UM10114

LPC21xx and LPC22xx User manual

Rev. 4 — 2 May 2012 User manual

Document information

Info	Content
Keywords	LPC2109/00, LPC2109/01, LPC2119, LPC2119/01, LPC2129, LPC2129/01, LPC2114, LPC2114/01, LPC2124, LPC2124/01, LPC2194, LPC2194/01, LPC2210, LPC2220, LPC2210/01, LPC2212, LPC2212/01, LPC2214, LPC2214/01, LPC2290, LPC2290/01, LPC2292, LPC2292/01, LPC2294, LPC2294/01, ARM, ARM7, 32-bit, Microcontroller
Abstract	User manual for LPC2109/19/29/14/24/94 and LPC2210/20/12/14/90/92/94 including /01 parts

Revision history

Rev	Date	Description
4.0	20120502	Modifications: • Device revision register added (see Section 21.9.11). • Max voltage on pin AINx limited to 3.3 V (see Table 292). • Lower limit for DLL = 3 (see Section 10.4.4 and Section 11.4.4). • Table 191 updated. • AFMR register description updated (see Table 278). • Full-CAN registers added for /01 parts (see Section 19.10). • VIC full-CAN interrupt added for /01 parts. • Length of write access updated in Section 4.5. • Bootloader version 1.7 added in Table 304. • Table 300 updated and reset memory map corrected for /01 devices (see Section 21.5.1). • SPI interrupt register bit description corrected (Table 199). • Location of the PCSSP bit in the PCONP register corrected (Table 74).
3.0	20080402	Modifications: • Flash chapter updated with correct boot process flowchart. • The Reinvoke ISP command has been removed from the ISP command description because it is not implemented in the LPC21xx/LPC22xx. • Description of CRP levels has been corrected, and CRP description for different bootloader code versions has been added. • Numbering of CAN controllers in the global CAN filter look-up table has been corrected for /01 devices. • Part ID’s have been updated for LPC2210/20 parts.
2.0	20080104	Integrated related parts into this manual and made numerous editorial and content updates throughout the document: • The format of this data sheet has been redesigned to comply with the new identity guidelines of NXP Semiconductors. • Legal texts have been adapted to the new company name where appropriate. • Parts LPC2109, LPC2119, LPC2129, LPC2114, LPC2124, LPC2194, LPC2212, LPC2214, LPC2290, LPC2292, LPC2294 and /01 parts added. • PWM mode description updated. • Fractional baud rate generator updated. • CTCR register updated. • ADC pin description updated. • SPI clock conditions updated. • JTAG pin description updated. • Startup sequence diagram added. • SPI master mode: SPI SSEL line conditioning for LPC2210/20 added in SPI pin description table.

Revision history *...continued*

Rev	Date	Description
1.0	20051012	Moved the UM document into the new structured FrameMaker template. Many changes were made to the format throughout the document. Here are the most important: • UART0 and UART1 description updated (fractional baudrate generator and hardware handshake features added - auto-CTS/RTS) • ADC chapter updated with the dedicated result registers • GPIO chapter updated with the description of the Fast IOs

Contact information

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

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

UM10114

Chapter 1: Introductory information

Rev. 4 — 2 May 2012 User manual

1.1 Introduction

The LPC21xx and LPC22xx are based on a 16/32 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, together with 64/128/256 kilobytes (kB) of embedded high speed flash memory. A 128-bit wide internal memory interface and a unique accelerator architecture enable 32-bit code execution at maximum clock rate. For critical code size applications, the alternative 16-bit Thumb Mode reduces code by more than 30% with minimal performance penalty.

With their compact 64 and 144 pin packages, low power consumption, various 32-bit timers, up to 12 external interrupt pins, and four channel 10-bit ADC and 46 GPIOs (64 pin packages), or 8-channel 10-bit ADC and 112 GPIOs (144 pin package), these microcontrollers are particularly targeted for industrial control, medical systems, access control, and point-of-sale. With a wide range of serial communications interfaces, they are also very well suited for communication gateways, protocol converters, and embedded soft modems as well as many other general-purpose applications.

1.2 How to read this manual

The LPC21xx and LPC22xx user manual covers the following parts and versions:

- LPC2109, LPC2119, LPC2129, /00 and /01 versions
- LPC2114, LPC2124, /00 and /01 versions
- LPC2194 and LPC2194/01
- LPC2210, LPC2210/01, and LPC2220
- LPC2212, LPC2214, /00 and /01 versions
- LPC2290 and LPC2290/01
- LPC2292, LPC2294, /00 and /01 versions

All parts exist in **legacy** versions and **enhanced** versions. Enhanced parts are equipped with enhanced GPIO, SSP, ADC, UART, and timer peripherals. They are also backward compatible to the “legacy” parts containing legacy versions of the same peripherals. Therefore, enhanced parts contain all features of legacy parts as well. See Table 16 for an overview.

To denote different versions the following suffixes are used (see Section 1.4 “Ordering options”); no suffix, /00, /01, and /G. All /01 versions and the LPC2220 (no suffix) contain enhanced features.

Table 1. LPC21xx and LPC22xx legacy/enhanced parts overview

Legacy parts	Enhanced parts
LPC2109/00 LPC2119, LPC2119/00 LPC2129, LPC2129/00	LPC2109/01 LPC2119/01 LPC2129/01
LPC2114, LPC2114/00 LPC2124, LPC2124/00	LPC2114/01 LPC2124/01
LPC2194, LPC2194/00	LPC2194/01
LPC2210	LPC2210/01 LPC2220, LPC2220/G
LPC2212, LPC2212/00 LPC2214, LPC2214/00	LPC2212/01 LPC2214/01
LPC2290	LPC2290/01
LPC2292, LPC2292/00 LPC2294, LPC2294/00	LPC2292/01 LPC2294/01

This user manual describes enhanced features together with legacy features for all LPC21xx and LPC22xx parts. Part specific and legacy/enhanced specific pinning, registers, and configurations are listed in a table at the beginning of each chapter (see for example Table 52 “LPC21xx/22xx part-specific register bits”). Use this table to determine which parts of the user manual apply.

1.3 Features

1.3.1 Legacy features common to all LPC21xx and LPC22xx parts

- 16-bit/32-bit ARM7TDMI-S microcontroller in a 64 or 144 pin package.
- 8/16/64 kB of on-chip static RAM and 64/128/256 kB of on-chip flash program memory (except for flashless LPC2210/20/90). 128-bit wide interface/accelerator enables high-speed 60 MHz operation.
- In-System/In-Application Programming (ISP/IAP) via on-chip boot loader software. Single flash sector or full chip erase in 100 ms and programming of 256 bytes in 1 ms.
- Diversified Code Read Protection (CRP) enables different security levels to be implemented.
- External 8, 16, or 32-bit bus (144 pin package).
- EmbeddedICE RT and Embedded Trace offer real-time debugging with the on-chip RealMonitor software and high speed tracing of instruction execution.
- Up to four interconnected CAN interfaces with advanced acceptance filters.
- 10-bit A/D converter providing four/eight analog inputs with conversion times as low as 2.44 ms per channel and dedicated result registers to minimize interrupt overhead.
- Two 32-bit timers/external event counters with four capture and four compare channels each, PWM unit (6 outputs), Real Time Clock (RTC), and watchdog.
- Multiple serial interfaces including two UARTs (16C550), a fast I^{2}C-bus (400 kbit/s), and two SPI interfaces.
- Vectored interrupt controller with configurable priorities and vector addresses.
- Up to forty-eight 5 V tolerant fast general purpose I/O pins.

- Up to 12 edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip PLL with a possible input frequency of 10 MHz to 25 MHz and a settling time of 100 ms.
- For flashless LPC2210/20/90 only: 60 MHz (LPC2210/90), 72 MHz (LPC2290/01), or 75 MHz (LPC2210/01 and LPC2220) maximum CPU clock available from programmable on-chip Phase-Locked Loop (PLL) with settling time of 100 μs.
- On-chip integrated oscillator operates with an external crystal in the range from 1 MHz to 25 MHz.
- Two power saving modes, Idle mode and Power-down mode.
- Peripheral clock scaling and individual enable/disable of peripheral functions for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt or CAN controllers.
- Dual power supply:
  - CPU operating voltage range of 1.65 V to 1.95 V (1.8 V ± 8.3 %).
  - I/O power supply range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

1.3.2 Enhanced features

- Fast GPIO ports enable port pin toggling up to 3.5 times faster than the original device. They also allow for a port pin to be read at any time regardless of its function.
- Dedicated result registers for ADC reduce interrupt overhead. The ADC pads are 5 V tolerant when configured for digital I/O functions.
- UART0/1 include fractional baud rate generator, auto-bauding capabilities, and handshake flow-control fully implemented in hardware.
- Buffered SSP serial controller supporting SPI, 4-wire SSI, and Microwire formats.
- SPI programmable data length and master mode enhancement.
- General purpose timers can operate as external event counters.

1.4 Ordering options

1.4.1 LPC2109/2119/2129

Table 2. LPC2109/2119/2129 Ordering information

Type number	Package		
	Name	Description	Version
LPC2109FBD64/00	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2109FBD64/01	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2119FBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2119FBD64/00	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2119FBD64/01	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2

Table 2. LPC2109/2119/2129 Ordering information *…continued*

Type number	Package		
	Name	Description	Version
LPC2129FBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2129FBD64/00	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2129FBD64/01	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2

Table 3. LPC2109/2119/2129 Ordering options

Type number	Flash memory	RAM	CAN	Fast GPIO/ SSP/ Enhanced UART, ADC, Timer	Temperature range
LPC2109FBD64/00	64 kB	8 kB	1 channel	no	–40 °C to +85 °C
LPC2109FBD64/01	64 kB	8 kB	1 channel	yes	–40 °C to +85 °C
LPC2119FBD64	128 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2119FBD64/00	128 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2119FBD64/01	128 kB	16 kB	2 channels	yes	–40 °C to +85 °C
LPC2129FBD64	256 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2129FBD64/00	256 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2129FBD64/01	256 kB	16 kB	2 channels	yes	–40 °C to +85 °C

1.4.2 LPC2114/2124

Table 4. LPC 2114/2124 Ordering information

Type number	Package		
	Name	Description	Version
LPC2114FBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2114FBD64/00	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2114FBD64/01	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2124FBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2124FBD64/00	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2124FBD64/01	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2

Table 5. LPC2114/2124 Ordering options

Type number	Flash memory	RAM	Fast GPIO/SSP/ Enhanced UART, ADC, Timer	Temperature range
LPC2114FBD64	128 kB	16 kB	no	–40 °C to +85 °C
LPC2114FBD64/00	128 kB	16 kB	no	–40 °C to +85 °C
LPC2114FBD64/01	128 kB	16 kB	yes	–40 °C to +85 °C
LPC2124FBD64	256 kB	16 kB	no	–40 °C to +85 °C
LPC2124FBD64/00	256 kB	16 kB	no	–40 °C to +85 °C
LPC2124FBD64/01	256 kB	16 kB	yes	–40 °C to +85 °C

1.4.3 LPC2194

Table 6. LPC2194 Ordering information

Type number	Package		
	Name	Description	Version
LPC2194HBD64	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2194HBD64/00	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
LPC2194HBD64/01	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2

Table 7. LPC2194 Ordering options

Type number	Flash memory	RAM	CAN	Fast GPIO/ SSP/ Enhanced UART, ADC, Timer	Temperature range
LPC2194HBD64	256 kB	16 kB	4 channels	no	–40 °C to +125 °C
LPC2194HBD64/00	256 kB	16 kB	4 channels	no	–40 °C to +125 °C
LPC2194HBD64/01	256 kB	16 kB	4 channels	yes	–40 °C to +125 °C

1.4.4 LPC2210/2220

Table 8. LPC2210/2220 Ordering information

Type number	Package		
	Name	Description	Version
LPC2210FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2210FBD144/01	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1

Table 8. LPC2210/2220 Ordering information *…continued*

Type number	Package		
	Name	Description	Version
LPC2220FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2220FET144	TFBGA144	plastic thin fine-pitch ball grid array package; 144 balls; body 12 × 12 × 0.8 mm	SOT569-1
LPC2220FET144/G	TFBGA144	plastic thin fine-pitch ball grid array package; 144 balls; body 12 × 12 × 0.8 mm	SOT569-1

Table 9. LPC2210/2220 Ordering options

Type number	RAM	Fast GPIO/ SSP/ Enhanced UART, ADC, Timer	Temperature range
LPC2210FBD144	16 kB	no	–40 °C to +85 °C
LPC2210FBD144/01	16 kB	yes	–40 °C to +85 °C
LPC2220FBD144	64 kB	yes	–40 °C to +85 °C
LPC2220FET144	64 kB	yes	–40 °C to +85 °C
LPC2220FET144/G	64 kB	yes	–40 °C to +85 °C

1.4.5 LPC2212/2214

Table 10. LPC2212/2214 Ordering information

Type number	Package		
	Name	Description	Version
LPC2212FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2212FBD144/00	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2212FBD144/01	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2214FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2214FBD144/00	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2214FBD144/01	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1

Table 11. LPC2212/2214 Ordering options

Type number	Flash memory	RAM	Fast GPIO/ SSP/ Enhanced UART, ADC, Timer	Temperature range
LPC2212FBD144	128 kB	16 kB	no	–40 °C to +85 °C
LPC2212FBD144/00	128 kB	16 kB	no	–40 °C to +85 °C
LPC2212FBD144/01	128 kB	16 kB	yes	–40 °C to +85 °C
LPC2214FBD144	256 kB	16 kB	no	–40 °C to +85 °C
LPC2214FBD144/00	256 kB	16 kB	no	–40 °C to +85 °C
LPC2214FBD144/01	256 kB	16 kB	yes	–40 °C to +85 °C

1.4.6 LPC2290

Table 12. LPC2290 Ordering information

Type number	Package		
	Name	Description	Version
LPC2290FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2290FBD144/01	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1

Table 13. LPC2290 Ordering options

Type number	RAM	CAN	Enhancements	Temperature range
LPC2290FBD144	16 kB	2 channels	None	–40 °C to +85 °C
LPC2290FBD144/01	64 kB	2 channels	Higher CPU clock, more on-chip SRAM, Fast I/Os, improved UARTs, added SSP, upgraded ADC	–40 °C to +85 °C

1.4.7 LPC2292/2294

Table 14. LPC2292/2294 Ordering information

Type number	Package		
	Name	Description	Version
LPC2292FBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2292FBD144/00	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2292FBD144/01	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2292FET144/00	TFBGA144	plastic thin fine-pitch ball grid array package; 144 balls; body 12 × 12 × 0.8 mm	SOT569-1
LPC2292FET144/01	TFBGA144	plastic thin fine-pitch ball grid array package; 144 balls; body 12 × 12 × 0.8 mm	SOT569-1
LPC2292FET144/G	TFBGA144	plastic thin fine-pitch ball grid array package; 144 balls; body 12 × 12 × 0.8 mm	SOT569-1

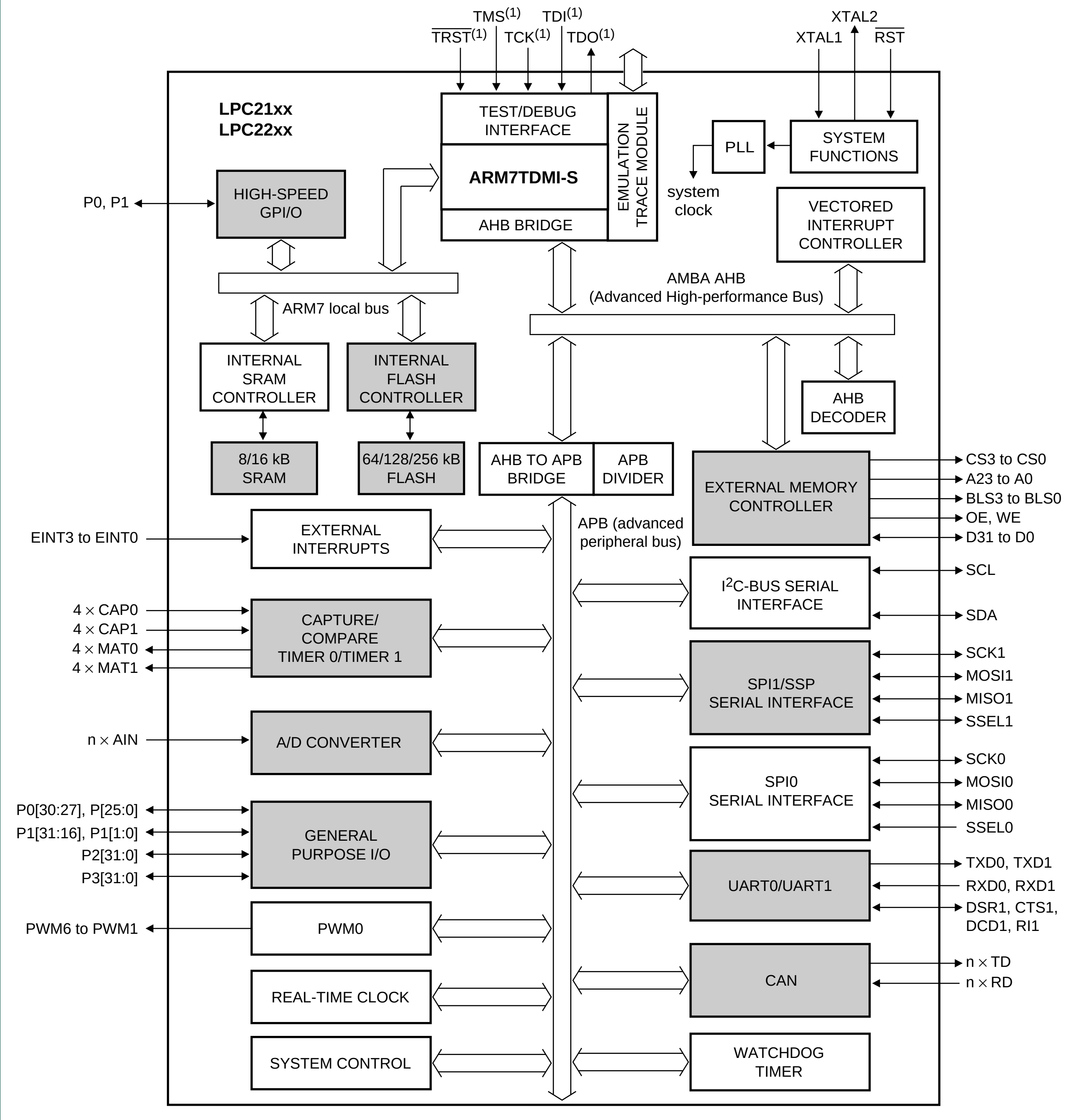
Table 14. LPC2292/2294 Ordering information *…continued*

Type number	Package		
	Name	Description	Version
LPC2294HBD144	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2294HBD144/00	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1
LPC2294HBD144/01	LQFP144	plastic low profile quad flat package; 144 leads; body 20 × 20 × 1.4 mm	SOT486-1

Table 15. LPC2292/2294 Ordering options

Type number	Flash memory	RAM	CAN	Fast GPIO/ SSP/ Enhanced UART, ADC, Timer	Temperature range
LPC2292FBD144	256 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2292FBD144/00	256 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2292FBD144/01	256 kB	16 kB	2 channels	yes	–40 °C to +85 °C
LPC2292FET144/00	256 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2292FET144/01	256 kB	16 kB	2 channels	yes	–40 °C to +85 °C
LPC2292FET144/G	256 kB	16 kB	2 channels	no	–40 °C to +85 °C
LPC2294HBD144	256 kB	16 kB	4 channels	no	–40 °C to +125 °C
LPC2294HBD144/00	256 kB	16 kB	4 channels	no	–40 °C to +125 °C
LPC2294HBD144/01	256 kB	16 kB	4 channels	yes	–40 °C to +125 °C

1.5 Block diagram



Grey-shaded blocks indicate configuration or pinout dependent on part and version number, see Table 16.

Fig 1. LPC21xx and LPC22xx block diagram

Table 16. LPC21xx/22xx part-specific configuration

Part	EMC	SRAM	Flash	Legacy GPIO	Fast GPIO	SSP	CAN channels	ADC channels/ enhanced ADC	Enhanced UART	Enhanced timers
No-suffix and /00 parts										
LPC2109	-	8 kB	64 kB	P0/1	-	-	1	4/-	-	-
LPC2119	-	16 kB	128 kB	P0/1	-	-	2	4/-	-	-
LPC2129	-	16 kB	256 kB	P0/1	-	-	2	4/-	-	-
LPC2114	-	16 kB	128 kB	P0/1	-	-	-	4/-	-	-
LPC2124	-	16 kB	256 kB	P0/1	-	-	-	4/-	-	-
LPC2194	-	16 kB	256 kB	P0/1	-	-	4	4/-	-	-
LPC2210	yes	16 kB	-	P0/1/2/3	-	-	-	4/-	-	-
LPC2220	yes	64 kB	-	P0/1/2/3	P0/1	yes	-	8/yes	yes	yes
LPC2212	yes	16 kB	128 kB	P0/1/2/3	-	-	-	8/-	-	-
LPC2214	yes	16 kB	256 kB	P0/1/2/3	-	-	-	8/-	-	-
LPC2290	yes	16 kB	-	P0/1/2/3	-	-	2	8/-	-	-
LPC2292	yes	16 kB	256 kB	P0/1/2/3	-	-	2	8/-	-	-
LPC2294	yes	16 kB	256 kB	P0/1/2/3	-	-	4	8/-	-	-
/01 parts										
LPC2109	-	8 kB	64 kB	P0/1	P0/1	yes	1	4/yes	yes	yes
LPC2119	-	16 kB	128 kB	P0/1	P0/1	yes	2	4/yes	yes	yes
LPC2129	-	16 kB	256 kB	P0/1	P0/1	yes	2	4/yes	yes	yes
LPC2114	-	16 kB	128 kB	P0/1	P0/1	yes	-	4/yes	yes	yes
LPC2124	-	16 kB	256 kB	P0/1	P0/1	yes	-	4/yes	yes	yes
LPC2194	-	16 kB	256 kB	P0/1	P0/1	yes	4	4/yes	yes	yes
LPC2210	yes	16 kB	-	P0/1/2/3	P0/1	yes	-	8/yes	yes	yes
LPC2212	yes	16 kB	128 kB	P0/1/2/3	P0/1	yes	-	8/yes	yes	yes
LPC2214	yes	16 kB	256 kB	P0/1/2/3	P0/1	yes	-	8/yes	yes	yes
LPC2290	yes	64 kB	-	P0/1/2/3	P0/1	yes	2	8/yes	yes	yes
LPC2292	yes	16 kB	256 kB	P0/1/2/3	P0/1	yes	2	8/yes	yes	yes
LPC2294	yes	16 kB	256 kB	P0/1/2/3	P0/1	yes	4	8/yes	yes	yes

1.6 Architectural overview

The LPC21xx/LPC22xx consist 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 ARM Peripheral Bus (APB, a compatible superset of ARM’s AMBA Advanced Peripheral Bus) for connection to on-chip peripheral functions. The LPC21xx/LPC22xx configures the ARM7TDMI-S processor in little-endian byte order.

AHB peripherals are allocated a 2 megabyte range of addresses at the very top of the 4 gigabyte ARM memory space. Each AHB peripheral is allocated a 16 kB address space within the AHB address space. LPC21xx/LPC22xx peripheral functions (other than the interrupt controller) are connected to the APB bus. The AHB to APB bridge interfaces the APB bus to the AHB bus. APB peripherals are also allocated a 2 megabyte range of addresses, beginning at the 3.5 gigabyte address point. Each APB peripheral is allocated a 16 kB address space within the APB address space.

The connection of on-chip peripherals to device pins is controlled by a Pin Connect Block (see Section 8.6). This must be configured by software to fit specific application requirements for the use of peripheral functions and pins.

1.7 ARM7TDMI-S processor

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

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

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

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

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

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

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

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

1.8 On-chip flash memory system

The LPC21xx/LPC22xx incorporate a 64 kB to 256 kB flash memory. This memory may be used for both code and data storage. Programming of the flash memory may be accomplished in several ways:

- using the serial built-in JTAG interface
- using In System Programming (ISP) and UART
- using In Application Programming (IAP) capabilities

The application program, using the IAP functions, may also erase and/or program the flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc. The entire flash memory is available for user code because the boot loader resides in a separate memory location.

The LPC21xx/LPC22xx flash memory provides minimum of 100,000 erase/write cycles and 20 years of data-retention.

1.9 On-chip Static RAM (SRAM)

On-chip Static RAM (SRAM) may be used for code and/or data storage. The on-chip SRAM may be accessed as 8-bits, 16-bits, and 32-bits.

The LPC21xx/LPC22xx SRAM is designed to be accessed as a byte-addressed memory. Word and halfword accesses to the memory ignore the alignment of the address and access the naturally-aligned value that is addressed (so a memory access ignores address bits 0 and 1 for word accesses, and ignores bit 0 for halfword accesses). Therefore valid reads and writes require data accessed as halfwords to originate from addresses with address line 0 being 0 (addresses ending with 0, 2, 4, 6, 8, A, C, and E in hexadecimal notation) and data accessed as words to originate from addresses with address lines 0 and 1 being 0 (addresses ending with 0, 4, 8, and C in hexadecimal notation).

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

UM10114

Chapter 2: LPC21xx/22xx Memory map

Rev. 4 — 2 May 2012 User manual

2.1 How to read this chapter

Remark: The LPC21xx and LPC22xx contain different memory configurations and APB/AHB peripherals. See Table 17 for part-specific memory and peripherals.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 "How to read this manual".

Table 17. LPC21xx and LPC22xx memory and peripheral configuration

Part	EMC Figure 2	SRAM Figure 2	Flash Figure 2	Fast GPIO Figure 2	SSP Table 18	CAN Table 18
	addresses	size/ addresses	size/ addresses	addresses	APB base addresses	
No suffix and /01 parts						
LPC2109	-	8 kB/ 0x4000 0000 - 0x4000 1FFF	64 kB/ 0x0000 0000 - 0x0000 FFFF	-	-	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000
LPC2119	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	128 kB/ 0x0000 0000 - 0x0001 FFFF	-	-	0xE003 8000 - 0xE004 0000 CAN1:0xE004 4000 CAN2: 0xE004 8000
LPC2129	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	-	-	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2114	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	128 kB/ 0x0000 0000 - 0x0001 FFFF	-	-	-
LPC2124	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	-	-	-
LPC2194	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	-	-	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000 CAN3: 0xE004 C000 CAN4: 0xE005 0000
LPC2210	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	-	-	-	-
LPC2220	0x8000 0000 - 0x83FF FFFF	64 kB/ 0x4001 0000 - 0x4000 FFFF	-	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	-
LPC2212	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	128 kB/ 0x0000 0000 - 0x0001 FFFF	-	-	-

Table 17. LPC21xx and LPC22xx memory and peripheral configuration

Part	EMC Figure 2	SRAM Figure 2	Flash Figure 2	Fast GPIO Figure 2	SSP Table 18	CAN Table 18
	addresses	size/ addresses	size/ addresses	addresses	APB base addresses	
LPC2214	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	-	-	-
LPC2290	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	-	-	-	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2292	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	-	-	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2294	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	-	-	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000 CAN3: 0xE004 C000 CAN4: 0xE005 0000
/01 parts						
LPC2109	-	8 kB/ 0x4000 0000 - 0x4000 1FFF	64 kB/ 0x0000 0000 - 0x0000 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000
LPC2119	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	128 kB/ 0x0000 0000 - 0x0001 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2129	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x003 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2114	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	128 kB/ 0x0000 0000 - 0x0001 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	-
LPC2124	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x003 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	-
LPC2194	-	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000 CAN3: 0xE004 C000 CAN4: 0xE005 0000
LPC2210	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	-	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	-
LPC2212	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	128 kB/ 0x0000 0000 - 0x0001 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	-

Table 17. LPC21xx and LPC22xx memory and peripheral configuration

Part	EMC Figure 2	SRAM Figure 2	Flash Figure 2	Fast GPIO Figure 2	SSP Table 18	CAN Table 18
	addresses	size/ addresses	size/ addresses	addresses	APB base addresses	
LPC2214	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	-
LPC2290	0x8000 0000 - 0x83FF FFFF	64 kB/ 0x4001 0000 - 0x4000 FFFF	-	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2292	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000
LPC2294	0x8000 0000 - 0x83FF FFFF	16 kB/ 0x4000 0000 - 0x4000 2FFF	256 kB/ 0x0000 0000 - 0x0003 FFFF	0x3FFF C000 - 0x3FFF FFFF	0xE005 C000	0xE003 8000 - 0xE004 0000 CAN1: 0xE004 4000 CAN2: 0xE004 8000 CAN3: 0xE004 C000 CAN4: 0xE005 0000

2.2 Memory maps

The LPC21xx and LPC22xx incorporate several distinct memory regions, shown in the following figures. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset. The interrupt vector area supports address remapping, which is described later in this section.

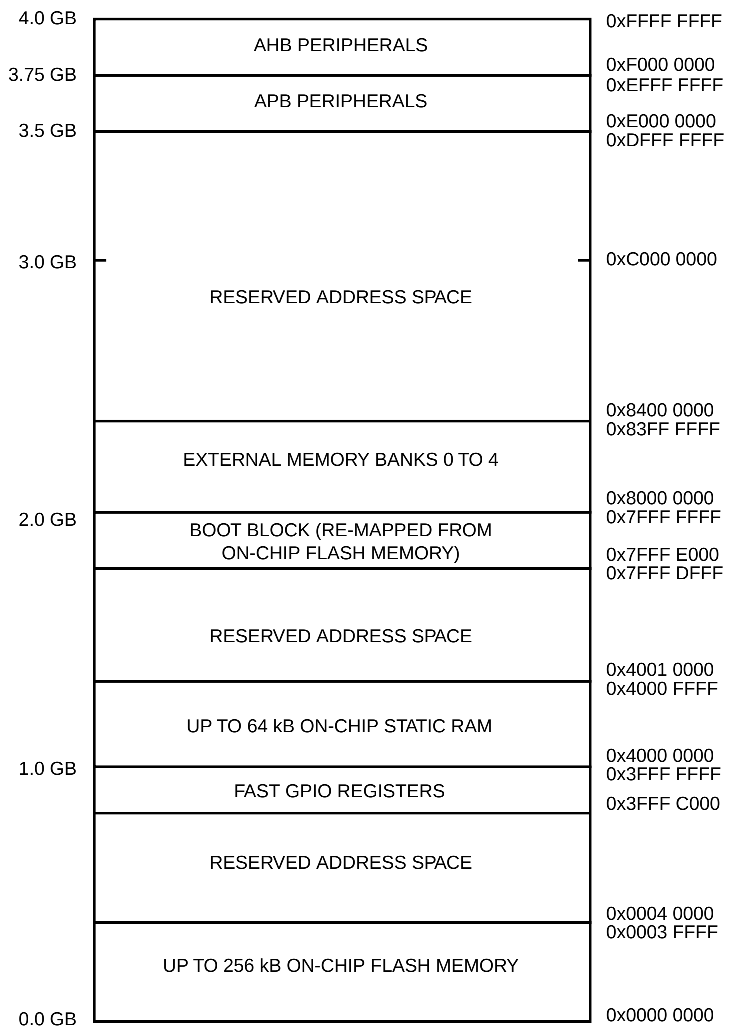
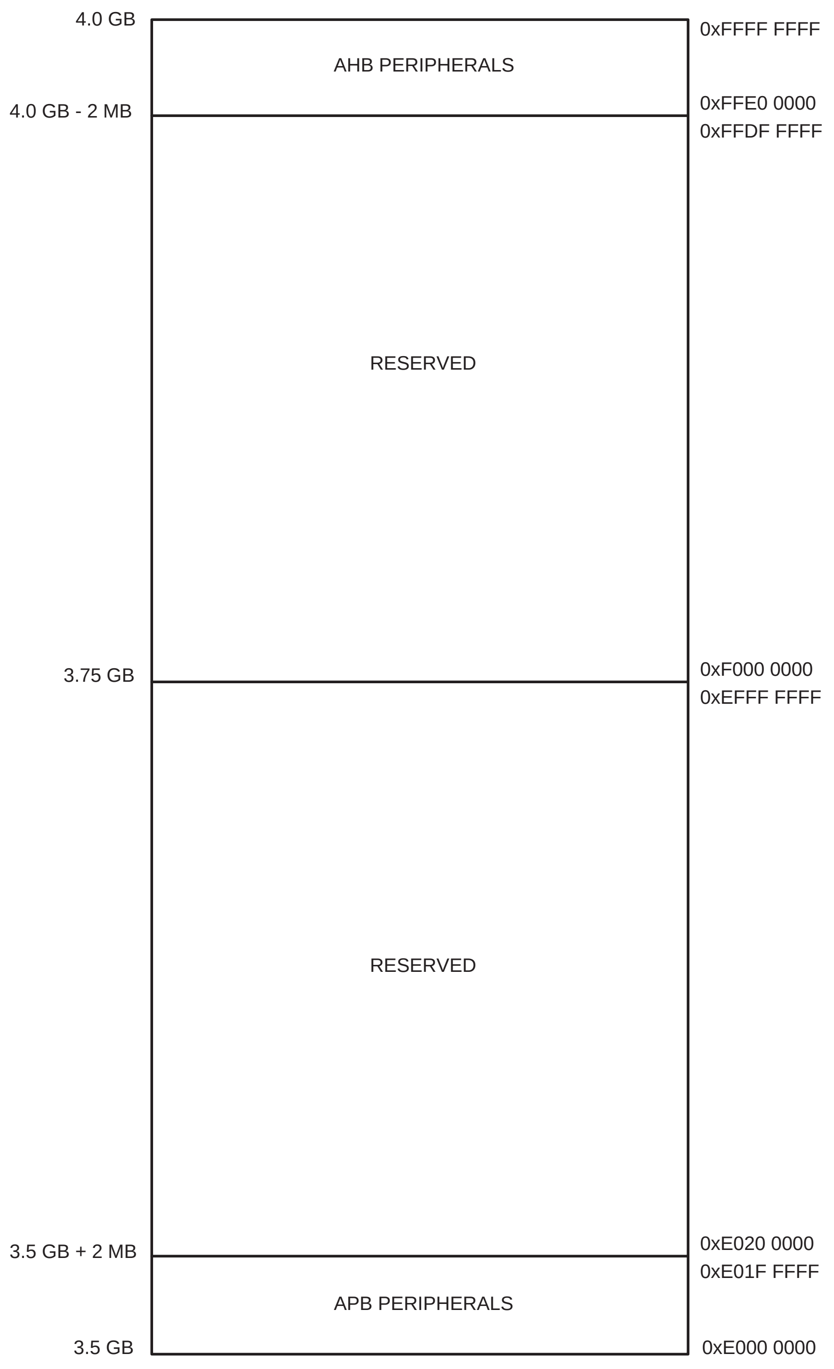


Fig 2. LPC21xx and LPC22xx system memory map



AHB section is 128 x 16 kB blocks (totaling 2 MB).

APB section is 128 x 16 kB blocks (totaling 2MB).

Fig 3. Peripheral memory map

Figures 3 through 4 and Table 18 show different views of the peripheral address space. Both the AHB and APB peripheral areas are 2 megabyte spaces which are divided up into 128 peripherals. Each peripheral space is 16 kilobytes in size. This allows simplifying the address decoding for each peripheral. All peripheral register addresses are word aligned (to 32-bit boundaries) regardless of their size. This eliminates the need for byte lane mapping hardware that would be required to allow byte (8-bit) or half-word (16-bit) accesses to occur at smaller boundaries. An implication of this is that word and half-word registers must be accessed all at once. For example, it is not possible to read or write the upper byte of a word register separately.

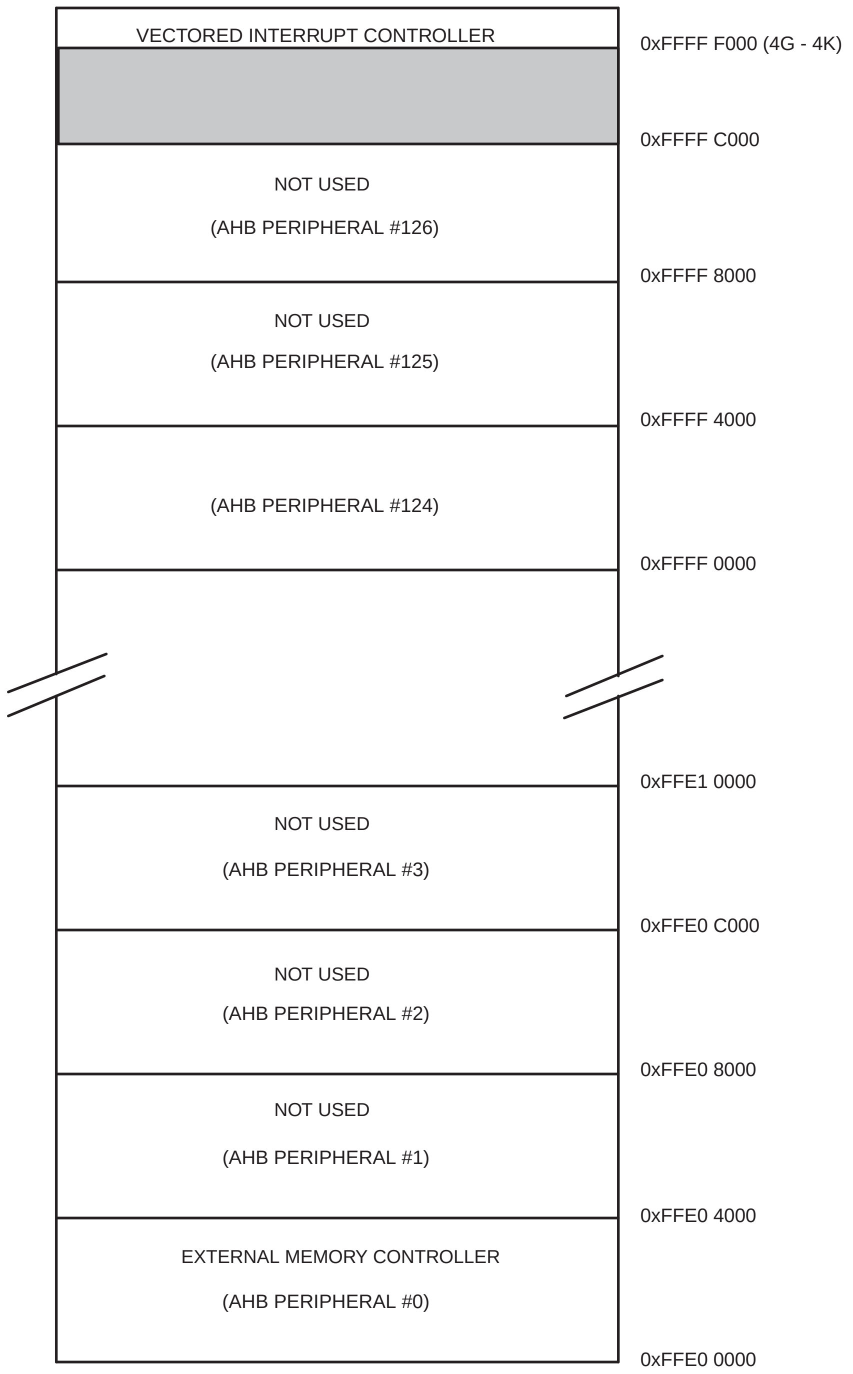


Fig 4. AHB peripheral map

Table 18. APB peripheries and base addresses

APB peripheral	Base address	Peripheral name
0	0xE000 0000	Watchdog timer
1	0xE000 4000	Timer 0
2	0xE000 8000	Timer 1
3	0xE000 C000	UART0
4	0xE001 0000	UART1
5	0xE001 4000	PWM
6	0xE001 8000	Not used
7	0xE001 C000	I^{2}C
8	0xE002 0000	SPI0
9	0xE002 4000	RTC
10	0xE002 8000	GPIO
11	0xE002 C000	Pin connect block
12	0xE003 0000	SPI1
13	0xE003 4000	10 bit ADC
14	0xE003 8000	CAN Acceptance Filter RAM
15	0xE003 C000	CAN Acceptance Filter Registers
16	0xE004 0000	CAN Common Registers
17	0xE004 4000	CAN Controller 1
18	0xE004 8000	CAN Controller 2
19	0xE004 C000	CAN Controller 3
20	0xE005 0000	CAN Controller 4
21 - 22	0xE005 4000 0xE005 8000	Not used
23	0xE005 C000	SSP
24 - 126	0xE006 0000 - 0xE01F 8000	Not used
127	0xE01F C000	System Control Block

2.3 LPC21xx and LPC22xx memory re-mapping and boot block

2.3.1 Memory map concepts and operating modes

The basic concept on the LPC21xx and LPC22xx 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 19 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 20. Re-mapping of the interrupts is accomplished via the Memory Mapping Control features. To select a specific memory mapping mode, see Table 62.

