the RBR. The RBR is always Read Only.

Table 49: Receiver Buffer Register (RBR - 0xE000C000 when DLAB = 0, Read Only)

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

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

The THR is the top byte of the 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) must be zero in order to access the THR. The THR is always Write Only.

Table 50: Transmit Holding Register (THR - 0xE000C000 when DLAB = 0, Write Only)

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

Divisor Latch LSB Register (DLL - 0xE000C000 when DLAB = 1)**Divisor Latch MSB Register (DLM - 0xE000C004 when DLAB = 1)**

The Divisor Latch is part of the 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 DLL and DLM registers together form a 16 bit divisor where DLL contains the lower 8 bits of the divisor and DLM 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) must be one in order to access the Divisor Latches.

Table 51: Divisor Latch LSB Register (DLL - 0xE000C000 when DLAB = 1)

DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The Divisor Latch LSB Register, along with the DLM register, determines the baud rate of the UART.	0x01

Table 52: Divisor Latch MSB Register (DLM - 0xE000C004 when DLAB = 1)

DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The Divisor Latch MSB Register, along with the DLL register, determines the baud rate of the UART.	0

Interrupt Enable Register (IER - 0xE00C004 when DLAB = 0)

The IER is used to enable the four interrupt sources that control the INTR output pin.

Table 53: Interrupt Enable Register Bit Descriptions (IER - 0xE00C004 when DLAB = 0)

IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. IER0 enables the Receive Data Available interrupt. In FIFO mode, IER0 also controls the Character Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. IER1 enables the THRE interrupt. The status of this interrupt can be read from LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. IER2 enables the Rx line status interrupts. The status of this interrupt can be read from LSR[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

Interrupt Identification Register (IIR - 0xE00C008, Read Only)

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

Table 54: Interrupt Identification Register Bit Descriptions (IIR - 0xE00C008, Read Only)

IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that IIR0 is active low. The pending interrupt can be determined by evaluating IER3:1.	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. IER3 identifies an interrupt corresponding to the Rx FIFO. All other combinations of IER3:1 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 FCR0.	0

Interrupts are handled as described in Table 55. Given the status of IIR[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 55. The IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

ARM-based Microcontroller

LPC2106/2105/2104

The RLS interrupt (IIR3:1=011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The Rx error condition that set the interrupt can be observed via LSR4:1. The interrupt is cleared upon an LSR read.

The RDA interrupt (IIR3:1=010) shares the second level priority with the CTI interrupt (IIR3:1=110). In '550 mode, the RDA is activated when the Rx FIFO reaches the trigger level defined in FCR7:6 and is reset when the Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active in '550 mode, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (IIR3:1=110) is a second level interrupt and is set when the Rx FIFO contains at least one character and no Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any Rx FIFO activity (read or write of RSR) will clear the interrupt. This interrupt is intended to flush the 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 55: Interrupt Handling

IIR[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	LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO mode (FCR0=1)	RBR Read or 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	RBR Read
0010	Third	THRE	THRE	IIR Read (if source of interrupt) or THR write
note: values "0000", "0011", "0101", "0111", "1000", "1001", "1010", "1011", "1101", "1110", "1111" are reserved.				

The THRE interrupt (IIR3:1=001) is a third level interrupt and is activated when the THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the 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 THR at one time since the last THRE=1 event. This delay is provided to give the CPU time to write data to THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the THR FIFO has held two or more characters at one time and currently, the THR is empty. The THRE interrupt is reset when a THR write occurs or a read of the IIR occurs and the THRE is the highest interrupt (IIR3:1=001).

FIFO Control Register (FCR - 0xE00C008)

The FCR controls the operation of the Rx and Tx FIFOs. The FCR is present when the FIFO depth parameter is greater than 1.

Table 56: FIFO Control Register Bit Descriptions (FCR - 0xE00C008)

FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both Rx and Tx FIFOs and FCR7:1 access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to FCR1 will clear all bytes in Rx FIFO and reset the pointer logic. This bit is self-clearing.	0
2	Tx FIFO Reset	Writing a logic 1 to FCR2 will clear all bytes in Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
3	DMA Mode Select	0: single transfer mode 1: multi-transfer mode FCR3 affects the operation of the RXRDY_N and TXRDY_N pins.	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	Rx Trigger Level Select	00: trigger level 0 (default='h1) 01: trigger level 1 (default='h4) 10: trigger level 2 (default='h8) 11: trigger level 3 (default='he) These two bits determine how many receiver FIFO characters must be written before an interrupt is activated. The four trigger levels are defined by the user at compilation allowing the user to tune the trigger levels to the FIFO depths chosen.	0

Line Control Register (LCR - 0xE00C00C)

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

Table 57: Line Control Register Bit Descriptions (LCR - 0xE00C00C)

LCR	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 LCR[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 TxD is forced to logic 0 when LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

Line Status Register (LSR - 0xE00C014, Read Only)

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

Table 58: Line Status Register Bit Descriptions (LSR - 0xE00C014, Read Only)

LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: RBR is empty 1: RBR contains valid data LSR0 is set when the RBR holds an unread character and is cleared when the 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 LSR read clears LSR1. LSR1 is set when RSR has a new character assembled and the RBR FIFO is full. In this case, the RBR FIFO will not be overwritten and the character in the 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 LSR read clears LSR2. Time of parity error detection is dependent on FCR0. A parity error is associated with the character being read from the 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 LSR read clears LSR2. The time of the framing error detection is dependent on FCR0. A framing error is associated with the character being read from the 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 RxD 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 RxD goes to marking state (all 1's). An LSR read clears this status bit. The time of break detection is dependent on FCR0. The break interrupt is associated with the character being read from the RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: THR contains valid data. 1: THR is empty. THRE is set immediately upon detection of an empty THR and is cleared on a THR write. The THR is a FIFO as deep as the FIFO_Word_depth parameter defines.	1
6	Transmitter Empty (TEMT)	0: THR and/or the TSR contains valid data. 1: THR and the TSR are empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the TSR or the THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: RBR contains no Rx errors or FCR0=0. 1: RBR contains at least one Rx error. LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the RBR. This bit is cleared when the LSR register is read and there are no subsequent errors in the FIFO.	0

Scratch Pad Register (SCR - 0xE000C01C)

The SCR has no effect on the UART 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 SCR has occurred.

Table 59: Scratchpad Register (SCR - 0xE000C01C)

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

ARCHITECTURE

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

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

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

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

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

The interrupt interface contains registers IER and IIR and controls the interrupt output pin, INTR. The interrupt interface receives several one clock wide enables from the Tx and Rx blocks.

Status information from the Tx and Rx is stored in the LSR. Control information for the Tx and Rx is stored in the LCR.

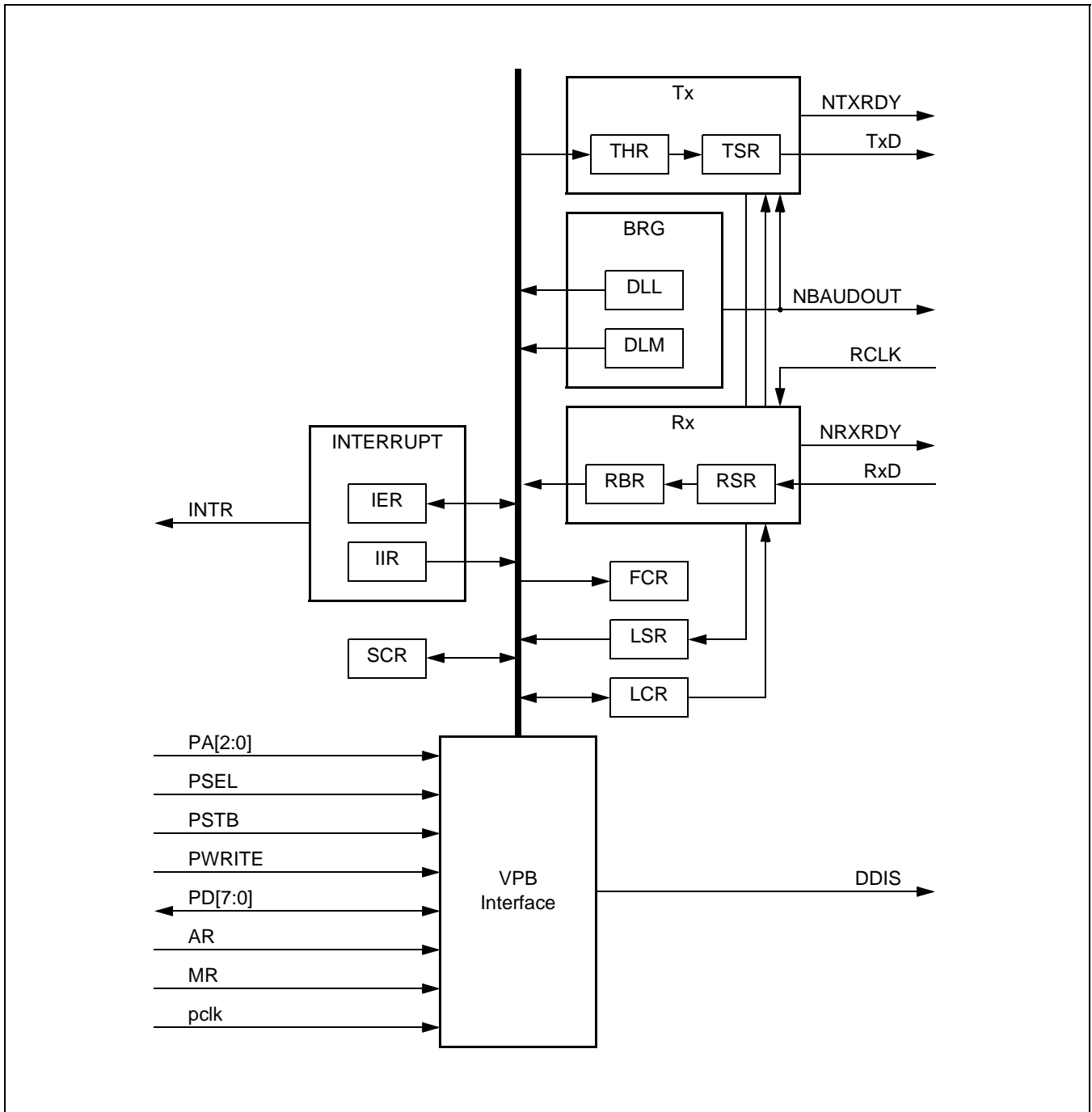


Figure 12: UART Block Diagram

8. UART1

FEATURES

- UART 1 is identical to UART 0, 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.

PIN DESCRIPTION

Table 60: UART1 Pin Description

Pin Name	Type	Description
RxD1	Input	Serial Input. Serial receive data.
TxD1	Output	Serial Output. Serial transmit data.
CTS1	Input	Clear To Send. Active low signal indicates if the external modem is ready to accept transmitted data via TxD from the UART. In normal operation of the modem interface (MCR4=0), the complement value of this signal is stored in MSR4. State change information is stored in MSR0 and is a source for a priority level 4 interrupt, if enabled (IER3=1).
DCD1	Input	Data Carrier Detect. Active low signal indicates if the external modem has established a communication link with the UART and data may be exchanged. In normal operation of the modem interface (MCR4=0), the complement value of this signal is stored in MSR7. State change information is stored in MSR3 and is a source for a priority level 4 interrupt, if enabled (IER3=1).
DSR1	Input	Data Set Ready. Active low signal indicates if the external modem is ready to establish a communications link with the UART. In normal operation of the modem interface (MCR4=0), the complement value of this signal is stored in MSR5. State change information is stored in MSR1 and is a source for a priority level 4 interrupt, if enabled (IER3=1).
DTR1	Output	Data Terminal Ready. Active low signal indicates that the UART is ready to establish connection with external modem. The complement value of this signal is stored in MCR0.
RI1	Input	Ring Indicator. Active low signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (MCR4=0), the complement value of this signal is stored in MSR6. State change information is stored in MSR2 and is a source for a priority level 4 interrupt, if enabled (IER3=1).
RTS1	Output	Request To Send. Active low signal indicates that the UART would like to transmit data to the external modem. The complement value of this signal is stored in MCR1.

REGISTER DESCRIPTION

Table 61: UART 1 Register Map

Address	Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value	
0xE0010000 DLAB = 0	RBR	Receiver Buffer Register	MSB READ DATA LSB								RO	un-defined	
0xE0010000 DLAB = 0	THR	Transmit Holding Register	MSB WRITE DATA LSB								WO	NA	
0xE0010004 DLAB = 0	IER	Interrupt Enable Register	0	0	0	0	Enable Modem Status Interrupt	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0	
0xE0010008	IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	
0xE0010008	FCR	FIFO Control Register	Rx Trigger		Reserved		DMA Mode Select	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0	
0xE001000C	LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0	
0xE0010010	MCR	Modem Control Register	0	0	0	Loop Back	0	0	RTS	DTR	R/W	0	
0xE0010014	LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	
0xE0010018	MSR	Modem Status Register	DCD	RI	DSR	CTS	Delta DCD	Trailing Edge RI	Delta DSR	Delta CTS	RO	0	
0xE001001C	SCR	Scratch Pad Register	MSB								LSB	R/W	0
0xE0010000 DLAB = 1	DLL	Divisor Latch LSB	MSB								LSB	R/W	0
0xE0010004 DLAB = 1	DLM	Divisor Latch MSB	MSB								LSB	R/W	0

UART1 contains twelve 8-bit registers as shown in Table 61. The Divisor Latch Access Bit (DLAB) is contained in LCR7 and enables access to the Divisor Latches.

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

The RBR is the top byte of the 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) must be zero in order to access the RBR. The RBR is always Read Only.

Table 62: Receiver Buffer Register (RBR - 0xE0010000 when DLAB = 0, Read Only)

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

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

The THR is the top byte of the 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) must be zero in order to access the THR. The THR is always Write Only.

Table 63: Transmit Holding Register (THR - 0xE0010000 when DLAB = 0, Write Only)

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

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

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

The Divisor Latch is part of the 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 DLL and DLM registers together form a 16 bit divisor where DLL contains the lower 8 bits of the divisor and DLM 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) must be one in order to access the Divisor Latches.

Table 64: Divisor Latch LSB Register (DLL - 0xE0010000 when DLAB = 1)

DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The Divisor Latch LSB Register, along with the DLM register, determines the baud rate of the UART.	0x01

Table 65: Divisor Latch MSB Register (DLM - 0xE0010004 when DLAB = 1)

DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The Divisor Latch MSB Register, along with the DLL register, determines the baud rate of the UART.	0

Interrupt Enable Register (IER - 0xE0010004 when DLAB = 0)

The IER is used to enable the four interrupt sources that control the INTR output pin.

Table 66: IER Bit Descriptions (IER - 0xE0010004 when DLAB = 0)

IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. IER0 enables the Receive Data Available interrupt. In FIFO mode, IER0 also controls the Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. IER1 enables the THRE interrupt. The status of this interrupt can be read from LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. IER2 enables the Rx line status interrupts. The status of this interrupt can be read from LSR[4:1].	0
3	Modem Status Interrupt Enable	0: Disable the modem interrupt. 1: Enable the modem interrupt. IER3 enables the modem interrupt. The status of this interrupt can be read from MSR[3:0].	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 Identification Register (IIR - 0xE0010008, Read Only)

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

Table 67: IIR Bit Descriptions (IIR - 0xE0010008, Read Only)

IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that IIR0 is active low. The pending interrupt can be determined by evaluating IIR3:1.	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. All other combinations of IER3:1 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.	NA
7:6	FIFO Enable	These bits are equivalent to FCR0.	0

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

The RLS interrupt (IIR3:1=011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The Rx error condition that set the interrupt can be observed via LSR4:1. The interrupt is cleared upon an LSR read.