Table 19. ARM exception vector locations

Address	Exception
0x0000 0000	Reset
0x0000 0004	Undefined Instruction
0x0000 0008	Software Interrupt
0x0000 000C	Prefetch Abort (instruction fetch memory fault)
0x0000 0010	Data Abort (data access memory fault)
0x0000 0014	Reserved **Note:** Identified as reserved in ARM documentation.
0x0000 0018	IRQ
0x0000 001C	FIQ

Table 20. LPC21xx and LPC22xx memory mapping modes

Mode	Activation	Usage
Boot Loader mode	Hardware activation by any Reset	The boot loader **always** executes after any reset. The boot block interrupt vectors are mapped to the bottom of memory to allow handling exceptions and using interrupts during the boot loading process.
User Flash mode	Software activation by Boot code	Activated by boot loader when a valid user program signature is recognized in memory and boot loader operation is not forced. Interrupt vectors are not re-mapped and are found in the bottom of the flash memory. **Remark:** This mode is not available on flashless parts (see Table 17).
User RAM mode	Software activation by User program	Activated by a user program as desired. Interrupt vectors are re-mapped to the bottom of the Static RAM.
User External mode	Activated by BOOT1:0 pins	Activated by the boot loader when one or both BOOT pins are LOW at the end of RESET LOW. Interrupt vectors are re-mapped from the bottom of the external memory map (see Section 8.6.5). **Remark:** This mode is available for parts with external memory controller only (see Table 17).

2.3.2 Memory re-mapping

In order to allow for compatibility with future derivatives, the entire boot block is mapped to the top of the on-chip memory space. Memory spaces other than the interrupt vectors remain in fixed locations. Figure 5 shows the on-chip memory mapping in the modes defined above.

The portion of memory that is re-mapped to allow interrupt processing in different modes includes the interrupt vector area (32 bytes) and an additional 32 bytes, for a total of 64 bytes. The re-mapped code locations overlay addresses 0x0000 0000 through 0x0000 003F. 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 two reasons this configuration was chosen:

1. Minimize the need for the SRAM and Boot Block vectors to deal with arbitrary boundaries in the middle of code space.

2. To provide space to store constants for jumping beyond the range of single word branch instructions.

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

Details on re-mapping and examples can be found in Section 6.8.1 “Memory Mapping control register (MEMMAP - 0xE01F C040)” on page 69.

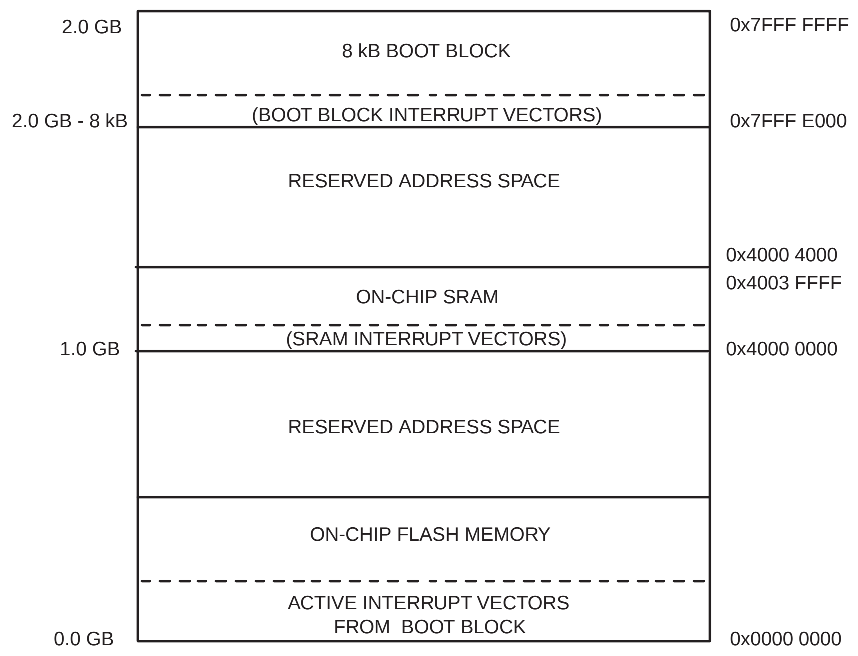


Fig 5. Map of lower memory is showing re-mapped and re-mappable areas for a part with on-chip flash memory

2.4 Prefetch Abort and Data Abort Exceptions

The LPC21xx and LPC22xx generate 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 LPC21xx and LPC22xx, those areas are:
  - Address space between the on-chip non-volatile memory and On-Chip SRAM, labelled "Reserved Address Space" in Figure 2 , and Figure 5. This is an address range from 0x0002 0000 to 0x3FFF FFFF for the 128 kB flash device and 0x0004 0000 to 0x3FFF FFFF for the 256 kB flash device.
  - Address space between on-chip SRAM and the boot block. This is the address range from 0x4000 4000 to 0x7FFF DFFF, labelled "Reserved Address Space" in Figure 2, and Figure 5.

- Address space between the top of the boot block and the APB peripheral space, except space used for external memory (LPC2292/2294 only). This is the address range from 0x8000 0000 to 0xDFFF FFFF, labelled "Reserved Address Space" in Figure 2, and Figure 5.
- Reserved regions of the AHB and APB spaces. See Figure 3 and Table 18.

- Unassigned AHB peripheral spaces. See Figure 4.
- Unassigned APB peripheral spaces. See Table 18.

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

Within the address space of an existing APB peripheral, a data abort exception is not generated in response to an access to an undefined address. Address decoding within each peripheral is limited to that needed to distinguish defined registers within the peripheral itself. For example, an access to address 0xE000 D000 (an undefined address within the UART0 space) may result in an access to the register defined at address 0xE000 C000. Details of such address aliasing within a peripheral space are not defined in the LPC21xx and LPC22xx documentation and are not a supported feature.

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

UM10114

Chapter 3: LPC21xx/22xx Memory Accelerator Module (MAM)

Rev. 4 — 2 May 2012 User manual

3.1 How to read this chapter

The MAM is identical for all parts with **flash memory**. It is available in the following parts:

- LPC2109, LPC2119, LPC2129, and /01 versions
- LPC2114, LPC2124, and /01 versions
- LPC2194 and LPC2194/01
- LPC2212, LPC2214, and /01 versions
- LPC2292, LPC2294, and /01 versions

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

3.2 Introduction

The MAM block in the LPC21xx and LPC22xx maximizes the performance of the ARM processor when it is running code in flash memory using a dual flash bank.

3.3 Operation

Simply put, the Memory Accelerator Module (MAM) attempts to have the next ARM instruction that will be needed in its latches in time to prevent CPU fetch stalls. The method used is to split the flash memory into two banks, each capable of independent accesses. Each of the two flash banks has its own prefetch buffer and branch trail buffer. The branch trail buffers for the two banks capture two 128-bit lines of flash data when an instruction fetch is not satisfied by either the prefetch buffer or branch trail buffer for its bank, and for which a prefetch has not been initiated. Each prefetch buffer captures one 128-bit line of instructions from its flash bank at the conclusion of a prefetch cycle initiated speculatively by the MAM.

Each 128 bit value includes four 32-bit ARM instructions or eight 16-bit Thumb instructions. During sequential code execution, typically one flash bank contains or is fetching the current instruction and the entire flash line that contains it. The other bank contains or is prefetching the next sequential code line. After a code line delivers its last instruction, the bank that contained it begins to fetch the next line in that bank.

Timing of flash read operations is programmable and is described in Section 3.9.

Branches and other program flow changes cause a break in the sequential flow of instruction fetches described above. When a backward branch occurs, there is a distinct possibility that a loop is being executed. In this case the branch trail buffers may already contain the target instruction. If so, execution continues without the need for a flash read cycle. For a forward branch, there is also a chance that the new address is already contained in one of the prefetch buffers. If it is, the branch is again taken with no delay.

When a branch outside the contents of the branch trail and prefetch buffers is taken, one flash access cycle is needed to load the branch trail buffers. Subsequently, there will typically be no further fetch delays until another such “Instruction Miss” occurs.

The flash memory controller detects data accesses to the flash memory and uses a separate buffer to store the results in a manner similar to that used during code fetches. This allows faster access to data if it is accessed sequentially. A single line buffer is provided for data accesses, as opposed to the two buffers per flash bank that are provided for code accesses. There is no prefetch function for data accesses.

3.4 MAM blocks

The Memory Accelerator Module is divided into several functional blocks:

- A flash address latch for each bank: An incrementor function is associated with the bank 0 flash address latch.
- Two flash memory banks
- Instruction latches, data latches, address comparison latches
- Control and wait logic

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

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

3.4.1 Flash memory bank

There are two banks of flash memory in order to allow parallel access and eliminate delays for sequential access.

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

The flash memories are wired so that each sector exists in both banks and that a sector erase operation acts on part of both banks simultaneously. In effect, the existence of two banks is transparent to the programming functions.

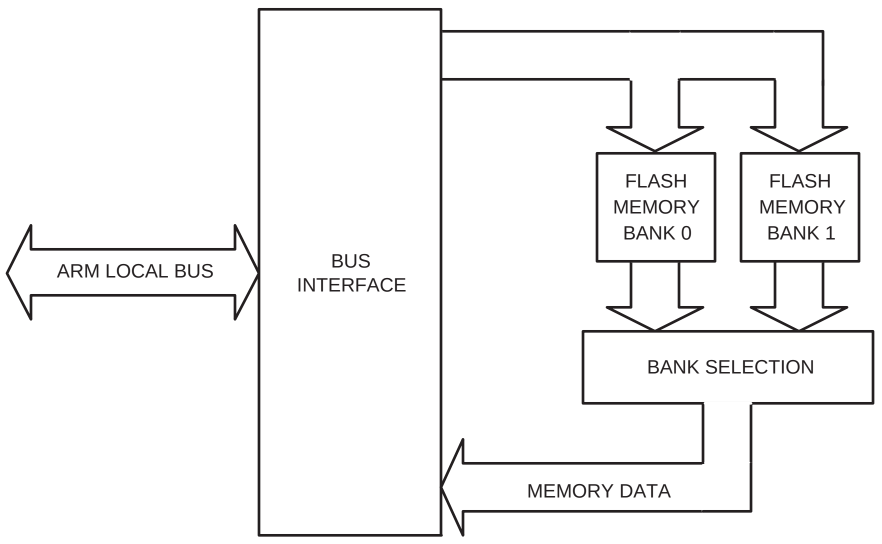


Fig 6. Simplified block diagram of the Memory Accelerator Module (MAM)

3.4.2 Instruction latches and data latches

Code and data accesses are treated separately by the Memory Accelerator Module.There are two sets of 128-bit instruction latches and 12-bit comparison address latches associated with each flash bank. One of the two sets, called the branch trail buffer, holds the data and comparison address for that bank from the last instruction miss. The other set, called the prefetch buffer, holds the data and comparison address from prefetches undertaken speculatively by the MAM. Each instruction latch holds 4 words of code (4 ARM instructions, or 8 Thumb instructions).

Similarly, there is a 128-bit data latch and 13-bit data address latch, that are used during data cycles. This single set of latches is shared by both flash banks. Each data access that is not in the data latch causes a flash fetch of 4 words of data, which are captured in the data latch. This speeds up sequential data operations, but has little or no effect on random accesses.

3.4.3 Flash programming Issues

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

In order to preclude the possibility of stale data being read from the flash memory, the LPC21xx and LPC22xx MAM holding latches are automatically invalidated at the beginning of any flash programming or erase operation. Any subsequent read from a flash address will cause a new fetch to be initiated after the flash operation has completed.

3.5 MAM operating modes

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

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

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

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

Table 21. MAM responses to program accesses of various types

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

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

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

Table 22. MAM responses to data accesses of various types

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

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

3.6 MAM configuration

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

3.7 Register description

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

Table 23. Summary of MAM registers

Name	Description	Access	Reset value[1]	Address
MAMCR	Memory Accelerator Module Control Register. Determines the MAM functional mode, that is, to what extent the MAM performance enhancements are enabled. See Table 24.	R/W	0x0	0xE01F C000
MAMTIM	Memory Accelerator Module Timing control. Determines the number of clocks used for flash memory fetches (1 to 7 processor clocks).	R/W	0x07	0xE01F C004

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

3.8 MAM Control Register (MAMCR - 0xE01F C000)

Two configuration bits select the three MAM operating modes, as shown in Table 24. Following Reset, MAM functions are disabled. Changing the MAM operating mode causes the MAM to invalidate all of the holding latches, resulting in new reads of flash information as required.

Table 24. MAM Control Register (MAMCR - address 0xE01F C000) bit description

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

3.9 MAM Timing register (MAMTIM - 0xE01F C004)

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

Table 25. MAM Timing register (MAMTIM - address 0xE01F C004) bit description

Bit	Symbol	Value	Description	Reset value
2:0	MAM_fetch_cycle_timing	000	0 - Reserved.	07
		001	1 - MAM fetch cycles are 1 processor clock (CCLK) in duration	
		010	2 - MAM fetch cycles are 2 CCLKs in duration	
		011	3 - MAM fetch cycles are 3 CCLKs in duration	
		100	4 - MAM fetch cycles are 4 CCLKs in duration	
		101	5 - MAM fetch cycles are 5 CCLKs in duration	
		110	6 - MAM fetch cycles are 6 CCLKs in duration	
		111	7 - MAM fetch cycles are 7 CCLKs in duration	
		Warning: These bits set the duration of MAM flash fetch operations as listed here. Improper setting of this value may result in incorrect operation of the device.		
7:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

3.10 MAM usage notes

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

For system clock slower than 20 MHz, MAMTIM can be 001. For system clock between 20 MHz and 40 MHz, flash access time is suggested to be 2 CCLKs, while in systems with system clock faster than 40 MHz, 3 CCLKs are proposed. For system clocks of 60 MHz and above, 4CCLK’s are needed.

Table 26. Suggestions for MAM timing selection

system clock	Number of MAM fetch cycles in MAMTIM
< 20 MHz	1 CCLK
20 MHz to 40 MHz	2 CCLK
40 MHz to 60 MHz	3 CCLK
> 60 MHz	4 CCLK

UM10114

Chapter 4: LPC21xx/22xx External Memory Controller (EMC)

Rev. 4 — 2 May 2012 User manual

4.1 How to read this chapter

This chapter applies to all parts with external memory controller. The EMC is identical for all these parts. It is available in the following parts (in all 144 pin packages):

- LPC2210, LPC2210/01, and LPC2220
- LPC2212, LPC2214, and /01 versions
- LPC2290 and LPC2290/01
- LPC2292, LPC2294, and /01 versions

The LPC21xx parts do **not** have an EMC controller.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

4.2 Features

- Support for various static memory-mapped devices including RAM, ROM, flash, burst ROM, and some external I/O devices
- Asynchronous page mode read operation in non-clocked memory subsystems
- Asynchronous burst mode read access to burst mode ROM devices
- Independent configuration for up to four banks, each up to 16 MB
- Programmable bus turnaround (idle) cycles (1 to 16)
- Programmable read and write WAIT states (up to 32) for static RAM devices
- Programmable initial and subsequent burst read WAIT state, for burst ROM devices
- Programmable write protection
- Programmable burst mode operation
- Programmable read byte lane enable control

4.3 Description

The external Static Memory Controller is an AMBA AHB slave module which provides an interface between an AMBA AHB system bus and external (off-chip) memory devices. It provides support for up to four independently configurable memory banks simultaneously. Each memory bank is capable of supporting SRAM, ROM, Flash EPROM, Burst ROM memory, or some external I/O devices.

Each memory bank may be 8, 16, or 32 bits wide.

Since the LPC22xx 144 pin packages pin out address lines A[23:0] only, the decoding among the four banks uses address bits A[25:24]. The native location of the four banks is at the start of the External Memory area identified in Figure 2, but Bank 0 can be used for initial booting under control of the state of the BOOT[1:0] pins.

Table 27. Address ranges of the external memory banks

Bank	Address range	Configuration register
0	0x8000 0000 - 0x80FF FFFF	BCFG0
1	0x8100 0000 - 0x81FF FFFF	BCFG1
2	0x8200 0000 - 0x82FF FFFF	BCFG2
3	0x8300 0000 - 0x83FF FFFF	BCFG3

4.4 Pin description

Table 28. External Memory Controller pin description

Pin name	Type	Pin description
D[31:0]	Input/Output	External memory Data lines
A[23:0]	Output	External memory Address lines
OE	Output	Low-active Output Enable signal
BLS[3:0]	Output	Low-active Byte Lane Select signals
WE	Output	Low-active Write Enable signal
CS[3:0]	Output	Low-active Chip-Select signals

4.5 Register description

The external memory controller contains 4 registers as shown in Table 29.

Table 29. External Memory Controller register map

Name	Description	Access	Reset value, see Table 30	Address
BCFG0	Configuration register for memory bank 0	R/W	0x0000 FBEF	0xFFE0 0000
BCFG1	Configuration register for memory bank 1	R/W	0x2000 FBEF	0xFFE0 0004
BCFG2	Configuration register for memory bank 2	R/W	0x1000 FBEF	0xFFE0 0008
BCFG3	Configuration register for memory bank 3	R/W	0x0000 FBEF	0xFFE0 000C

Each register selects the following options for its memory bank:

- The number of idle clock cycles inserted between read and write accesses in this bank, and between an access in another bank and an access in this bank, to avoid bus contention between devices (1 to 17 clocks)
- The length of read accesses, except for subsequent reads from a burst ROM (3 to 35 clocks)
- The length of write accesses (3 to 34 clocks)
- Whether the bank is write-protected or not
- Whether the bank is 8, 16, or 32 bits wide

4.5.1 Bank Configuration Registers 0-3 (BCFG0-3 - 0xFFE0 0000 to 0xFFE0 000C)

Table 30. Bank Configuration Registers 0-3 (BCFG0-3 - 0xFFE0 0000 to 0xFFE0 000C) address description

BCFG0-3	Name	Function	Reset value
3:0	IDCY	This field controls the minimum number of “idle” CCLK cycles that the EMC maintains between read and write accesses in this bank, and between an access in another bank and an access in this bank, to avoid bus contention between devices. The number of idle CCLK cycles between such accesses is the value in this field plus 1.	1111
4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9:5	WST1	This field controls the length of read accesses (except for subsequent reads from a burst ROM). The length of read accesses, in CCLK cycles, is this field value plus 3.	11111
10	RBLE	This bit should be 0 for banks composed of byte-wide or non-byte-partitioned devices, so that the EMC drives the BLS3:0 lines High during read accesses. This bit should be 1 for banks composed of 16-bit and 32-bit wide devices that include byte select inputs, so that the EMC drives the BLS3:0 lines Low during read accesses.	0
15:11	WST2	For SRAM banks, this field controls the length of write accesses, which consist of: One CCLK cycle of address setup with CS, BLS, and WE high This value plus 1, CCLK cycles with address valid and CS, BLS, and WE low AND One CCLK cycle with address valid, CS low, BLS and WE high. For burst ROM banks, this field controls the length of subsequent accesses, which are (this value plus 1) CCLK cycles long.	11111
23:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
24	BUSERR	The only known case in which this bit is set is if the EMC detects an AMBA request for more than 32 bits of data. The ARM7TDMI-S will not make such a request.	0
25	WPERR	This bit is set if software attempts to write to a bank that has the WP bit 1. Write a 1 to this bit to clear it.	0
26	WP	A 1 in this bit write-protects the bank.	0
27	BM	A 1 in this bit identifies a burst-ROM bank.	0
29:28	MW	This field controls the width of the data bus for this bank: 00=8 bit, 01=16 bit, 10=32 bit, 11=reserved	See Table 31
31:30	AT	Always write 00 to this field.	00

The table below shows the state of BCFG0[29:28] after the Boot Loader has run. The hardware reset state of these bits is 10.

Table 31. Default memory widths at reset

Bank	BOOT[1:0] during Reset	BCFG[29:28] Reset value	Memory width
0	LL	00	8 bits
0	LH	01	16 bits
0	HL	10	32 bits
0	HH	01	16 bits
1	XX	10	32 bits
2	XX	01	16 bits
3	XX	00	8 bits

4.5.2 Read Byte Lane Control (RBLE)

The External Memory Controller (EMC) generates byte lane control signals BLS[3:0] according to:

- External memory bank data bus width, defined within each configuration register (see MW field in BCFG register)
- External memory bank type, being either byte (8 bits), halfword (16 bits) or word (32 bits) (see RBLE field in BCFG register)

Each memory bank can either be 8, 16 or 32 bits wide. The type of memory used to configure a particular memory bank determines how the WE and BLS signals are connected to provide byte, halfword and word access. For read accesses, it is necessary to control the BLS signals by driving them either all HIGH, or all LOW.

This control is achieved by programming the Read Byte Lane Enable (RBLE) bit within each configuration register. The following two sections explain why different connections in respect of WE and BLS[3:0] are needed for different memory configurations.

4.5.3 Accesses to memory banks constructed from 8-bit or non byte-partitioned memory devices

For memory banks constructed from 8-bit or non byte-partitioned memory devices, it is important that the RBLE bit is cleared to zero within the respective memory bank configuration register. This forces all BLS[3:0] lines HIGH during a read access to that particular bank.

Figure 7 (a), Figure 8 (a) and Figure 9 show 8-bit memory being used to configure memory banks that are 8, 16 and 32 bits wide. In each of these configurations, the BLS[3:0] signals are connected to write enable (WE) inputs of each 8-bit memory.

Note: The WE signal from the EMC is not used. For write transfers, the relevant BLS[3:0] byte lane signals are asserted LOW and steer the data to the addressed bytes.

For read transfers, all of the BLS[3:0] lines are deasserted HIGH, which allows the external bus to be defined for at least the width of the accessed memory.

4.5.4 Accesses to memory banks constructed from 16 or 32 bit memory devices

For memory banks constructed from 16 bit or 32-bit memory devices, it is important that the RBLE bit is set to one within the respective memory bank configuration register. This asserts all BLS[3:0] lines LOW during a read access to that particular bank. For 16 and 32-bit wide memory devices, byte select signals exist and must be appropriately controlled as shown in Figure 7 and Figure 8.

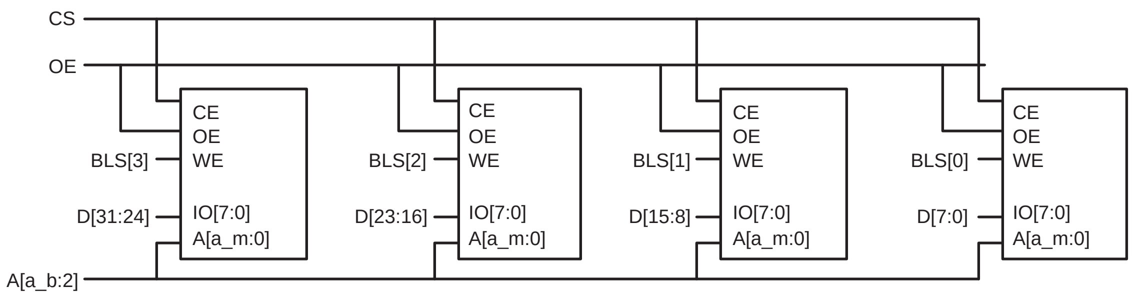
4.6 External memory interface

External memory interfacing depends on the bank width (32, 16 or 8 bit selected via MW bits in corresponding BCFG register). Furthermore, the memory chips require an adequate setup of RBLE bit in BCFG register. Memory accessed with an 8-bit wide data bus require RBLE = 0, while memory banks capable of accepting 16 or 32 bit wide data require RBLE = 1.

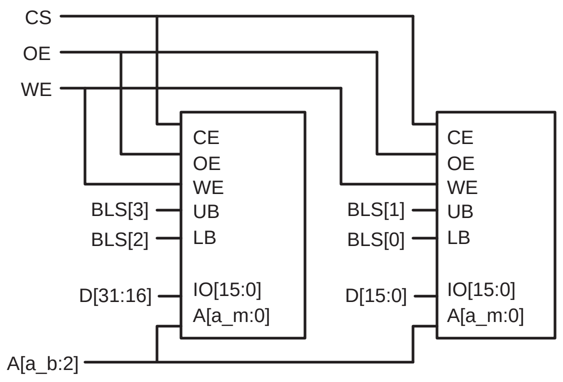
If a memory bank is configured to be 32 bits wide, address lines A0 and A1 can be used as non-address lines. If a memory bank is configured to 16 bits wide, A0 is not required. However, 8 bit wide memory banks do require all address lines down to A0. Configuring A1 and/or A0 lines to provide address or non-address function is accomplished using bits 23 and 24 in Pin Function Select Register 2 (PINSEL2 register, see Table 88).

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

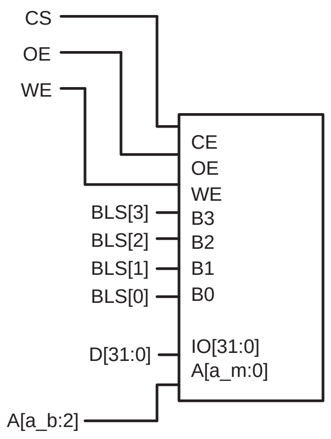
See Section 8.6.5 “Boot control for LPC22xx parts” for how to boot from external memory.



a. 32 bit wide memory bank interfaced to 8 bit memory chips (RBLE = 0)

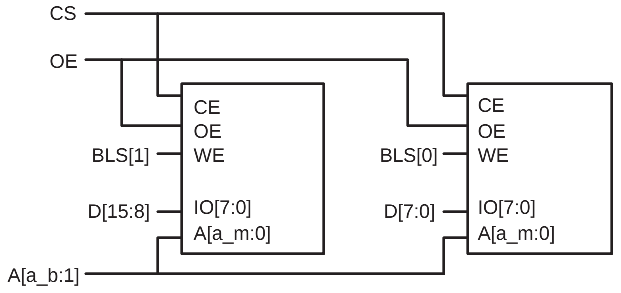


b. 32 bit wide memory bank interfaced to 16 bit memory chips (RBLE = 1)

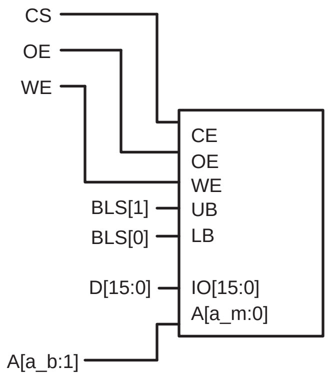


c. 32 bit wide memory bank interfaced to 32 bit memory chips (RBLE = 1)

Fig 7. 32 bit bank external memory interfaces (BGFGx Bits MW = 10)



a. 16 bit wide memory bank interfaced to 8 bit memory chips (RBLE = 0)



b. 16 bit wide memory bank interfaced to 16 bit memory chips (RBLE = 1)

Fig 8. 16 bit bank external memory interfaces (BCFGx bits MW = 01)

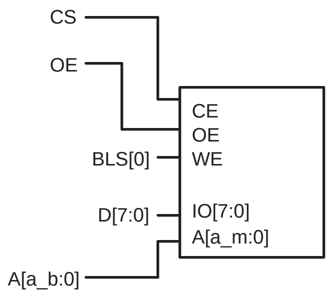


Fig 9. 8 bit bank external memory interface (BCFGx bits MW = 00 and RBLE = 0)

4.7 Typical bus sequences

The following figures show typical external read and write access cycles. XCLK is the clock signal available on P3.23. While not necessarily used by external memory, in these examples it is used to provide time reference (XCLK and CCLK are set to have the same frequency).

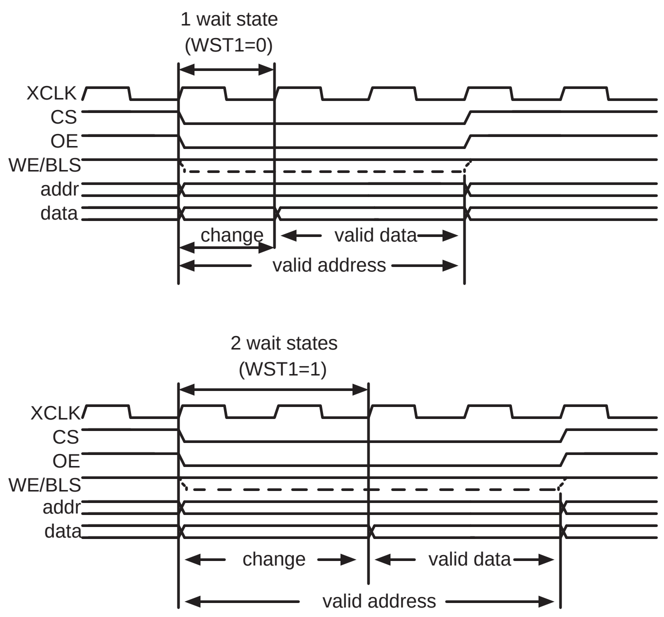


Fig 10. External memory read access (WST1 = 0 and WST1 = 1 examples)

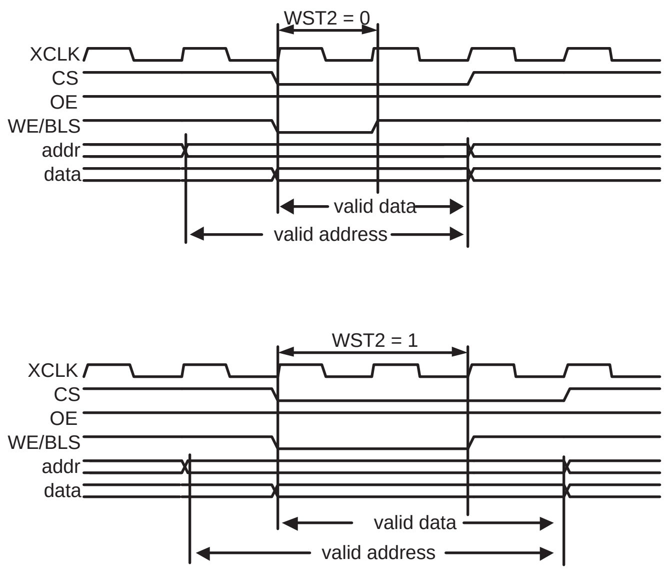


Fig 11. External memory write access (WST2 = 0 and WST2 = 1 examples)

Figure 10 and Figure 11 show typical read and write accesses to external memory. Dashed lines on Figure 10 correspond to memory banks using 16/32 bit memory chips having BLS lines connected to UB/LB or B[3:0] (see Section 4.5.4 and Figure 7 , Figure 8).

It is important to notice that some variations from Figure 10 and Figure 11 do exist in some particular cases.

For example, when the first read access to the memory bank that has just been selected is performed, CS and OE lines may become low one XCLK cycle earlier than it is shown in Figure 11.

Likewise, in a sequence of several consecutive write accesses to SRAM, the last write access will look like those shown in Figure 11. On the other hand, leading write cycles in that case will have data valid one cycle longer. Also, isolated write access will be identical to the one in Figure 11.

The EMC supports sequential access burst reads of up to four consecutive locations in 8, 16 or 32-bit memories. This feature supports burst mode ROM devices and increases the bandwidth by using reduced (configurable) access time for three sequential reads following a quad-location boundary read. Figure 12 shows an external memory burst read transfer. The first burst read access has two wait states and subsequent accesses have zero wait states.

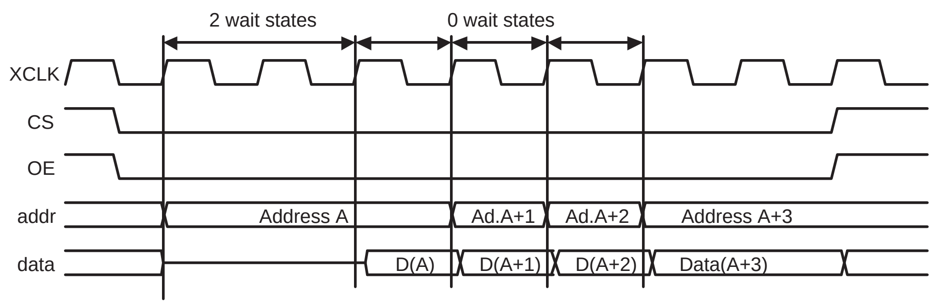


Fig 12. External burst memory read access (WST1 = 0 and WST1 = 1 examples)

4.8 External memory selection

Based on the description of the EMC operation and external memory in general (appropriate read and write access times tAA and tWRITE respectively), the following table can be constructed and used for external memory selection. tCYC is the period of a single CCLK cycle (see Figure 10 and Figure 11 where one XCLK cycle equals one CCLK cycle). fmax is the maximum CCLK frequency achievable in the system with selected external memory.

Table 32. External memory and system requirements

Access cycle	Maximum frequency	WST setting (WST>=0; round up to integer)	Required memory access time
Standard Read	$f_{MAX} \le \frac{2 + WST1}{t_{RAM} + 20ns}$	$WST1 \ge \frac{t_{RAM} + 20ns}{t_{CYC}} - 2$	$t_{RAM} \le t_{CYC} \times (2 + WST1) - 20ns$

Table 32. External memory and system requirements

Access cycle	Maximum frequency	WST setting (WST>=0; round up to integer)	Required memory access time
Standard Write	$f_{MAX} \le \frac{1 + WST2}{t_{WRITE} + 5ns}$	$WST2 \ge \frac{t_{WRITE} - t_{CYC} + 5}{t_{CYC}}$	$t_{WRITE} \le t_{CYC} \times (1 + WST2) - 5ns$
Burst read (initial)	$f_{MAX} \le \frac{2 + WST1}{t_{INIT} + 20ns}$	$WST1 \ge \frac{t_{INIT} + 20ns}{t_{CYC}} - 2$	$t_{INIT} \le t_{CYC} \times (2 + WST1) - 20ns$
Burst read subsequent 3x	$f_{MAX} \le \frac{1}{t_{ROM} + 20ns}$	N/A	$t_{ROM} \le t_{CYC} - 20ns$

UM10114

Chapter 5: LPC21xx/22xx Vectored Interrupt Controller (VIC)

Rev. 4 — 2 May 2012 **User manual**

5.1 How to read this chapter

The VIC is identical for all parts. However, the interrupts routed to the VIC depend on the peripherals implemented on a specific part. See Table 33 for part specific interrupt sources. All other interrupt sources in Table 33 are common to all parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

Table 33. LPC21xx/22xx part-specific interrupts

Part	SSP Registers: Table 51, Table 35	CAN	UART
no suffix and /00 parts			
LPC2109	-	CAN common, CAN1 TX, CAN1 RX	-
LPC2119	-	CAN common, CAN1/2 TX, CAN1/2 RX	-
LPC2129	-	CAN common, CAN1/2 TX, CAN1/2 RX	-
LPC2114	-	-	-
LPC2124	-	-	-
LPC2194	-	CAN common, CAN1/2/3/4 TX, CAN1/2/3/4 RX	-
LPC2210	-	-	-
LPC2220	TXRIS, RXRIS, RTRIS, RORRIS	-	ABTO, ABEO
LPC2212	-	-	-
LPC2214	-	-	-
LPC2290	-	CAN common, CAN1/2 TX, CAN1/2 RX	-
LPC2292	-	CAN common, CAN1/2 TX, CAN1/2 RX	-
LPC2294	-	CAN common, CAN1/2/3/4 TX, CAN1/2/3/4 RX	-
/01 parts			
LPC2109	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1 TX, CAN1 RX, FULLCAN	ABTO, ABEO
LPC2119	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1/2 TX, CAN1/2 RX, FULLCAN	ABTO, ABEO
LPC2129	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1/2 TX, CAN1/2 RX, FULLCAN	ABTO, ABEO
LPC2114	TXRIS, RXRIS, RTRIS, RORRIS	-	ABTO, ABEO
LPC2124	TXRIS, RXRIS, RTRIS, RORRIS	-	ABTO, ABEO
LPC2194	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1/2/3/4 TX, CAN1/2/3/4 RX, FULLCAN	ABTO, ABEO

Table 33. LPC21xx/22xx part-specific interrupts

Part	SSP Registers: Table 51, Table 35	CAN	UART
LPC2210	TXRIS, RXRIS, RTRIS, RORRIS	-	ABTO, ABEO
LPC2212	TXRIS, RXRIS, RTRIS, RORRIS	-	ABTO, ABEO
LPC2214	TXRIS, RXRIS, RTRIS, RORRIS	-	ABTO, ABEO
LPC2290	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1/2 TX, CAN1/2 RX, FULLCAN	ABTO, ABEO
LPC2292	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1/2 TX, CAN1/2 RX, FULLCAN	ABTO, ABEO
LPC2294	TXRIS, RXRIS, RTRIS, RORRIS	CAN common, CAN1/2/3/4 TX, CAN1/2/3/4 RX, FULLCAN	ABTO, ABEO

5.2 Features

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

5.3 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 sources is are requesting an interrupt.

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

Non-vectored IRQs have the lowest priority.

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

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

Additional information on the Vectored Interrupt Controller is available in the ARMPrimeCell Vectored Interrupt Controller (PL190) documentation.

5.4 Register description

The VIC implements the registers shown in Table 34. More detailed descriptions follow.

Table 34. VIC register map

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

Table 34. VIC register map

Name	Description	Access	Reset value[1]	Address
VICVectAddr1	Vector address 1 register.	R/W	0	0xFFFF F104
VICVectAddr2	Vector address 2 register.	R/W	0	0xFFFF F108
VICVectAddr3	Vector address 3 register.	R/W	0	0xFFFF F10C
VICVectAddr4	Vector address 4 register.	R/W	0	0xFFFF F110
VICVectAddr5	Vector address 5 register.	R/W	0	0xFFFF F114
VICVectAddr6	Vector address 6 register.	R/W	0	0xFFFF F118
VICVectAddr7	Vector address 7 register.	R/W	0	0xFFFF F11C
VICVectAddr8	Vector address 8 register.	R/W	0	0xFFFF F120
VICVectAddr9	Vector address 9 register.	R/W	0	0xFFFF F124
VICVectAddr10	Vector address 10 register.	R/W	0	0xFFFF F128
VICVectAddr11	Vector address 11 register.	R/W	0	0xFFFF F12C
VICVectAddr12	Vector address 12 register.	R/W	0	0xFFFF F130
VICVectAddr13	Vector address 13 register.	R/W	0	0xFFFF F134
VICVectAddr14	Vector address 14 register.	R/W	0	0xFFFF F138
VICVectAddr15	Vector address 15 register.	R/W	0	0xFFFF F13C
VICVectCntl0	Vector control 0 register. Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest.	R/W	0	0xFFFF F200
VICVectCntl1	Vector control 1 register.	R/W	0	0xFFFF F204
VICVectCntl2	Vector control 2 register.	R/W	0	0xFFFF F208
VICVectCntl3	Vector control 3 register.	R/W	0	0xFFFF F20C
VICVectCntl4	Vector control 4 register.	R/W	0	0xFFFF F210
VICVectCntl5	Vector control 5 register.	R/W	0	0xFFFF F214
VICVectCntl6	Vector control 6 register.	R/W	0	0xFFFF F218
VICVectCntl7	Vector control 7 register.	R/W	0	0xFFFF F21C
VICVectCntl8	Vector control 8 register.	R/W	0	0xFFFF F220
VICVectCntl9	Vector control 9 register.	R/W	0	0xFFFF F224
VICVectCntl10	Vector control 10 register.	R/W	0	0xFFFF F228
VICVectCntl11	Vector control 11 register.	R/W	0	0xFFFF F22C
VICVectCntl12	Vector control 12 register.	R/W	0	0xFFFF F230
VICVectCntl13	Vector control 13 register.	R/W	0	0xFFFF F234
VICVectCntl14	Vector control 14 register.	R/W	0	0xFFFF F238
VICVectCntl15	Vector control 15 register.	R/W	0	0xFFFF F23C

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

5.5 VIC registers

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

5.5.1 Software Interrupt register (VICSoftInt - 0xFFFF F018)

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

Table 35. Software Interrupt Register (VICSoftInt - address 0xFFFF F018) bit allocation
Reset value: 0x0000 0000

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Symbol	-	-	CAN4 RX	CAN3 RX	CAN2 RX	CAN1 RX	FULLCAN	-
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	CAN4 TX	CAN3 TX	CAN2 TX	CAN1 TX	CAN Common	ADC	EINT3	EINT2
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	EINT1	EINT0	RTC	PLL	SPI1/SSP	SPI0	I2C	PW\M0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 36. Software Interrupt Register (VICSoftInt - address 0xFFFF F018) bit description

Bit	Symbol	Reset value	Value	Description
31-0	See VICSoftInt bit allocation table.	0	0	Do not force the interrupt request with this bit number. Writing zeroes to bits in VICSoftInt has no effect, see VICSoftIntClear (Section 5.5.2).
			1	Force the interrupt request with this bit number.

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

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

Table 37. Software Interrupt Clear Register (VICSoftIntClear - 0xFFFF F01C)

VICSoftIntClear	Description	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

Table 38. Software Interrupt Clear Register (VICSoftIntClear - address 0xFFFF F01C) bit allocation
Reset value: 0x0000 0000

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Symbol	-	-	CAN4 RX	CAN3 RX	CAN2 RX	CAN1 RX	FULLCAN	-
Access	WO	WO	WO	WO	WO	WO	WO	WO
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	CAN4 TX	CAN3 TX	CAN2 TX	CAN1 TX	CAN Common	ADC	EINT3	EINT2
Access	WO	WO	WO	WO	WO	WO	WO	WO
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	EINT1	EINT0	RTC	PLL	SPI1/SSP	SPI0	I2C	PWM
Access	WO	WO	WO	WO	WO	WO	WO	WO
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
Access	WO	WO	WO	WO	WO	WO	WO	WO

Table 39. Software Interrupt Clear Register (VICSoftIntClear - address 0xFFFF F01C) bit description

Bit	Symbol	Reset value	Value	Description
31-0	See VICSoftIntClear bit allocation table.	0	0	Writing a 0 leaves the corresponding bit in VICSoftInt unchanged.
			1	Writing a 1 clears the corresponding bit in the Software Interrupt register, thus releasing the forcing of this request.