The RDA interrupt (IIR3:1=010) shares the second level priority with the CTI interrupt (IIR3:1=110). The RDA is activated when the Rx FIFO reaches the trigger level defined in FCR7:6 and is reset when the Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active in '550 mode, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (IIR3:1=110) is a second level interrupt and is set when the Rx FIFO contains at least one character and no Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any Rx FIFO activity (read or write of RSR) will clear the interrupt. This interrupt is intended to flush the 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: Interrupt Handling

IIR[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	LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO mode (FCR0=1)	RBR Read or 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	RBR Read
0010	Third	THRE	THRE	IIR Read (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read
note: values "0011", "0101", "0111", "1000", "1001", "1010", "1011", "1101", "1110", "1111" are reserved.				

The THRE interrupt (IIR3:1=001) is a third level interrupt and is activated when the THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the 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 THR at one time since the last THRE=1 event. This delay is provided to give the CPU time to write data to THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the THR FIFO has held two or more characters at one time and currently, the THR is empty. The THRE interrupt is reset when a THR write occurs or a read of the IIR occurs and the THRE is the highest interrupt (IIR3:1=001).

The modem interrupt (IIR3:1=000) 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 MSR3:0. A MSR read will clear the modem interrupt.

FIFO Control Register (FCR - 0xE0010008)

The FCR controls the operation of the Rx and Tx FIFOs. The FCR is present when the FIFO depth parameter is greater than 1.

Table 69: FCR Bit Descriptions (FCR - 0xE0010008)

FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both Rx and Tx FIFOs and FCR7:1 access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to FCR1 will clear all bytes in Rx FIFO and reset the pointer logic. This bit is self-clearing.	0
2	Tx FIFO Reset	Writing a logic 1 to FCR2 will clear all bytes in Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
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	Rx Trigger Level Select	00: trigger level 0 (default='h1) 01: trigger level 1 (default='h4) 10: trigger level 2 (default='h8) 11: trigger level 3 (default='he) These two bits determine how many receiver FIFO characters must be written before an interrupt is activated. The four trigger levels are defined by the user at compilation allowing the user to tune the trigger levels to the FIFO depths chosen.	0

Line Control Register (LCR - 0xE001000C)

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

Table 70: LCR Bit Descriptions (LCR - 0xE001000C)

LCR	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 LCR[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 TxD is forced to logic 0 when LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

Modem Control Register (MCR - 0xE0010010)

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

Table 71: MCR Bit Descriptions (MCR - 0xE0010010)

MCR	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, RxD, has no effect on loopback and output pin, TxD is held in marking state. The four modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the four modem outputs are connected to the four modem inputs. As a result of these connections, the upper four bits of the MSR will be driven by the lower four bits of the MCR rather than the four modem inputs in normal mode (). This permits modem status interrupts to be generated in loopback mode by writing the lower four bits of MCR.	0
7:5	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Line Status Register (LSR - 0xE0010014, Read Only)

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

Table 72: LSR Bit Descriptions (LSR - 0xE0010014, Read Only)

LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: RBR is empty 1: RBR contains valid data LSR0 is set when the RBR holds an unread character and is cleared when the 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 LSR read clears LSR1. LSR1 is set when RSR has a new character assembled and the RBR FIFO is full. In this case, the RBR FIFO will not be overwritten and the character in the 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 LSR read clears LSR2. Time of parity error detection is dependent on FCR0. A parity error is associated with the character being read from the 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 LSR read clears this bit. The time of the framing error detection is dependent on FCR0. A framing error is associated with the character being read from the 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 RxD 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 RxD goes to marking state (all 1's). An LSR read clears this status bit. The time of break detection is dependent on FCR0. The break interrupt is associated with the character being read from the RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: THR contains valid data. 1: THR is empty. THRE is set immediately upon detection of an empty THR and is cleared on a THR write. The THR is a FIFO as deep as the FIFO_Word_depth parameter defines.	1
6	Transmitter Empty (TEMT)	0: THR and/or the TSR contains valid data. 1: THR and the TSR are empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the TSR or the THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: RBR contains no Rx errors or FCR0=0. 1: RBR contains at least one Rx error. LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the RBR. This bit is cleared when the LSR register is read and there are no subsequent errors in the FIFO.	0

Modem Status Register (MSR - 0x0xE0010018)

The MSR is a read-only register that provides status information on the modem input signals. MSR3:0 is cleared on MSR read. Note that modem signals have no direct affect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 73: MSR Bit Descriptions (MSR - 0x0xE0010018)

MSR	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 MSR 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 MSR 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 MSR 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 MSR read.	0
4	CTS	Clear To Send State. Complement of input signal CTS. This bit is connected to MCR[1] in modem loopback mode.	0
5	DSR	Data Set Ready State. Complement of input signal DSR. This bit is connected to MCR[0] in modem loopback mode.	0
6	RI	Ring Indicator State. Complement of input RI. This bit is connected to MCR[2] in modem loopback mode.	0
7	DCD	Data Carrier Detect State. Complement of input DCD. This bit is connected to MCR[3] in modem loopback mode.	0

Scratch Pad Register (SCR - 0xE001001C)

The SCR has no effect on the UART 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 SCR has occurred.

Table 74: Scratchpad Register (SCR - 0xE001001C)

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

ARCHITECTURE

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

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

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

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

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

The modem interface contains registers MCR and MSR. This interface is responsible for handshaking between a modem peripheral and the UART.

The interrupt interface contains registers IER and IIR and controls the interrupt output pin, INTR. The interrupt interface receives several one clock wide enables from the Tx, Rx and modem blocks.

Status information from the Tx and Rx is stored in the LSR. Control information for the Tx and Rx is stored in the LCR.

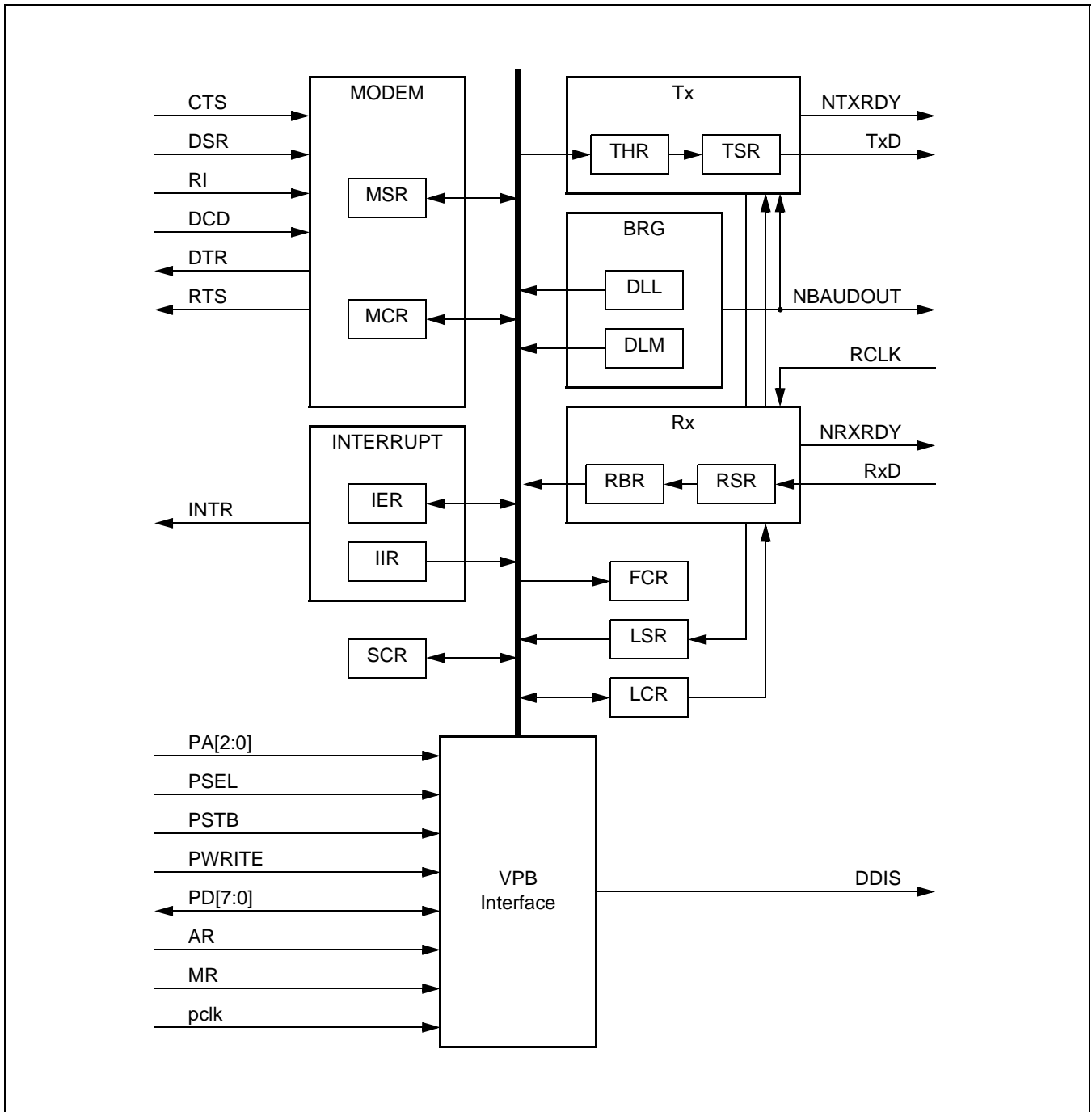


Figure 13: UART Block Diagram

9. I²C INTERFACE

FEATURES

- Standard I²C compliant bus interface.
- Easy to configure as Master, Slave, or Master/Slave.
- Programmable clocks allow versatile rate control.
- Bidirectional data transfer between masters and slaves.
- Multi-master bus (no central master).
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- 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 14. 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.

This device provides a byte oriented I²C interface. It has four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

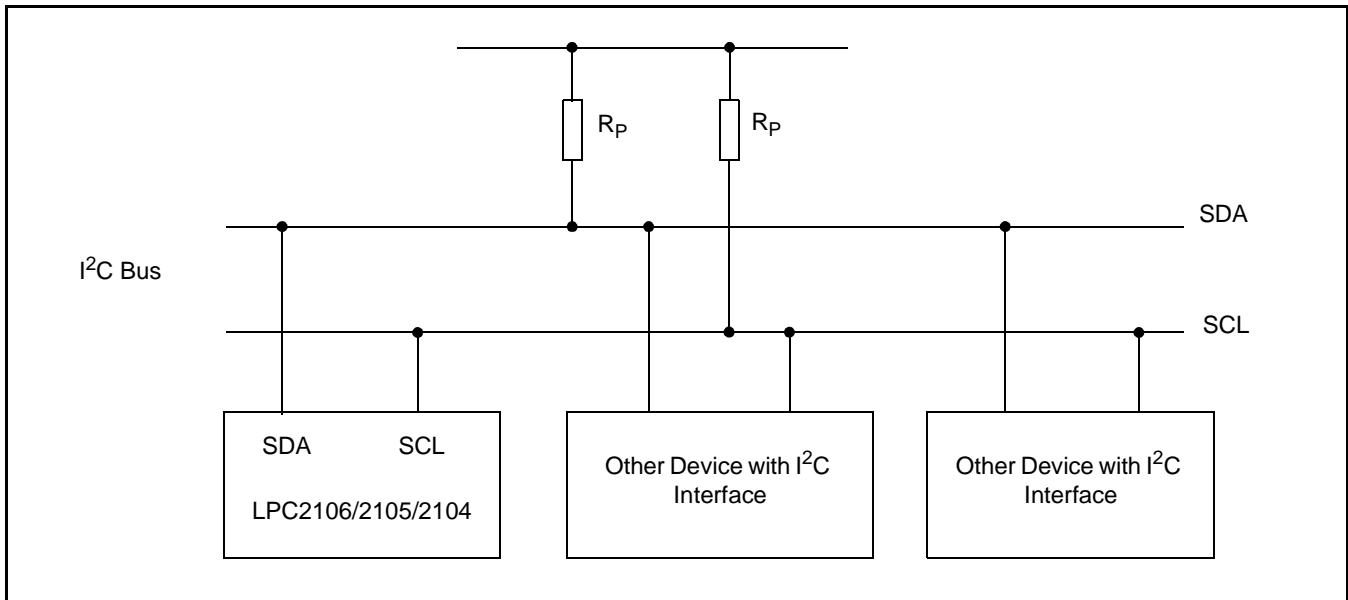


Figure 14: I²C Bus Configuration

I²C Operating Modes

Master Transmitter Mode:

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, I2CONSET must be initialized as shown in Figure 15. 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 15: Slave 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 I²C interface will enter master transmitter mode when software sets the STA bit. The I²C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in I2STAT should be 08h. This status code must be used to vector to an interrupt service routine which should load the slave address and Write bit to I2DAT (Data 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 3 to Table 6 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf.

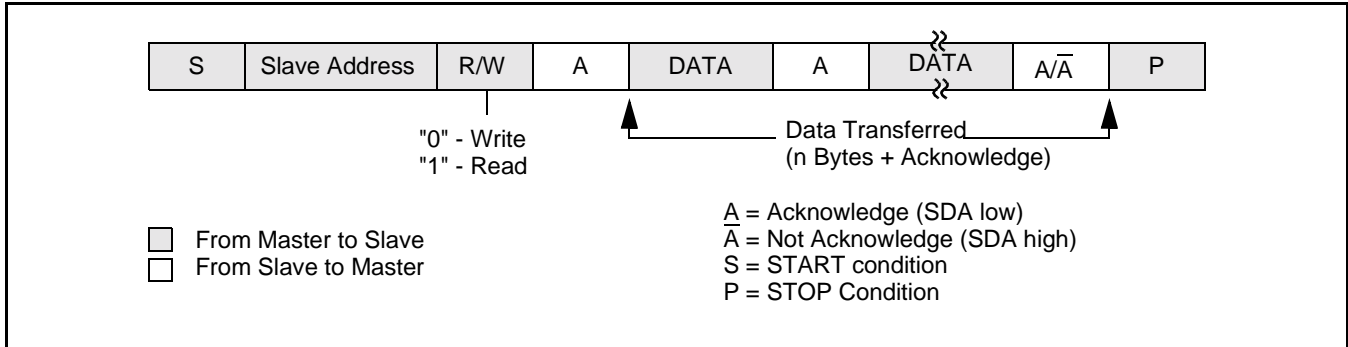


Figure 16: Format in the master transmitter mode

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 I²C Data Register (I2DAT), and then clear the SI bit.

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. Refer to Table 4 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

for details.

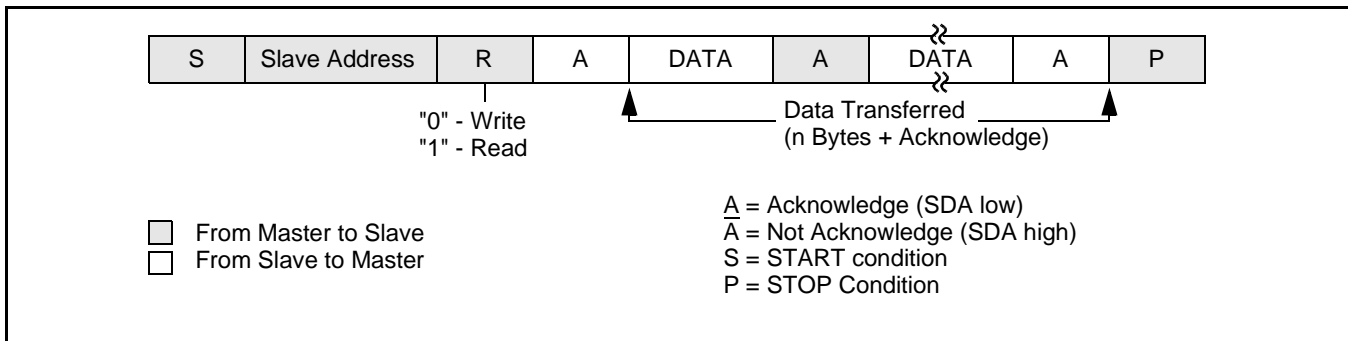


Figure 17: Format of master receiver mode

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

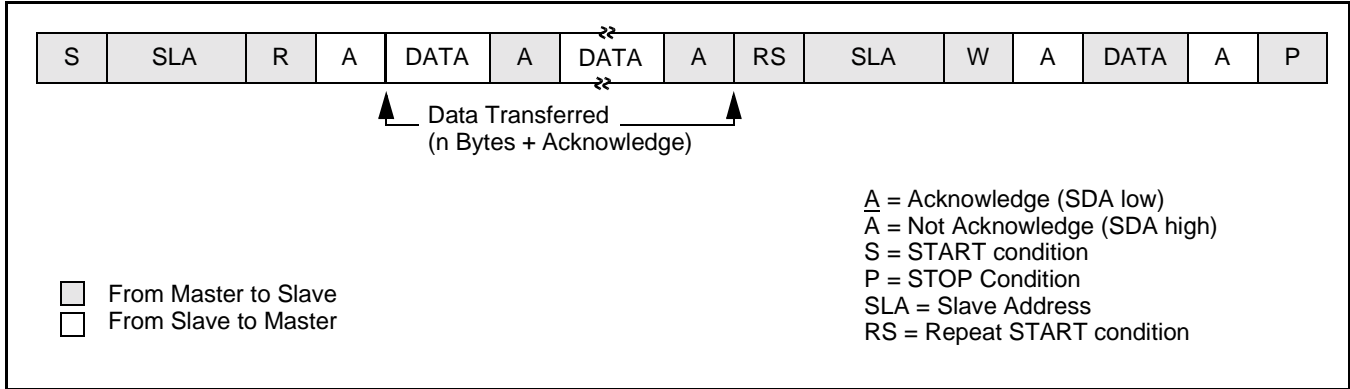


Figure 18: A master receiver switch to master transmitter after sending repeated START

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 I²C Control Set Register (I2CONSET) as shown in Figure 19.

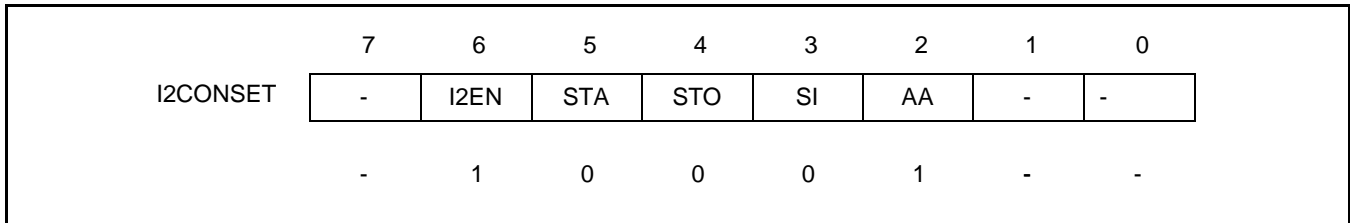


Figure 19: Slave Mode Configuration

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

After I2ADR and I2CONSET are initialized, the I²C interface waits until it is addressed by its own address or general address followed by the data direction bit. If the direction bit is 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 5 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

for the status codes and actions.

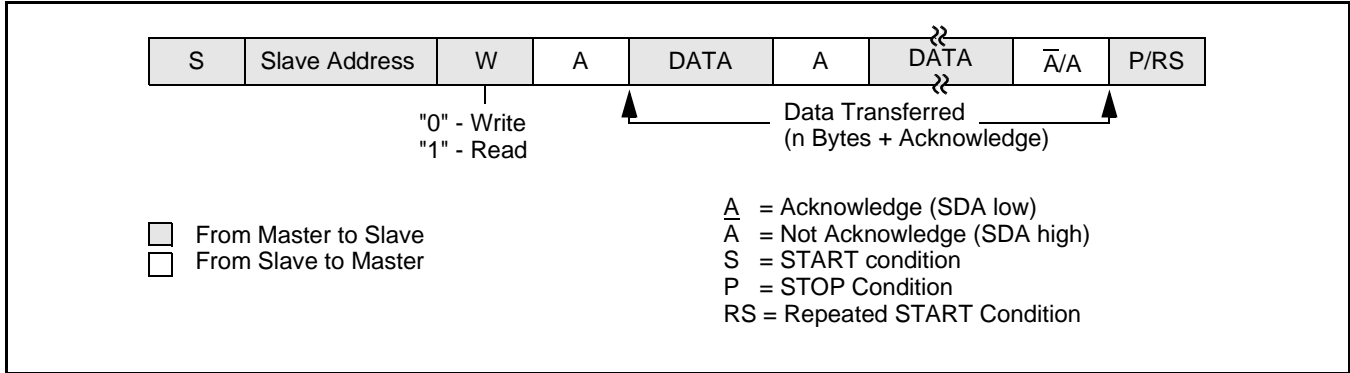


Figure 20: Format of slave receiver mode

Slave Transmitter Mode:

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I²C may operate as a master and as a slave. In the slave mode, the I²C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller 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, I²C switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

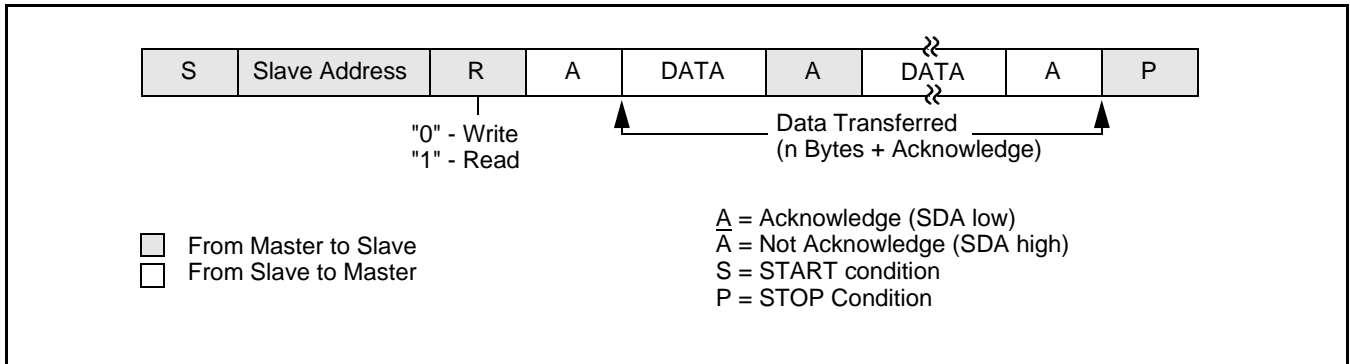


Figure 21: Format of slave transmitter mode

PIN DESCRIPTION

Table 75: I2C Pin Description

Pin Name	Type	Description
SDA	Input/Output	Serial Data. I ² C data input and output. The associated port pin has an open drain output in order to conform to I ² C specifications.
SCL	Input/Output	Serial Clock. I ² C clock input and output. The associated port pin has an open drain output in order to conform to I ² C specifications.

REGISTER DESCRIPTION

The I²C interface contains 7 registers as shown in Table 76. below.

Table 76: I²C Register Map

Address	Name	Description	Access	Reset Value
0xE001C000	I2CONSET	I ² C Control Set Register	Read/Set	0
0xE001C004	I2STAT	I ² C Status Register	Read Only	0xF8
0xE001C008	I2DAT	I ² C Data Register	Read/Write	0
0xE001C00C	I2ADR	I ² C Slave Address Register	Read/Write	0
0xE001C010	I2SCLH	SCL Duty Cycle Register High Half Word	Read/Write	0x04
0xE001C014	I2SCLL	SCL Duty Cycle Register Low Half Word	Read/Write	0x04
0xE001C018	I2CONCLR	I ² C Control Clear Register	Clear Only	NA

I²C Control Set Register (I2CONSET - 0xE001C000)

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. The address in the Slave Address Register has been received.
2. The general call address has been received while the general call bit(GC) in I2ADR is set.
3. A data byte has been received while the I²C is in the master receiver mode.
4. A data byte has been received while the I²C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I²C is in the master receiver mode.
2. A data byte has been received while the I²C is in the addressed slave receiver mode.

SI is the I²C Interrupt Flag. This bit is set when one of the 25 possible I²C states is entered. Typically, the I²C interrupt should only be used to indicate a start condition at an idle slave device, or a stop condition at an idle master device (if it is waiting to use the I²C bus). SI is cleared by writing a 1 to the SIC bit in I2CONCLR register.

STO is the STOP flag. Setting this bit causes the I²C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I²C bus. When the bus detects the STOP condition, STO is cleared automatically.

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

STA is the START flag. Setting this bit causes the I²C interface to enter master mode and transmit a START condition or transmit a repeated START condition if it is already in master mode.

When STA is 1 and the I²C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I²C interface is already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I²C interface is in an addressed slave mode.

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

If STA and STO are both set, then a STOP condition is transmitted on the I²C bus if the interface is in master mode, and transmits a START condition thereafter. If the I²C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

ARM-based Microcontroller

LPC2106/2105/2104

I2EN I²C Interface Enable. When I2EN is 1, the I²C function is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I²C function is disabled.

Table 77: I²C Control Set Register (I2CONSET - 0xE001C000)

I2CONSET	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.	NA
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	0
3	SI	I ² C interrupt flag	0
4	STO	STOP flag	0
5	STA	START flag	0
6	I2EN	I ² C interface enable	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I²C Control Clear Register (I2CONCLR - 0xE001C018)

Table 78: I²C Control Clear Register (I2CONCLR - 0xE001C018)

I2CONCLR	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.	NA
1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.	NA
3	SIC	I ² C Interrupt Clear Bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.	NA
4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	Start flag clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.	NA
6	I2ENC	I ² C interface disable. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.	NA
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I²C Status Register (I2STAT - 0xE001C004)

This is a read-only register. It contains the status code of the I²C interface. The least three bits are always 0. There are 26 possible status codes. When the code is F8H, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I²C states. When any of these states entered, SI bit will be set. Refer to Table 3 to Table 6 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

for a complete list of status codes.

Table 79: I²C Status Register (I2STAT - 0xE001C004)

I2STAT	Function	Description	Reset Value
2:0	Status	These bits are always 0	0
7:3	Status	Status bits	1

I²C Data Register (I2DAT - 0xE001C008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register while it is not in the process of shifting a byte. This register can be accessed only when 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 80: I²C Data Register (I2DAT - 0xE001C008)

I2DAT	Function	Description	Reset Value
7:0	Data	Transmit/Receive data bits	0

I²C Slave Address Register (I2ADR - 0xE001C00C)

This register is readable and writable, and is only used when the I²C 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 81: I²C Slave Address Register (I2ADR - 0xE001C00C)

I2ADR	Function	Description	Reset Value
0	GC	General Call bit	0
7:1	Address	Slave mode address	0

I²C SCL Duty Cycle Registers (I2SCLH - 0xE001C010 and I2SCLL - 0xE001C014)

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

$$\text{Bit Frequency} = f_{\text{CLK}} / (\text{I2SCLH} + \text{I2SCLL})$$

Where f_{CLK} is the frequency of pclk.

The values for I2SCLL and I2SCLH don't have to be the same. Software can set different duty cycles on SCL by setting these two registers. But the value of the register must ensure that the data rate is in the I²C data rate range of 0 through 400KHz. So the value of I2SCLL and I2SCLH has some restrictions. Each register value should be greater than or equal to 4.

Table 82: I²C SCL High Duty Cycle Register (I2SCLH - 0xE001C010)

I2SCLH	Function	Description	Reset Value
15:0	Count	Count for SCL HIGH time period selection	0x 0004

Table 83: I²C SCL Low Duty Cycle Register (I2SCLL - 0xE001C014)

I2SCLL	Function	Description	Reset Value
15:0	Count	Count for SCL LOW time period selection	0x 0004

Table 84: I2C Clock Rate Selections for VPB Clock Divider = 1

I2SCLL+ I2SCLH	Bit Frequency (kHz) At f_{CCLK} (MHz) & VPB Clock Divider = 1				
	16	20	40	60	80
8	-	-	-	-	-
10	-	-	-	-	-
25	-	-	-	-	-
50	320.0	400.0	-	-	-
75	213.333	266.667	-	-	-
100	160.0	200.0	400.0	-	-
160	100.0	125.0	250.0	375.0	-
200	80.0	100.0	200.0	300.0	400.0
320	50.0	62.5	125.0	187.5	250.0
400	40.0	50.0	100.0	150.0	200.0
510	31.373	39.216	78.431	117.647	156.863
800	20.0	25.0	50.0	75.0	100.0
1280	12.5	15.625	31.25	46.875	62.5

ARM-based Microcontroller

LPC2106/2105/2104

Table 85: I2C Clock Rate Selections for VPB Clock Divider = 2

I2SCLL+ I2SCLH	Bit Frequency (kHz) At f_{CCLK} (MHz) & VPB Clock Divider = 2				
	16	20	40	60	80
8	-	-	-	-	-
10	-	-	-	-	-
25	320.0	400.0	-	-	-
50	160.0	200.0	400.0	-	-
75	106.667	133.333	266.667	400.0	-
100	80.0	100.0	200.0	300.0	400.0
160	50.0	62.5	125.0	187.5	250.0
200	40.0	50.0	100.0	150.0	200.0
320	25.0	31.25	62.5	93.75	125.0
400	20.0	25.0	50.0	75.0	100.0
510	15.686	19.608	39.216	58.824	78.431
800	10.0	12.5	25.0	37.5	50.0
1280	6.25	7.813	15.625	23.438	31.25

Table 86: I2C Clock Rate Selections for VPB Clock Divider = 4

I2SCLL+ I2SCLH	Bit Frequency (kHz) At f_{CCLK} (MHz) & VPB Clock Divider = 4				
	16	20	40	60	80
8	500.0	-	-	-	-
10	400.0	-	-	-	-
25	160.0	200.0	400.0	-	-
50	80.0	100.0	200.0	300.0	400.0
75	53.333	66.667	133.333	200.0	266.667
100	40.0	50.0	100.0	150.0	200.0
160	25.0	31.25	62.5	93.75	125.0
200	20.0	25.0	50.0	75.0	100.0
320	12.5	15.625	31.25	46.875	62.5
400	10.0	12.5	25.0	37.5	50.0
510	7.843	9.804	19.608	29.412	39.216
800	5.0	6.25	12.5	18.75	25.0
1280	3.125	3.906	7.813	11.719	15.625