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

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

Table 40. Raw Interrupt Status Register (VICRawIntr - address 0xFFFF F008) bit description

VICRawIntr	Description	Reset value
31:0	1:The hardware or software interrupt request with this bit number is asserted. 0: Neither the hardware nor software interrupt request with this bit number is asserted.	0

5.5.4 Interrupt Enable Register (VICIntEnable - 0xFFFF F010)

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

Table 41. Interrupt Enable Register (VICINtEnable - address 0xFFFF F010) bit description

VICIntEnable	Description	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 Section 5.5.5 “Interrupt Enable Clear Register (VICIntEnClear - 0xFFFF F014)” on page 48 and Table 42 below for how to disable interrupts.	0

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

This is a write only register. This register allows software to clear one or more bits in the Interrupt Enable register (Section 5.5.4), without having to first read it.

Table 42. Software Interrupt Clear Register (VICIntEnClear - address 0xFFFF F014) bit description

VICIntEnClear	Description	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

5.5.6 Interrupt Select Register (VICIntSelect - 0xFFFF F00C)

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

Table 43. Interrupt Select Register (VICIntSelect - address 0xFFFF F00C) bit description

VICIntSelect	Description	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

5.5.7 IRQ Status Register (VICIRQStatus - 0xFFFF F000)

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

Table 44. IRQ Status Register (VICIRQStatus - address 0xFFFF F000) bit description

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

5.5.8 FIQ Status Register (VICFIQStatus - 0xFFFF F004)

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

Table 45. FIQ Status Register (VICFIQStatus - address 0xFFFF F004) bit description

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

5.5.9 Vector Control registers 0-15 (VICvectCntl0-15 - 0xFFFF F200-23C)

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

Table 46. Vector Control registers (VICVectCntl0-15 - addresses 0xFFFF F200-23C) bit description

VICVectCntl0-15	Description	Reset value
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
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
31:6	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

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

Table 47. Vector Address registers (VICVectAddr0-15 - addresses 0xFFFF F100-13C) bit description

VICVectAddr0-15	Description	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 (Section 5.5.10).	0

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

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

Table 48. Default Vector Address register (VICDefVectAddr - address 0xFFFF F034) bit description

VICDefVectAddr	Description	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

5.5.12 Vector Address register (VICVectAddr - 0xFFFF F030)

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

Table 49. Vector Address register (VICVectAddr - address 0xFFFF F030) bit description

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

5.5.13 Protection Enable register (VICProtection - 0xFFFF F020)

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

Table 50. Protection Enable register (VICProtection - address 0xFFFF F020) bit description

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

5.6 Interrupt sources

Table 51 lists the interrupt sources for each peripheral function. Each peripheral device has one interrupt line connected to the Vectored Interrupt Controller, but may have several internal interrupt flags. Individual interrupt flags may also represent more than one interrupt source. See Table 33 for which flags are implemented for which parts.

Table 51. Connection of interrupt sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Hex	VIC Channel # and Mask
WDT	Watchdog Interrupt (WDINT)	0	0x0000 0001
-	Reserved for software interrupts only	1	0x0000 0002
ARM Core	Embedded ICE, DbgCommRx	2	0x0000 0004
ARM Core	Embedded ICE, DbgCommTX	3	0x0000 0008
TIMER0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4	0x0000 0010
TIMER1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5	0x0000 0020
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Auto-Baud Time-Out (ABTO) End of Auto-Baud (ABEO)	6	0x0000 0040
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI) Auto-Baud Time-Out (ABTO) End of Auto-Baud (ABEO)	7	0x0000 0080
PWM	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8	0x0000 0100
I^{2}C	SI (state change)	9	0x0000 0200
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10	0x0000 0400
SPI1 (SSP)	**Source: SPI1** SPI Interrupt Flag (SPIF) Mode Fault (MODF) **Source: SSP** TX FIFO at least half empty (TXRIS) Rx FIFO at least half full (RXRIS) Receive Timeout condition (RTRIS) Receive overrun (RORRIS)	11	0x0000 0800
PLL	PLL Lock (PLOCK)	12	0x0000 1000
RTC	Counter Increment (RTCCIF) Alarm (RTCALF)	13	0x0000 2000
System Control	External Interrupt 0 (EINT0)	14	0x0000 4000
	External Interrupt 1 (EINT1)	15	0x0000 8000
	External Interrupt 2 (EINT2)	16	0x0001 0000
	External Interrupt 3 (EINT3)	17	0x0002 0000
ADC	A/D Converter end of conversion	18	0x0004 0000

Table 51. Connection of interrupt sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Hex	VIC Channel # and Mask
CAN	CAN common and acceptance filter (1 ORed CAN, LUTerr)	19	0x0008 0000
CAN	CAN1 TX	20	0x0010 0000
CAN	CAN2 TX	21	0x0020 0000
CAN	CAN3 TX	22	0x0040 0000
CAN	CAN4 TX	23	0x0080 0000
Reserved		24	0x0100 0000
CAN	FULLCAN	25	0x0200 0000
CAN	CAN1 RX	26	0x0400 0000
CAN	CAN2 RX	27	0x0800 0000
CAN	CAN3 RX	28	0x1000 0000
CAN	CAN4 RX	29	0x2000 0000

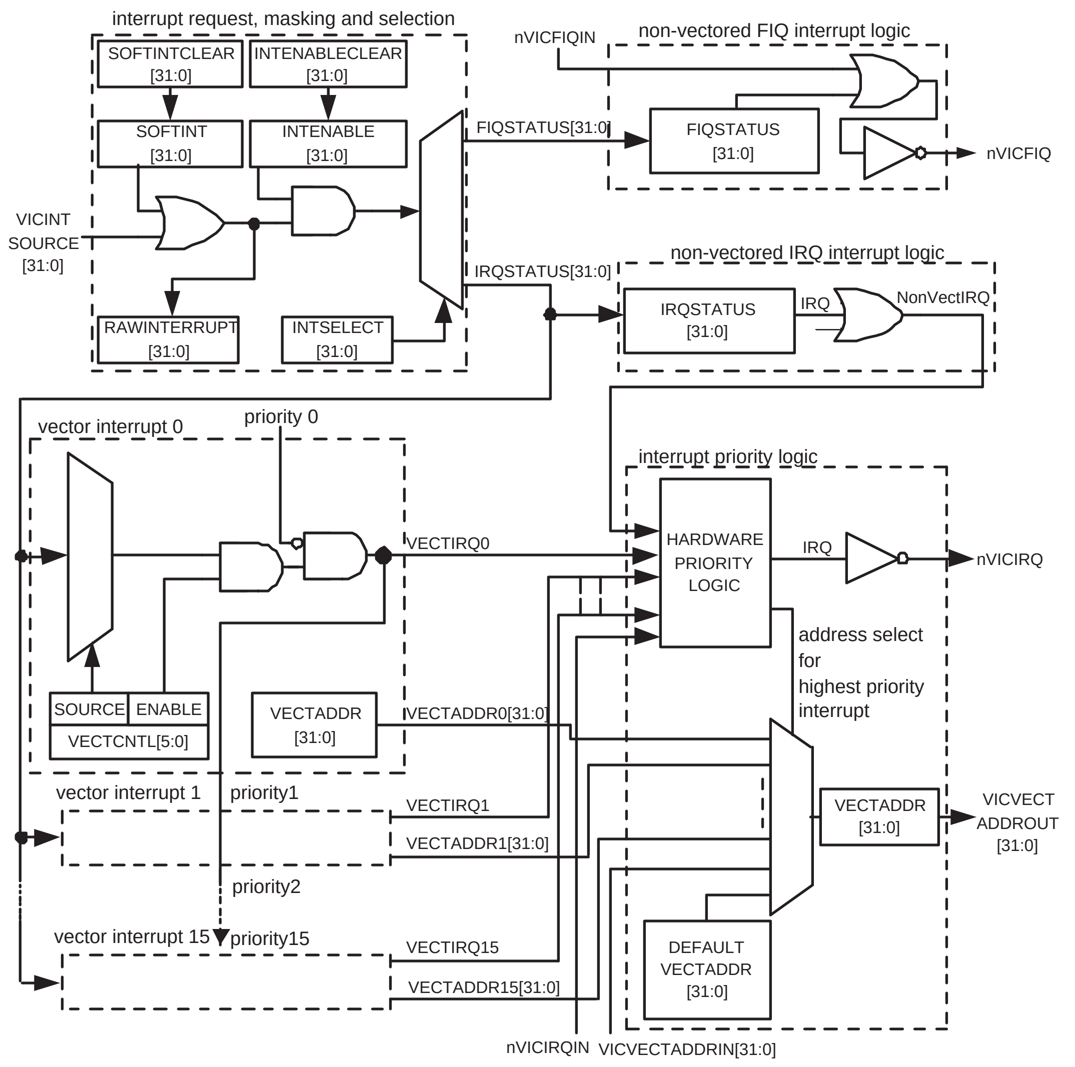


Fig 13. Block diagram of the Vectored Interrupt Controller

5.7 Spurious interrupts

Spurious interrupts are possible in the ARM7TDMI based microcontrollers such as the LPC21xx and LPC22xx due to asynchronous interrupt handling. The asynchronous character of the interrupt processing has its roots in the interaction of the core and the VIC. If the VIC state is changed between the moments when the core detects an interrupt, and the core actually processes an interrupt, problems may be generated.

Real-life applications may experience the following scenarios:

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

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

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

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

5.7.1 Details and case studies on spurious interrupts

This chapter contains details that can be obtained from the official ARM website , FAQ section under the "Technical Support":

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

Applies to: ARM7TDMI

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

For example, consider the following instruction sequence:

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

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

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

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

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

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

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

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

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

5.7.1.1 Workaround

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

5.7.1.1.1 Solution 1: Test for an IRQ received during a write to disable IRQs

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

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

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

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

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

5.7.1.1.2 Solution 2: Disable IRQs and FIQs using separate writes to the CPSR

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

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

5.7.1.1.3 Solution 3: Re-enable FIQs at the beginning of the IRQ handler

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

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

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

5.8 VIC usage notes

If user code is running from an on-chip RAM and an application uses interrupts, interrupt vectors must be re-mapped to on-chip address 0x0. This is necessary because all the exception vectors are located at addresses 0x0 and above. This is easily achieved by configuring the MEMMAP register (see Table 20) to User RAM mode. Application code should be linked such that at 0x4000 0000 the Interrupt Vector Table (IVT) will reside.

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

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

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

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

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

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

After any of IRQ requests (SPI0, I^{2}C, UART0 or UART1) is made, microcontroller will redirect code execution to the address specified at location 0x0000 0018. For vectored and non-vectored IRQ's the following instruction could be placed at 0x0000 0018:

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

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

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

6.1 How to read this chapter

Remark: The LPC21xx and LPC22xx have different features and peripherals enabled depending on part number and version. Refer to Table 52 for registers that need to be configured for each specific part and peripheral.

The following register descriptions include all LPC21xx and LPC22xx parts. Registers not listed in Table 52 are identical for all parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

Table 52. LPC21xx/22xx part-specific register bits

	Power control for peripherals	**Hi-Speed GPIO**	**Peripheral Clock**	**Memory mapping modes**
	PCONP bit, Table 74	**SCS bit, Table 61**	**APBDIV bit, Table 76**	**MEMMAP mode, Table 62**
no suffix and /00 parts				
LPC2109	all[1] + PCCAN1	n/a	APBDIV	Flash/ROM/RAM
LPC2119	all[1] + PCCAN1/2	n/a	APBDIV	Flash/ROM/RAM
LPC2129	all[1] + PCCAN1/2	n/a	APBDIV	Flash/ROM/RAM
LPC2114	all common peripherals[1]	n/a	APBDIV	Flash/ROM/RAM
LPC2124	all common peripherals[1]	n/a	APBDIV	Flash/ROM/RAM
LPC2194	all[1]+ PCCAN1/2/3/4	n/a	APBDIV	Flash/ROM/RAM
LPC2210	all[1] + PCEMC	n/a	APBDIV/XCLK	ROM/RAM/EMC
LPC2220	all[1] + PCEMC, PCSSP[2]	GPIO0/1M	APBDIV/XCLK	ROM/RAM/EMC
LPC2212	all[1] + PCEMC	n/a	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2214	all[1] + PCEMC	n/a	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2290	all[1] + PCEMC, PCAN1/2	n/a	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2292	all[1] + PCEMC, PCAN1/2	n/a	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2294	all[1] + PCEMC, PCCAN1/2/3/4	n/a	APBDIV/XCLK	Flash/ROM/RAM/EMC
/01 parts				
LPC2109	all[1] + PCSSP, PCCAN1	GPIO0/1M	APBDIV	Flash/ROM/RAM
LPC2119	all[1] + PCSSP, PCCAN1/2	GPIO0/1M	APBDIV	Flash/ROM/RAM
LPC2129	all[1] + PCSSP, PCCAN1/2	GPIO0/1M	APBDIV	Flash/ROM/RAM
LPC2114	all[1] + PCSSP	GPIO0/1M	APBDIV	Flash/ROM/RAM
LPC2124	all[1] + PCSSP	GPIO0/1M	APBDIV	Flash/ROM/RAM
LPC2194	all[1] + PCSSP, PCCAN1/2/3/4	GPIO0/1M	APBDIV	Flash/ROM/RAM
LPC2210	all[1] + PCEMC, PCSSP[2]	GPIO0/1M	APBDIV/XCLK	ROM/RAM/EMC
LPC2212	all[1] + PCEMC, PCSSP[2]	GPIO0/1M	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2214	all[1] + PCEMC, PCSSP[2]	GPIO0/1M	APBDIV/XCLK	Flash/ROM/RAM/EMC

Table 52. LPC21xx/22xx part-specific register bits

	Power control for peripherals	Hi-Speed GPIO	Peripheral Clock	Memory mapping modes
	PCONP bit, Table 74	**SCS bit, Table 61**	**APBDIV bit, Table 76**	**MEMMAP mode, Table 62**
LPC2290	all[1] + PCEMC, PCSSP[2], PCCAN1/2	GPIO0/1M	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2292	all[1] + PCEMC, PCSSP[2], PCCAN1/2	GPIO0/1M	APBDIV/XCLK	Flash/ROM/RAM/EMC
LPC2294	all[1] + PCEMC, PCSSP[2], PCCAN1/2/3/4	GPIO0/1M	APBDIV/XCLK	Flash/ROM/RAM/EMC

[1] The PCONP bits common to all parts are: PCTIM0/1, PCUART0/1, PCI2C, PCSPI0/1, PCRTC, PCAD.

[2] Use the PCSSP bit to configure the SPI1 interface as SSP interface.

6.2 Summary of system control block functions

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

- Crystal Oscillator
- External Interrupt Inputs
- Miscellaneous System Controls and Status
- Memory Mapping Control
- PLL
- Power Control
- Reset
- APB Divider
- Wake-up Timer

Each type of function has its own registers 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

6.3 Pin description

Table 53 shows pins that are associated with System Control block functions.

Table 53. Pin summary

Pin name	Pin direction	Pin description
XTAL1	Input	**Crystal Oscillator Input** - Input to the oscillator and internal clock generator circuits
XTAL2	Output	**Crystal Oscillator Output** - Output from the oscillator amplifier
EINT0	Input	**External Interrupt Input 0** - An active low/high level or falling/rising edge general purpose interrupt input. This pin may be used to wake up the processor from Idle or Power-down modes. Pins P0.1 and P0.16 can be selected to perform EINT0 function.

Table 53. Pin summary

Pin name	Pin direction	Pin description
EINT1	Input	**External Interrupt Input 1** - See the EINT0 description above. Pins P0.3 and P0.14 can be selected to perform EINT1 function. **Important:** LOW level on pin P0.14 immediately after reset is considered as an external hardware request to start the ISP command handler. More details on ISP and Serial Boot Loader can be found in Section 21.5 on page 311.
EINT2	Input	**External Interrupt Input 2** - See the EINT0 description above. Pins P0.7 and P0.15 can be selected to perform EINT2 function.
EINT3	Input	**External Interrupt Input 3** - See the EINT0 description above. Pins P0.9, P0.20 and P0.30 can be selected to perform EINT3 function.
$\overline{\text{RESET}}$	Input	**External Reset input** - A LOW on this pin resets the chip, causing I/O ports and peripherals to take on their default states and the processor to begin execution at address 0x0000 0000.

6.4 Register description

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

Table 54. Summary of system control registers

Name	Description	Access	Reset value[1]	Address
External Interrupts				
EXTINT	External Interrupt Flag Register	R/W	0	0xE01F C140
EXTWAKE	External Interrupt Wake-up Register	R/W	0	0xE01F C144
EXTMODE	External Interrupt Mode Register	R/W	0	0xE01F C148
EXTPOLAR	External Interrupt Polarity Register	R/W	0	0xE01F C14C
Memory Mapping Control				
MEMMAP	Memory Mapping Control	R/W	0	0xE01F C040
Phase Locked Loop				
PLLCON	PLL Control Register	R/W	0	0xE01F C080
PLLCFG	PLL Configuration Register	R/W	0	0xE01F C084
PLLSTAT	PLL Status Register	RO	0	0xE01F C088
PLLFEED	PLL Feed Register	WO	NA	0xE01F C08C
Power Control				
PCON	Power Control Register	R/W	0	0xE01F C0C0
PCONP	Power Control for Peripherals	R/W	0x1FBE	0xE01F C0C4
APB Divider				
APBDIV	APB Divider Control	R/W	0	0xE01F C100
Syscon Miscellaneous Registers				
SCS	System Controls and Status	R/W	0	0xE01F C1A0

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

6.5 Crystal oscillator

An input signal of 50-50 duty cycle within a frequency range from 1 MHz to 25 MHz should be used by the LPC21xx/22xx if supplied to its input XTAL1 pin. This microcontroller's onboard oscillator circuit supports external crystals in the range of 1 MHz to 25 MHz only. If the on-chip PLL system or the boot-loader is used, the input clock frequency is limited to an exclusive range of 10 MHz to 25 MHz.

The oscillator output frequency is called F_{OSC}, and the ARM processor clock frequency is referred to as CCLK for purposes of rate equations, etc. elsewhere in this document. F_{OSC} and CCLK are the same value unless the PLL is running and connected. Refer to the Section 6.9 "Phase Locked Loop (PLL)" on page 70 for details and frequency limitations.

The onboard oscillator in the LPC21xx/LPC22xx can operate in one of two modes: slave mode and oscillation mode.

In slave mode the input clock signal should be coupled by means of a capacitor of 100 pF (C_C in Figure 14, drawing a), with an amplitude of at least 200 mVrms. The XTAL2 pin in this configuration can be left not connected. If slave mode is selected, the F_{OSC} signal of 50-50 duty cycle can range from 1 MHz to 25 MHz.

External components and models used in oscillation mode are shown in Figure 14, drawings b and c, and in Table 55. Since the feedback resistance is integrated on chip, only a crystal and the capacitances C_{X1} and C_{X2} need to be connected externally in case of fundamental mode oscillation (the fundamental frequency is represented by L, C_L and R_S). Capacitance C_P in Figure 14, drawing c, represents the parallel package capacitance and should not be larger than 7 pF. Parameters F_C, C_L, R_S and C_P are supplied by the crystal manufacturer.

Choosing the oscillation mode as an on-board oscillator mode of operation, limits F_{OSC} clock selection to 1 MHz to 25 MHz.

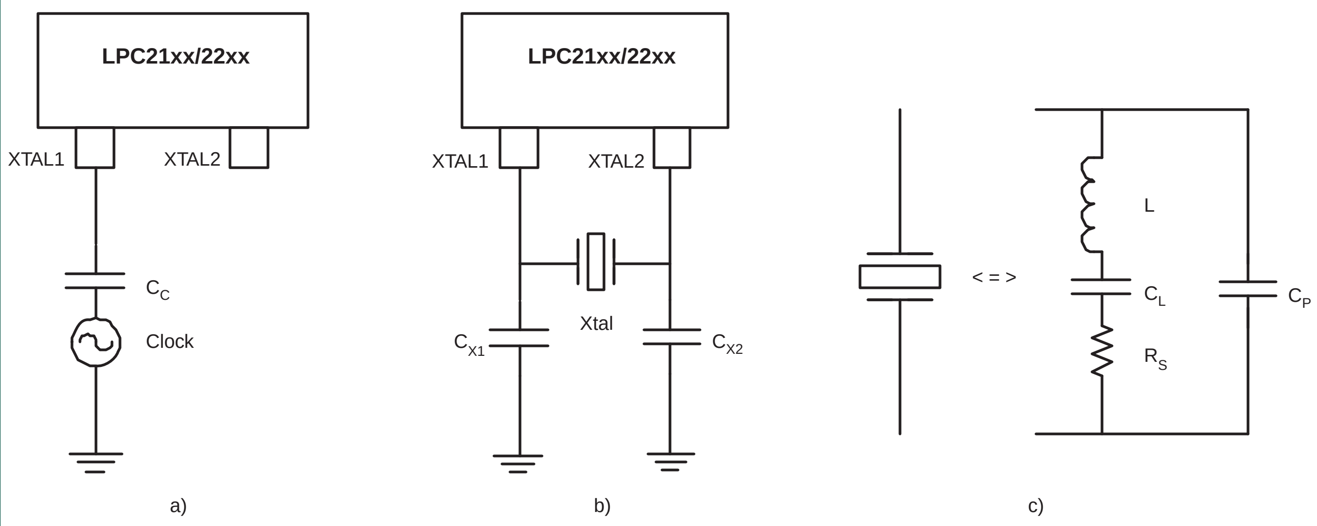


Fig 14. Oscillator modes and models: a) slave mode of operation, b) oscillation mode of operation, c) external crystal model used for $C_{X1}/_{X2}$ evaluation

Table 55. Recommended values for $C_{X1/X2}$ in oscillation mode (crystal and external components parameters)

Fundamental oscillation frequency F_{OSC}	Crystal load capacitance C_L	Maximum crystal series resistance R_S	External load capacitors C_{X1}, C_{X2}
1 MHz - 5 MHz	10 pF	NA	NA
	20 pF	NA	NA
	30 pF	< 300 Ω	58 pF, 58 pF
5 MHz - 10 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 300 Ω	38 pF, 38 pF
	30 pF	< 300 Ω	58 pF, 58 pF
10 MHz - 15 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 220 Ω	38 pF, 38 pF
	30 pF	< 140 Ω	58 pF, 58 pF
15 MHz - 20 MHz	10 pF	< 220 Ω	18 pF, 18 pF
	20 pF	< 140 Ω	38 pF, 38 pF
	30 pF	< 80 Ω	58 pF, 58 pF
20 MHz - 25 MHz	10 pF	< 160 Ω	18 pF, 18 pF
	20 pF	< 90 Ω	38 pF, 38 pF
	30 pF	< 50 Ω	58 pF, 58 pF

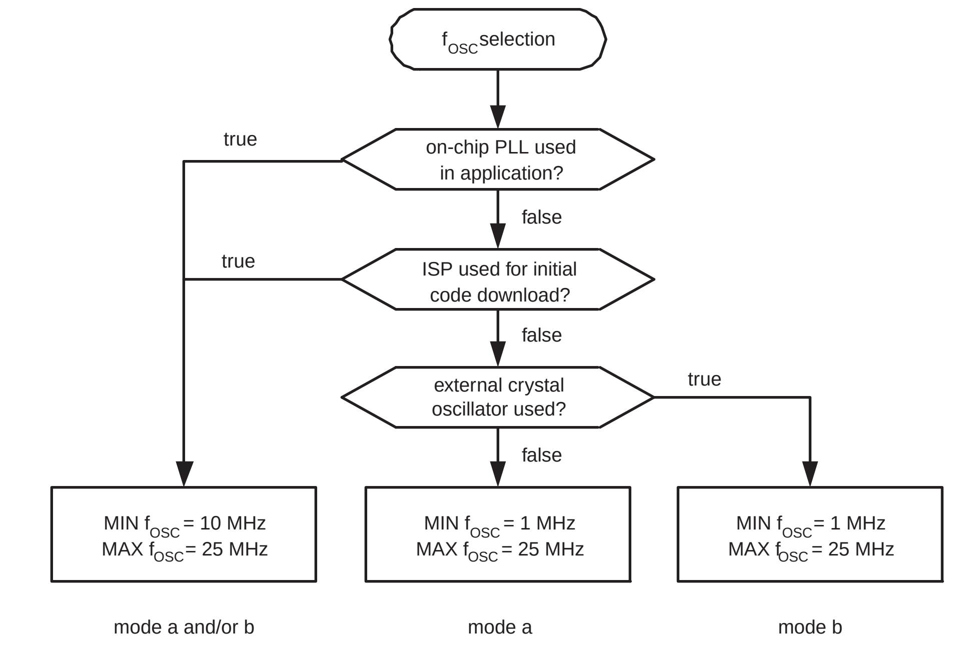

f_OSC selection
on-chip PLL used in application?
true
false
ISP used for initial code download?
true
false
external crystal oscillator used?
true
false
MIN f_OSC = 10 MHz MAX f_OSC = 25 MHz
MIN f_OSC = 1 MHz MAX f_OSC = 25 MHz
MIN f_OSC = 1 MHz MAX f_OSC = 25 MHz
mode a and/or b
mode a
mode b

Fig 15. F_{OSC} selection algorithm

6.6 External interrupt inputs

The LLPC21xx/LPC22xx includes four external interrupt inputs as selectable pin functions. The external interrupt inputs can optionally be used to wake up the processor from Power-down mode.

6.6.1 Register description

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

Table 56. External interrupt registers

Name	Description	Access	Reset value[1]	Address
EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, EINT2 and EINT3. See Table 57.	R/W	0	0xE01F C140
EXTWAKE	The External Interrupt Wake-up Register contains four enable bits that control whether each external interrupt will cause the processor to wake up from Power-down mode. See Table 58.	R/W	0	0xE01F C144
EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level sensitive.	R/W	0	0xE01F C148
EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt.	R/W	0	0xE01F C14C

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

6.6.2 External Interrupt Flag register (EXTINT - 0xE01F C140)

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

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

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

Important: whenever a change of external interrupt operating mode (i.e. active level/edge) is performed (including the initialization of an external interrupt), the corresponding bit in the EXTINT register must be cleared! For details see Section 6.6.4 “External Interrupt Mode register (EXTMODE - 0xE01F C148)” **and** Section 6.6.5 “External Interrupt Polarity register (EXTPOLAR - 0xE01F C14C)”.

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

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

Table 57. External Interrupt Flag register (EXTINT - address 0xE01F C140) bit description

Bit	Symbol	Description	Reset value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. Up to two pins can be selected to perform the EINT0 function (see P0.1 and P0.16 description in Section 7.2 and Section 7.3). This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT0 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. Up to two pins can be selected to perform the EINT1 function (see P0.3 and P0.14 description in Section 7.2 and Section 7.3). This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT1 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).	0
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. Up to two pins can be selected to perform the EINT2 function (see P0.7 and P0.15 description in Section 7.2 and Section 7.3). This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT2 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. Up to three pins can be selected to perform the EINT3 function (see P0.9, P0.20 and P0.30 description in Section 7.2 and Section 7.3). This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state (e.g. if EINT3 is selected to be low level sensitive and a low level is present on the corresponding pin, this bit can not be cleared; this bit can be cleared only when the signal on the pin becomes high).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.6.3 External interrupt Wake-up register (EXTWAKE - 0xE01F C144)

Enable bits in the EXTWAKE register allow the external interrupts and other sources to wake up the processor if it is in Power-down mode. The related EINTn function must be mapped to the pin in order for the wake-up process to take place. It is not necessary for the interrupt to be enabled in the Vectored Interrupt Controller for a wake-up 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 wake-up feature is not desirable in the application).

For an external interrupt pin to be a source that would wake up the microcontroller from Power-down mode, it is also necessary to clear the corresponding bit in the External Interrupt Flag register (Section 6.6.2 on page 63).

Table 58. Interrupt Wakeup register (INTWAKE - address 0xE01F C144) bit description

Bit	Symbol	Description	Reset value
0	EXTWAKE0	When one, assertion of EINT0 will wake up the processor from Power-down mode.	0
1	EXTWAKE1	When one, assertion of EINT1 will wake up the processor from Power-down mode.	0
2	EXTWAKE2	When one, assertion of EINT2 will wake up the processor from Power-down mode.	0
3	EXTWAKE3	When one, assertion of EINT3 will wake up the processor from Power-down mode.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

The bits in this register select whether each EINT pin is level- or edge-sensitive. Only pins that are selected for the EINT function (see Section 8.6) and enabled via the VICIntEnable register (Section 5.5.4 "Interrupt Enable Register (VICIntEnable - 0xFFFF F010)" on page 47) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

Table 59. External Interrupt Mode register (EXTMODE - address 0xE01F C148) bit description

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0	0	Level-sensitivity is selected for EINT0.	0
		1	EINT0 is edge sensitive.	
1	EXTMODE1	0	Level-sensitivity is selected for EINT1.	0
		1	EINT1 is edge sensitive.	
2	EXTMODE2	0	Level-sensitivity is selected for EINT2.	0

Table 59. External Interrupt Mode register (EXTMODE - address 0xE01F C148) bit description

Bit	Symbol	Value	Description	Reset value
		1	EINT2 is edge sensitive.	
3	EXTMODE3	0	Level-sensitivity is selected for EINT3.	0
		1	EINT3 is edge sensitive.	
7:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

In level-sensitive mode, the bits in this register select whether the corresponding pin is high- or low-active. In edge-sensitive mode, they select whether the pin is rising- or falling-edge sensitive. Only pins that are selected for the EINT function (see Section 8.6) and enabled in the VICIntEnable register (Section 5.5.4 “Interrupt Enable Register (VICIntEnable - 0xFFFF F010)” on page 47) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

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

Table 60. External Interrupt Polarity register (EXTPOLAR - address 0xE01F C14C) bit description

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

6.6.6 Multiple external interrupt pins

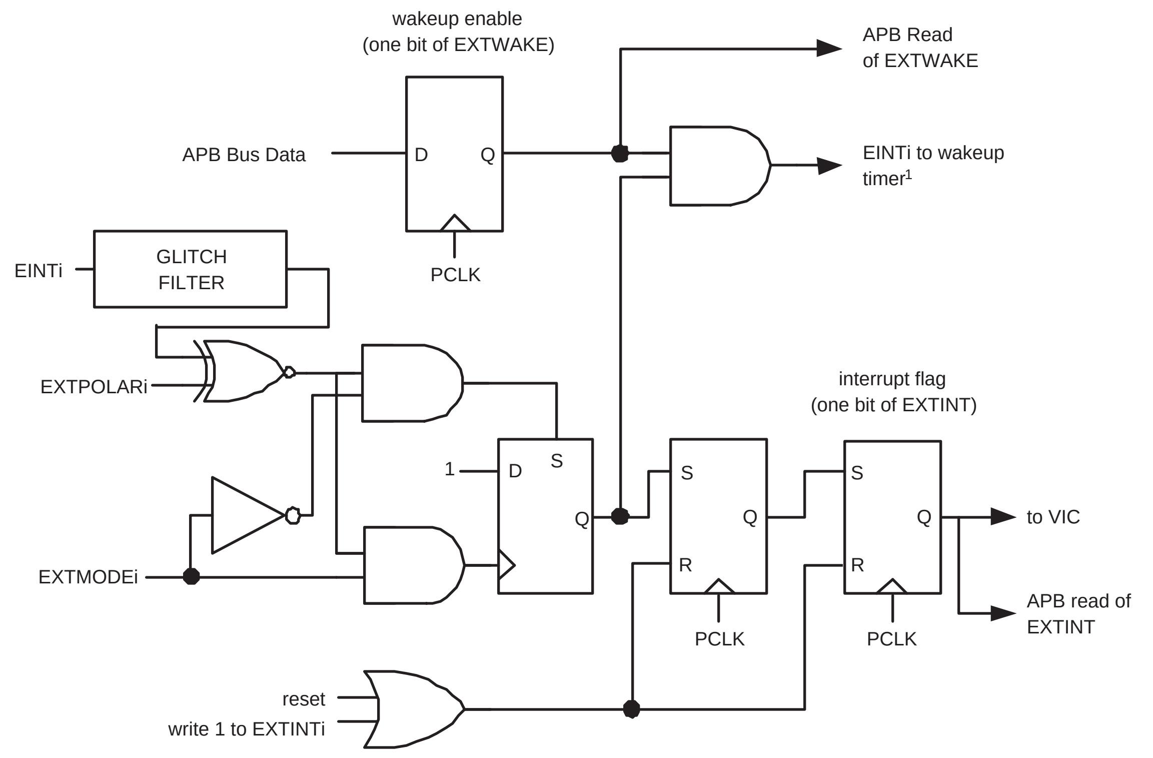
Software can select multiple pins for each of EINT3:0 in the Pin Select registers, which are described in Section 8.6. The external interrupt logic for each of EINT3:0 receives the state of all of its associated pins from the pins' receivers, along with signals that indicate whether each pin is selected for the EINT function.

The external interrupt logic handles the case when more than one pin is selected for a particular interrupt, depending on how the interrupt's mode and polarity bits are set:

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

The signal derived by this logic processing multiple external interrupt pins is the "EINTi to wake-up timer" signal in the following logic schematic Figure 16.

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



(1) See Figure 19.

Fig 16. External interrupt logic

6.7 Other system controls

Some aspects of controlling LPC21xx/LPC22xx operation that do not fit into peripheral or other registers are grouped here.

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

Table 61. System Control and Status flags register (SCS - address 0xE01F C1A0) bit description

Bit	Symbol	Value	Description	Reset value
0	GPIO0M		GPIO port 0 mode selection.	0
		0	GPIO port 0 is accessed via APB addresses in a fashion compatible with previous LCP2000 devices.	
		1	High speed GPIO is enabled on GPIO port 0, accessed via addresses in the on-chip memory range. This mode includes the port masking feature described in Section 9.5.5 “Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)”	

Table 61. System Control and Status flags register (SCS - address 0xE01F C1A0) bit description

Bit	Symbol	Value	Description	Reset value
1	GPIO1M		GPIO port 1 mode selection.	0
		0	GPIO port 1 is accessed via APB addresses in a fashion compatible with previous LCP2000 devices.	
		1	High speed GPIO is enabled on GPIO port 1, accessed via addresses in the on-chip memory range. This mode includes the port masking feature described in Section 9.5.5 "Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)"	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.8 Memory mapping control

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

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

Whenever an exception handling is necessary, the microcontroller will fetch an instruction residing on the exception corresponding address as described in Table 19 "ARM exception vector locations" on page 23. The MEMMAP register determines the source of data that will fill this table.

Table 62. Memory Mapping control register (MEMMAP - address 0xE01F C040) bit description

Bit	Symbol	Value	Description	Reset value
1:0	MAP	00	Boot Loader Mode. Interrupt vectors are re-mapped to Boot Block.	00[1]
		01	User flash mode. Interrupt vectors are not re-mapped and reside in Flash memory	
		10	User RAM Mode. Interrupt vectors are re-mapped to Static RAM.	
		11	User External memory Mode. Interrupt vectors are re-mapped to external memory. **Remark:** This mode is available in 144-pin parts with external memory controller only. This value is reserved for parts without external memory controller, and user software should not write ones to reserved bits.	
		Warning: Improper setting of this value may result in incorrect operation of the device.		
7:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] The hardware reset value of the MAP1:0 bits is 00 for LPC21xx/LPC22xx parts. The apparent reset value visible to the user is different because it is altered by the Boot Loader code, which always runs initially at reset.

6.8.2 Memory mapping control usage notes

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

For example, whenever a Software Interrupt request is generated, the ARM core will always fetch 32-bit data residing on 0x0000 0008 see Table 19 “ARM exception vector locations” on page 23. This means that when MEMMAP[1:0]=10 (User RAM Mode), a read/fetch from 0x0000 0008 will provide data stored in 0x4000 0008. In case of MEMMAP[1:0]=00 (Boot Loader Mode), a read/fetch from 0x0000 0008 will provide data available also at 0x7FFF E008 (Boot Block remapped from on-chip Bootloader). MEMMAP[1:1]=11 (User External Memory Mode) will result in fetching data from off-chip memory at location 0x8000 0008.

6.9 Phase Locked Loop (PLL)

The PLL accepts an input clock frequency in the range of 10 MHz to 25 MHz only. The input frequency is multiplied up the range of 10 MHz to 75 MHz for the CCLK clock using a Current Controlled Oscillators (CCO). The multiplier can be an integer value from 1 to 32 (in practice, the multiplier value cannot be higher than 7 on the LPC21xx/LPC22xx 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 17.

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

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

6.9.1 Register description

The PLL is controlled by the registers shown in Table 63. More detailed descriptions follow.

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

Table 63. PLL registers

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

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

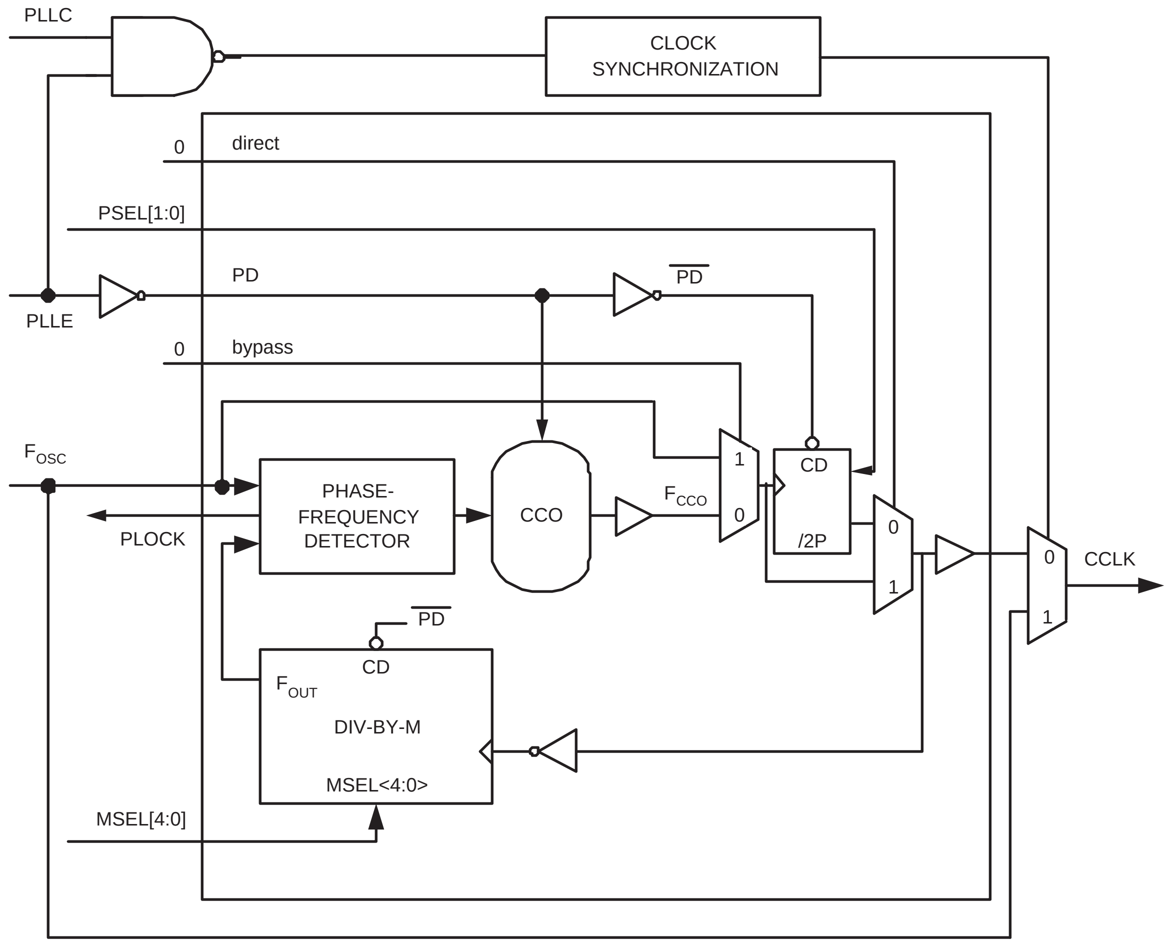


Fig 17. PLL block diagram

6.9.2 PLL Control register (PLLCON - 0xE01F C080)

The PLLCON register contains the bits that enable and connect the PLL. Enabling the PLL allows it to attempt to lock to the current settings of the multiplier and divider values. Connecting the PLL causes the processor and all chip functions to run from the PLL output clock. Changes to the PLLCON register do not take effect until a correct PLL feed sequence has been given (see Section 6.9.7 “PLL Feed register (PLLFEED - 0xE01F C08C)” and Section 6.9.3 “PLL Configuration register (PLLCFG - 0xE01F C084)” on page 72).

Table 64. PLL Control register (PLLCON - address 0xE01F C080) bit description

Bit	Symbol	Description	Reset value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 66.	0
1	PLLC	PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the microcontroller. Otherwise, the oscillator clock is used directly by the microcontroller. See PLLSTAT register, Table 66.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

6.9.3 PLL Configuration register (PLLCFG - 0xE01F C084)

The PLLCFG register contains the PLL multiplier and divider values. Changes to the PLLCFG register do not take effect until a correct PLL feed sequence has been given (see Section 6.9.7 “PLL Feed register (PLLFEED - 0xE01F C08C)” on page 74). Calculations for the PLL frequency, and multiplier and divider values are found in the PLL Frequency Calculation section on page 74.