ARCHITECTURE

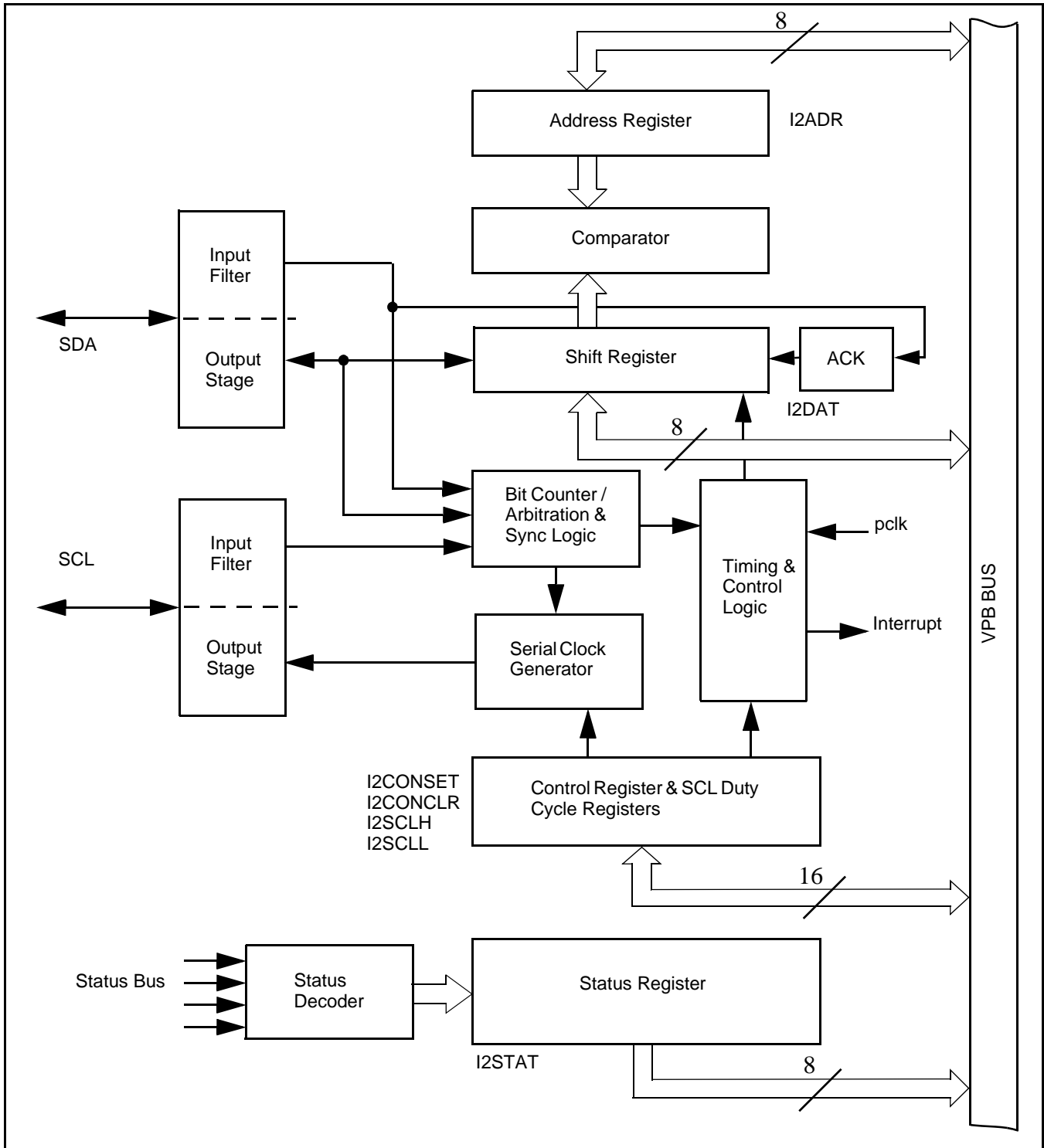


Figure 22: I²C Architecture

10. SPI INTERFACE

FEATURES

- 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

The SPI is a full duplex serial interface. It is designed to be able to 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 23 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 with CPOL set to both 0 and 1. The second point to note is the activation and de-activation of the SSEL signal. When CPHA = 1, the SSEL signal will always go inactive between data transfers. This is not guaranteed when CPHA = 0 (the signal can remain active).

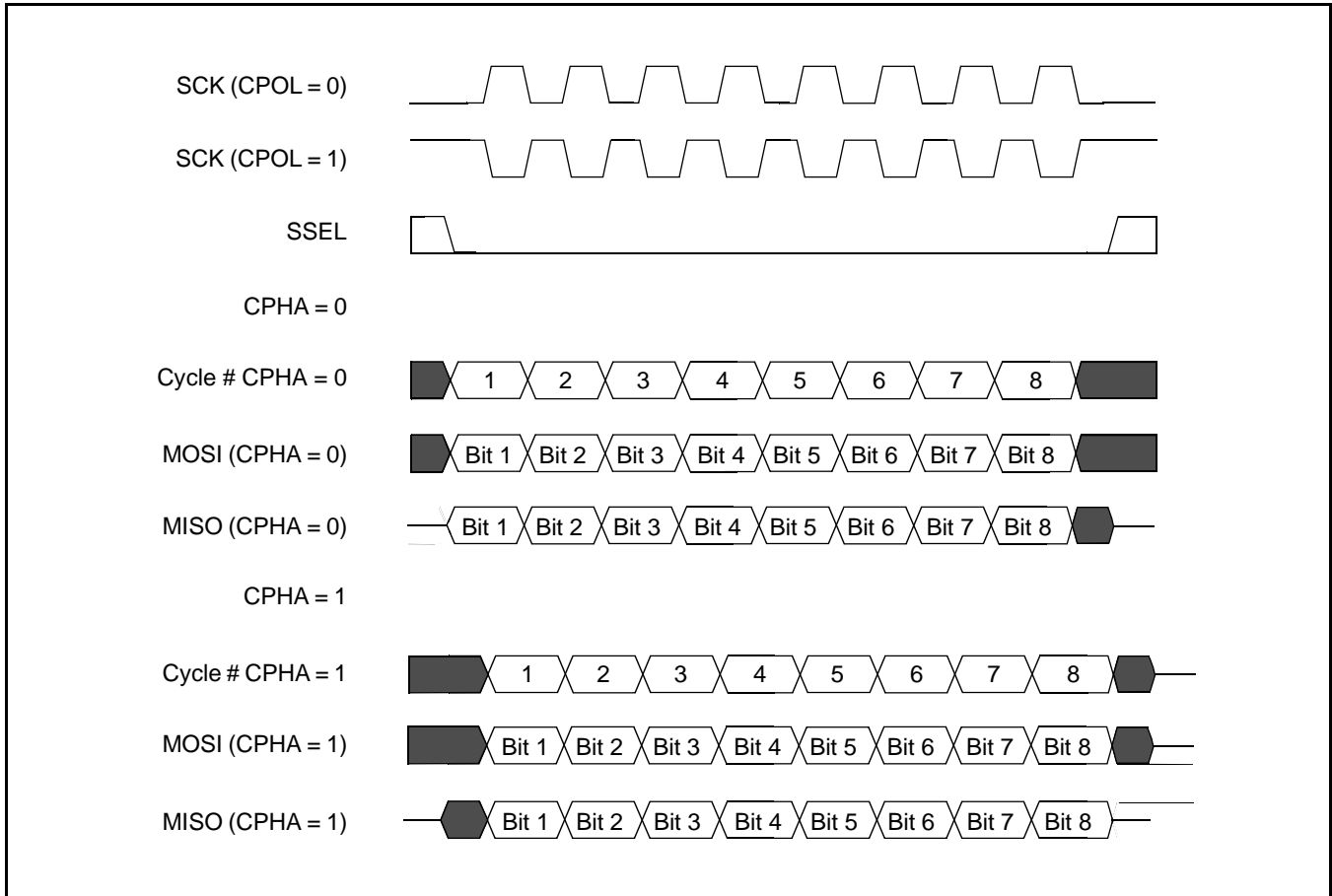


Figure 23: SPI Data Transfer Format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in Table 87. 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 87: 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 be 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 be 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 88: SPI Pin Description

Pin Name	Type	Pin Description
SCK	Input/ Output	Serial Clock. The SPI is a clock signal used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL	Input	Slave Select. The SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. Each slave has its own unique slave select signal input. The SSEL must be low before data transactions begin and normally stays low for the duration of the transaction. If the SSEL signal goes high any time during a data transfer, the transfer is considered to be aborted. In this event, the slave returns to idle, and any data that was received is thrown away. There are no other indications of this exception. This signal is not directly driven by the master. It could be driven by a simple general purpose I/O under software control.
MISO	Input/ Output	Master In Slave Out. The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. When a device is a slave, serial data is output on this signal. When a device is a master, serial data is input on this signal. When a slave device is not selected, the slave drives the signal high impedance.
MOSI	Input/ Output	Master Out Slave In. The MOSI signal is a unidirectional signal used to transfer serial data from the master to the slave. When a device is a master, serial data is output on this signal. When a device is a slave, serial data is input on this signal.

REGISTER DESCRIPTION

The SPI contains 7 registers as shown in Table 89. All registers are byte, half word and word accessible.

Table 89: SPI Register Map

Address	Name	Description	Access	Reset Value
0xE0020000	SPCR	SPI Control Register. This register controls the operation of the SPI.	Read/Write	0
0xE0020004	SPSR	SPI Status Register. This register shows the status of the SPI.	Read Only	0
0xE0020008	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
0xE002000C	SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK.	Read/Write	0
0xE0020010	SPTCR	SPI Test Control Register.	Read/Write	0
0xE0020014	SPTSR	SPI Test Status Register.	Read/Write	0
0xE0020018	SPTOR	SPI Test Observe Register.	Read Only	0
0xE002001C	SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	Read/Write	0

SPI Control Register (SPCR - 0xE0020000)

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

Table 90: SPI Control Register (SPCR - 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 (SPSR - 0xE0020004)

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

Table 91: SPI Status Register (SPSR - 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 (SPDR - 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 92: SPI Data Register (SPDR - 0xE0020008)

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

SPI Clock Counter Register (SPCCR - 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 93: SPI Clock Counter Register (SPCCR - 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 (SPINT - 0xE002001C)

This register contains the interrupt flag for the SPI interface.

Table 94: SPI Interrupt Register (SPINT - 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.	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

SPI Test Control Register (SPTCR - 0xE0020010)

Note that the bits in this register are intended for functional verification only. This register should not be used for normal operation.

Table 95: SPI Test Control Register (SPTCR - 0xE0020010)

SPTCR	Function	Description	Reset Value
0	Test	SPI test mode. When 0, the SPI operates normally. When 1, SCK will always be on, independent of master mode select, and data availability setting.	0
7:1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

SPI Test Status Register (SPTSR - 0xE0020014)

Note that the bits in this register are intended for functional verification only. This register should not be used for normal operation.

This register is a replication of the SPI status register. The difference between the registers is that a read of this register will not start the sequence of events required to clear these status bits. A write to this register will set an interrupt if the write data for the respective bit is a 1.

Table 96: SPI Test Status Register (SPTSR - 0xE0020014)

SPTSR	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	0

Table 96: SPI Test Status Register (SPTSR - 0xE0020014)

4	MODF	Mode fault	0
5	ROVR	Read overrun	0
6	WCOL	Write collision	0
7	SPIF	SPI transfer complete flag.	0

SPI Test Observe Register (SPTOR - 0xE0020018)

Note that the bits in this register are intended for functional verification only. This register should not be used for normal operation.

This read only register is used to observe the state of some of the internal signals and state machines in the SPI entity.

State	Bit 0	Bit 1
Master Transfer0	1	
Slave Transfer1	0	

Table 97: SPI Test Observe Register (SPTOR - 0xE0020018)

SPTOR	Function	Description	Reset Value
1:0	STATE	Current value of the internal state machine.	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

ARCHITECTURE

The block diagram of the SPI is shown in the Figure 24.

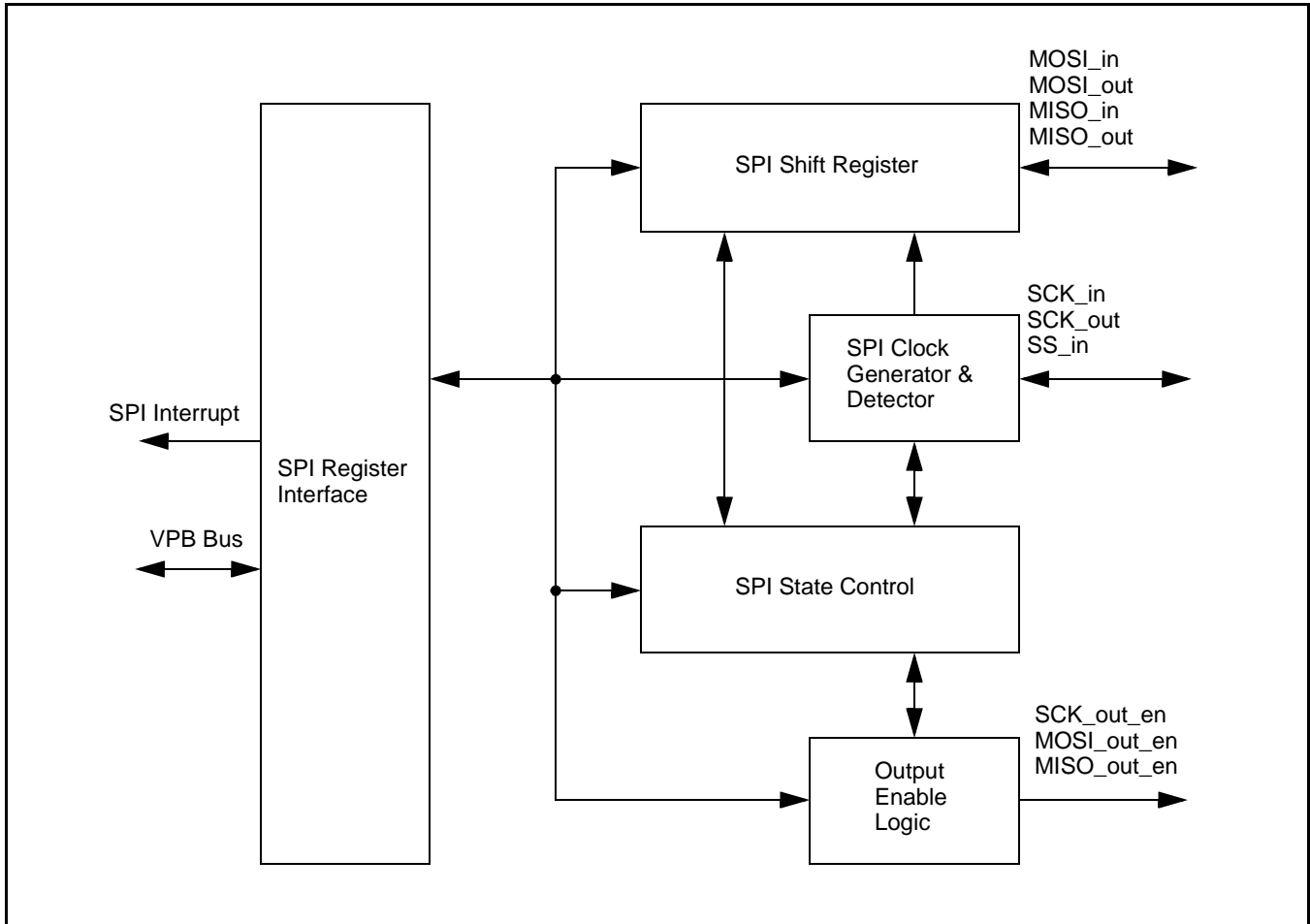


Figure 24: SPI Block Diagram

11. TIMER 0 AND TIMER 1

Timer 0 and Timer 1 are identical except for the peripheral base address.

FEATURES

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Up to four (Timer 1) and three (Timer 0) 32-bit capture channels, 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 (Timer 1) and three (Timer 0) 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 is designed to count cycles of the peripheral clock (pclk) and 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.

Due to the limited number of pins on the LPC2106/2105/2104, only three of the Capture inputs and Match outputs of Timer 0 are connected to device pins.

PIN DESCRIPTION

Table 98 gives a brief summary of each of the Timer related pins.

Table 98: Pin summary

Pin name	Pin direction	Pin Description
CAP0.2..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.
MAT0.0 MAT1.0	Output	External Match Output 0/1- When match register 0/1 (MR0/1) 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.
MAT0.1 MAT1.1	Output	External Match Output 1- See the MAT0/MAT1 description above.
MAT0.2 MAT1.2	Output	External Match Output 2- See the MAT0/MAT1 description above.
MAT1.3	Output	External Match Output 3- See the MAT1 description above.

REGISTER DESCRIPTION

Each Timer contains the registers shown in Table 99. More detailed descriptions follow.

Table 99: Timer Register Map

Timer 0 Address	Timer 1 Address	Name	Description	Access	Reset Value
0xE0004000	0xE0008000	IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight (Timer 1) or seven (Timer 0) possible interrupt sources are pending.	R/W	0
0xE0004004	0xE0008004	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
0xE0004008	0xE0008008	TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0
0xE000400C	0xE000800C	PR	Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0
0xE0004010	0xE0008010	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
0xE0004014	0xE0008014	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
0xE0004018	0xE0008018	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
0xE000401C	0xE000801C	MR1	Match Register 1. See MR0 description.	R/W	0
0xE0004020	0xE0008020	MR2	Match Register 2. See MR0 description.	R/W	0
0xE0004024	0xE0008024	MR3	Match Register 3. See MR0 description.	R/W	0
0xE0004028	0xE0008028	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
0xE000402C	0xE000802C	CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the capture[0] signal.	RO	0
0xE0004030	0xE0008030	CR1	Capture Register 1. See CR0 description.	RO	0
0xE0004034	0xE0008034	CR2	Capture Register 2. See CR0 description.	RO	0
0xE0004038	0xE0008038	CR3	Capture Register 3. See CR0 description. Not usable on Timer 0.	RO	0
0xE000403C	0xE000803C	EMR	External Match Register. The EMR controls the external match pins MATn.	R/W	0

Interrupt Register (IR - Timer 0: 0xE0004000; Timer 1: 0xE0008000)

The Interrupt Register consists of four bits for the match interrupts and four bits (three for Timer 0) 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 100: Interrupt Register (IR - Timer 0: 0xE0004000; Timer 1: 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 - Timer 0: 0xE0004004; Timer 1: 0xE0008004)

The Timer Control Register (TCR) is used to control the operation of the Timer Counter.

Table 101: Timer Control Register (TCR - Timer 0: 0xE0004004; Timer 1: 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

Timer Counter (TC - Timer 0: 0xE0004008; Timer 1: 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 - Timer 0: 0xE000400C; Timer 1: 0xE000800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Prescale Counter Register (PC - Timer 0: 0xE0004010; Timer 1: 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 - Timer 0: 0xE0004014; Timer 1: 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 102.

Table 102: Match Control Register (MCR - Timer 0: 0xE0004014; Timer 1: 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 - Timer 0: 0xE0004028; Timer 1: 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.

Table 103: Capture Control Register (CCR - Timer 0: 0xE0004028; Timer 1: 0xE0008028)

CCR	Function	Description	Reset Value
0	Capture on capture[0] rising edge	When one, a sequence of 0 then 1 on capture[0] will cause CR0 to be loaded with the contents of the TC. When zero this feature is disabled.	0
1	Capture on capture[0] falling edge	When one, a sequence of 1 then 0 on capture[0] will cause CR0 to be loaded with the contents of TC. When zero this feature is disabled.	0
2	Interrupt on capture[0] event	When one, a CR0 load due to a capture[0] event will generate an interrupt. When zero this feature is disabled.	0
3	Capture on capture[1] rising edge	When one, a sequence of 0 then 1 on capture[1] will cause CR1 to be loaded with the contents of the TC. When zero this feature is disabled.	0
4	Capture on capture[1] falling edge	When one, a sequence of 1 then 0 on capture[1] will cause CR1 to be loaded with the contents of TC. When zero this feature is disabled.	0
5	Interrupt on capture[1] event	When one, a CR1 load due to a capture[1] event will generate an interrupt. When zero this feature is disabled.	0
6	Capture on capture[2] rising edge	When one, a sequence of 0 then 1 on capture[2] will cause CR2 to be loaded with the contents of the TC. When zero this feature is disabled.	0
7	Capture on capture[2] falling edge	When one, a sequence of 1 then 0 on capture[2] will cause CR2 to be loaded with the contents of TC. When zero this feature is disabled.	0
8	Interrupt on capture[2] event	When one, a CR2 load due to a capture[2] event will generate an interrupt. When zero this feature is disabled.	0
9	Capture on capture[3] rising edge	(Timer 1 only.) When one, a sequence of 0 then 1 on capture[3] will cause CR3 to be loaded with the contents of TC. When zero this feature is disabled.	0
10	Capture on capture[3] falling edge	(Timer 1 only.) When one, a sequence of 1 then 0 on capture[3] will cause CR3 to be loaded with the contents of TC. When zero this feature is disabled.	0
11	Interrupt on capture[3] event	(Timer 1 only.) When one, a CR3 load due to a capture[3] event will generate an interrupt. When zero this feature is disabled.	0

External Match Register (EMR - Timer 0: 0xE000403C; Timer 1: 0xE000803C)

The External Match Register provides both control and status of the external match pins M(0-3).

Table 104: External Match Register (EMR - Timer 0: 0xE000403C; Timer 1: 0xE000803C)

EMR	Function	Description	Reset Value
0	External Match 0	This bit reflects the state of output MAT0/1, 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, 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 MAT0/1, 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
3	External Match 3	This bit reflects the state of output MAT0/1. In the case of Timer 0, this output cannot be connected to a device 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 105 shows the encoding of these bits.	0
7:6	External Match Control 1	Determines the functionality of External Match 1. Table 105 shows the encoding of these bits.	0
9:8	External Match Control 2	Determines the functionality of External Match 2. Table 105 shows the encoding of these bits.	0
11:10	External Match Control 3	Determines the functionality of External Match 3. Table 105 shows the encoding of these bits.	0

Table 105: 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 25 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 26 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.

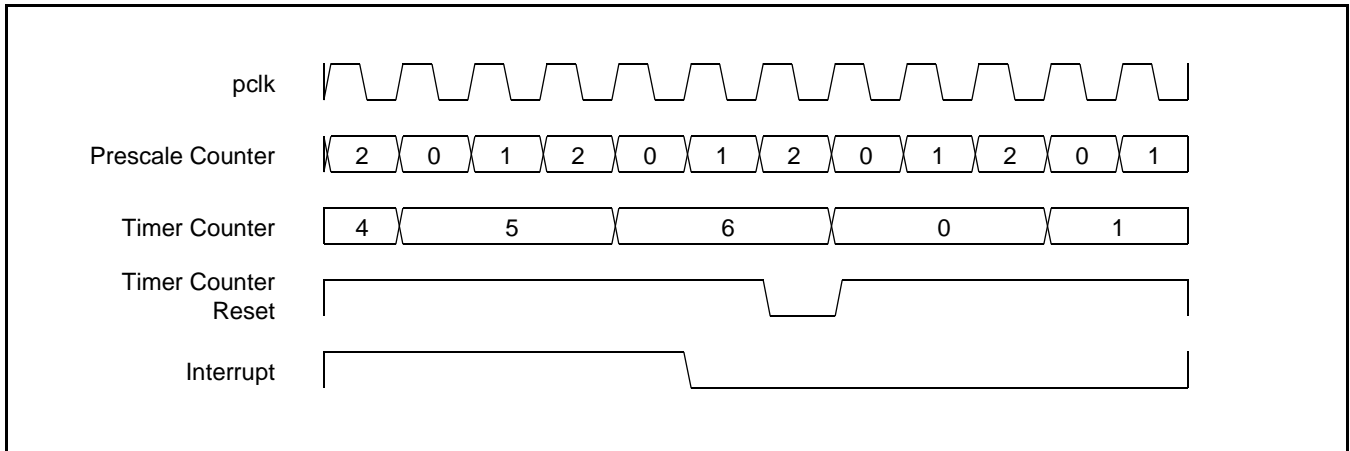


Figure 25: A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

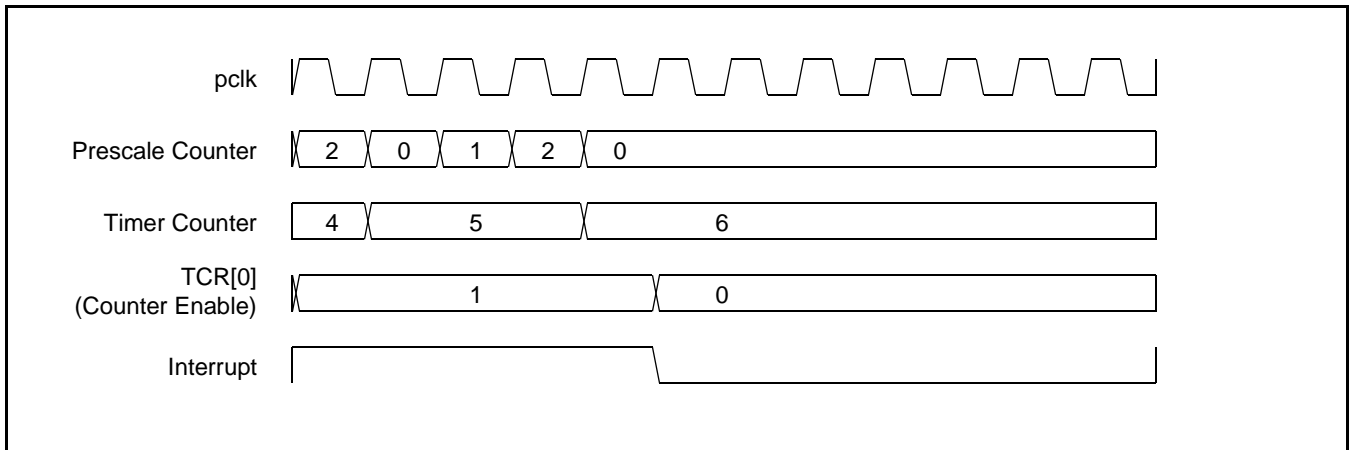
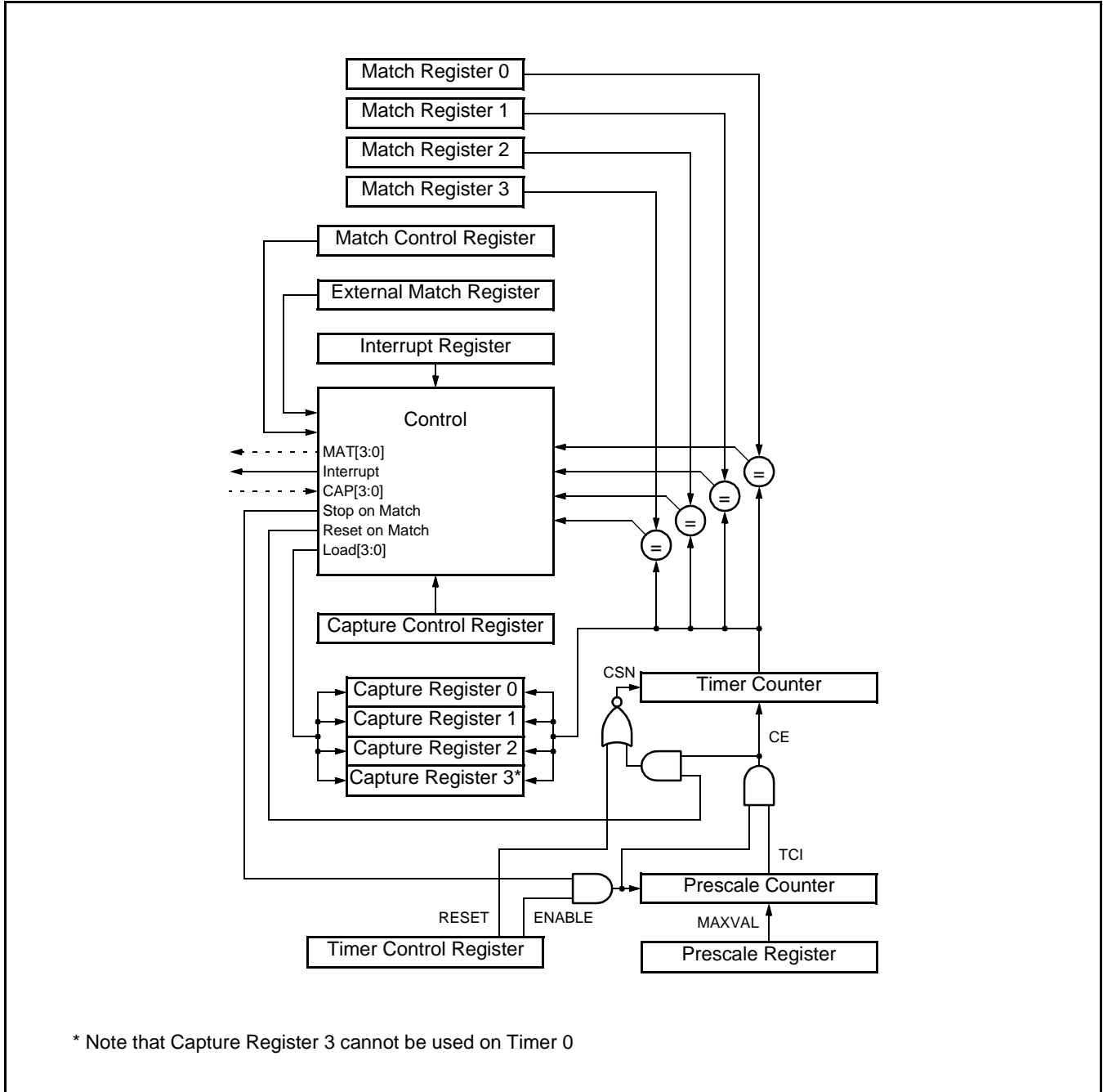


Figure 26: A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled.

ARCHITECTURE

The block diagram for Timer 0 and Timer 1 is shown in Figure 27.



* Note that Capture Register 3 cannot be used on Timer 0

Figure 27: Timer block diagram

12. PULSE WIDTH MODULATOR (PWM)

LPC2106/2105/2104 Pulse Width Modulator is based on standard Timer 0/1 described in 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 LPC2106/2105/2104. The Timer is designed to count cycles of the peripheral clock (pclk) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. It also includes four capture inputs to save the timer value when an input signal transitions, and optionally generate an interrupt when those events occur. The PWM function is in addition to these features, and is based on match register events.

The ability to separately control rising and falling edge locations allows the PWM to be used for more applications. For instance, multi-phase motor control typically requires three non-overlapping PWM outputs with individual control of all three pulse widths and positions.

Two match registers can be used to provide a single edge controlled PWM output. One match register (MR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position. Additional single edge controlled PWM outputs require only one match register each, since the repetition rate is the same for all PWM outputs. Multiple single edge controlled PWM outputs will all have a rising edge at the beginning of each PWM cycle, when an MR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the MR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions. Additional double edge controlled PWM outputs require only two match registers each, since the repetition rate is the same for all PWM outputs.

With double edge controlled PWM outputs, specific match registers control the rising and falling edge of the output. This allows both positive going PWM pulses (when the rising edge occurs prior to the falling edge), and negative going PWM pulses (when the falling edge occurs prior to the rising edge).

Figure 28 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.