Table 65. PLL Configuration register (PLLCFG - address 0xE01F C084) bit description

Bit	Symbol	Description	Reset value
4:0	MSEL	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations. **Note:** For details on selecting the right value for MSEL see Section 6.9.9 “PLL frequency calculation” on page 74.	0
6:5	PSEL	PLL Divider value. Supplies the value "P" in the PLL frequency calculations. **Note:** For details on selecting the right value for PSEL see Section 6.9.9 “PLL frequency calculation” on page 74.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.9.4 PLL Status register (PLLSTAT - 0xE01F C088)

The read-only PLLSTAT register provides the actual PLL parameters that are in effect at the time it is read, as well as the PLL status. PLLSTAT may disagree with values found in PLLCON and PLLCFG because changes to those registers do not take effect until a proper PLL feed has occurred (see Section 6.9.7 “PLL Feed register (PLLFEED - 0xE01F C08C)”).

Table 66. PLL Status register (PLLSTAT - address 0xE01F C088) bit description

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

6.9.5 PLL Interrupt

The PLOCK bit in the PLLSTAT register is connected to the interrupt controller. This allows for software to turn on the PLL and continue with other functions without having to wait for the PLL to achieve lock. When the interrupt occurs (PLOCK = 1), the PLL may be connected, and the interrupt disabled. For details on how to enable and disable the PLL interrupt, see Section 5.5.4 “Interrupt Enable Register (VICIntEnable - 0xFFFF F010)” on page 47 and Section 5.5.5 “Interrupt Enable Clear Register (VICIntEnClear - 0xFFFF F014)” on page 48.

6.9.6 PLL Modes

The combinations of PLLE and PLLC are shown in Table 67.

Table 67. PLL Control bit combinations

PLLC	PLLE	PLL Function
0	0	PLL is turned off and disconnected. The CCLK equals (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 00 combination. This prevents the possibility of the PLL being connected without also being enabled.
1	1	The PLL is active and has been connected as the system clock source. CCLK/system clock equals the PLL output.

6.9.7 PLL Feed register (PLLFEED - 0xE01F C08C)

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

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

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

Table 68. PLL Feed register (PLLFEED - address 0xE01F C08C) bit description

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

6.9.8 PLL and Power-down mode

Power-down mode automatically turns off and disconnects activated 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.

6.9.9 PLL frequency calculation

The PLL equations use the following parameters:

Table 69. Elements determining PLL's frequency

Element	Description
F_{OSC}	the frequency from the crystal oscillator/external oscillator
F_{CCO}	the frequency of the PLL current controlled oscillator

Table 69. Elements determining PLL’s frequency

Element	Description
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:

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

The CCO frequency can be computed as:

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

The PLL inputs and settings must meet the following:

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

6.9.10 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 Section 6.12 “APB divider” on page 81).
2. Choose an oscillator frequency (F_{OSC}). CCLK must be the whole (non-fractional) multiple of F_{OSC}.
3. Calculate the value of M to configure the MSEL bits. $M = CCLK / F_{OSC}$. M must be in the range of 1 to 32. The value written to the MSEL bits in PLLCFG is $M – 1$ (see Table 71.
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 70).

Table 70. PLL Divider values

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

Table 71. PLL Multiplier values

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

6.9.11 PLL configuring examples

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

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

Value for P can be derived from P = F_{CCO} / (CCLK x 2), using condition that F_{CCO} must be in range of 156 MHz to 320 MHz. Assuming the lowest allowed frequency for F_{CCO} = 156 MHz, P = 156 MHz / (2 x 60 MHz) = 1.3. The highest F_{CCO} frequency criteria produces P = 2.67. The only solution for P that satisfies both of these requirements and is listed in Table 70 is P = 2. Therefore, PLLCFG[6:5] = 1 will be used.

6.10 Power control

The LPC21xx/LPC22xx supports two reduced power modes: Idle mode and Power-down mode. In Idle mode, execution of instructions is suspended until either a reset or interrupt occurs. Peripheral functions continue operation during Idle mode and may generate interrupts to cause the processor to resume execution. Idle mode eliminates power used by the processor itself, memory systems and related controllers, and internal buses.

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

Entry to Power-down and Idle modes must be coordinated with program execution. Wakeup from Power-down or Idle modes via an interrupt resumes program execution in such a way that no instructions are lost, incomplete, or repeated. Wake up from Power-down mode is discussed further in Section 6.13 “Wakeup timer” on page 83.

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.

6.10.1 Register description

The Power Control function contains two registers, as shown in Table 72. More detailed descriptions follow.

Table 72. Power control registers

Name	Description	Access	Reset value[1]	Address
PCON	Power Control Register. This register contains control bits that enable the two reduced power operating modes of the microcontroller. See Table 73.	R/W	0x00	0xE01F C0C0
PCONP	Power Control for Peripherals Register. This register contains control bits that enable and disable individual peripheral functions, Allowing elimination of power consumption by peripherals that are not needed.	R/W	0x0000 1FBE	0xE01F C0C4

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

6.10.2 Power Control register (PCON - 0xE01F COCO)

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

Table 73. Power Control register (PCON - address 0xE01F COCO) bit description

Bit	Symbol	Description	Reset value
0	IDL	Idle mode - when 1, this bit causes the processor clock to be stopped, while on-chip peripherals remain active. Any enabled interrupt from a peripheral or an external interrupt source will cause the processor to resume execution.	0
1	PD	Power-down mode - when 1, this bit causes the oscillator and all on-chip clocks to be stopped. A wakeup condition from an external interrupt can cause the oscillator to restart, the PD bit to be cleared, and the processor to resume execution.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

The PCONP register allows turning off selected peripheral functions for the purpose of saving power. This is accomplished by gating off the clock source to the specified peripheral blocks. A few peripheral functions cannot be turned off (i.e. the Watchdog timer, GPIO, the Pin Connect block, and the System Control block). Some peripherals, particularly those that include analog functions, may consume power that is not clock dependent. These peripherals may contain a separate disable control that turns off additional circuitry to reduce power. Each bit in PCONP controls one of the peripherals. The bit numbers correspond to the related peripheral number as shown in the APB peripheral map Table 18 “APB peripheries and base addresses”.

If a peripheral control bit is 1, that peripheral is enabled. If a peripheral bit is 0, that peripheral is disabled to conserve power. For example, if bit 7 is 1, the I^{2}C interface is enabled. If bit 7 is 0, the I^{2}C1 interface is disabled.

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

Table 74. Power Control for Peripherals register (PCONP - address 0xE01F C0C4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	PCPWM0	PWM0 power/clock control bit.	1
6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	PCI2C	The I^{2}C interface power/clock control bit.	1
8	PCSPI0	The SPI0 interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSPI1	The SPI1 interface power/clock control bit.	1
11	PCEMC	The EMC power/clock control bit.	1
12	PCAD	A/D Converter (ADC) power/clock control bit. **Note:** Clear the PDN bit in the ADCR before clearing this bit, and set this bit before setting PDN.	1
13	PCCAN1	CAN1 controller bit.	1
14	PCCAN2	CAN2 controller bit.	1
15	PCCAN3	CAN3 controller bit.	1
16	PCCAN4	CAN4 controller bit.	1
20:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PCSSP	The SSP interface power/clock control bit **Remark:** Setting this bit to 1 and bit 10 (PSPI1) to 0, selects the SPI1 interface as SSP interface. At reset, SPI1 is enabled. See Section 14.3 on page 218.	0
31:22	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

6.10.4 Power control usage notes

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

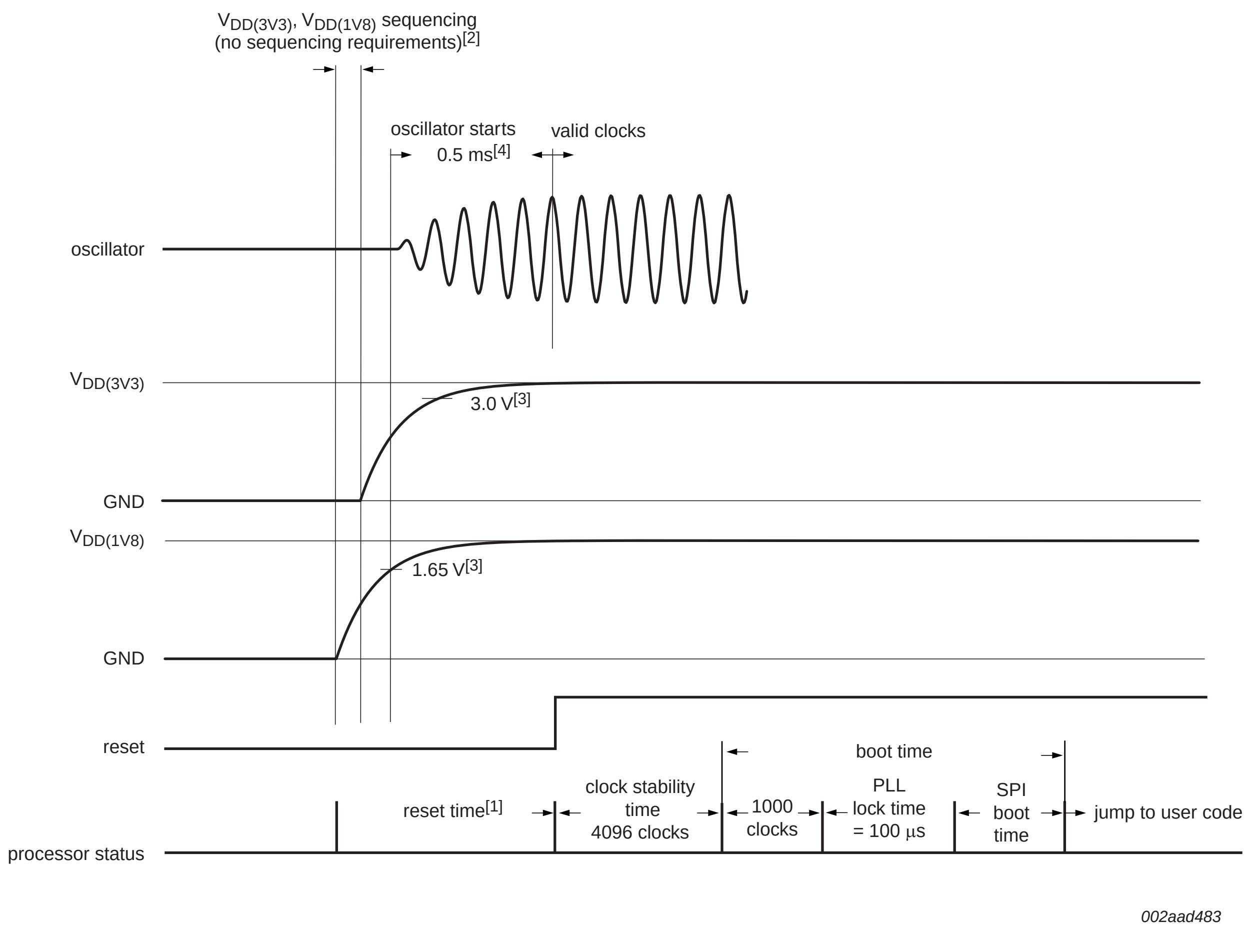
Power saving oriented systems should have 1s in the PCONP register only in positions that match peripherals really used in the application. All other bits, declared to be Reserved or dedicated to the peripherals not used in the current application, must be cleared to 0.

6.11 Reset

Reset has two sources on the LPC21xx/LPC22xx: the $\overline{\text{RESET}}$ pin and Watchdog reset. The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin with an additional glitch filter. Assertion of chip reset by any source starts the wakeup timer (see description in Section 6.13 “Wakeup timer” in this chapter), causing reset to remain asserted until the external reset is de-asserted, the oscillator is running, a fixed number of clocks have passed, and the on-chip circuitry has completed its initialization. The relationship between reset, the oscillator, and the wakeup timer during the startup sequence are shown in Figure 18. See Figure 19 for a block diagram of the Reset logic.

The reset glitch filter allows the processor to ignore external reset pulses that are very short, and also determines the minimum duration of $\overline{\text{RESET}}$ that must be asserted in order to guarantee a chip reset. Once asserted, $\overline{\text{RESET}}$ pin can be deasserted only when crystal oscillator is fully running and an adequate signal is present on the XTAL1 pin of the microcontroller. Assuming that an external crystal is used in the crystal oscillator subsystem, after power on, the $\overline{\text{RESET}}$ pin should be asserted for 10 ms. For all subsequent resets, when the crystal oscillator is already running and a stable signal is on the XTAL1 pin, the $\overline{\text{RESET}}$ pin needs to be asserted for 300 ns only.

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



(1) Reset time: The time reset needs to be held LOW. This time depends on system parameters such as $V_{DD(1V8)}$, V_{3V3} rise time, and the oscillator startup time. There are no restrictions from the microcontroller except that $V_{DD(1V8)}$, V_{3V3}, and the oscillator must be within the specific operating range.

(2) There are no sequencing requirements for V_{3V3} and $V_{DD(1V8)}$.

(3) When V_{3V3} and $V_{DD(1V8)}$ reach the minimum voltage, a reset is registered within two valid oscillator clocks.

(4) Typical startup time is 0.5 ms for a 12 MHz crystal.

Fig 18. Startup sequence diagram

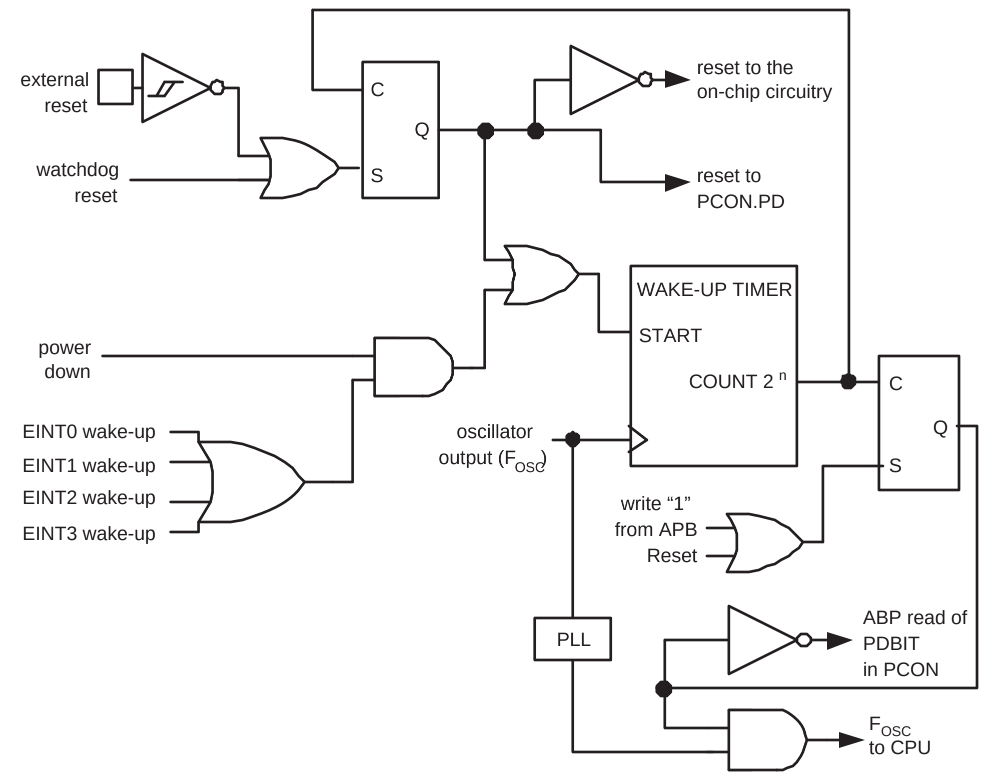


Fig 19. Reset block diagram including the wakeup timer

External and internal resets have some small differences. An external reset causes the value of certain pins to be latched to configure the part. External circuitry cannot determine when an internal reset occurs in order to allow setting up those special pins, so those latches are not reloaded during an internal reset. Pins that are examined during an external reset for various purposes are: P1.20/TRACESYNC, P1.26/RTCK (see Section 7.2, Section 7.3, and Section 8.6 . Pin P0.14 (see Section 21.5) is examined by on-chip bootloader when this code is executed after every reset.

6.12 APB divider

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

1. The first purpose is to provide peripherals with desired PCLK via APB bus so that they can operate at the speed chosen for the ARM processor. In order to achieve this, the APB bus may be slowed down to one half or one fourth of the processor clock rate. Because the APB bus must work properly at power up (and its timing cannot be altered if it does not work since the APB divider control registers reside on the APB bus), the default condition at reset is for the APB bus to run at one quarter speed.
2. The second purpose of the APB 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 APB Divider relative to the oscillator and the processor clock is shown in Figure 20. Because the APB Divider is connected to the PLL output, the PLL remains active (if it was running) during Idle mode.

6.12.1 Register description

Only one register is used to control the APB Divider.

Table 75. APB divider register map

Name	Description	Access	Reset value[1]	Address
APBDIV	Controls the rate of the APB clock in relation to the processor clock.	R/W	0x00	0xE01F C100

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

6.12.2 APB divider register (APBDIV - 0xE01F C100)

The APB Divider register contains two bits, allowing three divider values, as shown in Table 76.

Table 76. APB Divider register (APBDIV - address 0xE01F C100) bit description

Bit	Symbol	Value	Description	Reset value
1:0	APBDIV	00	APB bus clock is one fourth of the processor clock.	00
		01	APB bus clock is the same as the processor clock.	
		10	APB bus clock is one half of the processor clock.	
		11	Reserved. If this value is written to the APBDIV register, it has no effect (the previous setting is retained).	
3:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5:4	XCLKDIV		On the LPC22xx devices only, these bits control the clock that can be driven onto the P3.23/A23/XCLK pin. They have the same encoding as the APBDIV bits above. Bits 13 and 27:25 in the PINSEL2 register (Section 8.6.4) controls whether the pin carries A23 or the clock selected by this field. **Remark:** If this field and APBDIV have the same value, the same clock is used on the APB and XCLK. (This might be useful for external logic dealing with the APB peripherals).	00
		00	XCLK clock is one fourth of the processor clock.	
		01	XCLK clock is the same as the processor clock.	
		10	XCLK clock is one half of the processor clock.	
		11	Reserved. If this value is written to the APBDIV register, it has no effect (the previous setting is retained).	
7:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

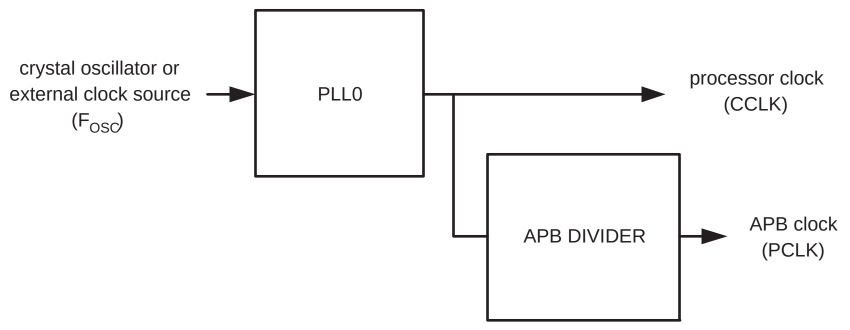


Fig 20. APB divider connections

6.13 Wakeup timer

On the LPC21xx/LPC22xx, 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.

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

Once a clock is detected, the wakeup timer counts 4096 clocks and 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 deasserted. If an external clock source is used in the system (as opposed to a crystal connected to the oscillator pins), the possibility that there could be little or no delay for oscillator start-up must be considered. The wakeup timer design then ensures that any other required chip functions will be operational prior to the beginning of program execution.

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

The pin multiplexing on the LPC21xx/LPC22xx (see Section 7.2, Section 7.3, and Section 8.6) allows peripherals that share pins with external interrupts to, in effect, bring the device out of Power-down mode. The following pin-function pairings allow interrupts from events relating to UART0 or 1, SPI 0 or 1, or the I^2C: RXD0 / EINT0, SDA / EINT1, SSEL0 / EINT2, RXD1 / EINT3, DCD1 / EINT1, RI1 / EINT2, SSEL1 / EINT3.

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

6.14 Code security vs. debugging

Applications in development typically need the debugging and tracing facilities in the LPC21xx/LPC22xx. Later in the life cycle of an application, it may be more important to protect the application code from observation by hostile or competitive eyes. The Code Read Protection feature of the LPC21xx/LPC22xx allows an application to control whether it can be debugged or protected from observation.

Details on the way Code Read Protection works can be found in Section 21.8 “Code Read Protection (CRP)”.

UM10114

Chapter 7: LPC21xx/22xx Pin configuration

Rev. 4 — 2 May 2012 User manual

7.1 How to read this chapter

The pin configurations are identical for all 64-pin packages and all 144-pin packages with the exception of the CAN pins which depend on the CAN configuration for each part, see Table 77 and Table 78. Pin configurations are identical for no-suffix, /00, and /01 versions.

Table 77. LPC21xx part-specific pin configurations 64-pin packages

Pin number Table 79	LPC209, LPC2109/01	LPC2119, LPC2119/01 LPC2129, LPC2129/01	LPC2114, LPC2114/01 LPC2124, LPC2124/01	LPC2194, LPC2194/01
1	P0[21]/PWM5/CAP1[3]	P0[21]/PWM5/CAP1[3]	P0[21]/PWM5/CAP1[3]	P0[21]/PWM5/RD3/CAP1[3]
2	P0[22]/CAP0[0]/MAT0[0]	P0[22]/CAP0[0]/MAT0[0]	P0[22]/CAP0[0]/MAT0[0]	P0[22]/TD3/CAP0[0]/MAT0[0]
3	P0[23]	P0[23]/RD2	P0[23]	P0[23]/RD2
5	P0[24]	P0[24]/TD2	P0[24]	P0[24]/TD2
9	P0[25]/RD1	P0[25]/RD1	P0[25]	P0[25]/RD1
10	TD1	TD1	n.c.	TD1
38	P0[12]/DSR1/MAT1[0]	P0[12]/DSR1/MAT1[0]	P0[12]/DSR1/MAT1[0]	P0[12]/DSR1/MAT1[0]/RD4
39	P0[13]/DTR1/MAT1[1]	P0[13]/DTR1/MAT1[1]	P0[13]/DTR1/MAT1[1]	P0[13]/DTR1/MAT1[1]/TD4

Table 78. LPC22xx part-specific pin configurations 144-pin packages

Pin number		LPC2210, LPC2210/01 LPC2220 LPC2212, LPC2212/01 LPC2214, LPC2214/01	LPC2290, LPC2290/01 LPC2292, LPC2292/01	LPC2294, LPC2294/01
LQFP144 Table 81	TFBGA144 Table 80			
4	C1	P0[21]/PWM5/CAP1[3]	P0[21]/PWM5/CAP1[3]	P0[21]/PWM5/RD3/CAP1[13]
5	D4	P0[22]/CAP0[0]/MAT0[0]	P0[22]/CAP0[0]/MAT0[0]	P0[22]/TD3/CAP0[0]/MAT0[0
6	D3	P0[23]	P0[23]/RD2	P0[23]/RD2
8	D1	P0[24]	P0[24]/TD2	P0[24]/TD2
21	H1	P0[25]	P0[25]/RD1	P0[25]/RD1
22	H2	n.c.	TD1	TD1
84	J13	P0[12]/DSR1/MAT1[0]	P0[12]/DSR1/MAT1[0]	P0[12]/DSR1/MAT1[0]/RD4
85	H10	P0[13]/DTR1/MAT1[1]	P0[13]/DTR1/MAT1[1]	P0[13]/DTR1/MAT1[1]/TD4

The SPI1 pins are shared with the SSP pins if the SSP interface is implemented. The following parts have an SSP interface:

- LPC2109/01, LPC2119/01, LPC2129/01
- LPC2114/01, LPC2124/01
- LPC2194/01

- LPC2210/01, LPC2220
- LPC2212/01, LPC2214/01
- LPC2290/01
- LPC2292/01, LPC2294/01

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 "How to read this manual".

7.2 Pin configuration for 64-pin packages

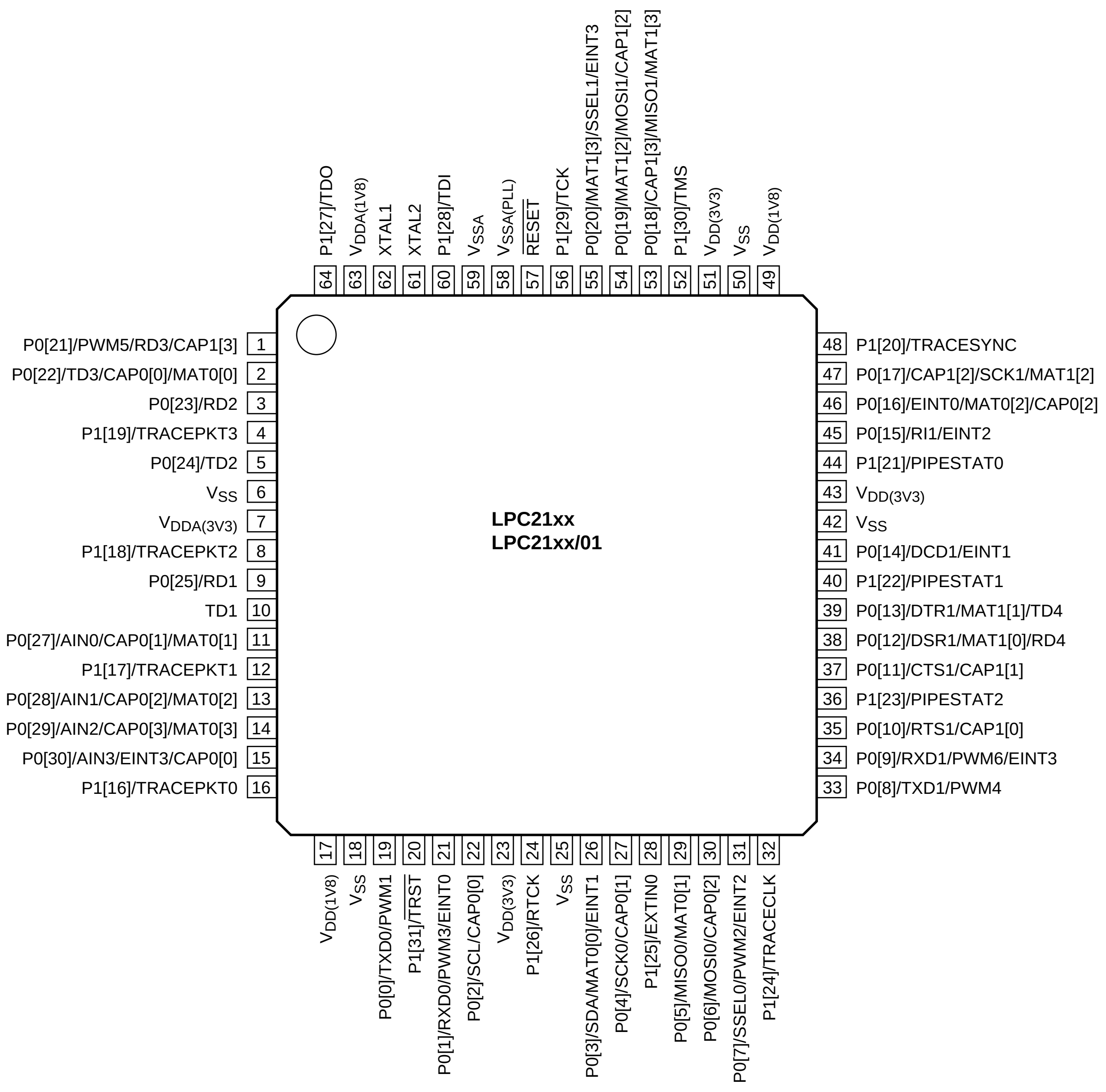


Fig 21. LPC21xx pin configuration (LQFP64 pin package)

Table 79. LPC21xx Pin description (64-pin packages)

Symbol	Pin	Type	Description
P0[0] to P0[31]		I/O	Port 0 is a 32-bit bidirectional I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the Pin Connect Block. Pins 26 and 31 of port 0 are not available.
P0[0]/TXD0/ PWM1	19[1]	O	**TXD0** — Transmitter output for UART0.
		O	**PWM1** — Pulse Width Modulator output 1.
P0[1]/RXD0/ PWM3/EINT0	21[2]	I	**RXD0** — Receiver input for UART0.
		O	**PWM3** — Pulse Width Modulator output 3.
		I	**EINT0** — External interrupt 0 input.
P0[2]/SCL/ CAP0[0]	22[3]	I/O	**SCL** — I^{2}C-bus clock input/output. Open-drain output (for I^{2}C-bus compliance).
		I	**CAP0[0]** — Capture input for Timer 0, channel 0.
P0[3]/SDA/ MAT0[0]/EINT1	26[3]	I/O	**SDA** — I^{2}C-bus data input/output. Open-drain output (for I^{2}C-bus compliance).
		O	**MAT0[0]** — Match output for Timer 0, channel 0.
		I	**EINT1** — External interrupt 1 input.
P0[4]/SCK0/ CAP0[1]	27[1]	I/O	**SCK0** — Serial clock for SPI0. SPI clock output from master or input to slave.
		I	**CAP0[1]** — Capture input for Timer 0, channel 1.
P0[5]/MISO0/ MAT0[1]	29[1]	I/O	**MISO0** — Master In Slave Out for SPI0. Data input to SPI master or data output from SPI slave.
		O	**MAT0[1]** — Match output for Timer 0, channel 1.
P0[6]/MOSI0/ CAP0[2]	30[1]	I/O	**MOSI0** — Master Out Slave In for SPI0. Data output from SPI master or data input to SPI slave.
		I	**CAP0[2]** — Capture input for Timer 0, channel 2.
P0[7]/SSEL0/ PWM2/EINT2	31[2]	I	**SSEL0** — Slave Select for SPI0. Selects the SPI interface as a slave.
		O	**PWM2** — Pulse Width Modulator output 2.
		I	**EINT2** — External interrupt 2 input.
P0[8]/TXD1/ PWM4	33[1]	O	**TXD1** — Transmitter output for UART1.
		O	**PWM4** — Pulse Width Modulator output 4.
P0[9]/RXD1/ PWM6/EINT3	34[2]	I	**RXD1** — Receiver input for UART1.
		O	**PWM6** — Pulse Width Modulator output 6.
		I	**EINT3** — External interrupt 3 input.
P0[10]/RTS1/ CAP1[0]	35[1]	O	**RTS1** — Request to Send output for UART1.
		I	**CAP1[0]** — Capture input for Timer 1, channel 0.
P0[11]/CTS1/ CAP1[1]	37[1]	I	**CTS1** — Clear to Send input for UART1.
		I	**CAP1[1]** — Capture input for Timer 1, channel 1.
P0[12]/DSR1/ MAT1[0]/RD4	38[1]	I	**DSR1** — Data Set Ready input for UART1.
		O	**MAT1[0]** — Match output for Timer 1, channel 0.
		O	**RD4** — CAN4 receiver input.
P0[13]/DTR1/ MAT1[1]/TD4	39[1]	O	**DTR1** — Data Terminal Ready output for UART1.
		O	**MAT1[1]** — Match output for Timer 1, channel 1.
		O	**TD4** — CAN4 transmitter output.

Table 79. LPC21xx Pin description (64-pin packages) *...continued*

Symbol	Pin	Type	Description
P0[14]/DCD1/ EINT1	41[2]	I	**DCD1 —** Data Carrier Detect input for UART1.
		I	**EINT1 —** External interrupt 1 input. **Note:** LOW on this pin while $\overline{\text{RESET}}$ is LOW forces on-chip bootloader to take control of the part after reset.
P0[15]/RI1/EINT2	45[2]	I	**RI1 —** Ring Indicator input for UART1.
		I	**EINT2 —** External interrupt 2 input.
P0[16]/EINT0/ MAT0[2]/CAP0[2]	46[2]	I	**EINT0 —** External interrupt 0 input.
		O	**MAT0[2] —** Match output for Timer 0, channel 2.
		I	**CAP0[2] —** Capture input for Timer 0, channel 2.
P0[17]/CAP1[2]/ SCK1/MAT1[2]	47[1]	I	**CAP1[2] —** Capture input for Timer 1, channel 2.
		I/O	**SCK1 —** Serial Clock for SPI1/SSP. SPI clock output from master or input to slave.
		O	**MAT1[2] —** Match output for Timer 1, channel 2.
P0[18]/CAP1[3]/ MISO1/MAT1[3]	53[1]	I	**CAP1[3] —** Capture input for Timer 1, channel 3.
		I/O	**MISO1 —** Master In Slave Out for SPI1/SSP. Data input to SPI master or data output from SPI slave.
		O	**MAT1[3] —** Match output for Timer 1, channel 3.
P0[19]/MAT1[2]/ MOSI1/CAP1[2]	54[1]	O	**MAT1[2] —** Match output for Timer 1, channel 2.
		I/O	**MOSI1 —** Master Out Slave In for SPI1/SSP. Data output from SPI master or data input to SPI slave.
		I	**CAP1[2] —** Capture input for Timer 1, channel 2.
P0[20]/MAT1[3]/ SSEL1/EINT3	55[2]	O	**MAT1[3] —** Match output for Timer 1, channel 3.
		I	**SSEL1 —** Slave Select for SPI1/SSP. Selects the SPI interface as a slave.
		I	**EINT3 —** External interrupt 3 input.
P0[21]/PWM5/ RD3/CAP1[3]	1[1]	O	**PWM5 —** Pulse Width Modulator output 5.
		I	**RD3 —** CAN3 receiver input.
		I	**CAP1[3] —** Capture input for Timer 1, channel 3.
P0[22]/TD3/ CAP0[0]/MAT0[0]	2[1]	O	**TD3 —** CAN3 transmitter output.
		I	**CAP0[0] —** Capture input for Timer 0, channel 0.
		O	**MAT0[0] —** Match output for Timer 0, channel 0.
P0[23]/RD2	3[1]	I	CAN2 receiver input.
P0[24]/TD2	5[1]	O	CAN2 transmitter output.
P0[25]/RD1	9[1]	O	CAN1 receiver input.
P0[27]/AIN0/ CAP0[1]/MAT0[1]	11[4]	I	**AIN0 —** A/D converter, input 0. This analog input is always connected to its pin.
		I	**CAP0[1] —** Capture input for Timer 0, channel 1.
		O	**MAT0[1] —** Match output for Timer 0, channel 1.
P0[28]/AIN1/ CAP0[2]/MAT0[2]	13[4]	I	**AIN1 —** A/D converter, input 1. This analog input is always connected to its pin.
		I	**CAP0[2] —** Capture input for Timer 0, channel 2.
		O	**MAT0[2] —** Match output for Timer 0, channel 2.
P0[29]/AIN2/ CAP0[3]/MAT0[3]	14[4]	I	**AIN2 —** A/D converter, input 2. This analog input is always connected to its pin.
		I	**CAP0[3] —** Capture input for Timer 0, Channel 3.
		O	**MAT0[3] —** Match output for Timer 0, channel 3.

Table 79. LPC21xx Pin description (64-pin packages) *…continued*

Symbol	Pin	Type	Description
P0[30]/AIN3/ EINT3/CAP0[0]	15[4]	I	**AIN3 —** A/D converter, input 3. This analog input is always connected to its pin.
		I	**EINT3 —** External interrupt 3 input.
		I	**CAP0[0] —** Capture input for Timer 0, channel 0.
P1[0] to P1[31]		I/O	Port 1 is a 32-bit bidirectional I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the Pin Connect Block. Pins 0 through 15 of port 1 are not available.
P1[16]/ TRACEPKT0	16[5]	O	Trace Packet, bit 0. Standard I/O port with internal pull-up.
P1[17]/ TRACEPKT1	12[5]	O	Trace Packet, bit 1. Standard I/O port with internal pull-up.
P1[18]/ TRACEPKT2	8[5]	O	Trace Packet, bit 2. Standard I/O port with internal pull-up.
P1[19]/ TRACEPKT3	4[5]	O	Trace Packet, bit 3. Standard I/O port with internal pull-up.
P1[20]/ TRACESYNC	48[5]	O	Trace Synchronization. Standard I/O port with internal pull-up. **Note:** LOW on this pin while $\overline{\text{RESET}}$ is LOW, enables pins P1[25:16] to operate as Trace port after reset.
P1[21]/ PIPESTAT0	44[5]	O	Pipeline Status, bit 0. Standard I/O port with internal pull-up.
P1[22]/ PIPESTAT1	40[5]	O	Pipeline Status, bit 1. Standard I/O port with internal pull-up.
P1[23]/ PIPESTAT2	36[5]	O	Pipeline Status, bit 2. Standard I/O port with internal pull-up.
P1[24]/ TRACECLK	32[5]	O	Trace Clock. Standard I/O port with internal pull-up.
P1[25]/EXTIN0	28[5]	I	External Trigger Input. Standard I/O with internal pull-up.
P1[26]/RTCK	24[5]	I/O	Returned Test Clock output. Extra signal added to the JTAG port. Assists debugger synchronization when processor frequency varies. Bidirectional pin with internal pull-up. **Note:** LOW on this pin while $\overline{\text{RESET}}$ is LOW, enables pins P1[31:26] to operate as Debug port after reset.
P1[27]/TDO	64[5]	O	Test Data out for JTAG interface.
P1[28]/TDI	60[5]	I	Test Data in for JTAG interface.
P1[29]/TCK	56[5]	I	Test Clock for JTAG interface. This clock must be slower than $^{1}?_{6}$ of the CPU clock (CCLK) for the JTAG interface to operate.
P1[30]/TMS	52[5]	I	Test Mode Select for JTAG interface.
P1[31]/$\overline{\text{TRST}}$	20[5]	I	Test Reset for JTAG interface.
TD1	10	O	CAN1 transmitter output.
$\overline{\text{RESET}}$	57	I	external reset input; a LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. TTL with hysteresis, 5 V tolerant.
XTAL1	62	I	input to the oscillator circuit and internal clock generator circuits.
XTAL2	61	O	output from the oscillator amplifier.
V_{SS}	6, 18, 25, 42, 50	I	ground: 0 V reference.

Table 79. LPC21xx Pin description (64-pin packages) *...continued*

Symbol	Pin	Type	Description
V_{SSA}	59	I	analog ground; 0 V reference. This should nominally be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{SSA(PLL)}$	58	I	PLL analog ground; 0 V reference. This should nominally be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{DD(1V8)}$	17, 49	I	1.8 V core power supply; this is the power supply voltage for internal circuitry.
$V_{DDA(1V8)}$	63	I	analog 1.8 V core power supply; this is the power supply voltage for internal circuitry. This should be nominally the same voltage as $V_{DD(1V8)}$ but should be isolated to minimize noise and error.
$V_{DD(3V3)}$	23, 43, 51	I	3.3 V pad power supply; this is the power supply voltage for the I/O ports.
$V_{DDA(3V3)}$	7	I	analog 3.3 V pad power supply; this should be nominally the same voltage as $V_{DD(3V3)}$ but should be isolated to minimize noise and error. The level on this pin also provides the voltage reference level for the ADC.

[1] 5 V tolerant pad providing digital I/O functions with TTL levels and hysteresis and 10 ns slew rate control.

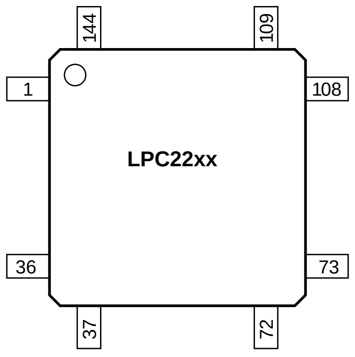
[2] 5 V tolerant pad providing digital I/O functions with TTL levels and hysteresis and 10 ns slew rate control. If configured for an input function, this pad utilizes built-in glitch filter that blocks pulses shorter than 3 ns.

[3] Open drain 5 V tolerant digital I/O I^2C-bus 400 kHz specification compatible pad. It requires external pull-up to provide an output functionality. Open-drain functionality applies to all output functions on this pin.

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

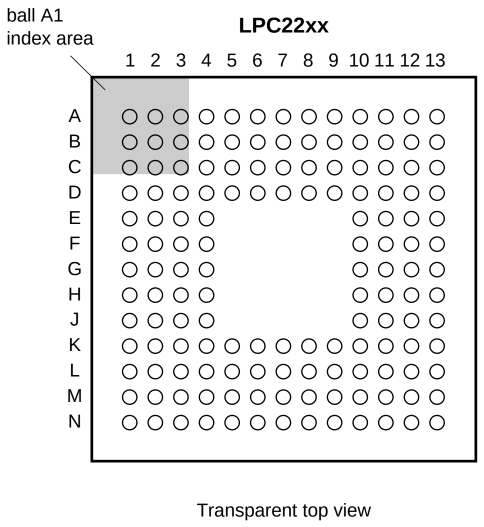
[5] 5 V tolerant pad with built-in pull-up resistor providing digital I/O functions with TTL levels and hysteresis and 10 ns slew rate control. The pull-up resistor's value ranges from 60 kΩ to 300 kΩ.

7.3 Pin configuration for 144-pin packages



(1) Pin configuration is identical for devices with and without /00 and /01 suffixes.

Fig 22. LQFP144 pinning



(1) Pin configuration is identical for devices with and without /00 and /01 suffixes.