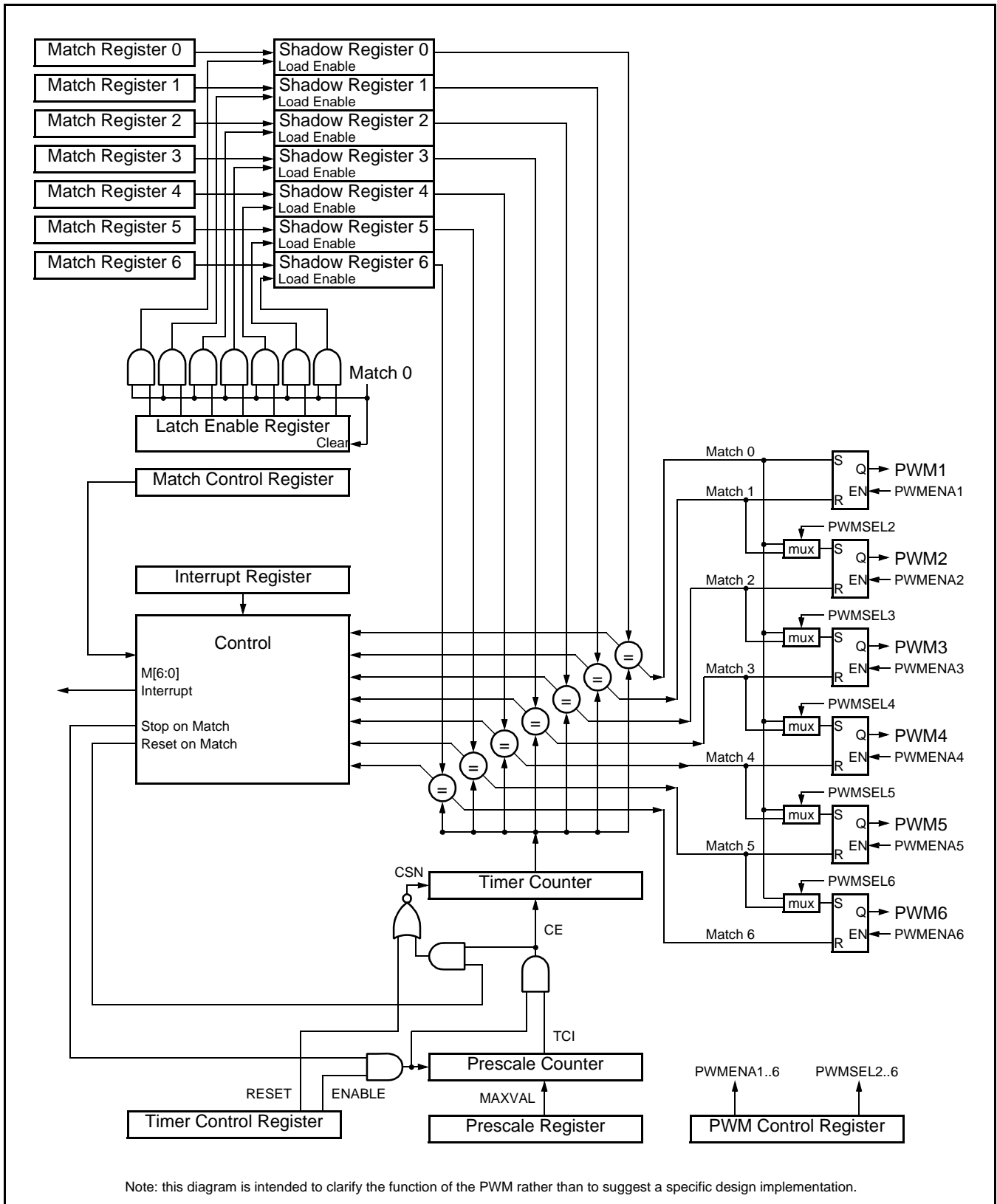


Figure 28: PWM block diagram

A sample of how PWM values relate to waveform outputs is shown in Figure 29. PWM output logic is shown in Figure 28 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 106. 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.

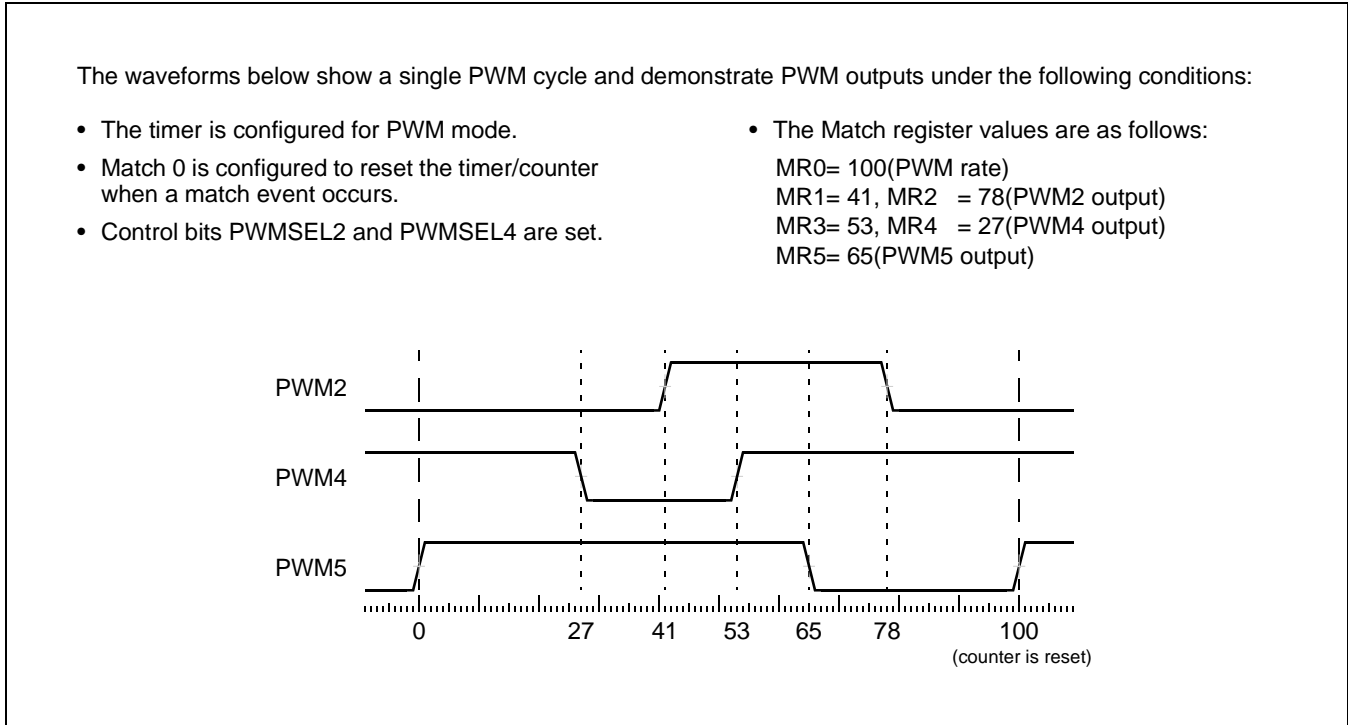


Figure 29: Sample PWM waveforms

Table 106: 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 107 gives a brief summary of each of PWM related pins.

Table 107: 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 108 below.

Table 108: Pulse Width Modulator Register Map

Address	Name	Description	Access	Reset Value
0xE0014000	IR	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
0xE0014004	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
0xE0014008	TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0
0xE001400C	PR	Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0
0xE0014010	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
0xE0014014	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
0xE0014018	MR0	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
0xE001401C	MR1	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
0xE0014020	MR2	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
0xE0014024	MR3	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
0xE0014040	MR4	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

Table 108: Pulse Width Modulator Register Map

Address	Name	Description	Access	Reset Value
0xE0014044	MR5	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
0xE0014048	MR6	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
0xE001404C	PCR	PWM Control Register. Enables PWM outputs and selects PWM channel types as either single edge or double edge controlled.	R/W	0
0xE0014050	LER	Latch Enable Register. Enables use of new PWM match values.	R/W	0

Interrupt Register (IR - 0xE0014000)

The Interrupt Register consists of eleven bits (Table 109), seven for the match interrupts and four reserved for the future use. 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 109: Interrupt Register (IR - 0xE0014000)

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	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	MR4 Interrupt	Interrupt flag for match channel 4.	0
9	MR5 Interrupt	Interrupt flag for match channel 5.	0
10	MR6 Interrupt	Interrupt flag for match channel 6.	0

Timer Control Register (TCR - 0xE0014004)

The Timer Control Register (TCR) is used to control the operation of the Timer Counter. The function of each of the bits is shown in Table 110.

Table 110: Timer Control Register (TCR - 0xE0014004)

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
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 LER has been set, followed by the occurrence of a Match 0 event. Note that the Match register that determines the PWM rate (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

Timer Counter (TC - Timer 0: 0xE0004008; Timer 1: 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 - Timer 0: 0xE000400C; Timer 1: 0xE000800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Prescale Counter Register (PC - Timer 0: 0xE0004010; Timer 1: 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 - MR6)

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 - 0xE0014014)

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 111.

Table 111: Match Control Register (MCR - 0xE0014014)

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
12	Interrupt on MR4	When one, an interrupt is generated when MR4 matches the value in the TC. When zero this interrupt is disabled.	0
13	Reset on MR4	When one, the TC will be reset if MR4 matches it. When zero this feature is disabled.	0
14	Stop on MR4	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR4 matches the TC. When zero this feature is disabled.	0
15	Interrupt on MR5	When one, an interrupt is generated when MR5 matches the value in the TC. When zero this interrupt is disabled.	0
16	Reset on MR5	When one, the TC will be reset if MR5 matches it. When zero this feature is disabled.	0
17	Stop on MR5	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR5 matches the TC. When zero this feature is disabled.	0
18	Interrupt on MR6	When one, an interrupt is generated when MR6 matches the value in the TC. When zero this interrupt is disabled.	0
19	Reset on MR6	When one, the TC will be reset if MR6 matches it. When zero this feature is disabled.	0
20	Stop on MR6	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR6 matches the TC. When zero this feature is disabled.	0

PWM Control Register (PCR - 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 112.

Table 112: PWM Control Register (PCR - 0xE001404C)

PCR	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

Latch Enable Register (LER - 0xE0014050)

The Latch Enable Register is used to control the update of the Match registers when they are used for PWM generation. When software writes to the location of a Match register while the Timer is in PWM mode, the value is held in a shadow register. When a 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 LER is set and a Match 0 event occurs, any value written to the 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 Match1 register.
- Write a new value to the Match2 register.
- Write to the LER, setting bits 1 and 2 at the same time.
- The altered values will become effective at the next reset of the timer (when a Match 0 event occurs).

The order of writing the two Match registers is not important, since neither value will be used until after the write to LER. This insures that both values go into effect at the same time, if that is required. A single value may be altered in the same way if needed.

The function of each of the bits in the LER is shown in Table 113.

Table 113: Latch Enable Register (LER - 0xE0014050)

LER	Function	Description	Reset Value
0	Enable Match 0 Latch	Writing a one to this bit allows the last value written to the Match 0 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
1	Enable Match 1 Latch	Writing a one to this bit allows the last value written to the Match 1 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
2	Enable Match 2 Latch	Writing a one to this bit allows the last value written to the Match 2 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
3	Enable Match 3 Latch	Writing a one to this bit allows the last value written to the Match 3 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
4	Enable Match 4 Latch	Writing a one to this bit allows the last value written to the Match 4 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
5	Enable Match 5 Latch	Writing a one to this bit allows the last value written to the Match 5 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
6	Enable Match 6 Latch	Writing a one to this bit allows the last value written to the Match 6 register to be become effective when the timer is next reset by a Match event. See the description of the Match Control Register (MCR).	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13. 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.
- Programmable Reference Clock Divider allows adjustment of the RTC to match various crystal frequencies.

DESCRIPTION

The Real Time Clock (RTC) is designed to provide a set of counters to measure time during system power on and off operation. The RTC has been designed to use little power, making it suitable for battery powered systems where the CPU is not running continuously (Idle mode).

ARCHITECTURE

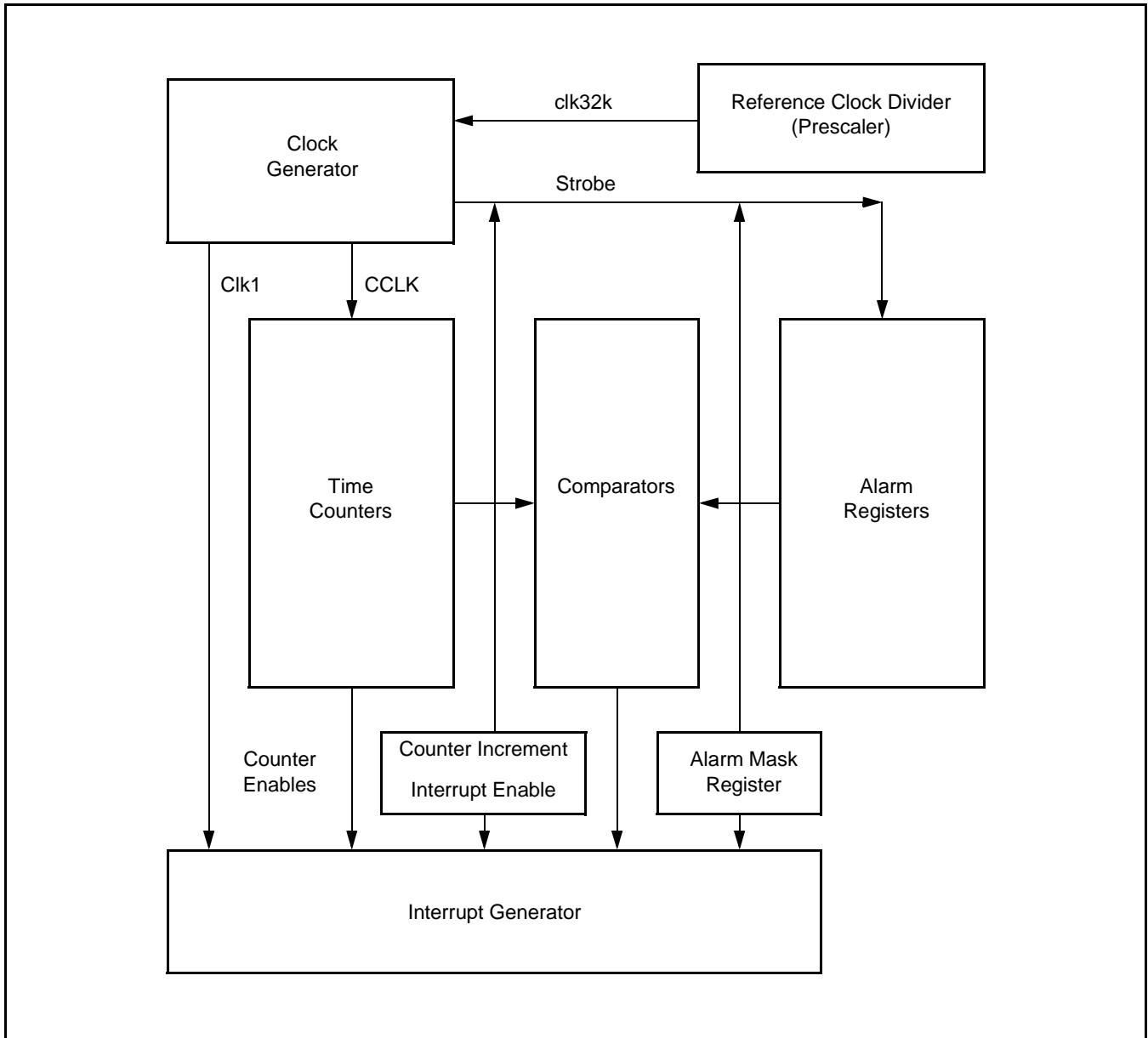


Figure 30: RTC block diagram

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 114. Detailed descriptions of the registers follow.

ARM-based Microcontroller

LPC2106/2105/2104

Table 114: Real Time Clock Register Map

Address	Name	Size	Description	Access	Reset Value
0xE0024000	ILR	2	Interrupt Location Register	R/W	*
0xE0024004	CTC	15	Clock Tick Counter.	RO	*
0xE0024008	CCR	4	Clock Control Register	R/W	*
0xE002400C	CIIR	8	Counter Increment Interrupt Register	R/W	*
0xE0024010	AMR	8	Alarm Mask Register	R/W	*
0xE0024014	CTIME0	(32)	Consolidated Time Register 0	RO	*
0xE0024018	CTIME1	(32)	Consolidated Time Register 1	RO	*
0xE002401C	CTIME2	(32)	Consolidated Time Register 2	RO	*
0xE0024020	SEC	6	Seconds Register	R/W	*
0xE0024024	MIN	6	Minutes Register	R/W	*
0xE0024028	HOUR	5	Hours Register	R/W	*
0xE002402C	DOM	5	Day of Month Register	R/W	*
0xE0024030	DOW	3	Day of Week Register	R/W	*
0xE0024034	DOY	9	Day of Year Register	R/W	*
0xE0024038	MONTH	4	Months Register	R/W	*
0xE002403C	YEAR	12	Years Register	R/W	*
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 Year	R/W	*
0xE0024080	PREINT	13	Prescale value, integer portion	R/W	0
0xE0024084	PREFRAC	15	Prescale value, fractional portion	R/W	0

* 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.