Fig 23. TFBGA144 pinning

Table 80. LPC22xx Ball allocation

Row	Column											
	1	2	3	4	5	6	7	8	9	10	11	1
A	P2[22]/ D22	$V_{DDA(1V8)}$	P1[28]/ TDI	P2[21]/ D21	P2[18]/ D18	P2[14]/ D14	P1[29]/ TCK	P2[11]/ D11	P2[10]/ D10	P2[7]/D7	$V_{DD(3V3)}$	V
B	$V_{DD(3V3)}$	P1[27]/ TDO	XTAL2	$V_{SSA(PLL)}$	P2[19]/ D19	P2[15]/ D15	P2[12]/ D12	P0[20]/ MAT1[3]/ SSEL1/ EINT3	$V_{DD(3V3)}$	P2[6]/D6	V_{SS}	P
C	P0[21]/ PWM5/ CAP1[3]	V_{SS}	XTAL1	V_{SSA}	$\overline{\text{RESET}}$	P2[16]/ D16	P2[13]/ D13	P0[19]/ MAT1[2]/ MOSI1/ CAP1[2]	P2[9]/D9	P2[5]/D5	P2[2]/D2	P
D	P0[24]/ TD2	P1[19]/ TRACE PKT3	P0[23]/ RD2	P0[22]/ CAP0[0]/ MAT0[0]	P2[20]/ D20	P2[17]/ D17	V_{SS}	P0[18]/ CAP1[3]/ MISO1/ MAT1[3]	P2[8]/D8	P1[30]/ TMS	V_{SS}	P T S
E	P2[25]/ D25	P2[24]/ D24	P2[23]/ D23	V_{SS}						P0[16]/ EINT0/ MAT0[2]/ CAP0[2]	P0[15]/ RI1/ EINT2	P
F	P2[27]/ D27/ BOOT1	P1[18]/ TRACE PKT2	$V_{DDA(3V3)}$	P2[26]/ D26/ BOOT0						P3[31]/ BLS0	P1[21]/ PIPE STAT0	V
G	P2[29]/ D29	P2[28]/ D28	P2[30]/ D30/AIN4	P2[31]/ D31/AIN5						P0[14]/ DCD1/ EINT1	P1[0]/CS0	P
H	P0[25]/ RD1	TD1	P0[27]/ AIN0/ CAP0[1]/ MAT0[1]	P1[17]/ TRACE PKT1						P0[13]/ DTR1/ MAT1[1]	P1[22]/ PIPE STAT1	P
J	P0[28]/ AIN1/ CAP0[2]/ MAT0[2]	V_{SS}	P3[29]/ BLS2/ AIN6	P3[28]/ BLS3/ AIN7						P3[3]/A3	P1[23]/ PIPE STAT2	P C C
K	P3[27]/ WE	P3[26]/ CS1	$V_{DD(3V3)}$	P3[22]/ A22	P3[20]/ A20	P0[1]/ RXD0/ PWM3/ EINT0	P3[14]/ A14	P1[25]/ EXTIN0	P3[11]/ A11	$V_{DD(3V3)}$	P0[10]/ RTS1/ CAP1[0]	V
L	P0[29]/ AIN2/ CAP0[3]/ MAT0[3]	P0[30]/ AIN3/ EINT3/ CAP0[0]	P1[16]/ TRACE PKT0	P0[0]/ TXD0/ PWM1	P3[19]/ A19	P0[2]/ SCL/ CAP0[0]	P3[15]/ A15	P0[4]/ SCK0/ CAP0[1]	P3[12]/ A12	V_{SS}	P1[24]/ TRACE CLK	P T P
M	P3[25]/ CS2	P3[24]/ CS3	$V_{DD(3V3)}$	P1[31]/ $\overline{\text{TRST}}$	P3[18]/ A18	$V_{DD(3V3)}$	P3[16]/ A16	P0[3]/ SDA/ MAT0[0]/ EINT1	P3[13]/ A13	P3[9]/A9	P0[7]/ SSEL0/ PWM2/ EINT2	P
N	$V_{DD(1V8)}$	V_{SS}	P3[23]/ A23/ XCLK	P3[21]/ A21	P3[17]/ A17	P1[26]/ RTCK	V_{SS}	$V_{DD(3V3)}$	P0[5]/ MISO0/ MAT0[1]	P3[10]/ A10	P0[6]/ MOSI0/ CAP0[2]	P

Table 81. LPC22xx Pin description (144 pin packages)

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
P0[0] to P0[31]			I/O	**Port 0:** Port 0 is a 32-bit bidirectional I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the Pin Connect Block. Pins 26 and 31 of port 0 are not available.
P0[0]/TXD0/ PWM1	42[1]	L4[1]	O	**TXD0** — Transmitter output for UART0.
			O	**PWM1** — Pulse Width Modulator output 1.
P0[1]/RXD0/ PWM3/EINT0	49[2]	K6[2]	I	**RXD0** — Receiver input for UART0.
			O	**PWM3** — Pulse Width Modulator output 3.
			I	**EINT0** — External interrupt 0 input
P0[2]/SCL/ CAP0[0]	50[3]	L6[3]	I/O	**SCL** — I^{2}C-bus clock input/output. Open-drain output (for I^{2}C-bus compliance).
			I	**CAP0[0]** — Capture input for Timer 0, channel 0.
P0[3]/SDA/ MAT0[0]/EINT1	58[3]	M8[3]	I/O	**SDA** — I^{2}C-bus data input/output. Open-drain output (for I^{2}C-bus compliance).
			O	**MAT0[0]** — Match output for Timer 0, channel 0.
			I	**EINT1** — External interrupt 1 input.
P0[4]/SCK0/ CAP0[1]	59[1]	L8[1]	I/O	**SCK0** — Serial clock for SPI0. SPI clock output from master or input to slave.
			I	**CAP0[1]** — Capture input for Timer 0, channel 1.
P0[5]/MISO0/ MAT0[1]	61[1]	N9[1]	I/O	**MISO0** — Master In Slave OUT for SPI0. Data input to SPI master or data output from SPI slave.
			O	**MAT0[1]** — Match output for Timer 0, channel 1.
P0[6]/MOSI0/ CAP0[2]	68[1]	N11[1]	I/O	**MOSI0** — Master Out Slave In for SPI0. Data output from SPI master or data input to SPI slave.
			I	**CAP0[2]** — Capture input for Timer 0, channel 2.
P0[7]/SSEL0/ PWM2/EINT2	69[2]	M11[2]	I	**SSEL0** — Slave Select for SPI0. Selects the SPI interface as a slave.
			O	**PWM2** — Pulse Width Modulator output 2.
			I	**EINT2** — External interrupt 2 input.
P0[8]/TXD1/ PWM4	75[1]	L12[1]	O	**TXD1** — Transmitter output for UART1.
			O	**PWM4** — Pulse Width Modulator output 4.
P0[9]/RXD1/ PWM6/EINT3	76[2]	L13[2]	I	**RXD1** — Receiver input for UART1.
			O	**PWM6** — Pulse Width Modulator output 6.
			I	**EINT3** — External interrupt 3 input.
P0[10]/RTS1/ CAP1[0]	78[1]	K11[1]	O	**RTS1** — Request to Send output for UART1.
			I	**CAP1[0]** — Capture input for Timer 1, channel 0.
P0[11]/CTS1/ CAP1[1]	83[1]	J12[1]	I	**CTS1** — Clear to Send input for UART1.
			I	**CAP1[1]** — Capture input for Timer 1, channel 1.
P0[12]/DSR1/ MAT1[0]/RD4	84[1]	J13[1]	I	**DSR1** — Data Set Ready input for UART1.
			O	**MAT1[0]** — Match output for Timer 1, channel 0.
			I	**RD4** — CAN4 receiver input (LPC2294 only).

Table 81. LPC22xx Pin description (144 pin packages) *…continued*

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
P0[13]/DTR1/ MAT1[1]/TD4	85[1]	H10[1]	O	**DTR1 —** Data Terminal Ready output for UART1.
			O	**MAT1[1] —** Match output for Timer 1, channel 1.
			O	**TD4 —** CAN4 transmitter output (LPC2294 only).
P0[14]/DCD1/ EINT1	92[2]	G10[2]	I	**DCD1 —** Data Carrier Detect input for UART1.
			I	**EINT1 —** External interrupt 1 input. **Note:** LOW on this pin while $\overline{\text{RESET}}$ is LOW forces on-chip bootloader to take over control of the part after reset.
P0[15]/RI1/ EINT2	99[2]	E11[2]	I	**RI1 —** Ring Indicator input for UART1.
			I	**EINT2 —** External interrupt 2 input.
P0[16]/EINT0/ MAT0[2]/ CAP0[2]	100[2]	E10[2]	I	**EINT0 —** External interrupt 0 input.
			O	**MAT0[2] —** Match output for Timer 0, channel 2.
			I	**CAP0[2] —** Capture input for Timer 0, channel 2.
P0[17]/CAP1[2]/ SCK1/MAT1[2]	101[1]	D13[1]	I	**CAP1[2] —** Capture input for Timer 1, channel 2.
			I/O	**SCK1 —** Serial Clock for SPI1/SSP. SPI clock output from master or input to slave.
			O	**MAT1[2] —** Match output for Timer 1, channel 2.
P0[18]/CAP1[3]/ MISO1/MAT1[3]	121[1]	D8[1]	I	**CAP1[3] —** Capture input for Timer 1, channel 3.
			I/O	**MISO1 —** Master In Slave Out for SPI1/SSP. Data input to SPI master or data output from SPI slave.
			O	**MAT1[3] —** Match output for Timer 1, channel 3.
P0[19]/MAT1[2]/ MOSI1/CAP1[2]	122[1]	C8[1]	O	**MAT1[2] —** Match output for Timer 1, channel 2.
			I/O	**MOSI1 —** Master Out Slave In for SPI1/SSP. Data output from SPI master or data input to SPI slave.
			I	**CAP1[2] —** Capture input for Timer 1, channel 2.
P0[20]/MAT1[3]/ SSEL1/EINT3	123[2]	B8[2]	O	**MAT1[3] —** Match output for Timer 1, channel 3.
			I	**SSEL1 —** Slave Select for SPI1/SSP. Selects the SPI interface as a slave.
			I	**EINT3 —** External interrupt 3 input.
P0[21]/PWM5/ RD3/CAP1[3]	4[1]	C1[1]	O	**PWM5 —** Pulse Width Modulator output 5.
			I	**RD3 —** CAN3 receiver input (LPC2294 only).
			I	**CAP1[3] —** Capture input for Timer 1, channel 3.
P0[22]/TD3/ CAP0[0]/ MAT0[0]	5[1]	D4[1]	O	**TD3 —** CAN3 transmitter output (LPC2294 only).
			I	**CAP0[0] —** Capture input for Timer 0, channel 0.
			O	**MAT0[0] —** Match output for Timer 0, channel 0.
P0[23]/RD2	6[1]	D3[1]	I	**RD2 —** CAN2 receiver input.
P0[24]/TD2	8[1]	D1[1]	O	**TD2 —** CAN2 transmitter output.
P0[25]/RD1	21[1]	H1[1]	I	**RD1 —** CAN1 receiver input.
P0[27]/AIN0/ CAP0[1]/ MAT0[1]	23[4]	H3[4]	I	**AIN0 —** ADC, input 0. This analog input is always connected to its pin.
			I	**CAP0[1] —** Capture input for Timer 0, channel 1.
			O	**MAT0[1] —** Match output for Timer 0, channel 1.

Table 81. LPC22xx Pin description (144 pin packages) *…continued*

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
P0[28]/AIN1/ CAP0[2]/ MAT0[2]	25[4]	J1[4]	I	**AIN1 —** ADC, input 1. This analog input is always connected to its pin.
			I	**CAP0[2] —** Capture input for Timer 0, channel 2.
			O	**MAT0[2] —** Match output for Timer 0, channel 2.
P0[29]/AIN2/ CAP0[3]/ MAT0[3]	32[4]	L1[4]	I	**AIN2 —** ADC, input 2. This analog input is always connected to its pin.
			I	**CAP0[3] —** Capture input for Timer 0, Channel 3.
			O	**MAT0[3] —** Match output for Timer 0, channel 3.
P0[30]/AIN3/ EINT3/CAP0[0]	33[4]	L2[4]	I	**AIN3 —** ADC, input 3. This analog input is always connected to its pin.
			I	**EINT3 —** External interrupt 3 input.
			I	**CAP0[0] —** Capture input for Timer 0, channel 0.
P1[0] to P1[31]			I/O	**Port 1:** Port 1 is a 32-bit bidirectional I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the Pin Connect Block. Pins 2 through 15 of port 1 are not available.
P1[0]/CS0	91[5]	G11[5]	O	**CS0 —** LOW-active Chip Select 0 signal. (Bank 0 addresses range 0x8000 0000 to 0x80FF FFFF)
P1[1]/OE	90[5]	G13[5]	O	**OE —** LOW-active Output Enable signal.
P1[16]/ TRACEPKT0	34[5]	L3[5]	O	**TRACEPKT0 —** Trace Packet, bit 0. Standard I/O port with internal pull-up.
P1[17]/ TRACEPKT1	24[5]	H4[5]	O	**TRACEPKT1 —** Trace Packet, bit 1. Standard I/O port with internal pull-up.
P1[18]/ TRACEPKT2	15[5]	F2[5]	O	**TRACEPKT2 —** Trace Packet, bit 2. Standard I/O port with internal pull-up.
P1[19]/ TRACEPKT3	7[5]	D2[5]	O	**TRACEPKT3 —** Trace Packet, bit 3. Standard I/O port with internal pull-up.
P1[20]/ TRACESYNC	102[5]	D12[5]	O	**TRACESYNC —** Trace Synchronization. Standard I/O port with internal pull-up. **Note:** LOW on this pin while $\overline{\text{RESET}}$ is LOW, enables pins P1[25:16] to operate as Trace port after reset.
P1[21]/ PIPESTAT0	95[5]	F11[5]	O	**PIPESTAT0 —** Pipeline Status, bit 0. Standard I/O port with internal pull-up.
P1[22]/ PIPESTAT1	86[5]	H11[5]	O	**PIPESTAT1 —** Pipeline Status, bit 1. Standard I/O port with internal pull-up.
P1[23]/ PIPESTAT2	82[5]	J11[5]	O	**PIPESTAT2 —** Pipeline Status, bit 2. Standard I/O port with internal pull-up.
P1[24]/ TRACECLK	70[5]	L11[5]	O	**TRACECLK —** Trace Clock. Standard I/O port with internal pull-up.
P1[25]/EXTIN0	60[5]	K8[5]	I	**EXTIN0 —** External Trigger Input. Standard I/O with internal pull-up.

Table 81. LPC22xx Pin description (144 pin packages) *…continued*

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
P1[26]/RTCK	52[5]	N6[5]	I/O	**RTCK —** Returned Test Clock output. Extra signal added to the JTAG port. Assists debugger synchronization when processor frequency varies. Bidirectional pin with internal pull-up. **Note:** LOW on this pin while $\overline{\text{RESET}}$ is LOW, enables pins P1[31:26] to operate as Debug port after reset.
P1[27]/TDO	144[5]	B2[5]	O	**TDO —** Test Data out for JTAG interface.
P1[28]/TDI	140[5]	A3[5]	I	**TDI —** Test Data in for JTAG interface.
P1[29]/TCK	126[5]	A7[5]	I	**TCK —** Test Clock for JTAG interface. This clock must be slower than $^{1}?_{6}$ of the CPU clock (CCLK) for the JTAG interface to operate.
P1[30]/TMS	113[5]	D10[5]	I	**TMS —** Test Mode Select for JTAG interface.
P1[31]/$\overline{\text{TRST}}$	43[5]	M4[5]	I	**$\overline{\text{TRST}}$ —** Test Reset for JTAG interface.
P2[0] to P2[31]			I/O	**Port 2 —** Port 2 is a 32-bit bidirectional I/O port with individual direction controls for each bit. The operation of port 2 pins depends upon the pin function selected via the Pin Connect Block.
P2[0]/D0	98[5]	E12[5]	I/O	**D0 —** External memory data line 0.
P2[1]/D1	105[5]	C12[5]	I/O	**D1 —** External memory data line 1.
P2[2]/D2	106[5]	C11[5]	I/O	**D2 —** External memory data line 2.
P2[3]/D3	108[5]	B12[5]	I/O	**D3 —** External memory data line 3.
P2[4]/D4	109[5]	A13[5]	I/O	**D4 —** External memory data line 4.
P2[5]/D5	114[5]	C10[5]	I/O	**D5 —** External memory data line 5.
P2[6]/D6	115[5]	B10[5]	I/O	**D6 —** External memory data line 6.
P2[7]/D7	116[5]	A10[5]	I/O	**D7 —** External memory data line 7.
P2[8]/D8	117[5]	D9[5]	I/O	**D8 —** External memory data line 8.
P2[9]/D9	118[5]	C9[5]	I/O	**D9 —** External memory data line 9.
P2[10]/D10	120[5]	A9[5]	I/O	**D10 —** External memory data line 10.
P2[11]/D11	124[5]	A8[5]	I/O	**D11 —** External memory data line 11.
P2[12]/D12	125[5]	B7[5]	I/O	**D12 —** External memory data line 12.
P2[13]/D13	127[5]	C7[5]	I/O	**D13 —** External memory data line 13.
P2[14]/D14	129[5]	A6[5]	I/O	**D14 —** External memory data line 14.
P2[15]/D15	130[5]	B6[5]	I/O	**D15 —** External memory data line 15.
P2[16]/D16	131[5]	C6[5]	I/O	**D16 —** External memory data line 16.
P2[17]/D17	132[5]	D6[5]	I/O	**D17 —** External memory data line 17.
P2[18]/D18	133[5]	A5[5]	I/O	**D18 —** External memory data line 18.
P2[19]/D19	134[5]	B5[5]	I/O	**D19 —** External memory data line 19.
P2[20]/D20	136[5]	D5[5]	I/O	**D20 —** External memory data line 20.
P2[21]/D21	137[5]	A4[5]	I/O	**D21 —** External memory data line 21.
P2[22]/D22	1[5]	A1[5]	I/O	**D22 —** External memory data line 22.
P2[23]/D23	10[5]	E3[5]	I/O	**D23 —** External memory data line 23.
P2[24]/D24	11[5]	E2[5]	I/O	**D24 —** External memory data line 24.
P2[25]/D25	12[5]	E1[5]	I/O	**D25 —** External memory data line 25.

Table 81. LPC22xx Pin description (144 pin packages) *…continued*

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
P2[26]/D26/ BOOT0	13[5]	F4[5]	I/O	**D26 —** External memory data line 26.
			I	**BOOT0 —** While $\overline{\text{RESET}}$ is low, together with BOOT1 controls booting and internal operation. Internal pull-up ensures high state if pin is left unconnected.
P2[27]/D27/ BOOT1	16[5]	F1[5]	I/O	**D27 —** External memory data line 27.
			I	**BOOT1 —** While $\overline{\text{RESET}}$ is low, together with BOOT0 controls booting and internal operation. Internal pull-up ensures high state if pin is left unconnected. BOOT1:0 = 00 selects 8-bit memory on CS0 for boot. BOOT1:0 = 01 selects 16-bit memory on CS0 for boot. BOOT1:0 = 10 selects 32-bit memory on CS0 for boot. BOOT1:0 = 11 selects internal flash memory or 16-bit memory for CS0 boot for flashless LPC22xx.
P2[28]/D28	17[5]	G2[5]	I/O	**D28 —** External memory data line 28.
P2[29]/D29	18[5]	G1[5]	I/O	**D29 —** External memory data line 29.
P2[30]/D30/ AIN4	19[4]	G3[2]	I/O	**D30 —** External memory data line 30.
			I	**AIN4 —** ADC, input 4. This analog input is always connected to its pin.
P2[31]/D31/ AIN5	20[4]	G4[2]	I/O	**D31 —** External memory data line 31.
			I	**AIN5 —** ADC, input 5. This analog input is always connected to its pin.
P3[0] to P3[31]			I/O	**Port 3 —** Port 3 is a 32-bit bidirectional I/O port with individual direction controls for each bit. The operation of port 3 pins depends upon the pin function selected via the Pin Connect Block.
P3[0]/A0	89[5]	G12[5]	O	**A0 —** External memory address line 0.
P3[1]/A1	88[5]	H13[5]	O	**A1 —** External memory address line 1.
P3[2]/A2	87[5]	H12[5]	O	**A2 —** External memory address line 2.
P3[3]/A3	81[5]	J10[5]	O	**A3 —** External memory address line 3.
P3[4]/A4	80[5]	K13[5]	O	**A4 —** External memory address line 4.
P3[5]/A5	74[5]	M13[5]	O	**A5 —** External memory address line 5.
P3[6]/A6	73[5]	N13[5]	O	**A6 —** External memory address line 6.
P3[7]/A7	72[5]	M12[5]	O	**A7 —** External memory address line 7.
P3[8]/A8	71[5]	N12[5]	O	**A8 —** External memory address line 8.
P3[9]/A9	66[5]	M10[5]	O	**A9 —** External memory address line 9.
P3[10]/A10	65[5]	N10[5]	O	**A10 —** External memory address line 10.
P3[11]/A11	64[5]	K9[5]	O	**A11 —** External memory address line 11.
P3[12]/A12	63[5]	L9[5]	O	**A12 —** External memory address line 12.
P3[13]/A13	62[5]	M9[5]	O	**A13 —** External memory address line 13.
P3[14]/A14	56[5]	K7[5]	O	**A14 —** External memory address line 14.
P3[15]/A15	55[5]	L7[5]	O	**A15 —** External memory address line 15.
P3[16]/A16	53[5]	M7[5]	O	**A16 —** External memory address line 16.
P3[17]/A17	48[5]	N5[5]	O	**A17 —** External memory address line 17.
P3[18]/A18	47[5]	M5[5]	O	**A18 —** External memory address line 18.

Table 81. LPC22xx Pin description (144 pin packages) *…continued*

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
P3[19]/A19	46[5]	L5[5]	O	**A19 —** External memory address line 19.
P3[20]/A20	45[5]	K5[5]	O	**A20 —** External memory address line 20.
P3[21]/A21	44[5]	N4[5]	O	**A21 —** External memory address line 21.
P3[22]/A22	41[5]	K4[5]	O	**A22 —** External memory address line 22.
P3[23]/A23/ XCLK	40[5]	N3[5]	I/O	**A23 —** External memory address line 23.
			O	**XCLK —** Clock output.
P3[24]/CS3	36[5]	M2[5]	O	**CS3 —** LOW-active Chip Select 3 signal. (Bank 3 addresses range 0x8300 0000 to 0x83FF FFFF)
P3[25]/CS2	35[5]	M1[5]	O	**CS2 —** LOW-active Chip Select 2 signal. (Bank 2 addresses range 0x8200 0000 to 0x82FF FFFF)
P3[26]/CS1	30[5]	K2[5]	O	**CS1 —** LOW-active Chip Select 1 signal. (Bank 1 addresses range 0x8100 0000 to 0x81FF FFFF)
P3[27]/WE	29[5]	K1[5]	O	**WE —** LOW-active Write enable signal.
P3[28]/BLS3/ AIN7	28[4]	J4[4]	O	**BLS3 —** LOW-active Byte Lane Select signal (Bank 3).
			I	**AIN7 —** ADC, input 7. This analog input is always connected to its pin.
P3[29]/BLS2/ AIN6	27[4]	J3[4]	O	**BLS2 —** LOW-active Byte Lane Select signal (Bank 2).
			I	**AIN6 —** ADC, input 6. This analog input is always connected to its pin.
P3[30]/BLS1	97[4]	E13[4]	O	**BLS1 —** LOW-active Byte Lane Select signal (Bank 1).
P3[31]/BLS0	96[4]	F10[4]	O	**BLS0 —** LOW-active Byte Lane Select signal (Bank 0).
TD1	22[5]	H2[5]	O	**TD1:** CAN1 transmitter output.
$\overline{\text{RESET}}$	135[6]	C5[6]	I	**External Reset input:** A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. TTL with hysteresis, 5 V tolerant.
XTAL1	142[7]	C3[7]	I	Input to the oscillator circuit and internal clock generator circuits.
XTAL2	141[7]	B3[7]	O	Output from the oscillator amplifier.
V_{SS}	3, 9, 26, 38, 54, 67, 79, 93, 103, 107, 111, 128	C2, E4, J2, N2, N7, L10, K12, F13, D11, B13, B11, D7	I	**Ground:** 0 V reference.
V_{SSA}	139	C4	I	**Analog ground:** 0 V reference. This should nominally be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{SSA(PLL)}$	138	B4	I	**PLL analog ground:** 0 V reference. This should nominally be the same voltage as V_{SS}, but should be isolated to minimize noise and error.
$V_{DD(1V8)}$	37, 110	N1, A12	I	**1.8 V core power supply:** This is the power supply voltage for internal circuitry.

Table 81. LPC22xx Pin description (144 pin packages) *…continued*

Symbol	Pin (LQFP)	Pin (TFBGA)	Type	Description
$V_{DDA(1V8)}$	143	A2	I	**Analog 1.8 V core power supply:** This is the power supply voltage for internal circuitry. This should be nominally the same voltage as $V_{DD(1V8)}$ but should be isolated to minimize noise and error.
$V_{DD(3V3)}$	2, 31, 39, 51, 57, 77, 94, 104, 112, 119	B1, K3, M3, M6, N8, K10, F12, C13, A11, B9	I	**3.3 V pad power supply:** This is the power supply voltage for the I/O ports.
$V_{DDA(3V3)}$	14	F3	I	**Analog 3.3 V pad power supply:** This should be nominally the same voltage as $V_{DD(3V3)}$ but should be isolated to minimize noise and error. The level on this pin also provides the voltage reference level for the ADC.

[1] 5 V tolerant pad providing digital I/O functions with TTL levels and hysteresis and 10 ns slew rate control.

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

[3] Open drain 5 V tolerant digital I/O I^2C-bus 400 kHz specification compatible pad. It requires external pull-up to provide an output functionality. Open-drain functionality applies to all output functions on this pin.

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

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

[6] 5 V tolerant pad providing digital input (with TTL levels and hysteresis) function only.

[7] Pad provides special analog functionality.

UM10114

Chapter 8: LPC21xx/22xx Pin connect block

Rev. 4 — 2 May 2012 **User manual**

8.1 How to read this chapter

The pin connect blocks are identical for all LPC21xx and LPC22xx parts, respectively. The LPC22xx use additional bits in the PINSEL2 register to select the EMC, additional ADC pins, and for boot control (see Table 83). For parts with CAN interface, see Table 82 for which bits select the CAN pins in the PINSEL registers. The CAN bit settings are reserved for parts without CAN interfaces.

Table 82. CAN configuration in the LPC21xx/22xx pin connect registers

Pin	available in part	PINSEL register	Bits
	no suffix, /00, /01		
RD1[1]	LPC2109 LPC2119 LPC2129 LPC2194 LPC2290 LPC2292 LPC2294	PINSEL1 Table 87	19:18
RD2/TD2	LPC2119 LPC2129 LPC2194 LPC2290 LPC2292 LPC2294	PINSEL1 Table 87	15:14/17:16
RD3/TD3	LPC2194 LPC2294	PINSEL1 Table 87	11:10/13:12
RD4/TD4	LPC2194 LPC2294	PINSEL0 Table 86	25:24/27:26

[1] The TD1 output, if available, is not shared with other pins.

Table 83. Pin select registers for 64-pin (LPC21xx) and 144-pin (LPC22xx) configurations

Parts	PINSEL0	PINSEL1	PINSEL2	Boot control
all LPC21xx	Table 86	Table 87	Table 88	n/a
all LPC22xx	Table 86	Table 87	Table 89	Table 90

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

8.2 Features

Allows individual pin configuration.

8.3 Applications

The purpose of the Pin connect block is to configure the microcontroller pins to the desired functions.

8.4 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 interrupts being enabled. Activity of any enabled peripheral function that is not mapped to a related pin should be considered undefined.

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

The only exception are the inputs to the A/D converter. Regardless of the function that is selected for the port pin that also hosts the A/D input, this A/D input can be read at any time, and variations of the voltage level on this pin will be reflected in the A/D readings. However, valid analog readings can be obtained if and only if the analog input function is selected. Only then the proper interface circuit is active in between the physical pin and the A/D module. In all other cases, the logic necessary for the digital function will be active and will disrupt proper behavior of the A/D.

8.5 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 84. Pin function Select register bits

PINSEL0 & PINSEL1 values	Function	Value after reset
00	Primary (default) function, typically GPIO port	00
01	First alternate function	
10	Second alternate function	
11	Third alternate function	

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

8.6 Register description

The Pin Control Module contains 3 registers as shown in Table 85 below.

Table 85. Pin connect block register map

Name	Description	Access	Reset value	Address
PINSEL0	Pin function select register 0	R/W	0x0000 0000	0xE002 C000
PINSEL1	Pin function select register 1	R/W	0x1540 0000	0xE002 C004
PINSEL2	Pin function select register 2	R/W	See Table 88.	0xE002 C014

8.6.1 Pin function Select register 0 (PINSEL0 - 0xE002 C000)

The PINSEL0 register controls the functions of the pins using the settings listed in Table 86. The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

The CAN bit settings are reserved for parts without CAN interfaces (see Table 82).

Table 86. Pin function Select register 0 (PINSEL0 - address 0xE002 C000) bit description)

Bit	Symbol	Value	Function	Reset value
1:0	P0.0	00	GPIO Port 0.0	0
		01	TXD (UART0)	
		10	PWM1	
		11	Reserved	
3:2	P0.1	00	GPIO Port 0.1	0
		01	RxD (UART0)	
		10	PWM3	
		11	EINT0	
5:4	P0.2[1]	00	GPIO Port 0.2	0
		01	SCL (I^2C)	
		10	Capture 0.0 (Timer 0)	
		11	Reserved	
7:6	P0.3[1]	00	GPIO Port 0.3	0
		01	SDA (I^2C)	
		10	Match 0.0 (Timer 0)	
		11	EINT1	
9:8	P0.4	00	GPIO Port 0.4	0
		01	SCK0 (SPI0)	
		10	Capture 0.1 (Timer 0)	
		11	Reserved	
11:10	P0.5	00	GPIO Port 0.5	0
		01	MISO0 (SPI0)	
		10	Match 0.1 (Timer 0)	
		11	Reserved	

Table 86. Pin function Select register 0 (PINSEL0 - address 0xE002 C000) bit description)

Bit	Symbol	Value	Function	Reset value
13:12	P0.6	00	GPIO Port 0.6	0
		01	MOSI0 (SPI0)	
		10	Capture 0.2 (Timer 0)	
		11	Reserved	
15:14	P0.7	00	GPIO Port 0.7	0
		01	SSEL0 (SPI0)	
		10	PWM2	
		11	EINT2	
17:16	P0.8	00	GPIO Port 0.8	0
		01	TXD UART1	
		10	PWM4	
		11	Reserved	
19:18	P0.9	00	GPIO Port 0.9	0
		01	RxD (UART1)	
		10	PWM6	
		11	EINT3	
21:20	P0.10	00	GPIO Port 0.10	0
		01	RTS1 (UART1)	
		10	Capture 1.0 (Timer 1)	
		11	Reserved	
23:22	P0.11	00	GPIO Port 0.11	0
		01	CTS1 (UART1)	
		10	Capture 1.1 (Timer 1)	
		11	Reserved	
25:24	P0.12	00	GPIO Port 0.12	0
		01	DSR1 (UART1)	
		10	Match 1.0 (Timer 1)	
		11	RD4 (CAN 4)	
27:26	P0.13	00	GPIO Port 0.13	0
		01	DTR1 (UART1)	
		10	Match 1.1 (Timer 1)	
		11	TD4 (CAN 4)	
29:28	P0.14	00	GPIO Port 0.14	0
		01	DCD1 (UART1)	
		10	EINT1	
		11	Reserved	
31:30	P0.15	00	GPIO Port 0.15	0
		01	RI1 (UART1)	
		10	EINT2	
		11	Reserved	

[1] All functions on this pin are open-drain outputs for I^{2}C-bus compliance.

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

The PINSEL1 register controls the functions of the pins using the settings listed in Table 87. The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

The CAN bit settings are reserved for parts without CAN interfaces (see Table 82).

Table 87. Pin function Select register 1 (PINSEL1 - address 0xE002 C004) bit description

Bit	Symbol	Value	Function	Reset value
1:0	P0.16	00	GPIO Port 0.16	0
		01	EINT0	
		10	Match 0.2 (Timer 0)	
		11	Capture 0.2 (Timer 0)	
3:2	P0.17	00	GPIO Port 0.17	0
		01	Capture 1.2 (Timer 1)	
		10	SCK1 (SSP)	
		11	Match 1.2 (Timer 1)	
5:4	P0.18	00	GPIO Port 0.18	0
		01	Capture 1.3 (Timer 1)	
		10	MISO1 (SSP)	
		11	Match 1.3 (Timer 1)	
7:6	P0.19	00	GPIO Port 0.19	0
		01	Match 1.2 (Timer 1)	
		10	MOSI1 (SSP)	
		11	Capture 1.2 (Timer 1)	
9:8	P0.20	00	GPIO Port 0.20	0
		01	Match 1.3 (Timer 1)	
		10	SSEL1 (SSP)	
		11	EINT3	
11:10	P0.21	00	GPIO Port 0.21	0
		01	PWM5	
		10	RD3 (CAN 3)	
		11	Capture 1.3 (Timer 1)	
13:12	P0.22	00	GPIO Port 0.22	0
		01	TD3 (CAN 3)	
		10	Capture 0.0 (Timer 0)	
		11	Match 0.0 (Timer 0)	
15:14	P0.23	00	GPIO Port 0.23	0
		01	RD2 (CAN2)	
		10	Reserved	
		11	Reserved	

Table 87. Pin function Select register 1 (PINSEL1 - address 0xE002 C004) bit description

Bit	Symbol	Value	Function	Reset value
17:16	P0.24	00	GPIO Port 0.24	0
		01	TD2 (CAN2)	
		10	Reserved	
		11	Reserved	
19:18	P0.25	00	GPIO Port 0.25	0
		01	RD1 (CAN1)	
		10	Reserved	
		11	Reserved	
21:20	P0.26	00	Reserved	0
		01	Reserved	
		10	Reserved	
		11	Reserved	
23:22	P0.27	00	GPIO Port 0.27	01
		01	AIN0	
		10	CAP0.1 (Timer 0)	
		11	MAT0.1 (Timer 0)	
25:24	P0.28	00	GPIO Port 0.28	01
		01	AIN1	
		10	Capture 0.2 (Timer 0)	
		11	Match 0.2 (Timer 0)	
27:26	P0.29	00	GPIO Port 0.29	01
		01	AIN2	
		10	Capture 0.3 (Timer 0)	
		11	Match 0.3 (Timer 0)	
29:28	P0.30	00	GPIO Port 0.30	01
		01	AIN3	
		10	EINT3	
		11	Capture 0.0 (Timer 0)	
31:30	P0.31	00	Reserved	0
		01	Reserved	
		10	Reserved	
		11	Reserved	

8.6.3 LPC21xx Pin function Select register 2 (PINSEL2 - 0xE002 C014)

The PINSEL2 register controls the functions of the pins using the settings listed in Table 88. The direction control bit in the IO1DIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

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

Table 88. Pin function Select register 2 (PINSEL2 - 0xE002 C014) bit description

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

8.6.4 LPC22xx Pin function Select register 2 (PINSEL2 - 0xE002 C014)

The PINSEL2 register controls the functions of the pins using the settings listed in Table 89. 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.

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

Table 89. Pin function Select register 2 (PINSEL2 - 0xE002 C014) bit description

Bit	Symbol	Value	Function	Value after reset
1:0	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	GPIO/ DEBUG	Controls the use of P1.31-26 pins.		
		0	Pins P1.31-26 are used as GPIO pins.	$\overline{\text{P1.26}}$/$\overline{\text{RTCK}}$
		1	Pins P1.31-26 are used as a Debug port.	
3	GPIO/ TRACE	Controls the use of P1.25-16 pins.		$\overline{\text{P1.20}}$/ $\overline{\text{TRACESYNC}}$
		0	Pins P1.25-16 are used as GPIO pins.	
		1	Pins P1.25-16 are used as a Trace port.	

Table 89. Pin function Select register 2 (PINSEL2 - 0xE002 C014) bit description

Bit	Symbol	Value	Function	Value after reset
5:4	CTRLDBP	Controls the use of the data bus and strobe pins. At a reset triggered via the RESET pin, these bits are loaded with the content from lines BOOT1:0; if a watchdog reset occurs, these two bits are loaded with the BOOT10_SAVE register content (see Section 8.6.5 “Boot control for LPC22xx parts” on page 109).		BOOT1:0 or BOOT10_SAVE

	Functions available based on PINSEL2[5:4] values			
Pins	10	01	00	11
P1.1	OE			P1.1
P2.7:0	D7:0			P2.7:0
P2.15:8	D15:8		P2.15:8	
P2.27:16	D27:16	P2.27:16		
P2.29:28	D29:28	P2.29:28 or reserved (see bit 20)		
P2.30	D30	P2.30 or AIN4 (see bit 21)		
P2.31	D31	P2.31 or AIN5 (see bit 22)		
P1.0	CS0			P1.0
P3.31	BLS0			P3.31
P3.30	BLS1		P3.30	
P3.28	BLS2	P3.28 or AIN7 (see bit 7)		
P3.29	BLS3	P3.29 or AIN6 (see bit 6)		

6	CTRLP329	If bits 5:4 are not 10, controls the use of pin P3.29:		1
		0	P3.29 is a GPIO pin.	
		1	P3.29 is an ADC input pin (AIN6).	
7	CTRLP328	If bits 5:4 are not 10, controls the use of pin P3.28:		1
		0	P3.28 is a GPIO pin.	
		1	P3.28 is an ADC input pin (AIN7).	
8	CTRLP327	Controls the use of pin P3.27:		0
		0	P3.27 is a GPIO pin.	
		1	P3.27 is a Write Enable pin (WE).	
10:9	-		Reserved	-
11	CTRLP326	Controls the use of pin P3.26:		0
		0	P3.26 is a GPIO pin.	
		1	P3.26 is a chip/memory bank select pin (CS1).	
12	-	NA	Reserved	-
13	CTRLP323	If bits 27:25 are not 111, controls the use of pin P3.23/A23/XCLK:		0
		0	P3.23 is a GPIO/address line pin (see bits 27:25).	
		1	P3.23 is XCLK output pin.	

Table 89. Pin function Select register 2 (PINSEL2 - 0xE002 C014) bit description

Bit	Symbol	Value	Function	Value after reset
15:14	CTRLP325		Controls the use of pin P3.25:	00
		00	P3.25 is a GPIO pin.	
		01	P3.25 is a chip/memory bank select pin (CS2).	
		10	Reserved	
		11	Reserved	
17:16	CTRLP324		Controls the use of pin P3.24:	00
		00	P3.24 is a GPIO pin.	
		01	P3.24 is a chip/memory bank select pin (CS3).	
		10	Reserved	
		11	Reserved	
19:18	-	NA	Reserved	-
20	CTRLP229_28		If bits PINSEL2[5:4] are not 10, controls the use of pin P2.29:28:	0
		0	P2.29 and P2.28 are GPIO pins.	
		1	Reserved	
21	CTRLP230		If bits PINSEL2[5:4] are not 10, controls the use of pin P2.30:	1
		0	P2.30 is a GPIO pin.	
		1	P2.30 is an ADC input pin (AIN4).	
22	CTRLP231		If bits PINSEL2[5:4] are not 10, controls the use of pin P2.31:	1
		0	P2.31 is a GPIO pin.	
		1	P2.31 is an ADC input pin (AIN5).	
23	CTRLP300		Controls the use of pin P3.0:	1 if BOOT1:0 = 00 at RESET = 0, 0 otherwise
		0	P3.0/A0 is a GPIO pin.	
		1	P3.0/A0 is an address line.	
24	CTRLP301		Controls the use of pin P3.1:	$\overline{\text{BOOT1}}$ during Reset
		0	3.1/A1 is a GPIO pin.	
		1	3.1/A1 is an address line.	
27:25	CTRLAB		Controls the number of pins among P3.23/A23/XCLK and P3.22:2/A2.22:2 that are address lines:	000 if BOOT1:0 = 11 at Reset; 111 otherwise
		000	None	
		001	A3:2 are address lines.	
		010	A5:2 are address lines.	
		011	A7:2 are address lines.	
		100	A11:2 are address lines.	
		101	A15:2 are address lines.	
		110	A19:2 are address lines.	
		111	A23:2 are address lines.	
31:28	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	00

8.6.5 Boot control for LPC22xx parts

The state of the BOOT1:0 pins (P2.26 and P2.27) while RESET is low controls booting and initial operation. Internal pull-up resistors in the receivers ensure high state if a pin is left unconnected. Board designers can connect weak pull-down resistors (10 kΩ) or transistors that drive low while RESET is low to these pins to select among the following options:

Table 90. Boot control on BOOT1:0

P2.27/D27/BOOT1	P2.26/D26/BOOT0	Boot from
0	0	8 bit memory on CS0[1]
0	1	16 bit memory on CS0[1]
1	0	32 bit memory on CS0[1]
1	1	internal flash memory or 16 bit memory on CS0[1] for flashless parts LPC2210/20/90

[1] See Section 4.6 on how to connect external memory to the LPC22xx.

When the LPC22xx hardware detects a rising edge on the Reset pin, it latches content from BOOT[1:0] pins and stores it into bits 5 and 4 of the BOOT10_SAVE register (0x3FFF 8030). Once this register is written, it is accessible for reading only.

Whenever the boot loader is executed, it reads the content of the BOOT10_SAVE register, and configures the PINSEL2 (address and data bus structure) together with other resources. For the boot loader flowchart details, see Figure 73 for parts with flash and Figure 76 for flashless parts.

UM10114

Chapter 9: LPC21xx/22xx General Purpose I/O (GPIO) controller

Rev. 4 — 2 May 2012 **User manual**

9.1 How to read this chapter

For port 0 and port 1 , the GPIO can be selected to be Fast GPIO or legacy GPIO (see Section 9.5). Port 2 and port 3 are available in the 144-pin packages only and are always legacy GPIO. See table Table 91 for a list of LPC21xx and LPC22xx parts and their GPIO pins and available ports.