MISCELLANEOUS REGISTER GROUP

Table 115 summarizes the registers located from 0 to 7 of A[6:2]. More detailed descriptions follow.

Table 115: 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 116). 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 116: Interrupt Location Register Bits (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) [called Prescale in the HDL template]

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 117: Clock Tick Counter Bits (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) [called Clock Register in the HDL template]

The clock register is a 4-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 118.

Table 118: Clock Control Register Bits (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.

Counter Increment Interrupt

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 119: Counter Increment Interrupt Register Bits (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

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 120 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 120: Alarm Mask Register Bits (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 121, 122, and 123. 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 121: Consolidated Time Register 0 Bits (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 122: Consolidated Time Register 1 Bits (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 123: Consolidated Time Register 2 Bits (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 124 and 125. These counters can be read or written at the locations shown in Table 125.

Table 124: Time Counter Relationships and Values

Counter	Size	Enabled by	Min value	Maximum value
Second	6	Clk1 (see Figure 30)	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 125: 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 126. The values in these registers are compared with the time counters. If all the unmasked (See "Alarm Mask" on page 139.) 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 126: 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

Since the RTC operates from the VPB clock (pclk), any interruption of that 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.

No provision is made in the LPC2106/2105/2104 to retain RTC status upon power loss, or to maintain time incrementation if the clock source is lost, interrupted, or altered. Loss of chip power will result in complete loss of all RTC register contents. Entry to Power Down mode will cause a lapse in the time update. Altering the RTC timebase during system operation (by reconfiguring the PLL, the VPB timer, or the RTC prescaler) will result in some form of accumulated time error.

REFERENCE CLOCK DIVIDER (PRESCALER)

The reference clock divider (hereafter referred to as the Prescaler) allows generation of a 32.768 kHz reference clock from any peripheral clock frequency greater than or equal to 65.536 kHz (2×32.768 kHz). This permits the RTC to always run at the proper rate regardless of the peripheral clock rate. Basically, the Prescaler divides the peripheral clock (pclk) by a value which contains both an integer portion and a fractional portion. The result is not a continuous output at a constant frequency, some clock periods will be one pclk longer than others. However, the overall result can always be 32,768 counts per second.

The reference clock divider consists of a 13-bit integer counter and a 15-bit fractional counter. The reasons for these counter sizes are as follows:

1. For frequencies that are expected to be supported by the LPC2106/2105/2104, a 13-bit integer counter is required. This can be calculated as $160 \text{ MHz} / 32,768 = 4881$ with a remainder of 26,624. Thirteen bits are needed to hold the value 4881, but actually supports frequencies up to 268.4 MHz ($32,768 \times 8192$).
2. The remainder value could be as large as 32,767, which requires 15 bits.

Table 127: 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:

$\text{PREINT} = \text{int}(\text{pclk} / 32768) - 1$. The value of PREINT must be greater than or equal to 1.

Table 128: 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:

$\text{PREFRAC} = \text{pclk} - ((\text{PREINT} + 1) \times 32768)$.

Table 129: 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:

$$PREINT = \text{int}(\text{pclk} / 32768) - 1 = 1 \quad \text{and} \quad PREFRAC = \text{pclk} - ((PREINT + 1) \times 32768) = 1$$

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,

$$PREINT = \text{int}(\text{pclk} / 32768) - 1 = 304 \quad \text{and} \quad PREFRAC = \text{pclk} - ((PREINT + 1) \times 32768) = 5,760.$$

In this case, 5,760 of the prescaler output clocks will be 306 (305+1) pclks long, the rest will be 305 pclks long.

In a similar manner, any pclk rate greater than 65.536 kHz (as long as it is an even number of cycles per second) may be turned into a 32 kHz reference clock for the RTC. The only caveat is that if PREFRAC does not contain a zero, then not all of the 32,768 per second clocks are of the same length. Some of the clocks are one pclk longer than others. While the longer pulses are distributed as evenly as possible among the remaining pulses, this "jitter" could possibly be of concern in an application that wishes to observe the contents of the Clock Tick Counter (CTC) directly.

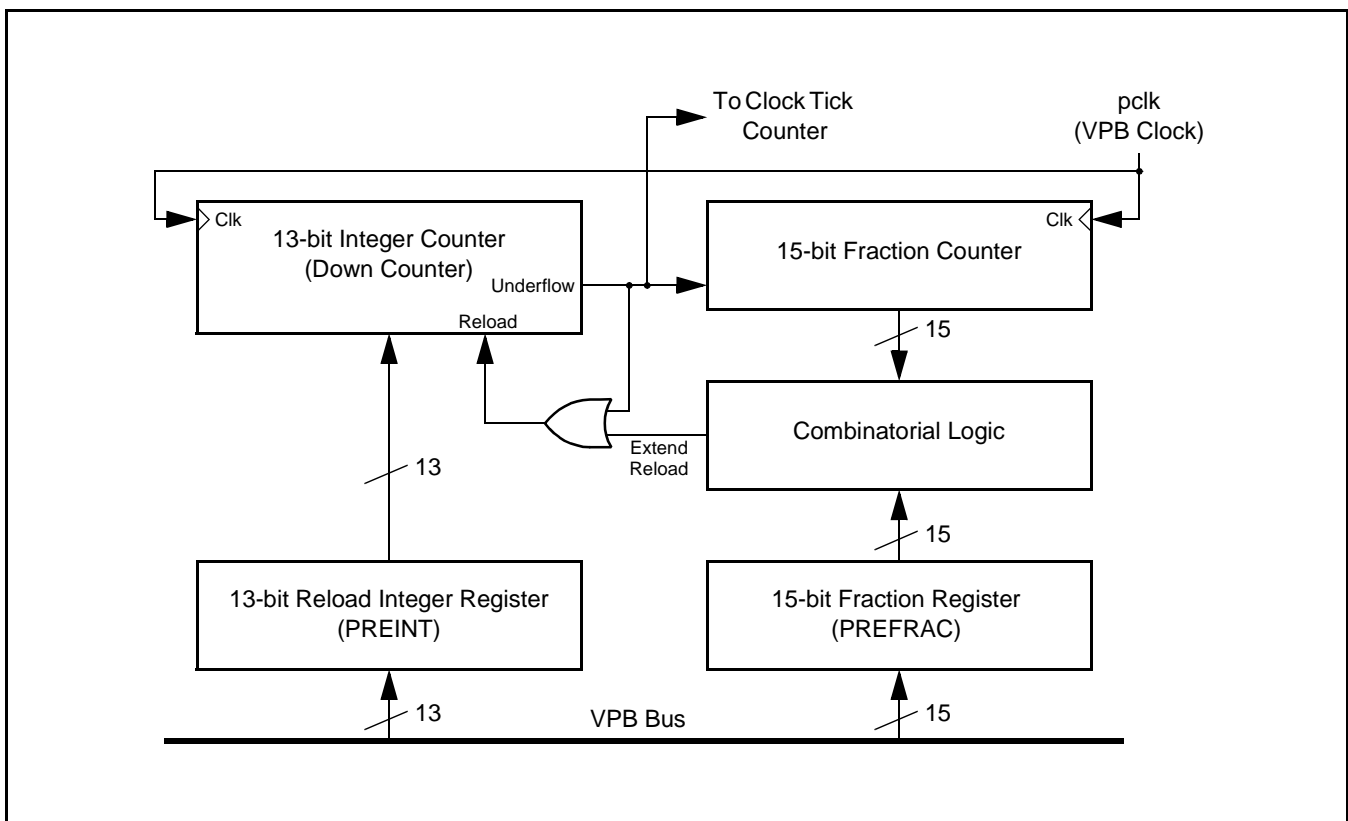


Figure 31: RTC Prescaler block diagram

14. 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 ($t_{\text{pclk}} \times 256 \times 4$) to ($t_{\text{pclk}} \times 2^{32} \times 4$) in multiples of $t_{\text{pclk}} \times 4$

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.

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 ($t_{\text{pclk}} \times 256 \times 4$) and the maximum Watchdog interval is ($t_{\text{pclk}} \times 2^{32} \times 4$) in multiples of ($t_{\text{pclk}} \times 4$). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup mode in WDMOD register.
- Start the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register.
- Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.

When the Watchdog counter underflows, the program counter will start from 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 130 below.

Table 130: Watchdog Register Map

Address	Name	Description	Access	Reset Value
0xE0000000	WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	Read/Set	0
0xE0000004	WDTC	Watchdog timer constant register. This register determines the time-out value.	Read/Write	0xFF
0xE0000008	WDFEED	Watchdog feed sequence register. Writing AAh followed by 55h to this register reloads the Watchdog timer to its preset value.	Write Only	NA
0xE000000C	WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	Read Only	0xFF

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 131: 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 $t_{pclk} \times 256 \times 4$.

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 WDFEED 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 132: 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 133: 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 32.

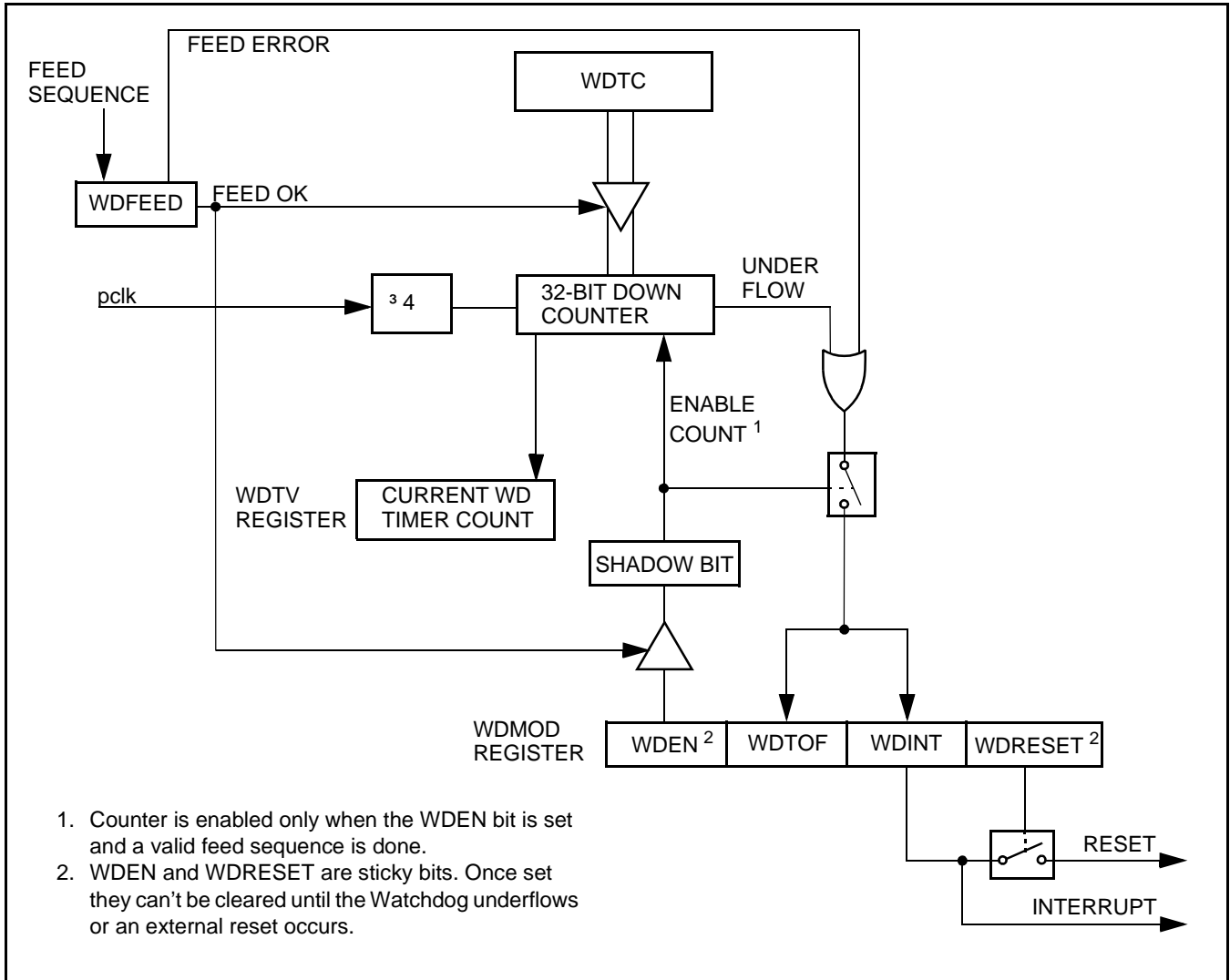


Figure 32: Watchdog Block Diagram

15. 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 134: 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.
DBGSEL	Input	Debug Select. When low, the P0.17 - P0.21 pins are configured for alternate functions via the Pin Connect Block. When high, debug mode is entered.
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)". Also used during entry into debug mode to select primary or secondary JTAG pins on the 48-pin package.

REGISTER DESCRIPTION

The EmbeddedICE logic contains 16 registers as shown in Table 135. 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 135: EmbeddedICE Logic Registers

Address	Width	Name	Description
00000	6	Debug Control	Force debug state, disable interrupts
00001	5	Debug status	Status of debug
00100	32	Debug Comms Control Register	Debug communication control register
00101	32	Debug Comms Data Register	Debug communication data register
01000	32	Watchpoint 0 Address Value	Holds watchpoint 0 address value
01001	32	Watchpoint 0 Address Mask	Holds watchpoint 0 address mask
01010	32	Watchpoint 0 Data Value	Holds watchpoint 0 data value
01011	32	Watchpoint 0 Data Mask	Holds watchpoint 0 data Mask
01100	9	Watchpoint 0 Control Value	Holds watchpoint 0 control value
01101	8	Watchpoint 0 Control Mask	Holds watchpoint 0 control mask
10000	32	Watchpoint 1 Address Value	Holds watchpoint 1 address value
10001	32	Watchpoint 1 Address Mask	Holds watchpoint 1 address mask
10010	32	Watchpoint 1 Data Value	Holds watchpoint 1 data value
10011	32	Watchpoint 1 Data Mask	Holds watchpoint 1 data Mask
10100	9	Watchpoint 1 Control Value	Holds watchpoint 1 control value
10101	8	Watchpoint 1 Control Mask	Holds watchpoint 1 control mask

BLOCK DIAGRAM

The block diagram of the debug environment is shown below in Figure 33.

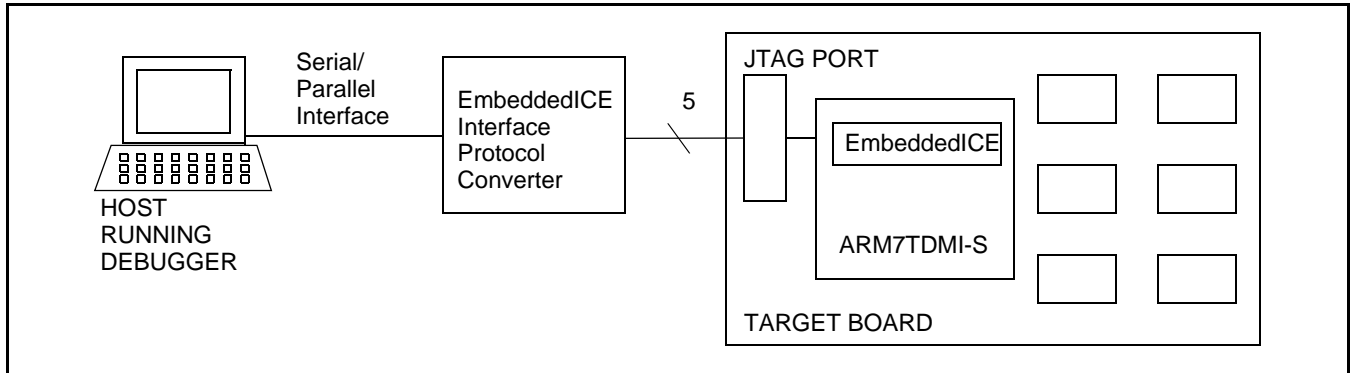


Figure 33: EmbeddedICE Debug Environment Block Diagram

16. 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 136: 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 136: 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 137: 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.