Not all pins are available on port 0 and port 1. The respective bits in the GPIO registers are reserved.

Table 91. GPIO features

Part	Legacy I/O ports Register base address				Fast GPIO ports Register base address	
	P0	P1	P2	P3	P0	P1
	0xE002 8000	0xE002 8010	0xE002 8020	0xE002 8030	0x3FFF C000	0x3FFF C020
no suffix and /00 parts						
LPC2109	P0[30:27], P0[25:0]	P1[31:16]	-	-	-	-
LPC2119	P0[30:27], P0[25:0]	P1[31:16]	-	-	-	-
LPC2129	P0[30:27], P0[25:0]	P1[31:16]	-	-	-	-
LPC2114	P0[30:27], P0[25:0]	P1[31:16]	-	-	-	-
LPC2124	P0[30:27], P0[25:0]	P1[31:16]	-	-	-	-
LPC2194	P0[30:27], P0[25:0]	P1[31:16]	-	-	-	-
LPC2210	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	-	-
LPC2220	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]
LPC2212	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	-	-
LPC2214	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	-	-
LPC2290	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	-	-
LPC2292	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	-	-
LPC2294	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	-	-
/01 parts						
LPC2109	P0[30:27], P0[25:0]	P1[31:16]	-	-	P0[30:27], P0[25:0]	P1[31:16]

Table 91. GPIO features

Part	Legacy I/O ports Register base address				Fast GPIO ports Register base address	
	P0	P1	P2	P3	P0	P1
	0xE002 8000	0xE002 8010	0xE002 8020	0xE002 8030	0x3FFF C000	0x3FFF C020
LPC2119	P0[30:27], P0[25:0]	P1[31:16]	-	-	P0[30:27], P0[25:0]	P1[31:16]
LPC2129	P0[30:27], P0[25:0]	P1[31:16]	-	-	P0[30:27], P0[25:0]	P1[31:16]
LPC2114	P0[30:27], P0[25:0]	P1[31:16]	-	-	P0[30:27], P0[25:0]	P1[31:16]
LPC2124	P0[30:27], P0[25:0]	P1[31:16]	-	-	P0[30:27], P0[25:0]	P1[31:16]
LPC2194	P0[30:27], P0[25:0]	P1[31:16]	-	-	P0[30:27], P0[25:0]	P1[31:16]
LPC2210	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]
LPC2212	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]
LPC2214	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]
LPC2290	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]
LPC2292	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]
LPC2294	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]	P2[31:0]	P3[31:0]	P0[30:27], P0[25:0]	P1[31:16], P1[1:0]

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 "How to read this manual".

9.2 Features

- Every physical GPIO port can be accessed either through registers providing enhanced features and accelerated port access or through legacy registers providing backward compatibility to earlier LPC2000 devices.
- Accelerated Fast GPIO functions (see Table 91):
  – GPIO registers are relocated to the ARM local bus so that the fastest possible I/O timing can be achieved.
  – Mask registers allow treating sets of port bits as a group, leaving other bits unchanged.
  – All registers are byte, half-word, and word addressable.
  – The entire port value can be written in one instruction.
- Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.
- Direction of each pin can be controlled individually.
- All I/O default to inputs after reset.

- Backward compatibility with other earlier devices is maintained with legacy registers appearing at the original addresses on the APB bus.

9.3 Applications

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

9.4 Pin description

Table 92. GPIO pin description

Pin	Type	Description
P031:0] P1[31:0]	Input/ Output	General purpose input/output. The number of GPIOs actually available depends on the use of alternate functions.
P2[31:0] P3[31:0]	Input/ Output	External bus data/address lines shared with GPIO, digital and analog functions. The number of GPIOs/digital and analog functions available depends on the selected bus structure.

9.5 Register description

LPC21xx/LPC22xx devices have two 32-bit General Purpose I/O ports. PORT0 and PORT1 are controlled by two groups of 4 registers as shown in Table 93 and Table 94. LPC22xx devices have two additional 32-bit ports, PORT2 and PORT3. These ports can be configured either as external memory data address and data bus or as GPIOs sharing pins with a handful of digital and analog functions. Details on PORT2 and PORT3 usage can be found in Section 8.6.4.

Legacy registers shown in Table 93 allow backward compatibility with earlier family devices, using existing code. The functions and relative timing of older GPIO implementations is preserved.

The registers in Table 94 represent the enhanced Fast GPIO features available on the PORT0 and PORT1 only. All of these registers are located directly on the local bus of the CPU for the fastest possible read and write timing. An additional feature has been added that provides byte and half-word addressability of all GPIO registers. A mask register allows treating groups of bits in a single GPIO port separately from other bits on the same port.

When PORT0 and/or PORT1 are used, the user must select whether a these ports will be accessed via registers that provide enhanced features or a legacy set of registers (see Section 6.7.1). While both of a port’s fast and legacy GPIO registers are controlling the same physical pins, these two port control branches are mutually exclusive and operate independently. For example, changing a pin’s output through a fast register will not be observable trough the corresponding legacy register.

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

The "slow", legacy registers are word accessible only. The “fast” GPIO registers are byte, half-word, and word accessible.

Table 93. GPIO register map (legacy APB accessible registers)

Generic Name	Description	Access	Reset value[1]	PORT0 Address & Name	PORT1 Address & Name	PORT2 Address & name	PORT3 Address & name
IOPIN	GPIO Port Pin value register. The current state of the GPIO configured port pins can always be read from this register, regardless of pin direction.	R/W	NA	0xE002 8000 IO0PIN	0xE002 8010 IO1PIN	0xE002 8020 IO2PIN	0xE002 8030 IO3PIN
IOSET	GPIO Port Output Set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces HIGHs at the corresponding port pins. Writing zeroes has no effect.	R/W	0x0000 0000	0xE002 8004 IO0SET	0xE002 8014 IO1SET	0xE002 8024 IO2SET	0xE002 8034 IO3SET
IODIR	GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0x0000 0000	0xE002 8008 IO0DIR	0xE002 8018 IO1DIR	0xE002 8028 IO2DIR	0xE002 8038 IO3DIR
IOCLR	GPIO Port Output Clear register. This register controls the state of output pins. Writing ones produces LOWs at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	WO	0x0000 0000	0xE002 800C IO0CLR	0xE002 801C IO1CLR	0xE002 802C IO2CLR	0xE002 803C IO3CLR

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

Table 94. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORT0 Address & Name	PORT1 Address & Name
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0x0000 0000	0x3FFF C000 FIO0DIR	0x3FFF C020 FIO1DIR
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN). Only the bits enabled by zeroes in this register are altered or cleared.	R/W	0x0000 0000	0x3FFF C010 FIO0MASK	0x3FFF C030 FIO1MASK

Table 94. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORT0 Address & Name	PORT1 Address & Name
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins is not configured as an input to ADC). The value read is masked by ANDing with FIOMASK. Writing to this register places corresponding values in all bits enabled by zeroes in FIOMASK.	R/W	0x0000 0000	0x3FFF C014 FIO0PIN	0x3FFF C034 FIO1PIN
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by zeroes in FIOMASK can be altered.	R/W	0x0000 0000	0x3FFF C018 FIO0SET	0x3FFF C038 FIO1SET
FIOCLR	Fast Port Output Clear register using FIOMASK0. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by zeroes in FIOMASK can be altered.	WO	0x0000 0000	0x3FFF C01C FIO0CLR	0x3FFF C03C FIO1CLR

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

9.5.1 GPIO port Direction register IODIR (IO0DIR - 0xE002 8008, IO1DIR - 0xE002 8018, IO2DIR - 0xE002 8028, IO3DIR - 0xE002 8038, FIO0DIR - 0x3FFF C000, FIO1DIR - 0x3FFF C020)

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

Legacy registers are the IO0DIR, IO1DIR, IO2DIR and IO3DIR while the enhanced GPIO functions are supported via the FIO0DIR and FIO1DIR registers.

Table 95. GPIO port 0 Direction register (IO0DIR - address 0xE002 8008) bit description

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

Table 96. GPIO port 1 Direction register (IO1DIR - address 0xE002 8018) bit description

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

Table 97. GPIO port 2 Direction register (IO2DIR - address 0xE002 8028) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P2xDIR		Slow GPIO Direction control bits. Bit 0 in IO2DIR controls P2.0 ... Bit 31 in IO2DIR controls P2.31.	0x0000 0000
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Table 98. GPIO port 3 Direction register (IO3DIR - address 0xE002 8038) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P3xDIR		Slow GPIO Direction control bits. Bit 0 in IO3DIR controls P3.0 ... Bit 31 in IO3DIR controls P3.31.	0x0000 0000
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Table 99. Fast GPIO port 0 Direction register (FIO0DIR - address 0x3FFF C000) bit description

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

Table 100. Fast GPIO port 1 Direction register (FIO1DIR - address 0x3FFF C020) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP1xDIR		Fast GPIO Direction control bits. Bit 0 in FIO1DIR controls P1.0 ... Bit 31 in FIO1DIR controls P1.31.	0x0000 0000
		0	Controlled pin is input.	
		1	Controlled pin is output.	

In addition to the 32-bit long and word only accessible FIODIR register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 101 and Table 102. Next to providing the same functions as the FIODIR register, these additional registers allow easier and faster access to the physical port pins.

Table 101. Fast GPIO port 0 Direction control byte and half-word accessible register description

Register name	Register length (bits) & access	Address	Description	Reset value
FIO0DIR0	8 (byte)	0x3FFF C000	Fast GPIO Port 0 Direction control register 0. Bit 0 in FIO0DIR0 register corresponds to P0.0 ... bit 7 to P0.7.	0x00
FIO0DIR1	8 (byte)	0x3FFF C001	Fast GPIO Port 0 Direction control register 1. Bit 0 in FIO0DIR1 register corresponds to P0.8 ... bit 7 to P0.15.	0x00
FIO0DIR2	8 (byte)	0x3FFF C002	Fast GPIO Port 0 Direction control register 2. Bit 0 in FIO0DIR2 register corresponds to P0.16 ... bit 7 to P0.23.	0x00

Table 101. Fast GPIO port 0 Direction control byte and half-word accessible register description

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

Table 102. Fast GPIO port 1 Direction control byte and half-word accessible register description

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

9.5.2 GPIO port output Set register IOSET (IO0SET - 0xE002 8004, IO1SET - 0xE002 8014, IO2SET - 0xE002 8024, IO3SET - 0xE002 8034, FIO0SET - 0x3FFF C018, FIO1SET - 0x3FFF C038)

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

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

Legacy registers are the IO0SET, IO1SET, IO2SET and IO3SET while the enhanced GPIOs are supported via the FIO0SET and FIO1SET registers. Access to a port pins via the FIOSET register is conditioned by the corresponding FIOMASK register (see Section 9.5.5 “Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)”).

Table 103. GPIO port 0 output Set register (IO0SET - address 0xE002 8004 bit description

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

Table 104. GPIO port 1 output Set register (IO1SET - address 0xE002 8014) bit description

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

Table 105. GPIO port 2 output Set register (IO2SET - address 0xE002 8024) bit description

Bit	Symbol	Description	Reset value
31:0	P2xSET	Slow GPIO output value Set bits. Bit 0 in IO2SET corresponds to P2.0 ... Bit 31 in IO2SET corresponds to P2.31.	0x0000 0000

Table 106. GPIO port 3 output Set register (IO3SET - address 0xE002 8034) bit description

Bit	Symbol	Description	Reset value
31:0	P3xSET	Slow GPIO output value Set bits. Bit 0 in IO3SET corresponds to P3.0 ... Bit 31 in IO3SET corresponds to P3.31.	0x0000 0000

Table 107. Fast GPIO port 0 output Set register (FIO0SET - address 0x3FFF C018) bit description

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

Table 108. Fast GPIO port 1 output Set register (FIO1SET - address 0x3FFF C038) bit description

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

Aside from the 32-bit long and word only accessible FIOSET register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 109 and Table 110. Next to providing the same functions as the FIOSET register, these additional registers allow easier and faster access to the physical port pins.

Table 109. Fast GPIO port 0 output Set byte and half-word accessible register description

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

Table 110. Fast GPIO port 1 output Set byte and half-word accessible register description

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

9.5.3 GPIO port output Clear register IOCLR (IO0CLR - 0xE002 800C, IO1CLR - 0xE002 801C, IO2CLR - 0xE002 802C, IO3CLR - 0xE002 803C, FIO0CLR - 0x3FFF C01C, FIO1CLR - 0x3FFF C03C)

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

Legacy registers are the IO0CLR, IO1CLR, IO2CLR and IO3CLR while the enhanced GPIOs are supported via the FIO0CLR and FIO1CLR registers. Access to a port pins via the FIOCLR register is conditioned by the corresponding FIOMASK register (see Section 9.5.5 “Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)”).

Table 111. GPIO port 0 output Clear register 0 (IO0CLR - address 0xE002 800C) bit description

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

Table 112. GPIO port 1 output Clear register 1 (IO1CLR - address 0xE002 801C) bit description

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

Table 113. GPIO port 2 output Clear register 2 (IO2CLR - address 0xE002 802C) bit description

Bit	Symbol	Description	Reset value
31:0	P2xCLR	Slow GPIO output value Clear bits. Bit 0 in IO2CLR corresponds to P1.0 ... Bit 31 in IO2CLR corresponds to P2.31.	0x0000 0000

Table 114. GPIO port 3 output Clear register 3 (IO3CLR - address 0xE002 803C) bit description

Bit	Symbol	Description	Reset value
31:0	P3xCLR	Slow GPIO output value Clear bits. Bit 0 in IO3CLR corresponds to P1.0 ... Bit 31 in IO3CLR corresponds to P2.31.	0x0000 0000

Table 115. Fast GPIO port 0 output Clear register 0 (FIO0CLR - address 0x3FFF C01C) bit description

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

Table 116. Fast GPIO port 1 output Clear register 1 (FIO1CLR - address 0x3FFF C03C) bit description

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

Aside from the 32-bit long and word only accessible FIOCLR register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 117 and Table 118. Next to providing the same functions as the FIOCLR register, these additional registers allow easier and faster access to the physical port pins.

Table 117. Fast GPIO port 0 output Clear byte and half-word accessible register description

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

Table 118. Fast GPIO port 1 output Clear byte and half-word accessible register description

Register name	Register length (bits) & access	Address	Description	Reset value
FIO1CLR0	8 (byte)	0x3FFF C03C	Fast GPIO Port 1 output Clear register 0. Bit 0 in FIO1CLR0 register corresponds to P1.0 ... bit 7 to P1.7.	0x00
FIO1CLR1	8 (byte)	0x3FFF C03D	Fast GPIO Port 1 output Clear register 1. Bit 0 in FIO1CLR1 register corresponds to P1.8 ... bit 7 to P1.15.	0x00
FIO1CLR2	8 (byte)	0x3FFF C03E	Fast GPIO Port 1 output Clear register 2. Bit 0 in FIO1CLR2 register corresponds to P1.16 ... bit 7 to P1.23.	0x00

Table 118. Fast GPIO port 1 output Clear byte and half-word accessible register description

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

9.5.4 GPIO port Pin value register IOPIN (IO0PIN - 0xE002 8000, IO1PIN - 0xE002 8010, IO2PIN - 0xE002 8020, IO3PIN - 0xE002 8030, FIO0PIN - 0x3FFF C014, FIO1PIN - 0x3FFF C034)

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

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

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

Legacy registers are the IO0PIN, IO1PIN, IO2PIN and IO3PIN while the enhanced GPIOs are supported via the FIO0PIN and FIO1PIN registers. Access to a port pins via the FIOPIN register is conditioned by the corresponding FIOMASK register (see Section 9.5.5 “Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)”).

Only pins masked with zeros in the Mask register (see Section 9.5.5 “Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)”) will be correlated to the current content of the Fast GPIO port pin value register.

Table 119. GPIO port 0 Pin value register (IO0PIN - address 0xE002 8000) bit description

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

Table 120. GPIO port 1 Pin value register (IO1PIN - address 0xE002 8010) bit description

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

Table 121. GPIO port 2 Pin value register (IO2PIN - address 0xE002 8020) bit description

Bit	Symbol	Description	Reset value
31:0	P2xVAL	Slow GPIO pin value bits. Bit 0 in IO2PIN corresponds to P1.0 ... Bit 31 in IO2PIN corresponds to P2.31.	NA

Table 122. GPIO port 3 Pin value register (IO3PIN - address 0xE002 8030) bit description

Bit	Symbol	Description	Reset value
31:0	P3xVAL	Slow GPIO pin value bits. Bit 0 in IO3PIN corresponds to P3.0 ... Bit 31 in IO3PIN corresponds to P3.31.	NA

Table 123. Fast GPIO port 0 Pin value register (FIO0PIN - address 0x3FFF C014) bit description

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

Table 124. Fast GPIO port 1 Pin value register (FIO1PIN - address 0x3FFF C034) bit description

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

Aside from the 32-bit long and word only accessible FIOPIN register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 125 and Table 126. Next to providing the same functions as the FIOPIN register, these additional registers allow easier and faster access to the physical port pins.

Table 125. Fast GPIO port 0 Pin value byte and half-word accessible register description

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

Table 126. Fast GPIO port 1 Pin value byte and half-word accessible register description

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

9.5.5 Fast GPIO port Mask register FIOMASK(FIO0MASK - 0x3FFF C010, FIO1MASK - 0x3FFF C030)

This register is available in the enhanced group of registers only. It is used to select the port pins that will and will not be affected by a write accesses to the FIOPIN, FIOSET or FIOSLR register. The mask register also filters the port’s content when the FIOPIN register is read.

A zero in this register’s bit enables an access to the corresponding physical pin via a read or write access. If a bit in this register is one, the corresponding pin will not be changed with write access and if read, will not be reflected in the updated FIOPIN register. For software examples, see Section 9.6 “GPIO usage notes” on page 124

Table 127. Fast GPIO port 0 Mask register (FIO0MASK - address 0x3FFF C010) bit description

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

Table 128. Fast GPIO port 1 Mask register (FIO1MASK - address 0x3FFF C030) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FP1xMASK		Fast GPIO physical pin access control.	0x0000 0000
		0	Pin is affected by writes to the FIOSET, FIOCLR, and FIOPIN registers. Current state of the pin will be observable in the FIOPIN register.	
		1	Physical pin is unaffected by writes into the FIOSET, FIOCLR and FIOPIN registers. When the FIOPIN register is read, this bit will not be updated with the state of the physical pin.	

Aside from the 32-bit long and word only accessible FIOMASK register, every fast GPIO port can also be controlled via several byte and half-word accessible registers listed in Table 129 and Table 130. Next to providing the same functions as the FIOMASK register, these additional registers allow easier and faster access to the physical port pins.

Table 129. Fast GPIO port 0 Mask byte and half-word accessible register description

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

Table 130. Fast GPIO port 1 Mask byte and half-word accessible register description

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

9.6 GPIO usage notes

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

The state of a GPIO pin configured as output is determined by writes into the pin's port IOSET and IOCLR registers. The last access to the IOSET/IOCLR register will determine the final output of the pin.

In the following code example

```
IO0DIR = 0x0000 0080 ;pin P0.7 configured as output
```

```
IO0CLR = 0x0000 0080 ;P0.7 goes LOW
IO0SET = 0x0000 0080 ;P0.7 goes HIGH
IO0CLR = 0x0000 0080 ;P0.7 goes LOW
```

pin P0.7 is configured as an output pin (write to IO0DIR register). Then, the P0.7 output pin is set to low (first write to IO0CLR register). A short high pulse follows on P0.7 (write access to IO0SET), and the second write to IO0CLR register sets pin P0.7 back to low level.

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

Writing 1’s to the port’s IOSET register (setting port output to HIGH) followed by writing 1’s to the IOCLR register (setting port output to LOW) causes a slight delay between the HIGH and LOW output at the port’s pins.

There are systems that can tolerate this delay of a valid output, but for some applications simultaneous output of a binary content (mixed 0s and 1s) within a group of pins on a single GPIO port is required. This can be accomplished by writing to the port’s IOPIN register.

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

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

The same outcome can be obtained using the fast port access.

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

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

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

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

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

```
FIO0PIN1  = 0xA5;
```

9.6.3 Writing to IOSET/IOCLR .vs. IOPIN

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

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

9.6.4 Output signal frequency considerations when using the legacy and enhanced GPIO registers

The enhanced features of fast GPIO ports available on this microcontroller make the performance of the GPIO pins more dependent on the details of the application code. In particular, software access to a GPIO pin is 3.5 times faster through the fast GPIO registers than through the legacy set of registers. As a result, the maximum output frequency of the digital pin is increased 3.5 times if the fast GPIO registers are used. This tremendous increase of the output frequency is less noticeable when plain C code is used. The portion of an application handling the fast port output should be written in assembly code and executed in the ARM mode to take full advantage of the fast GPIO access.

The following is a code example in which the pin control section is written in assembly language for ARM. It illustrates the difference between the fast and slow GPIO port output capabilities. For the best performances, compile this code in the ARM mode and execute from the on-chip SRAM memory.

```
        ldr   r0,=0xe01fc1a0  /*register address--enable fast port*/
        mov   r1,#0x1
        str   r1,[r0]         /*enable fast port0*/
        ldr   r1,=0xffffffff
        ldr   r0,=0x3fffc000  /*direction of fast port0*/
        str   r1,[r0]
        ldr   r0,=0xe0028018  /*direction of slow port 1*/
        str   r1,[r0]
        ldr   r0,=0x3fffc018  /*FIO0SET -- fast port0 register*/
        ldr   r1,=0x3fffc01c  /*FIO0CLR0 -- fast port0 register*/
        ldr   r2,=0x00001000  /*select fast port 0.12 for toggle*/
        ldr   r3,=0xE0028014  /*IO1SET -- slow port1 register*/
        ldr   r4,=0xE002801C  /*IO1CLR -- slow port1 register*/
        ldr   r5,=0x00100000  /*select slow port 1.20 for toggle*/
        /*Generate 2 pulses on the fast port*/
        str   r2,[r0]
        str   r2,[r1]
        str   r2,[r0]
        str   r2,[r1]
        /*Generate 2 pulses on the slow port*/
        str   r5,[r3]
        str   r5,[r4]
        str   r5,[r3]
        str   r5,[r4]
loop:   b     loop
```

Figure 24 illustrates the code from above executed from the LPC21xx/LPC22xx on-chip SRAM. The PLL generated F_{CCLK} =60 MHz out of external F_{OSC} = 12 MHz and VPBDIV = 1 (PCLK = CCLK).

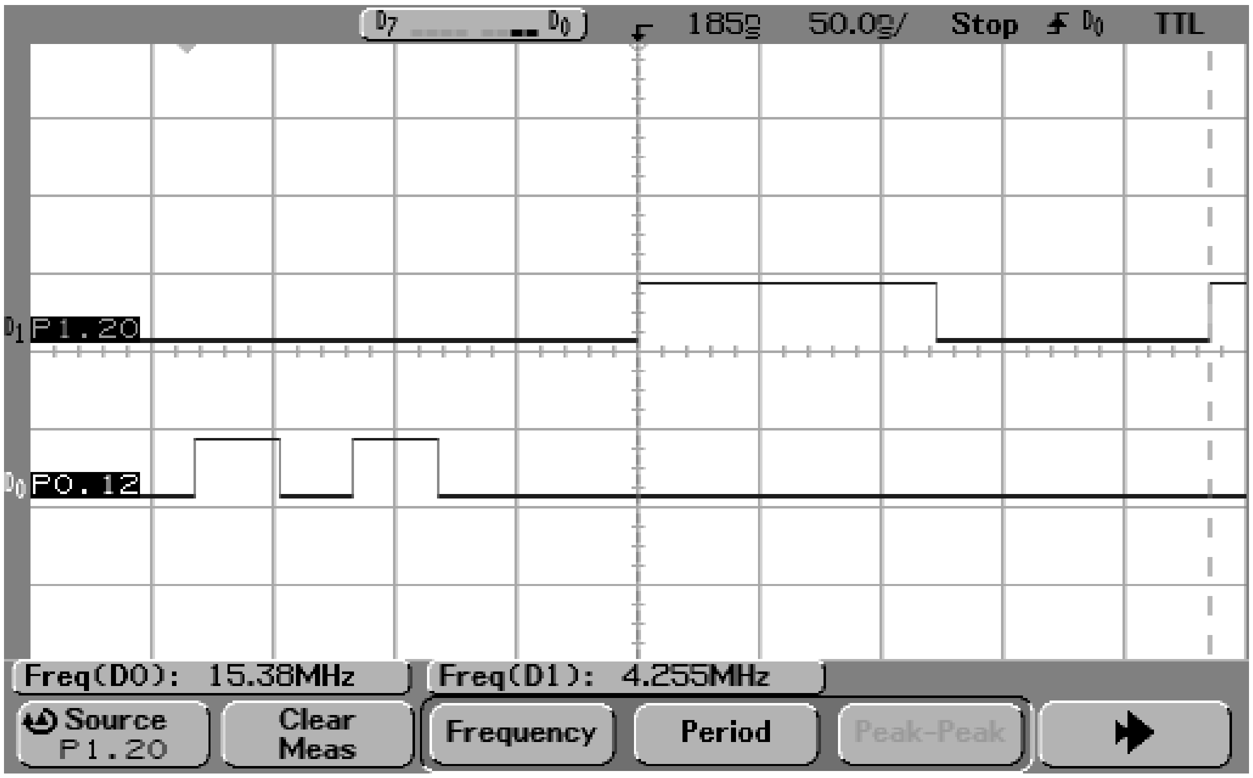


Fig 24. Illustration of the fast and slow GPIO access and output showing 3.5 x increase of the pin output frequency

UM10114

Chapter 10: LPC21xx/22xx Universal Asynchronous Receiver/Transmitter 0 (UART0)

Rev. 4 — 2 May 2012 **User manual**

10.1 How to read this chapter

The following features on the LPC21xx and LPC22xx are available in parts with enhanced features only:

- Fractional baud rate controller
- Auto-baud control
- Software flow control

Therefore, the registers controlling enhanced features are available only for /01 parts and LPC2220 (see Table 131).

The baud rate is determined by the register values U0DLL and U0DLM. Enhanced parts also include a fractional baud rate generator for fine-tuning the baud rate. The fractional baud rate settings are determined by the content of the U0FDR register.

Table 131. LPC21xx/22xx part-specific registers

Part	Baud rate			Auto-baud control			Software flow control
	Section 10.4.3		Section 10.4.4	Section 10.4.11	Section 10.4.5	Section 10.4.6	Section 10.4.12
no suffix and /00 parts							
LPC2109	U0DLL	U0DLM	-	-	-	-	-
LPC2119	U0DLL	U0DLM	-	-	-	-	-
LPC2129	U0DLL	U0DLM	-	-	-	-	-
LPC2114	U0DLL	U0DLM	-	-	-	-	-
LPC2124	U0DLL	U0DLM	-	-	-	-	-
LPC2194	U0DLL	U0DLM	-	-	-	-	-
LPC2210	U0DLL	U0DLM	-	-	-	-	-
LPC2220	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2212	U0DLL	U0DLM	-	-	-	-	-
LPC2214	U0DLL	U0DLM	-	-	-	-	-
LPC2290	U0DLL	U0DLM	-	-	-	-	-
LPC2292	U0DLL	U0DLM	-	-	-	-	-
LPC2294	U0DLL	U0DLM	-	-	-	-	-
/01 parts							
LPC2109	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2119	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2129	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2114	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2124	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2194	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2210	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER

Table 131. LPC21xx/22xx part-specific registers

Part	Baud rate			Auto-baud control			Software flow control
	Section 10.4.3		Section 10.4.4	Section 10.4.11	Section 10.4.5	Section 10.4.6	Section 10.4.12
LPC2212	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2214	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2290	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2292	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER
LPC2294	U0DLL	U0DLM	U0FDR	U0ACR	U0IER, bits 9:8	U0IIR, bits 9:8	U0TER

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

10.2 Features

- 16 byte Receive and Transmit FIFOs
- Register locations conforming to ‘550 industry standard
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes
- Built-in fractional baud rate generator with autobauding capabilities.
- Mechanism that enables software and hardware flow control implementation

10.3 Pin description

Table 132: UART0 pin description

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

10.4 Register description

UART0 contains registers organized as shown in Table 133. The Divisor Latch Access Bit (DLAB) is contained in U0LCR[7] and enables access to the Divisor Latches.

The divisor latches are used to determine the baud rate for all UART transfers. When setting up the part, follow these steps:

1. Set DLAB = 1 in U0LCR (Section 10.4.8).
2. Set baud rate by writing values to registers DLL and DLM at address 0xE000 C000 Section 10.4.3).
3. Set DLAB = 0 in U0LCR (Section 10.4.8).
4. Read at address 0xE000 C000 accesses the U0RBR register (Section 10.4.1).
5. Write at address 0xE000 C000 accesses the U0THR register (Section 10.4.2).

Table 133. UART0 register map

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

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

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

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

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

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

Table 134: UART0 Receiver Buffer Register (U0RBR - address 0xE000 C000, when DLAB = 0, Read Only) bit description

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

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

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

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

Table 135: UART0 Transmit Holding Register (U0THR - address 0xE000 C000, when DLAB = 0, Write Only) bit description

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

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

The UART0 Divisor Latch is part of the UART0 Baud Rate Generator and holds the value used to divide the clock in order to produce the baud rate clock, which must be 16x the desired baud rate (Equation 1). The U0DLL and U0DLM registers together form a 16 bit divisor where U0DLL contains the lower 8 bits of the divisor and U0DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U0LCR must be one in order to access the UART0 Divisor Latches.

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL)} \tag{1}$$

Details on how to select the right value for U0DLL and U0DLM if the part includes a fractional divider (see Table 131) can be found later on in this chapter.

Table 136: UART0 Divisor Latch LSB register (U0DLL - address 0xE000 C000, when DLAB = 1) bit description

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

Table 137: UART0 Divisor Latch MSB register (U0DLM - address 0xE000 C004, when DLAB = 1) bit description

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

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

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

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be 3 or greater.

Table 138: UARTn Fractional Divider Register (U0FDR - address 0xE000 C028, U2FDR - 0xE007 8028, U3FDR - 0xE007 C028) bit description

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

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

The UART0 baudrate can be calculated as (n = 0):

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)} \tag{2}$$

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

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $0 <$ MULVAL ≤ 15
2. $0 \leq$ DIVADDVAL < 15
3. DIVADDVAL<MULVAL

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

If the U0FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

10.4.4.1 Baudrate calculation

UART can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baudrate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baudrate with a relative error of less than 1.1% from the desired one.

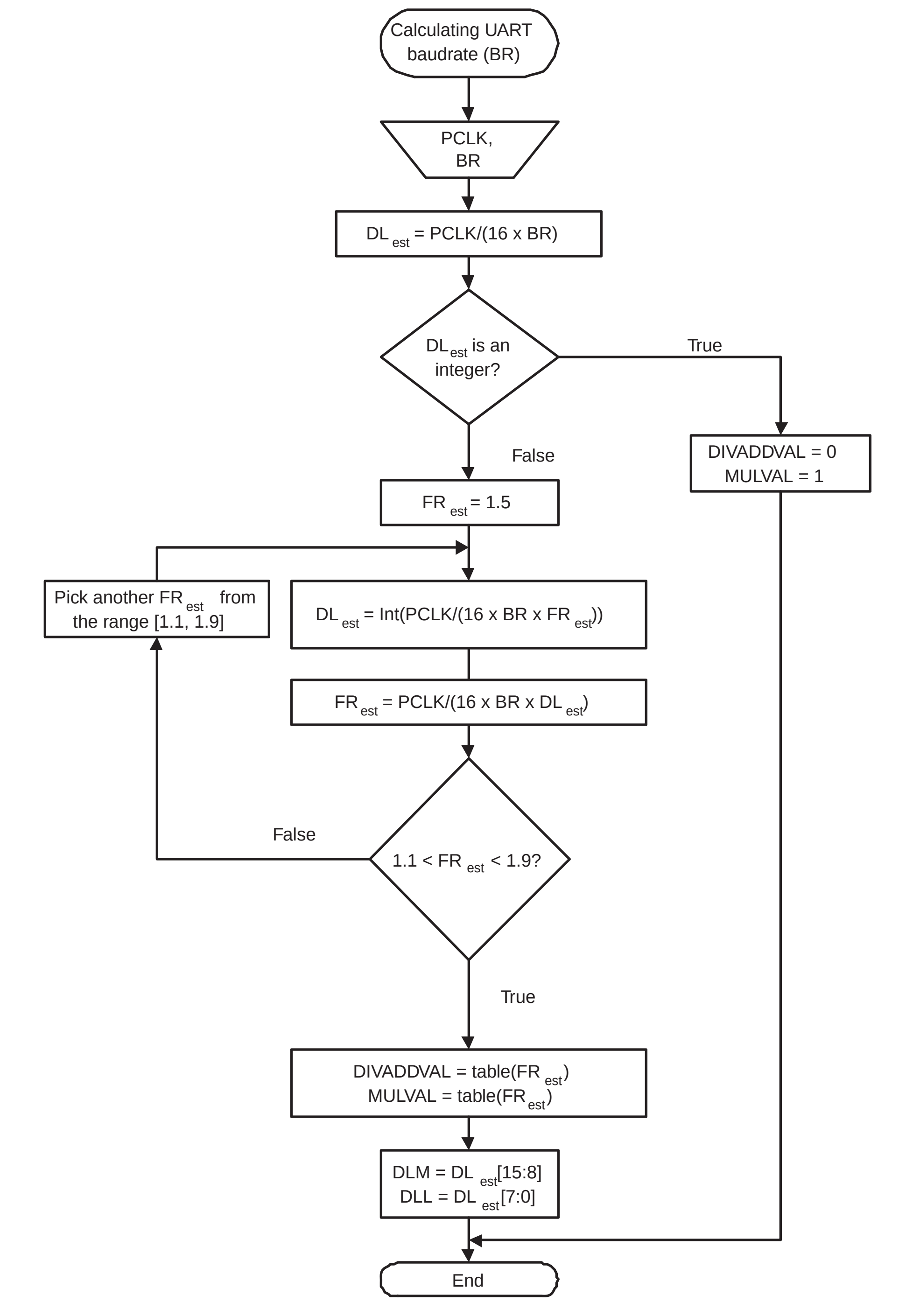


Fig 25. Algorithm for setting UART dividers

Table 139. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

10.4.4.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

10.4.4.1.2 Example 2: PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 139 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 2 the UART's baud rate is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

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

The U0IER is used to enable UART0 interrupt sources.

Table 140. UART0 Interrupt Enable Register (U0IER - address 0xE000 C004, when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset value
0	RBR Interrupt Enable		U0IER[0] enables the Receive Data Available interrupt for UART0. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THRE Interrupt Enable		U0IER[1] enables the THRE interrupt for UART0. The status of this can be read from U0LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RX Line Status Interrupt Enable		U0IER[2] enables the UART0 RX line status interrupts. The status of this interrupt can be read from U0LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
7:3	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	ABEOIntEn		Enables the end of auto-baud interrupt.	0
		0	Disable End of Auto-baud Interrupt.	
		1	Enable End of Auto-baud Interrupt.	
9	ABTOIntEn		Enables the auto-baud time-out interrupt.	0
		0	Disable Auto-baud Time-out Interrupt.	
		1	Enable Auto-baud Time-out Interrupt.	
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

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

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

Table 141: UART0 Interrupt Identification Register (U0IIR - address 0xE000 C008, read only) bit description

Bit	Symbol	Value	Description	Reset value
0	Interrupt Pending		Note that U0IIR[0] is active LOW. The pending interrupt can be determined by evaluating U0IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No pending interrupts.	

Table 141: UART0 Interrupt Identification Register (U0IIR - address 0xE000 C008, read only) bit description

Bit	Symbol	Value	Description	Reset value
3:1	Interrupt Identification		U0IER[3:1] identifies an interrupt corresponding to the UART0 Rx FIFO. All other combinations of U0IER[3:1] not listed above are reserved (000,100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		These bits are equivalent to U0FCR[0].	0
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Interrupts are handled as described in Table 142. Given the status of U0IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U0IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

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

The UART0 RDA interrupt (U0IIR[3:1] = 010) shares the second level priority with the CTI interrupt (U0IIR[3:1] = 110). The RDA is activated when the UART0 Rx FIFO reaches the trigger level defined in U0FCR[7:6] and is reset when the UART0 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (U0IIR[3:1] = 110) is a second level interrupt and is set when the UART0 Rx FIFO contains at least one character and no UART0 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART0 Rx FIFO activity (read or write of UART0 RSR) will clear the interrupt. This interrupt is intended to flush the UART0 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 142: UART0 interrupt handling

U0IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	U0LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U0FCR0=1)	U0RBR Read[3] or UART0 FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) $\times$ 7 – 2] $\times$ 8 + [(trigger level – number of characters) $\times$ 8 + 1] RCLKs	U0RBR Read[3]
0010	Third	THRE	THRE[2]	U0IIR Read (if source of interrupt) or THR write[4]

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 10.4.9 “UART0 Line Status Register (U0LSR - 0xE000 C014, Read Only)”

[3] For details see Section 10.4.1 “UART0 Receiver Buffer register (U0RBR - 0xE000 C000, when DLAB = 0, Read Only)”

[4] For details see Section 10.4.6 “UART0 Interrupt Identification Register (U0IIR - 0xE000 C008, Read Only)” and Section 10.4.2 “UART0 Transmit Holding Register (U0THR - 0xE000 C000, when DLAB = 0, Write Only)”

The UART0 THRE interrupt (U0IIR[3:1] = 001) is a third level interrupt and is activated when the UART0 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART0 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE=1 and there have not been at least two characters in the U0THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U0THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART0 THR FIFO has held two or more characters at one time and currently, the U0THR is empty. The THRE interrupt is reset when a U0THR write occurs or a read of the U0IIR occurs and the THRE is the highest interrupt (U0IIR[3:1] = 001).

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

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

Table 143: UART0 FIFO Control Register (U0FCR - address 0xE000 C008) bit description

Bit	Symbol	Value	Description	Reset value
0	FIFO Enable	0	UART0 FIFOs are disabled. Must not be used in the application.	0
		1	Active HIGH enable for both UART0 Rx and TX FIFOs and U0FCR[7:1] access. This bit must be set for proper UART0 operation. Any transition on this bit will automatically clear the UART0 FIFOs.	

Table 143: UART0 FIFO Control Register (U0FCR - address 0xE000 C008) bit description

Bit	Symbol	Value	Description	Reset value
1	RX FIFO Reset	0	No impact on either of UART0 FIFOs.	0
		1	Writing a logic 1 to U0FCR[1] will clear all bytes in UART0 Rx FIFO and reset the pointer logic. This bit is self-clearing.	
2	TX FIFO Reset	0	No impact on either of UART0 FIFOs.	0
		1	Writing a logic 1 to U0FCR[2] will clear all bytes in UART0 TX FIFO and reset the pointer logic. This bit is self-clearing.	
5:3	-	0	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RX Trigger Level		These two bits determine how many receiver UART0 FIFO characters must be written before an interrupt is activated.	0
		00	trigger level 0 (1 character or 0x01).	
		01	trigger level 1 (4 characters or 0x04).	
		10	trigger level 2 (8 characters or 0x08).	
		11	trigger level 3 (14 characters or 0x0E).	

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

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

Table 144: UART0 Line Control Register (U0LCR - address 0xE000 C00C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	Word Length Select	00	5 bit character length	0
		01	6 bit character length	
		10	7 bit character length	
		11	8 bit character length	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if U0LCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UART0 TXD is forced to logic 0 when U0LCR[6] is active HIGH.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

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

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

Table 145: UART0 Line Status Register (U0LSR - address 0xE000 C014, read only) bit description

Bit	Symbol	Value	Description	Reset value
0	Receiver Data Ready (RDR)		U0LSR0 is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty.	0
		0	U0RBR is empty.	
		1	U0RBR contains valid data.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR[2]. Time of parity error detection is dependent on U0FCR[0]. **Note:** A parity error is associated with the character at the top of the UART0 RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR[3]. The time of the framing error detection is dependent on U0FCR0. Upon detection of a framing error, the Rx will attempt to re-synchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UART0 RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXD0 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD0 goes to marking state (all 1's). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART0 RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE))		THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.	1
		0	U0THR contains valid data.	
		1	U0THR is empty.	

Table 145: UART0 Line Status Register (U0LSR - address 0xE000 C014, read only) bit description

Bit	Symbol	Value	Description	Reset value
6	Transmitter Empty (TEMT)		TEMT is set when both U0THR and U0TSR are empty; TEMT is cleared when either the U0TSR or the U0THR contain valid data.	1
		0	U0THR and/or the U0TSR contains valid data.	
		1	U0THR and the U0TSR are empty.	
7	Error in RX FIFO (RXFE)		U0LSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO.	0
		0	U0RBR contains no UART0 RX errors or U0FCR[0]=0.	
		1	UART0 RBR contains at least one UART0 RX error.	

10.4.10 UART0 Scratch Pad Register (U0SCR - 0xE000 C01C)

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

Table 146: UART0 Scratch Pad Register (U0SCR - address 0xE000 C01C) bit description

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

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

The UART0 Auto-baud Control Register (U0ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user’s discretion.

Table 147: Auto-baud Control Register (U0ACR - 0xE000 C020) bit description

Bit	Symbol	Value	Description	Reset value
0	Start		This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running).Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	Mode		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AutoRestart	0	No restart	0
		1	Restart in case of time-out (counter restarts at next UART0 Rx falling edge)	
7:3	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

Table 147: Auto-baud Control Register (U0ACR - 0xE000 C020) bit description

Bit	Symbol	Value	Description	Reset value
8	ABEOIntClr		End of auto-baud interrupt clear bit (write only accessible). Writing a 1 will clear the corresponding interrupt in the U0IIR. Writing a 0 has no impact.	0
9	ABTOIntClr		Auto-baud time-out interrupt clear bit (write only accessible). Writing a 1 will clear the corresponding interrupt in the U0IIR. Writing a 0 has no impact.	0
31:10	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

10.4.11.1 Auto-baud

The UART0 auto-baud function can be used to measure the incoming baud-rate based on the ”AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers U0DLM and U0DLL accordingly.

Auto-baud is started by setting the U0ACR Start bit. Auto-baud can be stopped by clearing the U0ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the U0ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART0 Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UART0 Rx pin (the length of the start bit).

The U0ACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART0 Rx pin.

The auto-baud function can generate two interrupts.

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

The auto-baud interrupts have to be cleared by setting the corresponding U0ACR ABTOIntClr and ABEOIntEn bits.

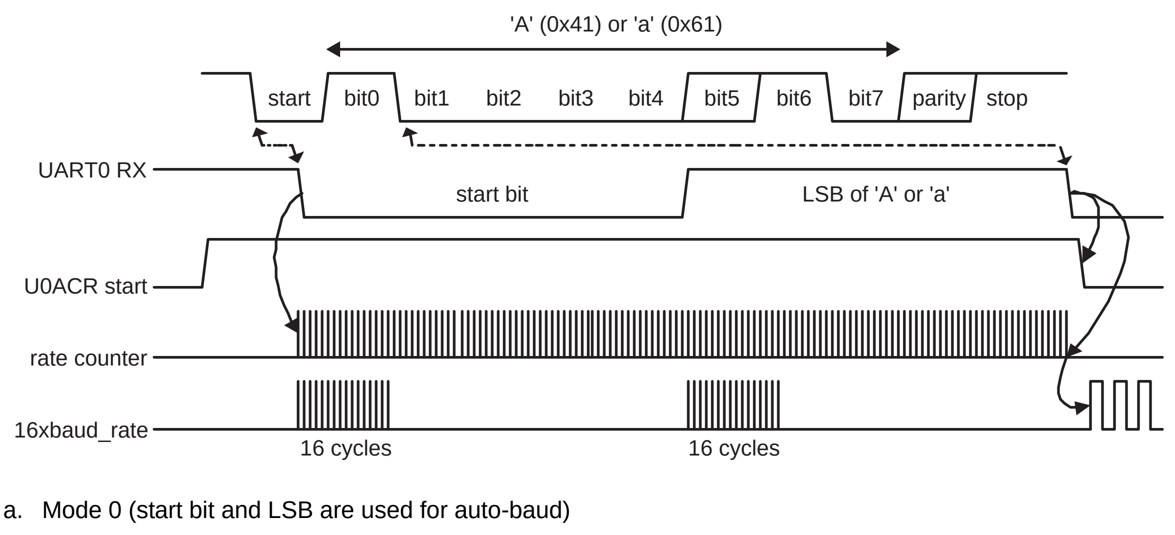
Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART0 Rx pin baud-rate, but the value of the U0FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U0DLM and U0DLL registers should be done before U0ACR register write. The minimum and the maximum baudrates supported by UART0 are function of PCLK, number of data bits, stop-bits and parity bits.

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART0_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax \quad (3)$$

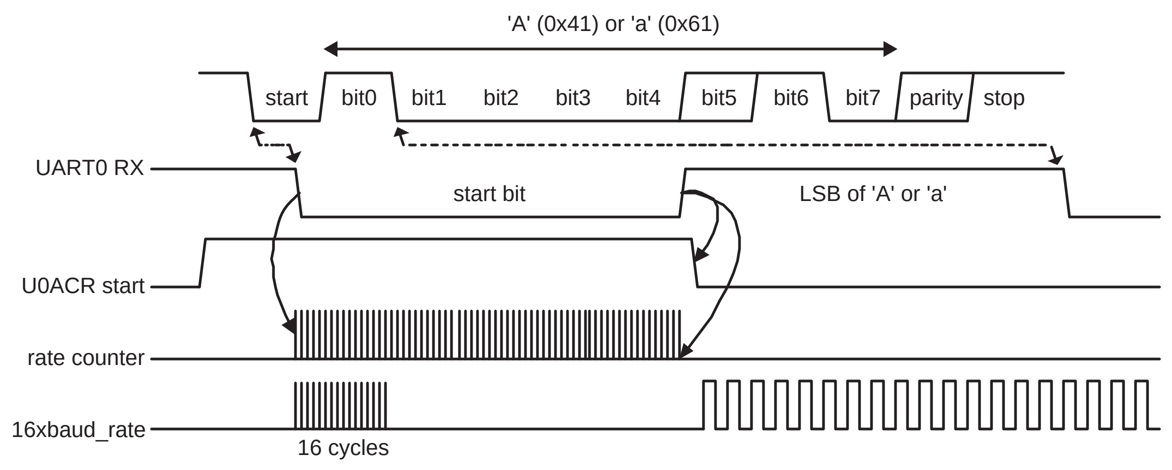
10.4.11.2 Auto-baud modes

When the software is expecting an ”AT" command, it configures the UART0 with the expected character format and sets the U0ACR Start bit. The initial values in the divisor latches U0DLM and U0DLM don‘t care. Because of the ”A" or ”a" ASCII coding (”A" = 0x41, ”a" = 0x61), the UART0 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U0ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U0ACR Start bit setting, the baud-rate measurement counter is reset and the UART0 U0RSR is reset. The U0RSR baud rate is switch to the highest rate.
2. A falling edge on UART0 Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting PCLK cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UART0 input clock, guaranteeing the start bit is stored in the U0RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART0 input clock (PCLK).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART0 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART0 Rx pin.
6. The rate counter is loaded into U0DLM/U0DLL and the baud-rate will be switched to normal operation. After setting the U0DLM/U0DLL the end of auto-baud interrupt U0IIR ABEOInt will be set, if enabled. The U0RSR will now continue receiving the remaining bits of the ”A/a" character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 26. Autobaud a) mode 0 and b) mode 1 waveform.

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

The U0TER enables implementation of software flow control. When TXEn=1, UART0 transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART0 transmission will stop.

Table 148 describes how to use TXEn bit in order to achieve software flow control.

Table 148: UART0 Transmit Enable Register (U0TER - address 0xE000 C030) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit is cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software implementing software-handshaking can clear this bit when it receives an XOFF character (DC3). Software can set this bit again when it receives an XON (DC1) character.	1

10.5 Architecture

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

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

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

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

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

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

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

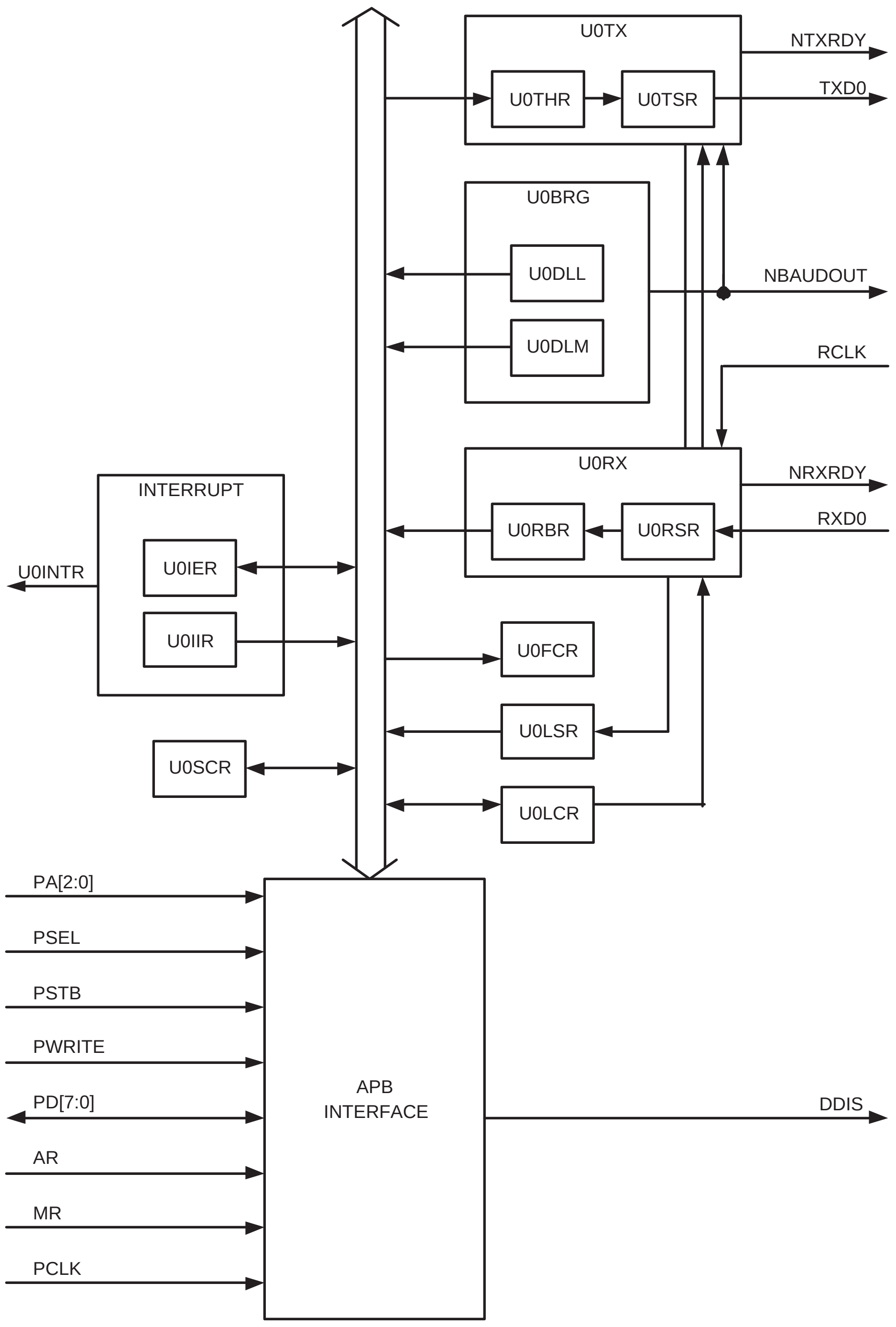


Fig 27. UART0 block diagram

UM10114

Chapter 11: LPC21xx/22xx Universal Asynchronous Receiver/Transmitter 1 (UART1)

Rev. 4 — 2 May 2012 **User manual**

11.1 How to read this chapter

The following features on the LPC21xx and LPC22xx are available in parts with enhanced features only:

- Fractional baud rate controller
- Auto-baud control
- Software flow control

Therefore, the registers controlling enhanced features are available only for /01 parts and LPC2220 (see Table 149).

The baud rate is determined by the register values U1DLL and U1DLM. Enhanced parts also include a fractional baud rate generator for fine-tuning the baud rate. The fractional baud rate settings are determined by the content of the U1FDR register.

Table 149. LPC21xx/22xx part-specific registers

Part	Baud rate			Auto-baud control			Software flow control
	Section 11.4.3		Section 11.4.4	Section 11.4.13	Section 11.4.5	Section 11.4.6	Section 11.4.16
no suffix and /00 parts							
LPC2109	U1DLL	U1DLM	-	-	-	-	-
LPC2119	U1DLL	U1DLM	-	-	-	-	-
LPC2129	U1DLL	U1DLM	-	-	-	-	-
LPC2114	U1DLL	U1DLM	-	-	-	-	-
LPC2124	U1DLL	U1DLM	-	-	-	-	-
LPC2194	U1DLL	U1DLM	-	-	-	-	-
LPC2210	U1DLL	U1DLM	-	-	-	-	-
LPC2220	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2212	U1DLL	U1DLM	-	-	-	-	-
LPC2214	U1DLL	U1DLM	-	-	-	-	-
LPC2290	U1DLL	U1DLM	-	-	-	-	-
LPC2292	U1DLL	U1DLM	-	-	-	-	-
LPC2294	U1DLL	U1DLM	-	-	-	-	-
/01 parts							
LPC2109	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2119	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2129	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2114	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2124	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2194	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2210	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER

Table 149. LPC21xx/22xx part-specific registers

Part	Baud rate			Auto-baud control			Software flow control
	Section 11.4.3		Section 11.4.4	Section 11.4.13	Section 11.4.5	Section 11.4.6	Section 11.4.16
LPC2212	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2214	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2290	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2292	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER
LPC2294	U1DLL	U1DLM	U1FDR	U1ACR	U1IER, bits 9:8	U1IIR, bits 9:8	U1TER

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 "How to read this manual".

11.2 Features

- UART1 is identical to UART0 with the addition of a modem interface.
- UART1 contains 16 byte Receive and Transmit FIFOs.
- Register locations conform to '550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Fractional baud rate generator with autobauding capabilities is built-in.
- Mechanism enables software and hardware flow control implementation.
- Standard modem interface signals are included, and flow control (auto-CTS/RTS) is fully supported in hardware.

11.3 Pin description

Table 150. UART1 pin description

Pin	Type	Description
RXD1	Input	**Serial Input.** Serial receive data.
TXD1	Output	**Serial Output.** Serial transmit data.
CTS1	Input	**Clear To Send.** Active LOW signal indicates if the external modem is ready to accept transmitted data via TXD1 from the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[4]. State change information is stored in U1MSR[0] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
DCD1	Input	**Data Carrier Detect.** Active LOW signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged. In normal operation of the modem interface (U1MCR[4]=0), the complement value of this signal is stored in U1MSR[7]. State change information is stored in U1MSR3 and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
DSR1	Input	**Data Set Ready.** Active LOW signal indicates if the external modem is ready to establish a communications link with the UART1. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[5]. State change information is stored in U1MSR[1] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).

Table 150. UART1 pin description

Pin	Type	Description
DTR1	Output	Data Terminal Ready. Active LOW signal indicates that the UART1 is ready to establish connection with external modem. The complement value of this signal is stored in U1MCR[0].
RI1	Input	**Ring Indicator.** Active LOW signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (U1MCR[4] = 0), the complement value of this signal is stored in U1MSR[6]. State change information is stored in U1MSR[2] and is a source for a priority level 4 interrupt, if enabled (U1IER[3] = 1).
RTS1	Output	**Request To Send.** Active LOW signal indicates that the UART1 would like to transmit data to the external modem. The complement value of this signal is stored in U1MCR[1].

11.4 Register description

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

The divisor latches are used to determine the baud rate for all UART transfers. When setting up the part, follow these steps:

1. Set DLAB = 1 in U1LCR (Section 11.4.10).
2. Set baud rate by writing values to registers DLL and DLM at address 0xE000 C000 Section 11.4.3).
3. Set DLAB = 0 in U1LCR (Section 11.4.8).
4. Read at address 0xE000 C000 accesses the U1RBR register (Section 11.4.1).
5. Write at address 0xE000 C000 accesses the U1THR register (Section 11.4.2).

Table 151. UART1 register map

Name	Description	Bit functions and addresses								Access	Reset value[1]	Address
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U1RBR	Receiver Buffer Register	8-bit Read Data								RO	NA	0xE001 0000 (DLAB=0)
U1THR	Transmit Holding Register	8-bit Write Data								WO	NA	0xE001 0000 (DLAB=0)
U1DLL	Divisor Latch LSB	8-bit Data								R/W	0x01	0xE001 0000 (DLAB=1)
U1DLM	Divisor Latch MSB	8-bit Data								R/W	0x00	0xE001 0004 (DLAB=1)
U1IER	Interrupt Enable Register	-	-	-	-	-	-	En.ABTO	En.ABEO	R/W	0x00	0xE001 0004 (DLAB=0)
		En.CTS Int	-	-	-	E.Modem St.Int	En. RX Lin.St. Int	Enable THRE Int	En. RX Dat.Av.Int			
U1IIR	Interrupt ID Reg.	-	-	-	-	-	-	ABTO Int	ABEO Int	RO	0x01	0xE001 0008
		FIFOs Enabled		-	-	IIR3	IIR2	IIR1	IIR0			
U1FCR	FIFO Control Register	RX Trigger		-	-	-	TX FIFO Reset	RX FIFO Reset	FIFO Enable	WO	0x00	0xE001 0008
U1LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Par.Selct.	Parity Enable	No. of Stop Bits	Word Length Select		R/W	0x00	0xE001 000C
U1MCR	Modem Ctrl. Reg.	CTSen	RTSen	-	LoopBck.	-	-	RTS	DTR	R/W	0x00	0xE001 0010
U1LSR	Line Status Register	RX FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE001 0014
U1MSR	Modem Status Register	DCD	RI	DSR	CTS	Delta DCD	Trailing Edge RI	Delta DSR	Delta CTS	RO	0x00	0xE001 0018
U1SCR	Scratch Pad Reg.	8-bit Data								R/W	0x00	0xE001 001C
U1ACR	Auto-baud Control Register	-	-	-	-	-	-	ABTO IntClr	ABEO IntClr	R/W	0x00	0xE001 0020
		-	-	-	-	-	Aut.Rstrt.	Mode	Start			
U1FDR	Fractional Divider Register	Reserved[31:8]								R/W	0x10	0xE001 0028
		MulVal				DivAddVal						
U1TER	TX. Enable Reg.	TXEN	-	-	-	-	-	-	-	R/W	0x80	0xE001 0030

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

11.4.1 UART1 Receiver Buffer Register (U1RBR - 0xE001 0000, when DLAB = 0 Read Only)

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

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

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

Table 152. UART1 Receiver Buffer Register (U1RBR - address 0xE001 0000, when DLAB = 0 Read Only) bit description

Bit	Symbol	Description	Reset value
7:0	RBR	The UART1 Receiver Buffer Register contains the oldest received byte in the UART1 RX FIFO.	undefined

11.4.2 UART1 Transmitter Holding Register (U1THR - 0xE001 0000, when DLAB = 0 Write Only)

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

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

Table 153. UART1 Transmitter Holding Register (U1THR - address 0xE001 0000, when DLAB = 0 Write Only) bit description

Bit	Symbol	Description	Reset value
7:0	THR	Writing to the UART1 Transmit Holding Register causes the data to be stored in the UART1 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA

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

The UART0 Divisor Latch is part of the UART0 Baud Rate Generator and holds the value used to divide the clock in order to produce the baud rate clock, which must be 16x the desired baud rate (Equation 4). The U1DLL and U1DLM registers together form a 16 bit divisor where U1DLL contains the lower 8 bits of the divisor and U1DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U1LCR must be one in order to access the UART1 Divisor Latches.

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL)} \tag{4}$$

Details on how to select the right value for U1DLL and U1DLM if the part includes a fractional divider (see Table 149) can be found later on in this chapter.

Table 154: UART1 Divisor Latch LSB register (U1DLL - address 0xE001 C000, when DLAB = 1) bit description

Bit	Symbol	Description	Reset value
7:0	DLL	The UART0 Divisor Latch LSB Register, along with the U1DLM register, determines the baud rate of the UART1.	0x01

Table 155: UART0 Divisor Latch MSB register (U1DLM - address 0xE001 C004, when DLAB = 1) bit description

Bit	Symbol	Description	Reset value
7:0	DLM	The UART1 Divisor Latch MSB Register, along with the U1DLL register, determines the baud rate of the UART1.	0x00

11.4.4 UART1 Fractional Divider Register (U1FDR - 0xE001 0028)

The UART1 Fractional Divider Register (U1FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be 3 or greater.

Table 156. UART1 Fractional Divider Register (U1FDR - address 0xE001 0028) bit description

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

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

UART1 baudrate can be calculated as (n = 1):

(5)

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

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

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. 0 < MULVAL ≤ 15
2. 0 ≤ DIVADDVAL < 15
3. DIVADDVAL<MULVAL

The value of the U1FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the U1FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

11.4.4.1 Baudrate calculation

UART can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baudrate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baudrate with a relative error of less than 1.1% from the desired one.

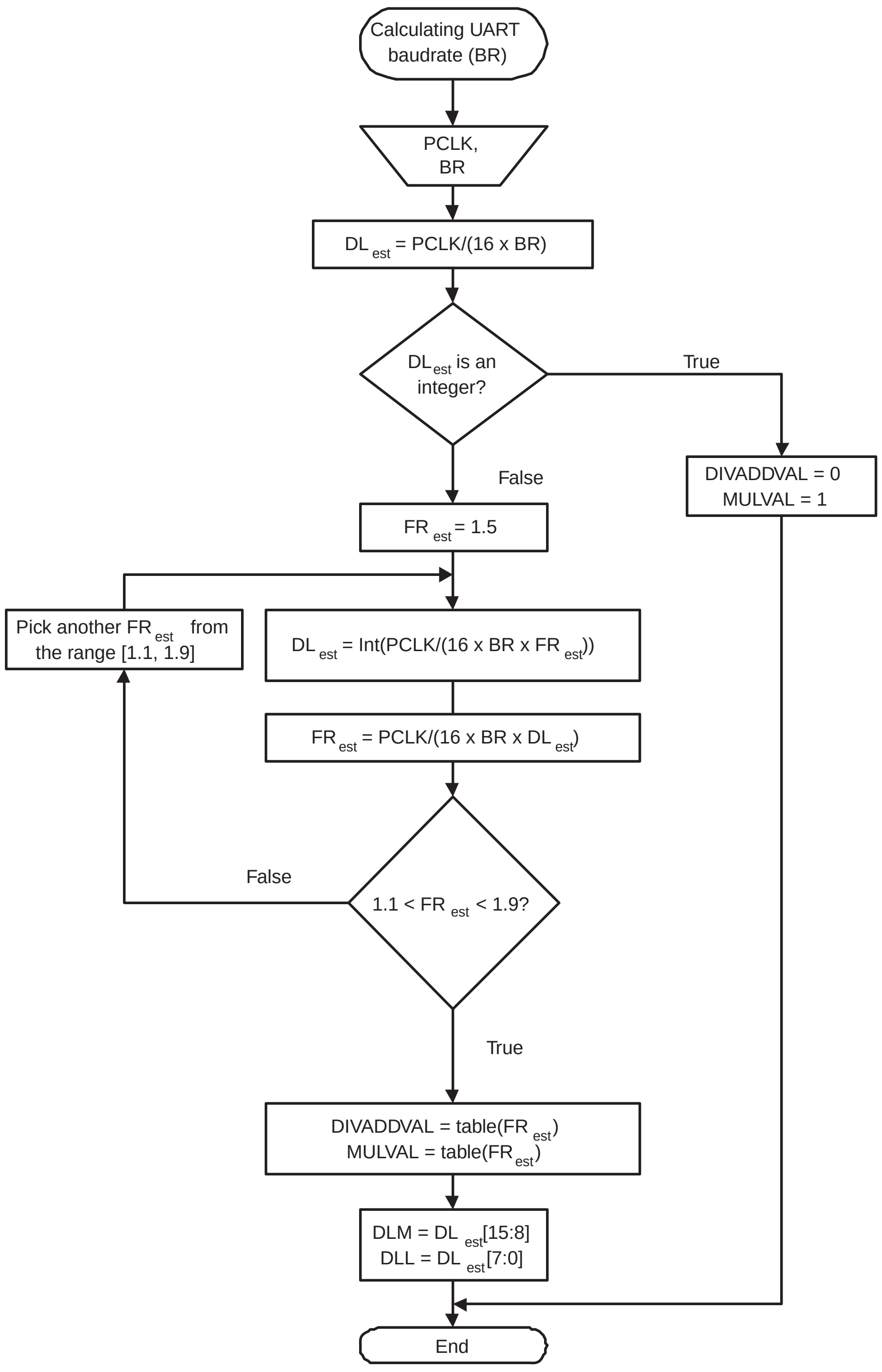


Fig 28. Algorithm for setting UART dividers

Table 157. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

11.4.4.1.1 Example 1: PCLK = 14.7456 MHz, BR = 9600 Bd

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

11.4.4.1.2 Example 2: PCLK = 12 MHz, BR = 115200 Bd

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 157 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 5 the UART’s baud rate is 115384 Bd. This rate has a relative error of 0.16% from the originally specified 115200 Bd.

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

The U1IER is used to enable UART1 interrupt sources.

Table 158. UART1 Interrupt Enable Register (U1IER - address 0xE001 0004, when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset value
0	RBR Interrupt Enable		U1IER[0] enables the Receive Data Available interrupt for UART1. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THRE Interrupt Enable		U1IER[1] enables the THRE interrupt for UART1. The status of this interrupt can be read from U1LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RX Line Interrupt Enable		U1IER[2] enables the UART1 RX line status interrupts. The status of this interrupt can be read from U1LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
3	Modem Status Interrupt Enable		U1IER[3] enables the modem interrupt. The status of this interrupt can be read from U1MSR[3:0].	0
		0	Disable the modem interrupt.	
		1	Enable the modem interrupt.	
6:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	CTS Interrupt Enable		If auto-CTS mode is enabled this bit enables/disables the modem status interrupt generation on a CTS1 signal transition. If auto-CTS mode is disabled a CTS1 transition will generate an interrupt if Modem Status Interrupt Enable (U1IER[3]) is set. In normal operation a CTS1 signal transition will generate a Modem Status Interrupt unless the interrupt has been disabled by clearing the U1IER[3] bit in the U1IER register. In auto-CTS mode a transition on the CTS1 bit will trigger an interrupt only if both the U1IER[3] and U1IER[7] bits are set.	0
		0	Disable the CTS interrupt.	
		1	Enable the CTS interrupt.	
8	ABEOIntEn		Enables the end of auto-baud interrupt.	0
		0	Disable End of Auto-baud Interrupt.	
		1	Enable End of Auto-baud Interrupt.	
9	ABTOIntEn		Enables the auto-baud time-out interrupt.	0
		0	Disable Auto-baud Time-out Interrupt.	
		1	Enable Auto-baud Time-out Interrupt.	
31:10	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11.4.6 UART1 Interrupt Identification Register (U1IIR - 0xE001 0008, Read Only)

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

Table 159. UART1 Interrupt Identification Register (U1IIR - address 0xE001 0008, read only) bit description

Bit	Symbol	Value	Description	Reset value
0	Interrupt Pending		Note that U1IIR[0] is active LOW. The pending interrupt can be determined by evaluating U1IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	Interrupt Identification		U1IER[3:1] identifies an interrupt corresponding to the UART1 Rx FIFO. All other combinations of U1IER[3:1] not listed above are reserved (100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt.	
		000	4 - Modem Interrupt.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		These bits are equivalent to U1FCR[0].	0
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Interrupts are handled as described in Table 160. Given the status of U1IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U1IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART1 RLS interrupt (U1IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART1RX input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART1 Rx error condition that set the interrupt can be observed via U1LSR[4:1]. The interrupt is cleared upon an U1LSR read.

The UART1 RDA interrupt (U1IIR[3:1] = 010) shares the second level priority with the CTI interrupt (U1IIR[3:1] = 110). The RDA is activated when the UART1 Rx FIFO reaches the trigger level defined in U1FCR7:6 and is reset when the UART1 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (U1IIR[3:1] = 110) is a second level interrupt and is set when the UART1 Rx FIFO contains at least one character and no UART1 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART1 Rx FIFO activity (read or write of UART1 RSR) will clear the interrupt. This interrupt is intended to flush the UART1 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 160. UART1 interrupt handling

U1IIR[3:0] value[1]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	U1LSR Read[2]
0100	Second	RX Data Available	Rx data available or trigger level reached in FIFO (U1FCR0=1)	U1RBR Read[3] or UART1 FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the RX FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) $\times$ 7 - 2] $\times$ 8 + [(trigger level - number of characters) $\times$ 8 + 1] RCLKs	U1RBR Read[3]
0010	Third	THRE	THRE[2]	U1IIR Read[4] (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read

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

[2] For details see Section 11.4.10 “UART1 Line Status Register (U1LSR - 0xE001 0014, Read Only)”

[3] For details see Section 11.4.1 “UART1 Receiver Buffer Register (U1RBR - 0xE001 0000, when DLAB = 0 Read Only)”

[4] For details see Section 11.4.6 “UART1 Interrupt Identification Register (U1IIR - 0xE001 0008, Read Only)” and Section 11.4.2 “UART1 Transmitter Holding Register (U1THR - 0xE001 0000, when DLAB = 0 Write Only)”

The UART1 THRE interrupt (U1IIR[3:1] = 001) is a third level interrupt and is activated when the UART1 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART1 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the U1THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to U1THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART1 THR FIFO has held two or more characters at one time and currently, the U1THR is empty. The THRE interrupt is reset when a U1THR write occurs or a read of the U1IIR occurs and the THRE is the highest interrupt (U1IIR[3:1] = 001).

The modem interrupt (U1IIR[3:1] = 000) is available in LPC2101/02/03. It is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a LOW to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining U1MSR[3:0]. A U1MSR read will clear the modem interrupt.

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

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

Table 161. UART1 FIFO Control Register (U1FCR - address 0xE001 0008) bit description

Bit	Symbol	Value	Description	Reset value
0	FIFO Enable	0	UART1 FIFOs are disabled. Must not be used in the application.	0
		1	Active HIGH enable for both UART1 Rx and TX FIFOs and U1FCR[7:1] access. This bit must be set for proper UART1 operation. Any transition on this bit will automatically clear the UART1 FIFOs.	
1	RX FIFO Reset	0	No impact on either of UART1 FIFOs.	0
		1	Writing a logic 1 to U1FCR[1] will clear all bytes in UART1 Rx FIFO and reset the pointer logic. This bit is self-clearing.	
2	TX FIFO Reset	0	No impact on either of UART1 FIFOs.	0
		1	Writing a logic 1 to U1FCR[2] will clear all bytes in UART1 TX FIFO and reset the pointer logic. This bit is self-clearing.	
5:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RX Trigger Level		These two bits determine how many receiver UART1 FIFO characters must be written before an interrupt is activated.	0
		00	trigger level 0 (1 character or 0x01).	
		01	trigger level 1 (4 characters or 0x04).	
		10	trigger level 2 (8 characters or 0x08).	
		11	trigger level 3 (14 characters or 0x0E).	

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

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

Table 162. UART1 Line Control Register (U1LCR - address 0xE001 000C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	Word Length Select	00	5 bit character length.	0
		01	6 bit character length.	
		10	7 bit character length.	
		11	8 bit character length.	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if U1LCR[1:0]=00).	

Table 162. UART1 Line Control Register (U1LCR - address 0xE001 000C) bit description

Bit	Symbol	Value	Description	Reset value
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UART1 TXD is forced to logic 0 when U1LCR[6] is active HIGH.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

11.4.9 UART1 Modem Control Register (U1MCR - 0xE001 0010)

The U1MCR enables the modem loop-back mode and controls the modem output signals.

Table 163. UART1 Modem Control Register (U1MCR - address 0xE001 0010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTR Control		Source for modem output pin, DTR. This bit reads as 0 when modem loopback mode is active.	0
1	RTS Control		Source for modem output pin RTS. This bit reads as 0 when modem loopback mode is active.	0
3:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	Loop-back Mode Select		The modem loop-back mode provides a mechanism to perform diagnostic loop-back testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RXD1, has no effect on loop-back and output pin, TXD1 is held in marking state. The four modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the four modem outputs are connected to the four modem inputs. As a result of these connections, the upper four bits of the U1MSR will be driven by the lower four bits of the U1MCR rather than the four modem inputs in normal mode. This permits modem status interrupts to be generated in loop-back mode by writing the lower four bits of U1MCR.	0
		0	Disable modem loopback mode.	
		1	Enable modem loopback mode.	
5:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
6	RTSen		Auto-RTS control bit.	0
		0	Disable auto-RTS flow control.	
		1	Enable auto-RTS flow control.	

Table 163. UART1 Modem Control Register (U1MCR - address 0xE001 0010) bit description

Bit	Symbol	Value	Description	Reset value
7	CTSen		Auto-CTS control bit.	0
		0	Disable auto-CTS flow control.	
		1	Enable auto-CTS flow control.	

11.4.9.1 Auto-flow control

If auto-RTS mode is enabled the UART1‘s receiver FIFO hardware controls the RTS1 output of the UART1. If the auto-CTS mode is enabled the UART1‘s U1TSR hardware will only start transmitting if the CTS1 input signal is asserted.

11.4.9.1.1 Auto-RTS

The auto-RTS function is enabled by setting the RTSen bit. Auto-RTS data flow control originates in the U1RBR module and is linked to the programmed receiver FIFO trigger level. If auto-RTS is enabled, the data-flow is controlled as follows:

When the receiver FIFO level reaches the programmed trigger level, RTS1 is deasserted (to a high value). It is possible that the sending UART sends an additional byte after the trigger level is reached (assuming the sending UART has another byte to send) because it might not recognize the deassertion of RTS1 until after it has begun sending the additional byte. RTS1 is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The reassertion of RTS1 signals to the sending UART to continue transmitting data.

If Auto-RTS mode is disabled, the RTS Control bit controls the RTS1 output of the UART1. If Auto-RTS mode is enabled, hardware controls the RTS1 output, and the actual value of RTS1 will be copied in the RTS Control bit of the UART1. As long as Auto-RTS is enabled, the value of the RTS Control bit is read-only for software.

Example: Suppose the UART1 operating in type 550 has trigger level in U1FCR set to 0x2 then if Auto-RTS is enabled the UART1 will deassert the RTS1 output as soon as the receive FIFO contains 8 bytes (Table 161). The RTS1 output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

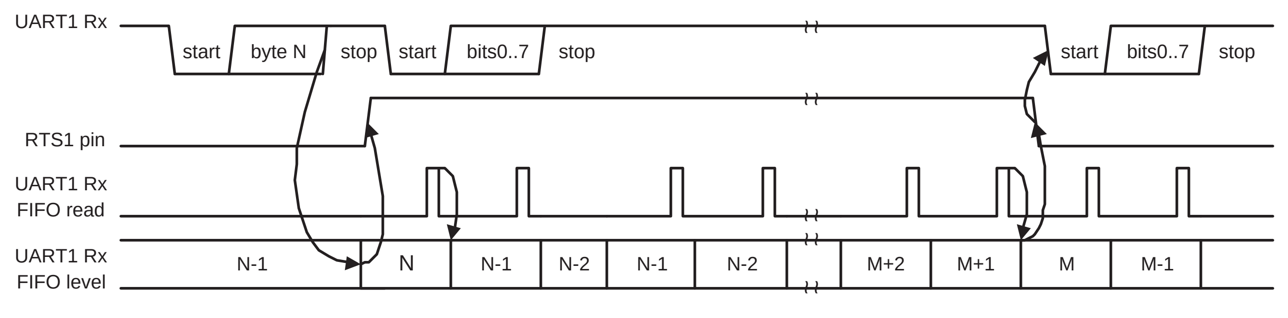


Fig 29. Auto-RTS functional timing

11.4.9.1.2 Auto-CTS

The auto-CTS function is enabled by setting the CTSen bit. If auto-CTS is enabled the transmitter circuitry in the U1TSR module checks CTS1 input before sending the next data byte. When CTS1 is active (LOW), the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS1 must be released before the middle of the last stop bit that is currently being sent. In auto-CTS mode a change of the CTS1 signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, Delta CTS bit in the U1MSR will be set though. Table 164 lists the conditions for generating a Modem Status interrupt.

Table 164. Modem status interrupt generation

Enable Modem Status Interrupt (U1IER[3])	CTSen (U1MCR[7])	CTS Interrupt Enable (U1IER[7])	Delta CTS (U1MSR[0])	Delta DCD or Trailing Edge RI or Delta DSR (U1MSR[3] or U1MSR[2] or (U1MSR[1]))	Modem Status Interrupt
0	x	x	x	x	no
1	0	x	0	0	no
1	0	x	1	x	yes
1	0	x	x	1	yes
1	1	0	x	0	no
1	1	0	x	1	yes
1	1	1	0	0	no
1	1	1	1	x	yes
1	1	1	x	1	yes

The auto-CTS function reduces interrupts to the host system. When flow control is enabled, a CTS1 state change does not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. Figure 30 illustrates the auto-CTS functional timing.

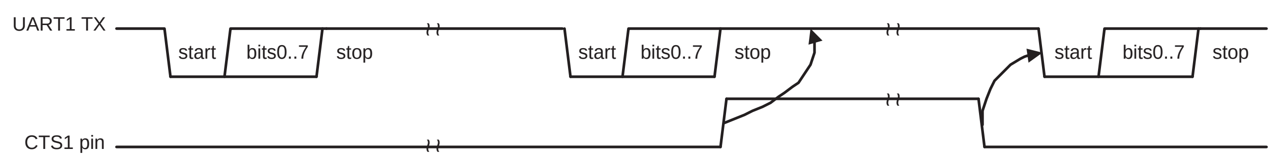


Fig 30. Auto-CTS functional timing

While starting transmission of the initial character the CTS1 signal is asserted. Transmission will stall as soon as the pending transmission has completed. The UART will continue transmitting a 1 bit as long as CTS1 is deasserted (HIGH). As soon as CTS1 gets deasserted transmission resumes and a start bit is sent followed by the data bits of the next character.

11.4.10 UART1 Line Status Register (U1LSR - 0xE001 0014, Read Only)

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

Table 165. UART1 Line Status Register (U1LSR - address 0xE001 0014, read only) bit description

Bit	Symbol	Value	Description	Reset value
0	Receiver Data Ready (RDR)		U1LSR[0] is set when the U1RBR holds an unread character and is cleared when the UART1 RBR FIFO is empty.	0
		0	U1RBR is empty.	
		1	U1RBR contains valid data.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An U1LSR read clears U1LSR[1]. U1LSR[1] is set when UART1 RSR has a new character assembled and the UART1 RBR FIFO is full. In this case, the UART1 RBR FIFO will not be overwritten and the character in the UART1 RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An U1LSR read clears U1LSR[2]. Time of parity error detection is dependent on U1FCR[0]. **Note:** A parity error is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An U1LSR read clears U1LSR[3]. The time of the framing error detection is dependent on U1FCR0. Upon detection of a framing error, the RX will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXD1 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all 1's). An U1LSR read clears this status bit. The time of break detection is dependent on U1FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART1 RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE)		THRE is set immediately upon detection of an empty UART1 THR and is cleared on a U1THR write.	1
		0	U1THR contains valid data.	
		1	U1THR is empty.	
6	Transmitter Empty (TEMT)		TEMT is set when both U1THR and U1TSR are empty; TEMT is cleared when either the U1TSR or the U1THR contain valid data.	1
		0	U1THR and/or the U1TSR contains valid data.	
		1	U1THR and the U1TSR are empty.	

Table 165. UART1 Line Status Register (U1LSR - address 0xE001 0014, read only) bit description

Bit	Symbol	Value	Description	Reset value
7	Error in RX FIFO (RXFE)		U1LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the U1RBR. This bit is cleared when the U1LSR register is read and there are no subsequent errors in the UART1 FIFO.	0
		0	U1RBR contains no UART1 RX errors or U1FCR[0]=0.	
		1	UART1 RBR contains at least one UART1 RX error.	

11.4.11 UART1 Modem Status Register (U1MSR - 0xE001 0018)

The U1MSR is a read-only register that provides status information on the modem input signals. U1MSR[3:0] is cleared on U1MSR read. Note that modem signals have no direct affect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 166. UART1 Modem Status Register (U1MSR - address 0xE001 0018) bit description

Bit	Symbol	Value	Description	Reset value
0	Delta CTS		Set upon state change of input CTS. Cleared on an U1MSR read.	0
		0	No change detected on modem input, CTS.	
		1	State change detected on modem input, CTS.	
1	Delta DSR		Set upon state change of input DSR. Cleared on an U1MSR read.	0
		0	No change detected on modem input, DSR.	
		1	State change detected on modem input, DSR.	
2	Trailing Edge RI		Set upon LOW to HIGH transition of input RI. Cleared on an U1MSR read.	0
		0	No change detected on modem input, RI.	
		1	LOW-to-HIGH transition detected on RI.	
3	Delta DCD		Set upon state change of input DCD. Cleared on an U1MSR read.	0
		0	No change detected on modem input, DCD.	
		1	State change detected on modem input, DCD.	
4	CTS		Clear To Send State. Complement of input signal CTS. This bit is connected to U1MCR[1] in modem loopback mode.	0
5	DSR		Data Set Ready State. Complement of input signal DSR. This bit is connected to U1MCR[0] in modem loopback mode.	0
6	RI		Ring Indicator State. Complement of input RI. This bit is connected to U1MCR[2] in modem loopback mode.	0
7	DCD		Data Carrier Detect State. Complement of input DCD. This bit is connected to U1MCR[3] in modem loopback mode.	0

11.4.12 UART1 Scratch Pad Register (U1SCR - 0xE001 001C)

The U1SCR has no effect on the UART1 operation. This register can be written and/or read at user’s discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the U1SCR has occurred.

Table 167. UART1 Scratch Pad Register (U1SCR - address 0xE001 0014) bit description

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

11.4.13 UART1 Auto-baud Control Register (U1ACR - 0xE001 0020)

The UART1 Auto-baud Control Register (U1ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 168. Auto-baud Control Register (U1ACR - 0xE001 0020) bit description

Bit	Symbol	Value	Description	Reset value
0	Start		This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running).Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	Mode		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AutoRestart	0	No restart	0
		1	Restart in case of time-out (counter restarts at next UART1 Rx falling edge)	
7:3	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOIntClr		End of auto-baud interrupt clear bit (write only accessible). Writing a 1 will clear the corresponding interrupt in the U1IIR. Writing a 0 has no impact.	0
9	ABTOIntClr		Auto-baud time-out interrupt clear bit (write only accessible). Writing a 1 will clear the corresponding interrupt in the U1IIR. Writing a 0 has no impact.	0
31:10	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

11.4.14 Auto-baud

The UART1 auto-baud function can be used to measure the incoming baud-rate based on the ”AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers U1DLM and U1DLL accordingly.