REGISTER DESCRIPTION

The ETM contains 29 registers as shown in Table 138. 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 138: ETM Registers

Register encoding	Name	Description	Access
000 0000	ETM Control	Controls the general operation of the ETM	Read/Write
000 0001	ETM Configuration Code	Allows a debugger to read the number of each type of resource	Read Only
000 0010	Trigger Event	Holds the controlling event	Write Only
000 0011	Memory Map Decode Control	Eight-bit register, used to statically configure the memory map decoder	Write Only
000 0100	ETM Status	Holds the pending overflow status bit	Read Only
000 0101	System Configuration	Holds the configuration information using the SYSOPT bus	Read Only
000 0110	Trace Enable Control 3	Holds the trace on/off addresses	Write Only
000 0111	Trace Enable Control 2	Holds the address of the comparison	Write Only
000 1000	Trace Enable Event	Holds the enabling event	Write Only
000 1001	Trace Enable Control 1	Holds the include and exclude regions	Write Only
000 1010	FIFOFULL Region	Holds the include and exclude regions	Write Only
000 1011	FIFOFULL Level	Holds the level below which the FIFO is considered full	Write Only
000 1100	ViewData event	Holds the enabling event	Write Only
000 1101	ViewData Control 1	Holds the include/exclude regions	Write Only
000 1110	ViewData Control 2	Holds the include/exclude regions	Write Only
000 1111	ViewData Control 3	Holds the include/exclude regions	Write Only
001 xxxx	Address Comparator 1 to 16	Holds the address of the comparison	Write Only
010 xxxx	Address Access Type 1 to 16	Holds the type of access and the size	Write Only
000 xxxx	reserved	-	-
100 xxxx	reserved	-	-
101 00xx	Initial Counter Value 1 to 4	Holds the initial value of the counter	Write Only
101 01xx	Counter Enable 1 to 4	Holds the counter clock enable control and event	Write Only
101 10xx	Counter reload 1 to 4	Holds the counter reload event	Write Only
101 11xx	Counter Value 1 to 4	Holds the current counter value	Read Only
110 00xx	Sequencer State and Control	Holds the next state triggering events.	-
110 10xx	External Output 1 to 4	Holds the controlling events for each output	Write Only
110 11xx	Reserved	-	-
111 0xxx	Reserved	-	-
111 1xxx	Reserved	-	-

BLOCK DIAGRAM

The block diagram of the ETM debug environment is shown below in Figure 34.

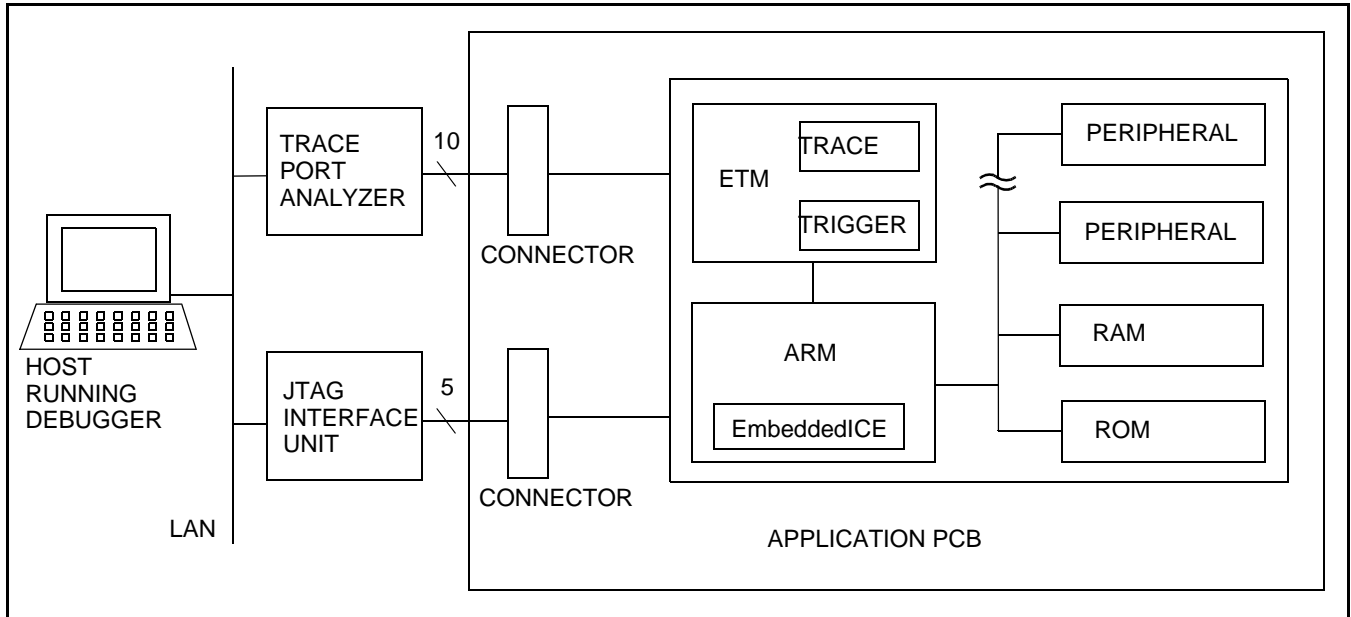


Figure 34: ETM Debug Environment Block Diagram

17. 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 flash 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 Flash 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 35, 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 Flash 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)*.

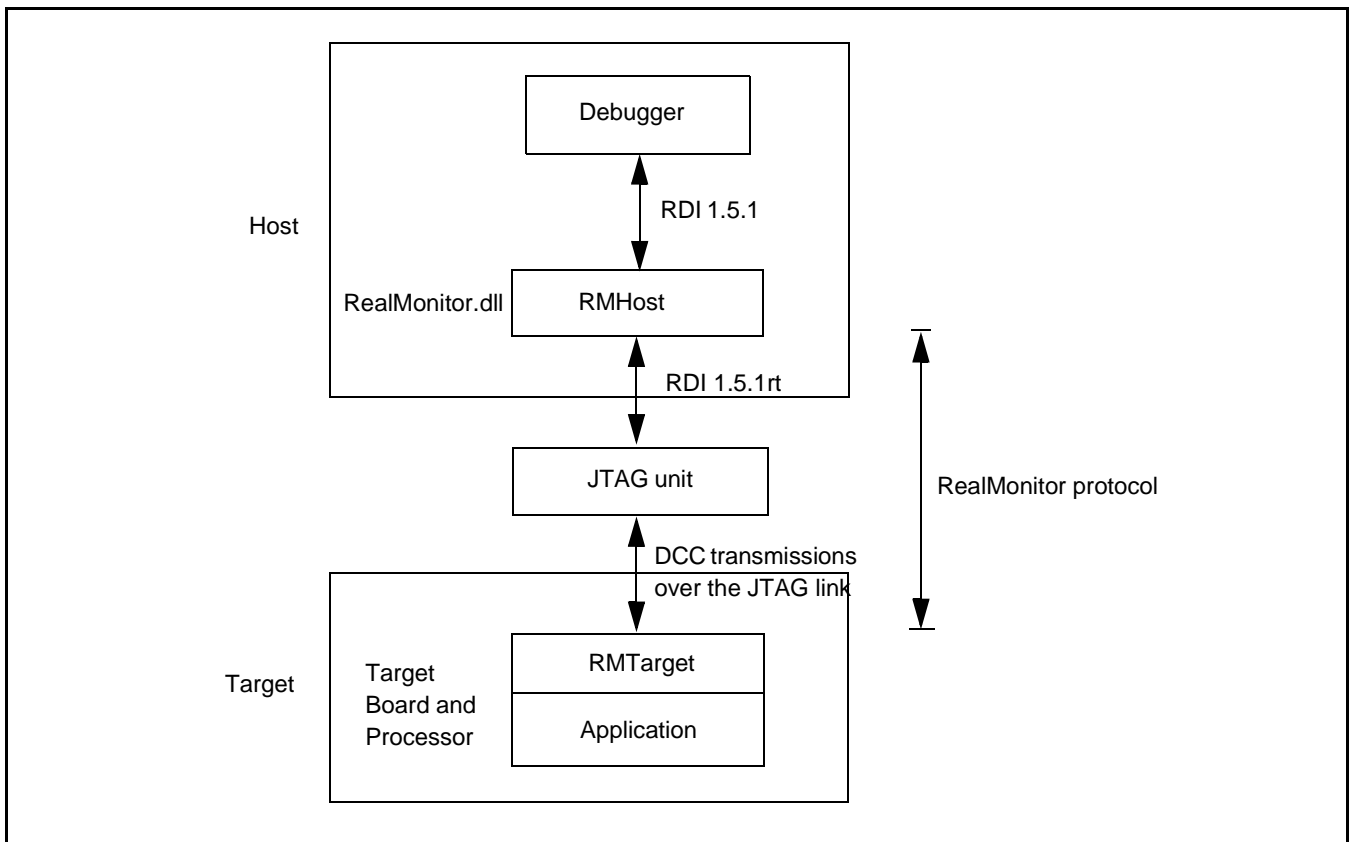


Figure 35: RealMonitor components

How RealMonitor works

In general terms, the RealMonitor operates as a state machine, as shown in Figure 36. 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.

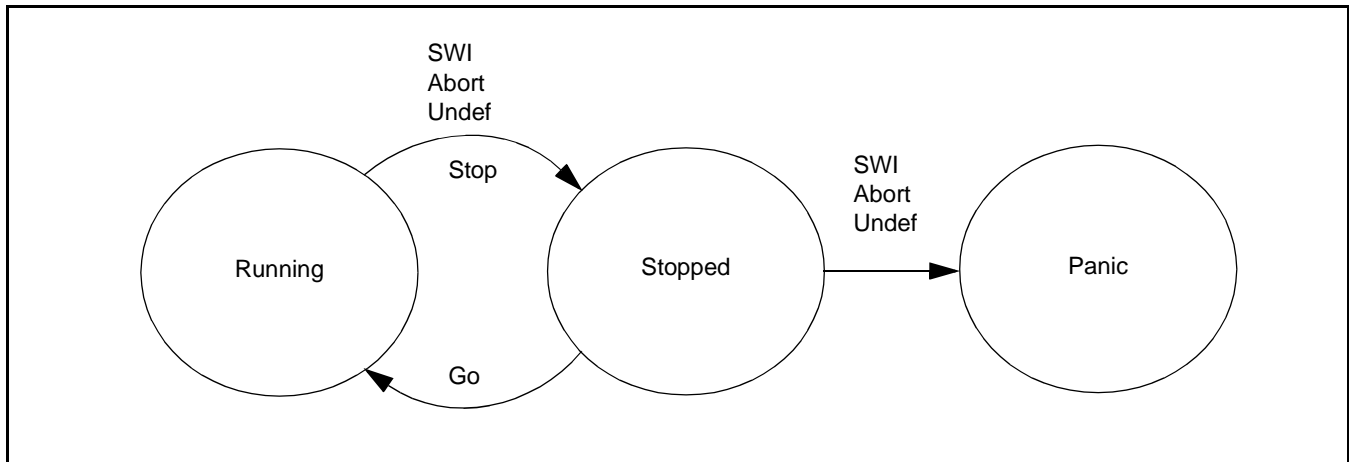


Figure 36: RealMonitor as a state machine

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 35.

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:

ARM-based Microcontroller**LPC2106/2105/2104**

- 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 139: 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 37 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.

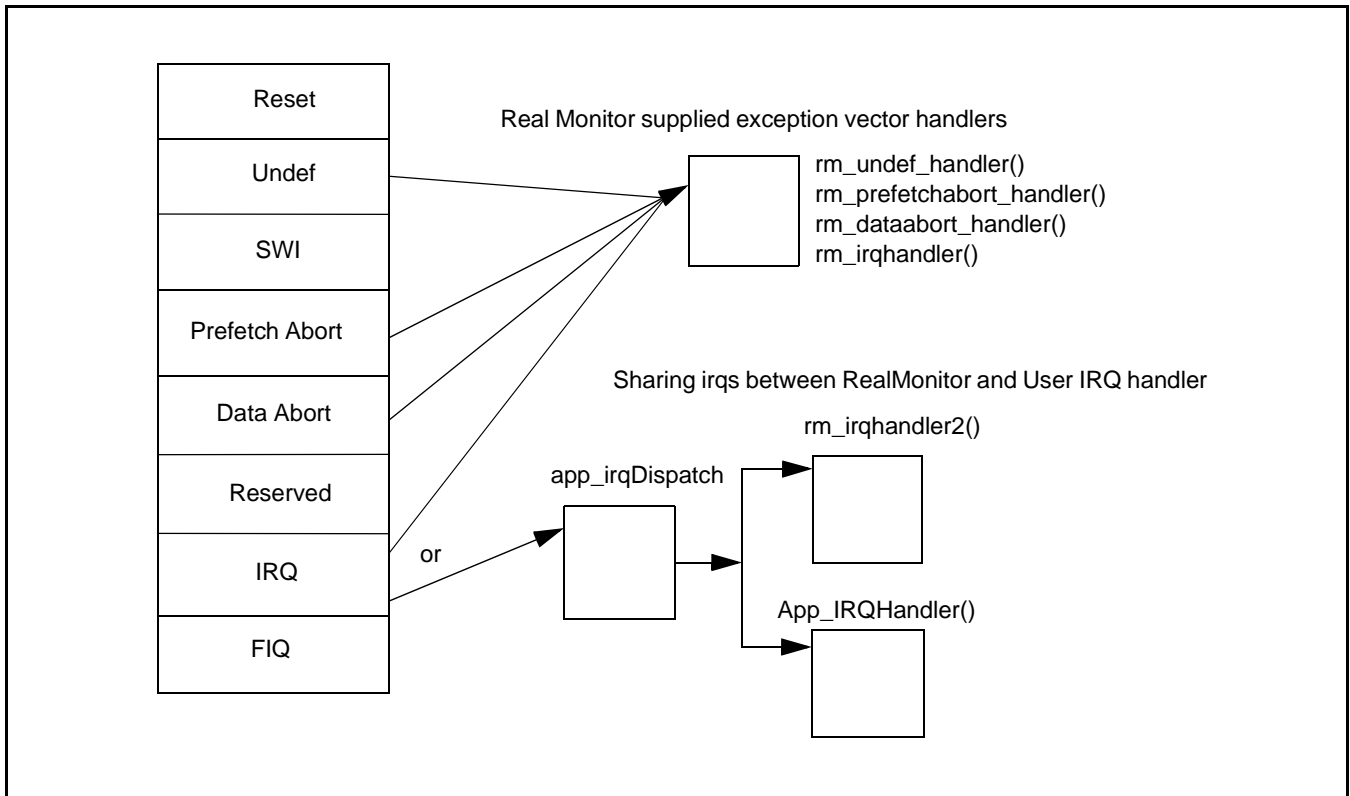


Figure 37: Exception Handlers

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

```

ARM-based Microcontroller

LPC2106/2105/2104

```

; 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

```

ARM-based Microcontroller

LPC2106/2105/2104

```
;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 μ HAL (ARM HW abstraction API).