Auto-baud is started by setting the U1ACR Start bit. Auto-baud can be stopped by clearing the U1ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the U1ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART1 Rx pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UART1 Rx pin (the length of the start bit).

The U1ACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART1 Rx pin.

The auto-baud function can generate two interrupts.

- The U1IIR ABTOInt interrupt will get set if the interrupt is enabled (U1IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The U1IIR ABEOInt interrupt will get set if the interrupt is enabled (U1IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding U1ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART1 Rx pin baud-rate, but the value of the U1FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to U1DLM and U1DLL registers should be done before U1ACR register write. The minimum and the maximum baudrates supported by UART1 are function of PCLK, number of data bits, stop-bits and parity bits.

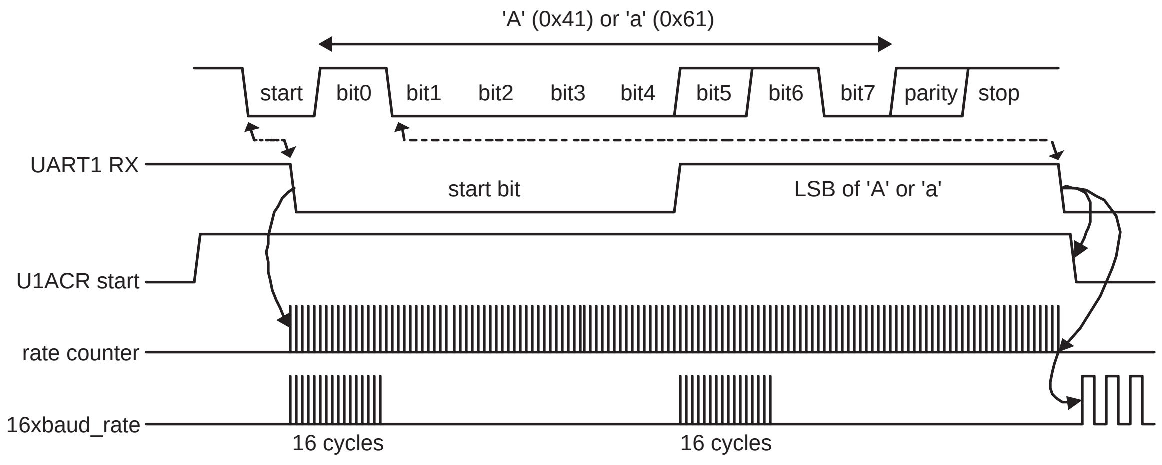
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART1_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax \quad (6)$$

11.4.15 Auto-baud modes

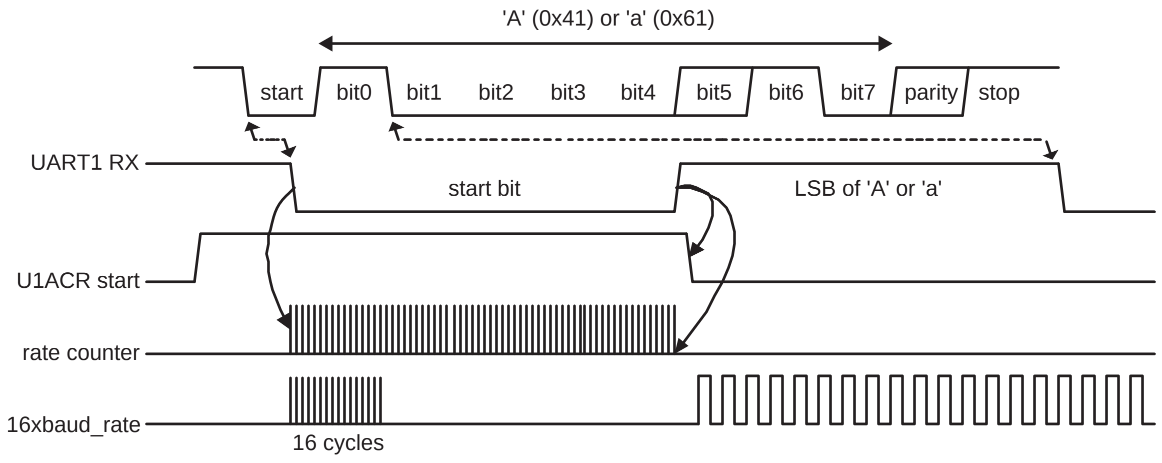
When the software is expecting an ”AT" command, it configures the UART1 with the expected character format and sets the U1ACR Start bit. The initial values in the divisor latches U1DLM and U1DLM don‘t care. Because of the ”A" or ”a" ASCII coding (”A" = 0x41, ”a" = 0x61), the UART1 Rx pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the U1ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On U1ACR Start bit setting, the baud-rate measurement counter is reset and the UART1 U1RSR is reset. The U1RSR baud rate is switch to the highest rate.
2. A falling edge on UART1 Rx pin triggers the beginning of the start bit. The rate measuring counter will start counting PCLK cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UART1 input clock, guaranteeing the start bit is stored in the U1RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART1 input clock (PCLK).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART1 Rx pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART1 Rx pin.

6. The rate counter is loaded into U1DLM/U1DLL and the baud-rate will be switched to normal operation. After setting the U1DLM/U1DLL the end of auto-baud interrupt U1IIR ABEOInt will be set, if enabled. The U1RSR will now continue receiving the remaining bits of the ”A/a" character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 31. Autobaud a) mode 0 and b) mode 1 waveform

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

LPC2101/2102/2103’s U1TER enables implementation of software and hardware flow control. When TXEn=1, UART1 transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART1 transmission will stop.

Table 169 describes how to use TXEn bit in order to achieve software flow control.

Table 169. UART1 Transmit Enable Register (U1TER - address 0xE001 0030) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software can clear this bit when it detects that the a hardware-handshaking TX-permit signal CTS has gone false, or it can clear this bit with software handshaking, when it receives an XOFF character (DC3). Software can set this bit again when it detects that the TX-permit signal has gone true, or when it receives an XON (DC1) character.	1

11.5 Architecture

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

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

The UART1 receiver block, U1RX, monitors the serial input line, RXD1, for valid input. The UART1 RX Shift Register (U1RSR) accepts valid characters via RXD1. After a valid character is assembled in the U1RSR, it is passed to the UART1 RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

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

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 TX block. The U1BRG clock input source is the APB clock (PCLK). The main clock is divided down per the divisor specified in the U1DLL and U1DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

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

The interrupt interface contains registers U1IER and U1IIR. The interrupt interface receives several one clock wide enables from the U1TX and U1RX blocks.

Status information from the U1TX and U1RX is stored in the U1LSR. Control information for the U1TX and U1RX is stored in the U1LCR.

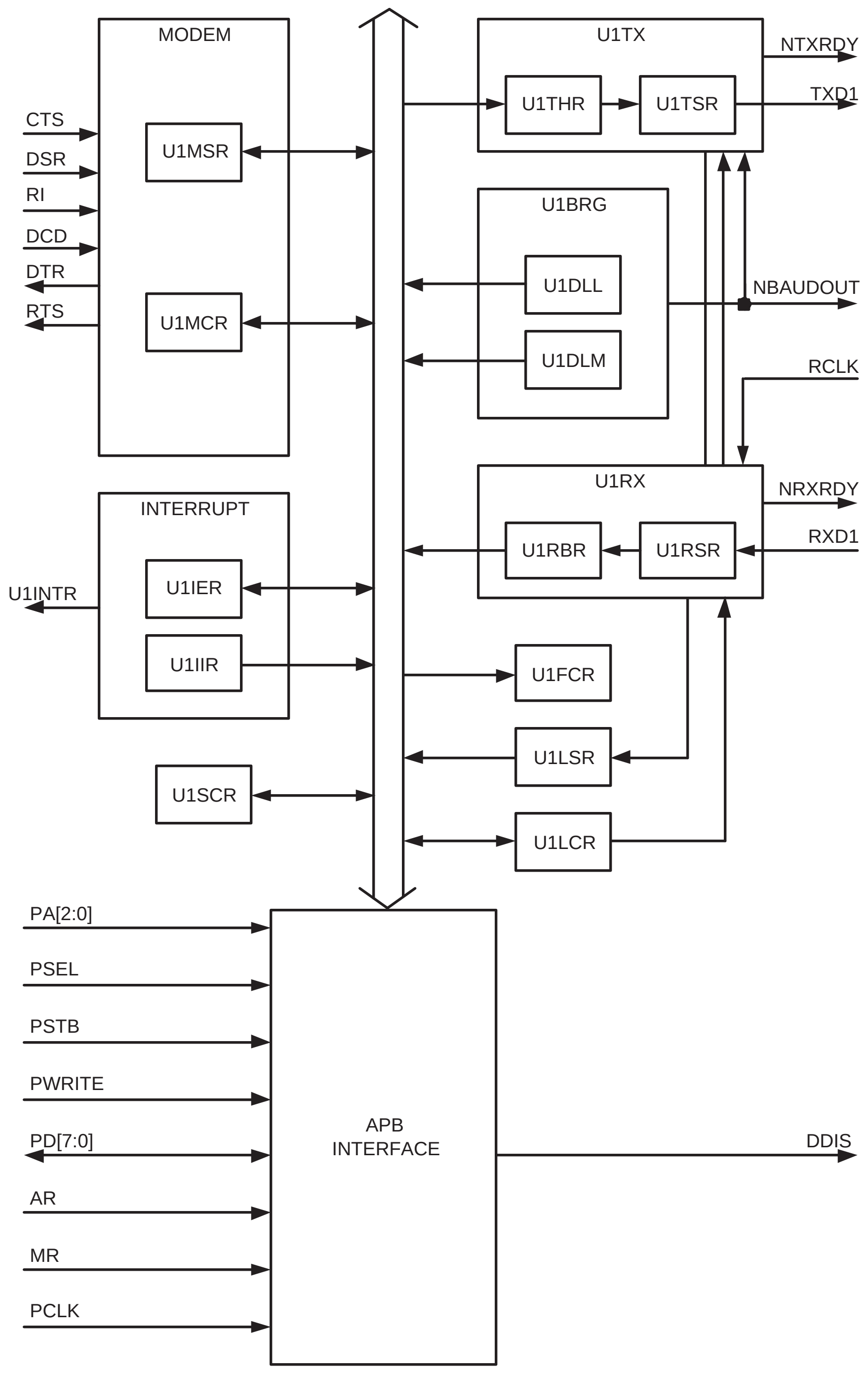


Fig 32. UART1 block diagram

UM10114

Chapter 12: LPC21xx/22xx I2C interface

Rev. 4 — 2 May 2012 **User manual**

12.1 How to read this chapter

The I^{2}C-bus interface is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

12.2 Features

- Standard I^{2}C compliant bus interfaces that may be configured as Master, Slave, or Master/Slave.
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock to allow adjustment of I^{2}C transfer rates.
- Bidirectional data transfer between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.
- The I^{2}C-bus may be used for test and diagnostic purposes.

12.3 Applications

Interfaces to external I^{2}C standard parts, such as serial RAMs, LCDs, tone generators, etc.

12.4 Description

A typical I^{2}C-bus configuration is shown in Figure 33. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^{2}C-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 I^{2}C-bus will not be released.

The LPC21xx/22xx I^{2}C interface is byte oriented, and have four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The I^{2}C interface complies with entire I^{2}C specification, supporting the ability to turn power off to the LPC21xx/22xx without causing a problem with other devices on the same I^{2}C-bus. This is sometimes a useful capability, but intrinsically limits alternate uses for the same pins if the I^{2}C interface is not used.

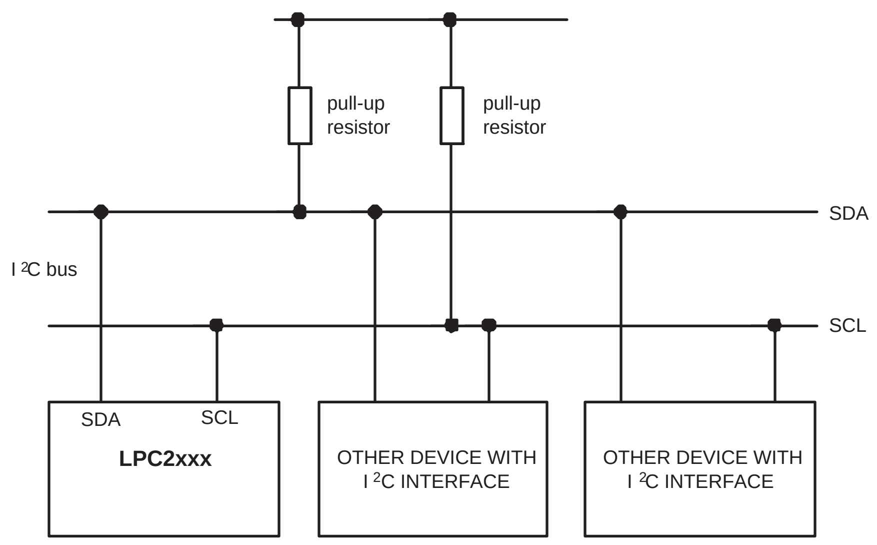

pull-up resistor
pull-up resistor
SDA
I 2C bus
SCL
SDA
SCL
LPC2xxx
OTHER DEVICE WITH I 2C INTERFACE
OTHER DEVICE WITH I 2C INTERFACE

Fig 33. I^{2}C-bus Configuration

12.5 Pin description

Table 170. I^{2}C Pin Description

Pin	Type	Description
SDA	Input/Output	I^{2}C serial data
SCL	Input/Output	I^{2}C Serial clock

Remark: The SDA and SCL outputs are open-drain outputs for I^{2}C-bus compliance.

12.6 I^{2}C operating modes

In a given application, the I^{2}C block may operate as a master, a slave, or both. In the slave mode, the I^{2}C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I^{2}C block switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

12.6.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 171. I2EN must be set to 1 to enable the I^{2}C function. If the AA bit is 0, the I^{2}C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI Bit is cleared by writing 1 to the SIC bit in the I2CONCLR register.

Table 171. I2CCONSET used to configure Master mode

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

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

The I^{2}C interface will enter master transmitter mode when software sets the STA bit. The I^{2}C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the I2STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register. The STA bit should be cleared after writing the slave address.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 0x18, 0x20, or 0x38 for the master mode, or 0x68, 0x78, or 0xB0 if the slave mode was enabled (by setting AA to 1). The appropriate actions to be taken for each of these status codes are shown in Table 186 to Table 189.

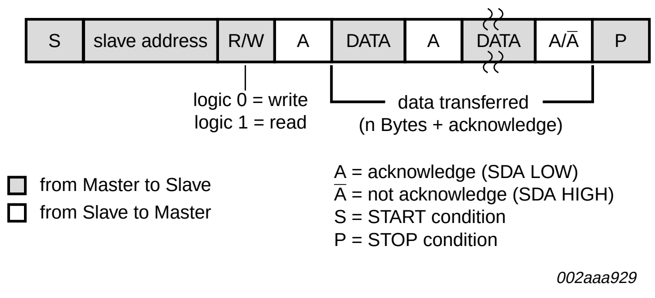


Fig 34. Format in the Master Transmitter mode

12.6.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^{2}C Data register (I2DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 187.

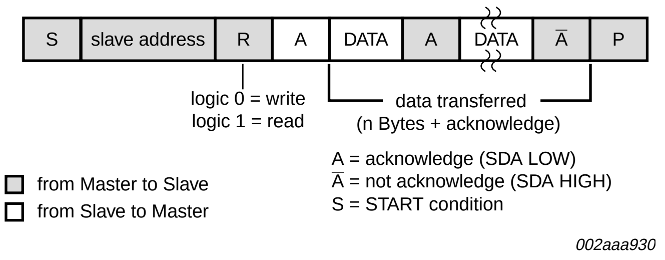


Fig 35. Format of Master Receiver mode

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

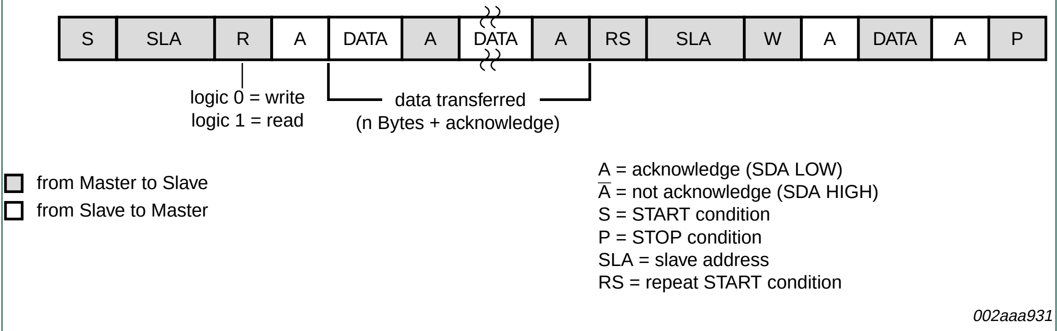


Fig 36. A Master Receiver switches to Master Transmitter after sending Repeated START

12.6.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, user write the Slave Address register (I2ADR) and write the I^{2}C Control Set register (I2CONSET) as shown in Table 172.

Table 172. I2CONSET used to configure Slave mode

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

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

After I2ADR and I2CONSET are initialized, the I^{2}C interface waits until it is addressed by its own address or general address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status register (I2STAT). Refer to Table 188 for the status codes and actions.

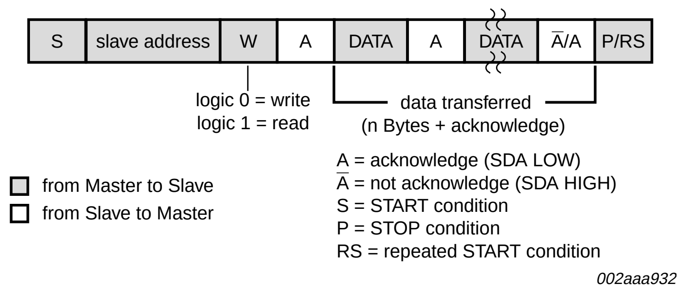


Fig 37. Format of Slave Receiver mode

12.6.4 Slave Transmitter mode

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

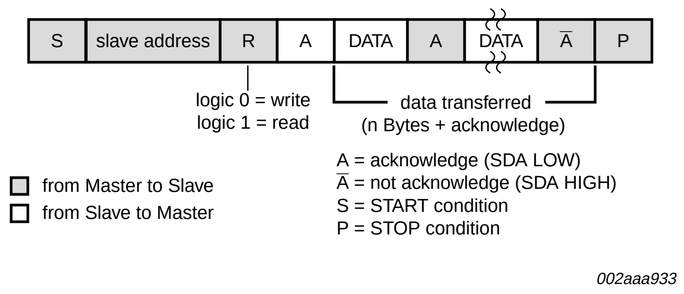


Fig 38. Format of Slave Transmitter mode

12.7 I^{2}C Implementation and operation

Figure 39 shows how the on-chip I2C-bus interface is implemented, and the following text describes the individual blocks.

12.7.1 Input filters and output stages

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

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

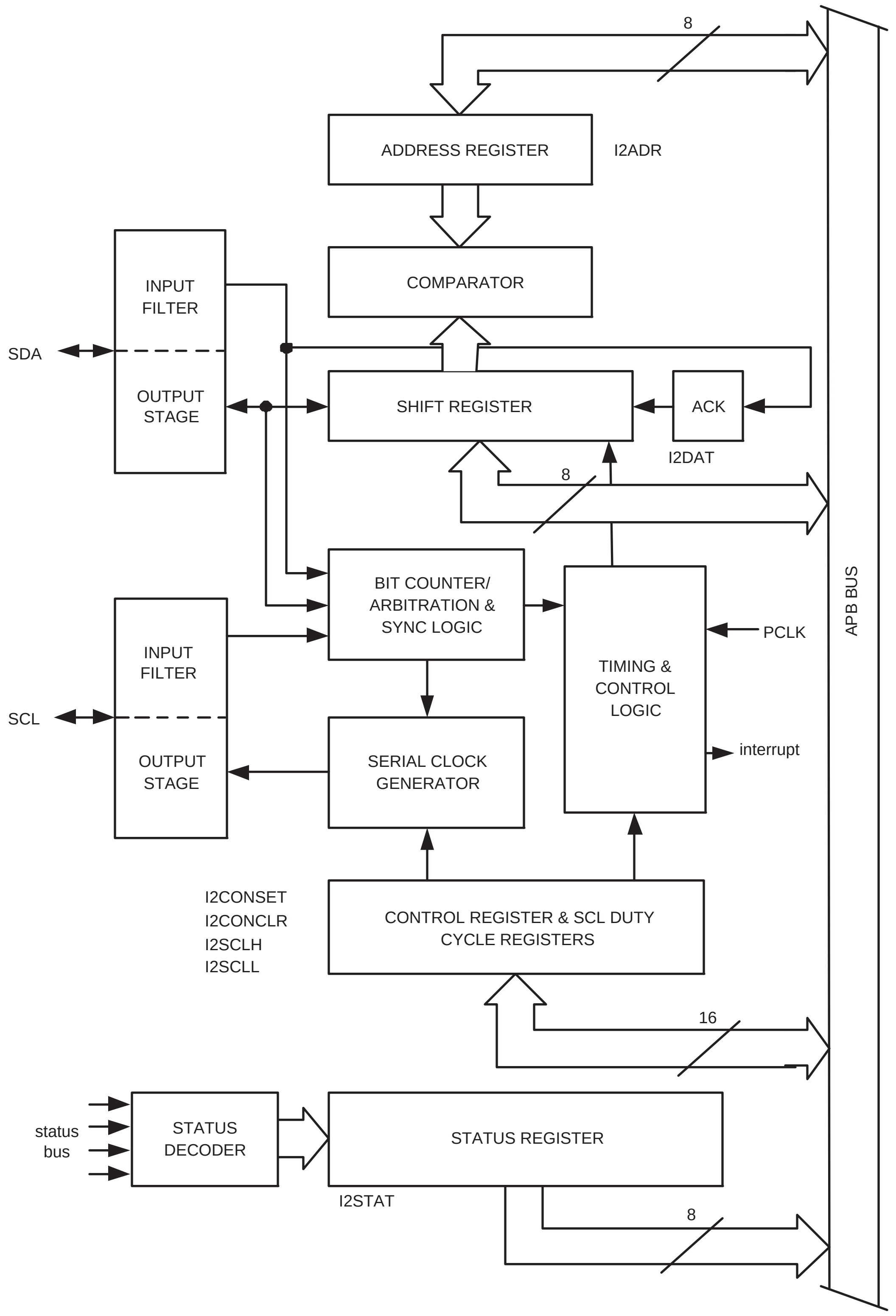


Fig 39. I^{2}C serial interface block diagram

12.7.2 Address Register, I2ADDR

This register may be loaded with the 7-bit slave address (7 most significant bits) to which the I^{2}C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable general call address (0x00) recognition.

12.7.3 Comparator

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in I2ADR). It also compares the first received 8-bit byte with the general call address (0x00). If an equality is found, the appropriate status bits are set and an interrupt is requested.

12.7.4 Shift register, I2DAT

This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in I2DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in I2DAT.

12.7.5 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^{2}C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^{2}C block immediately changes from master transmitter to slave receiver. The I^{2}C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^{2}C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, the I^{2}C block generates no further clock pulses. Figure 40 shows the arbitration procedure.

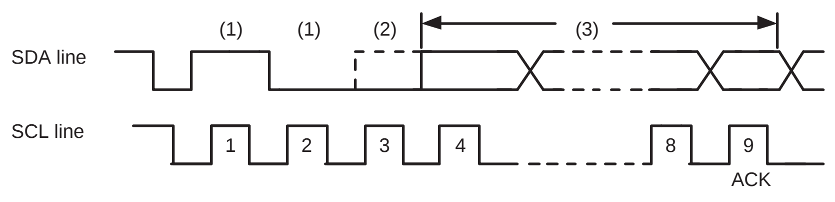


Fig 40. Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 41 shows the synchronization procedure.

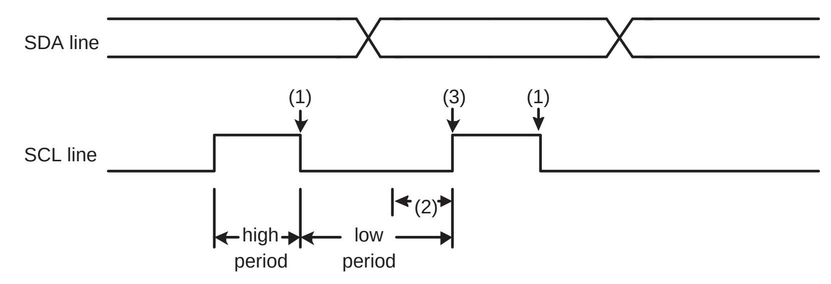


Fig 41. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I^{2}C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

12.7.6 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I^{2}C block is in the master transmitter or master receiver mode. It is switched off when the I^{2}C block is in a slave mode. The I^{2}C output clock frequency and duty cycle is programmable via the I^{2}C Clock Control Registers. See the description of the I2CSCLL and I2CSCLH registers for details. The output clock pulses have a duty cycle as programmed unless the bus is synchronizing with other SCL clock sources as described above.

12.7.7 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for I2DAT, enables the comparator, generates and detects start and stop conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I^{2}C-bus status.

12.7.8 Control register, I2CONSET and I2CONCLR

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

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

12.7.9 Status decoder and Status register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I^{2}C-bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I^{2}C block are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

12.8 Register description

Each I^{2}C interface contains 7 registers as shown in Table 173 below.

Table 173. I^{2}C register map

Name	Description	Access	Reset value[1]	Address
I2CONSET	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I^{2}C control register is set. Writing a zero has no effect on the corresponding bit in the I^{2}C control register.	R/W	0x00	0xE001 C000
I2STAT	**I2C Status Register.** During I^{2}C operation, this register provides detailed status codes that allow software to determine the next action needed.	RO	0xF8	0xE001 C004
I2DAT	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	R/W	0x00	0xE001 C008
I2ADR	**I2C Slave Address Register.** Contains the 7-bit slave address for operation of the I^{2}C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the general call address.	R/W	0x00	0xE001 C00C
I2SCLH	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I^{2}C clock.	R/W	0x04	0xE001 C010
I2SCLL	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I^{2}C clock. I2SCLL and I2SCLH together determine the clock frequency generated by an I^{2}C master and certain times used in slave mode.	R/W	0x04	0xE001 C014
I2CONCLR	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I^{2}C control register is cleared. Writing a zero has no effect on the corresponding bit in the I^{2}C control register.	WO	NA	0xE001 C018

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

12.8.1 I^{2}C Control Set register (I2CONSET - 0xE001 C000)

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I^{2}C interface. Writing a one to a bit of this register causes the corresponding bit in the I^{2}C control register to be set. Writing a zero has no effect.

Table 174. I^{2}C Control Set register (I2CONSET - address 0xE001 C000) bit description

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

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

When I2EN is “0”, the SDA and SCL input signals are ignored, the I^{2}C block is in the “not addressed” slave state, and the STO bit is forced to “0”.

I2EN should not be used to temporarily release the I^{2}C-bus since, when I2EN is reset, the I^{2}C-bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I^{2}C interface to enter master mode and transmit a START condition or transmit a repeated START condition if it is already in master mode.

When STA is 1 and the I^{2}C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I^{2}C interface is already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I^{2}C interface is in an addressed slave mode.

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

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

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

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

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

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

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

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

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

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

12.8.2 I^{2}C Control Clear register (I2CONCLR - 0xE001 C018)

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I^{2}C interface. Writing a one to a bit of this register causes the corresponding bit in the I^{2}C control register to be cleared. Writing a zero has no effect.

Table 175. I^{2}C Control Set register (I2CONCLR - address 0xE001 C018) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert acknowledge Clear bit.	
3	SIC	I^{2}C interrupt Clear bit.	0
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	START flag Clear bit.	0
6	I2ENC	I^{2}C interface Disable bit.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

AAC is the Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.

SIC is the 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.

STAC is the Start flag Clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.

I2ENC is the I^{2}C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.

12.8.3 I^{2}C Status register (I2STAT - 0xE001 C004)

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

Table 176. I^{2}C Status register (I2STAT - address 0xE001) bit description

Bit	Symbol	Description	Reset value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I^{2}C interface.	0x1F

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I^{2}C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 186 to Table 189.

12.8.4 I^{2}C Data register (I2DAT - 0xE001 C008)

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

Table 177. I^{2}C Data register (I2DAT - address 0xE001 C008) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds data values that have been received, or are to be transmitted.	0

12.8.5 I^{2}C Slave Address register (I2ADR - 0xE001 C00C)

These registers are readable and writable, and is only used when an I^{2}C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the general call bit. When this bit is set, the general call address (0x00) is recognized.

Table 178. I^{2}C Slave Address register (I2ADR - address 0xE001 C00C) bit description

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

12.8.6 I^{2}C SCL High duty cycle register (I2SCLH - 0xE001 C010)

Table 179. I^{2}C SCL High Duty Cycle register (I2SCLH - address 0xE001 C010) bit description

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

12.8.7 I^{2}C SCL Low duty cycle register (I2SCLL - 0xE001 C014)

Table 180. I^{2}C SCL Low Duty Cycle register (I2SCLL - address 0xE001 C014) bit description

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

12.8.8 Selecting the appropriate I^{2}C data rate and duty cycle

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

$$I^2C_{bitfrequency} = \frac{PCLK}{I2CSCLH + I2CSCLL} \tag{7}$$

The values for I2SCLL and I2SCLH should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I^{2}C-bus specification defines the SCL low time and high time at different values for a 400 kHz I^{2}C rate. The value of the register must ensure that the data rate is in the I^{2}C data rate range of 0 through 400 kHz. Each register value must be greater than or equal to 4. Table 181 gives some examples of I^{2}C-bus rates based on PCLK frequency and I2SCLL and I2SCLH values.

Table 181. Example I^{2}C clock rates

I2SCLL + I2SCLH	I^{2}C Bit Frequency (kHz) at PCLK (MHz)						
	1	5	10	16	20	40	60
8	125						
10	100						
25	40	200	400				
50	20	100	200	320	400		
100	10	50	100	160	200	400	
160	6.25	31.25	62.5	100	125	250	375
200	5	25	50	80	100	200	300
400	2.5	12.5	25	40	50	100	150
800	1.25	6.25	12.5	20	25	50	75

12.9 Details of I^{2}C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver

- Slave Transmitter

Data transfers in each mode of operation are shown in Figures 42 to 46. Table 182 lists abbreviations used in these figures when describing the I^{2}C operating modes.

Table 182. Abbreviations used to describe an I^{2}C operation

Abbreviation	Explanation
S	Start Condition
SLA	7-bit slave address
R	Read bit (high level at SDA)
W	Write bit (low level at SDA)
A	Acknowledge bit (low level at SDA)
$\overline{A}$	Not acknowledge bit (high level at SDA)
Data	8-bit data byte
P	Stop condition

In Figures 42 to 46, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the I2STAT register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in I2STAT is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in tables from Table 186 to Table 190.

12.9.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 42). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 183. I2CONSET used to initialize Master Transmitter mode

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

The I^{2}C rate must also be configured in the I2SCLL and I2SCLH registers. I2EN must be set to logic 1 to enable the I^{2}C block. If the AA bit is reset, the I^{2}C block will not acknowledge its own slave address or the general call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I^{2}C interface cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I^{2}C logic will now test the I^{2}C-bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (I2STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads I2DAT with the slave address and the data direction bit (SLA+W). The SI bit in I2CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 186. After a repeated start condition (state 0x10). The I^{2}C block may switch to the master receiver mode by loading I2DAT with SLA+R).

12.9.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 43). The transfer is initialized as in the master transmitter mode. When the start condition has been transmitted, the interrupt service routine must load I2DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in I2CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. These are 0x40, 0x48, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 187. After a repeated start condition (state 0x10), the I^{2}C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

12.9.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 44). To initiate the slave receiver mode, I2ADR and I2CON must be loaded as follows:

Table 184. I2CADR usage in Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	own slave 7-bit address							GC

The upper 7 bits are the address to which the I^{2}C block will respond when addressed by a master. If the LSB (GC) is set, the I^{2}C block will respond to the general call address (0x00); otherwise it ignores the general call address.

Table 185. I2CONSET used to initialize Slave Receiver mode

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

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

When I2ADR and I2CON have been initialized, the I^{2}C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^{2}C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 104. The slave receiver mode may also be entered if arbitration is lost while the I^{2}C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^{2}C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^{2}C block does not respond to its own slave address or a general call address. However, the I^{2}C-bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^{2}C block from the I^{2}C-bus.

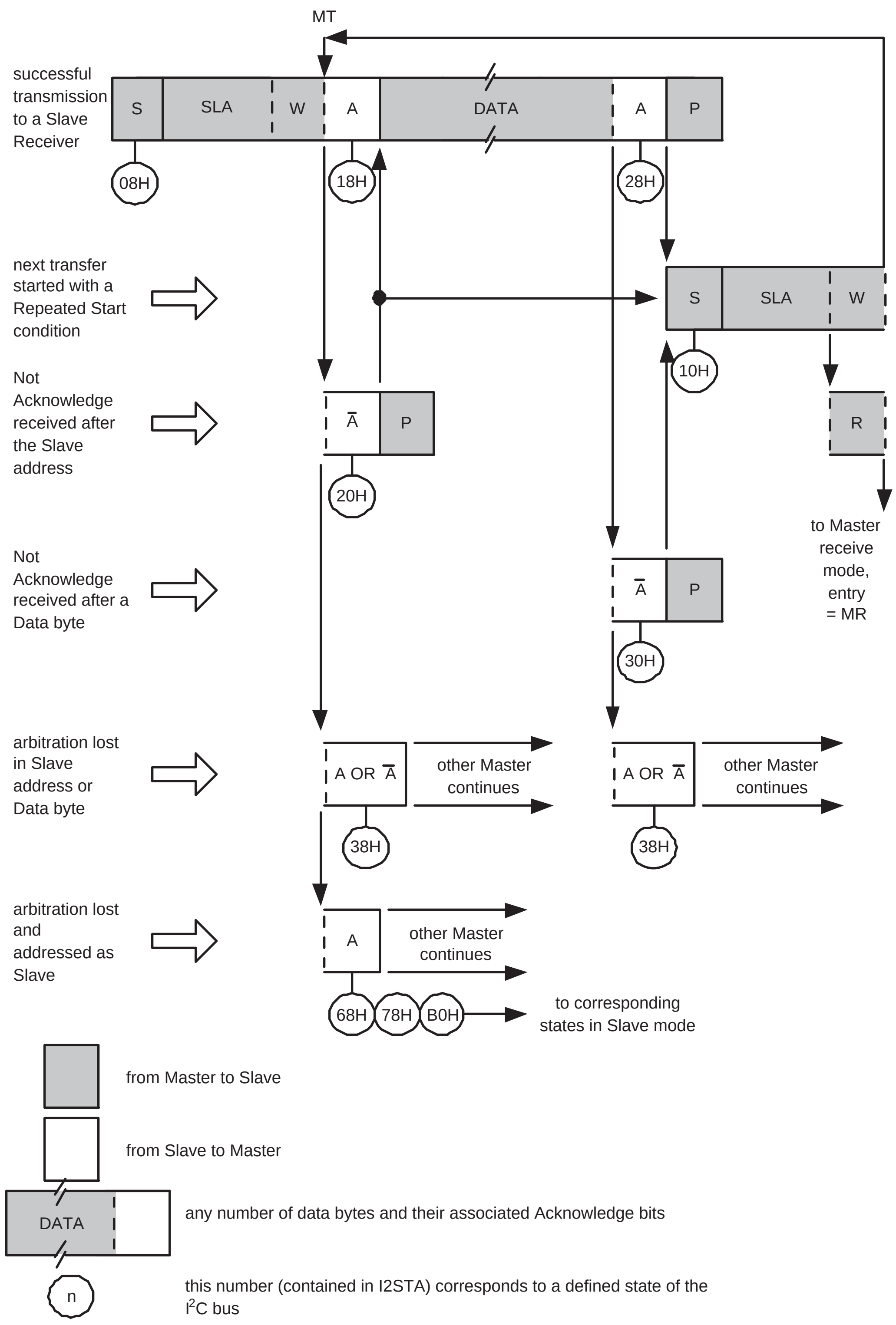

MT
successful transmission to a Slave Receiver
S
SLA
W
A
DATA
A
P
08H
18H
28H
next transfer started with a Repeated Start condition
S
SLA
W
10H
R
to Master receive mode, entry = MR
Not Acknowledge received after the Slave address
Ā
P
20H
Not Acknowledge received after a Data byte
Ā
P
30H
arbitration lost in Slave address or Data byte
A OR Ā
other Master continues
38H
A OR Ā
other Master continues
38H
arbitration lost and addressed as Slave
A
other Master continues
68H
78H
B0H
to corresponding states in Slave mode
from Master to Slave
from Slave to Master
DATA
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 42. Format and States in the Master Transmitter mode

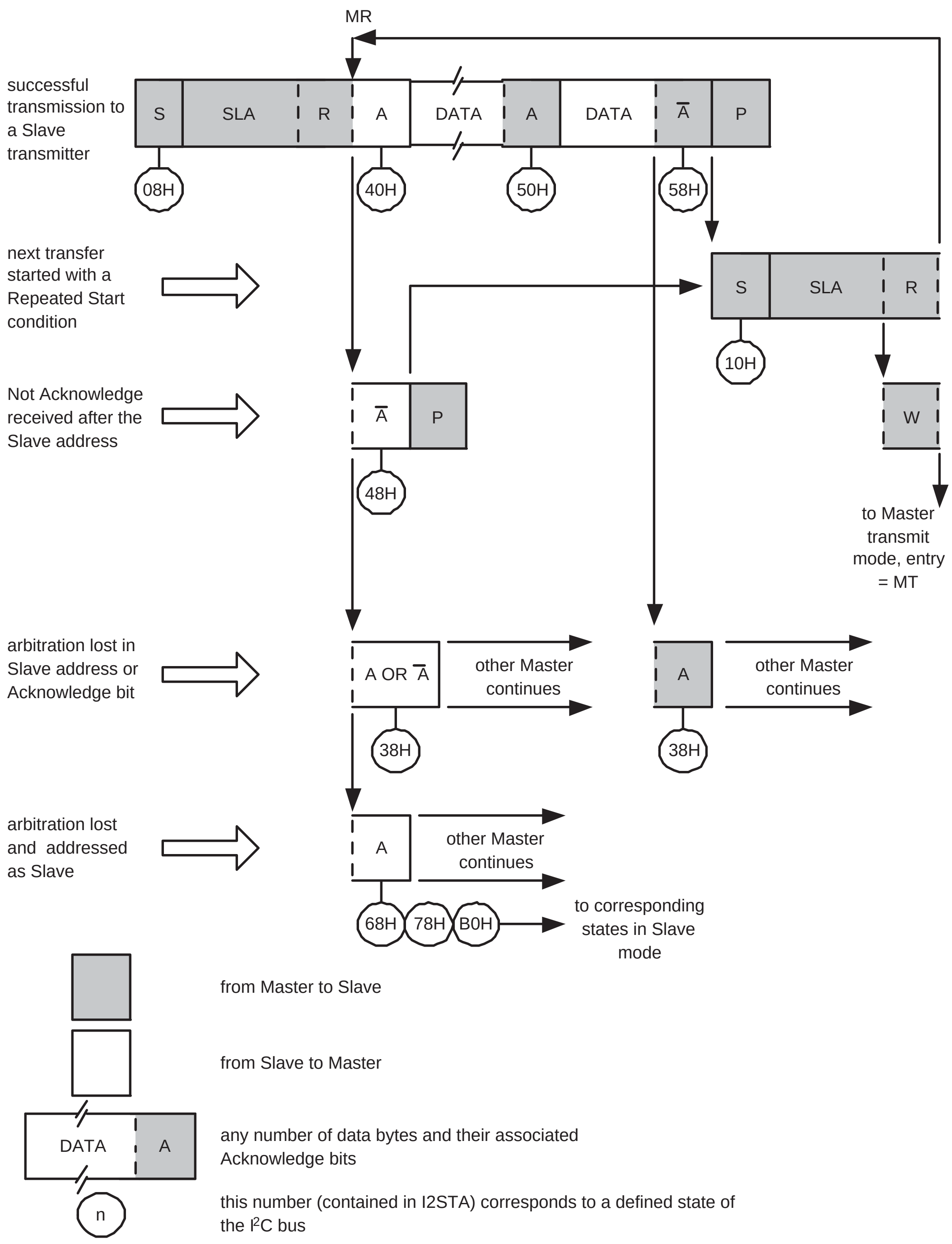

MR
successful transmission to a Slave transmitter
S
SLA
R
A
DATA
A
DATA
A̅
P
08H
40H
50H
58H
next transfer started with a Repeated Start condition
S
SLA
R
10H
Not Acknowledge received after the Slave address
A̅
P
48H
W
to Master transmit mode, entry = MT
arbitration lost in Slave address or Acknowledge bit
A OR A̅
other Master continues
A
other Master continues
38H
38H
arbitration lost and addressed as Slave
A
other Master continues
68H
78H
B0H
to corresponding states in Slave mode
from Master to Slave
from Slave to Master
DATA
A
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 43. Format and States in the Master Receiver mode

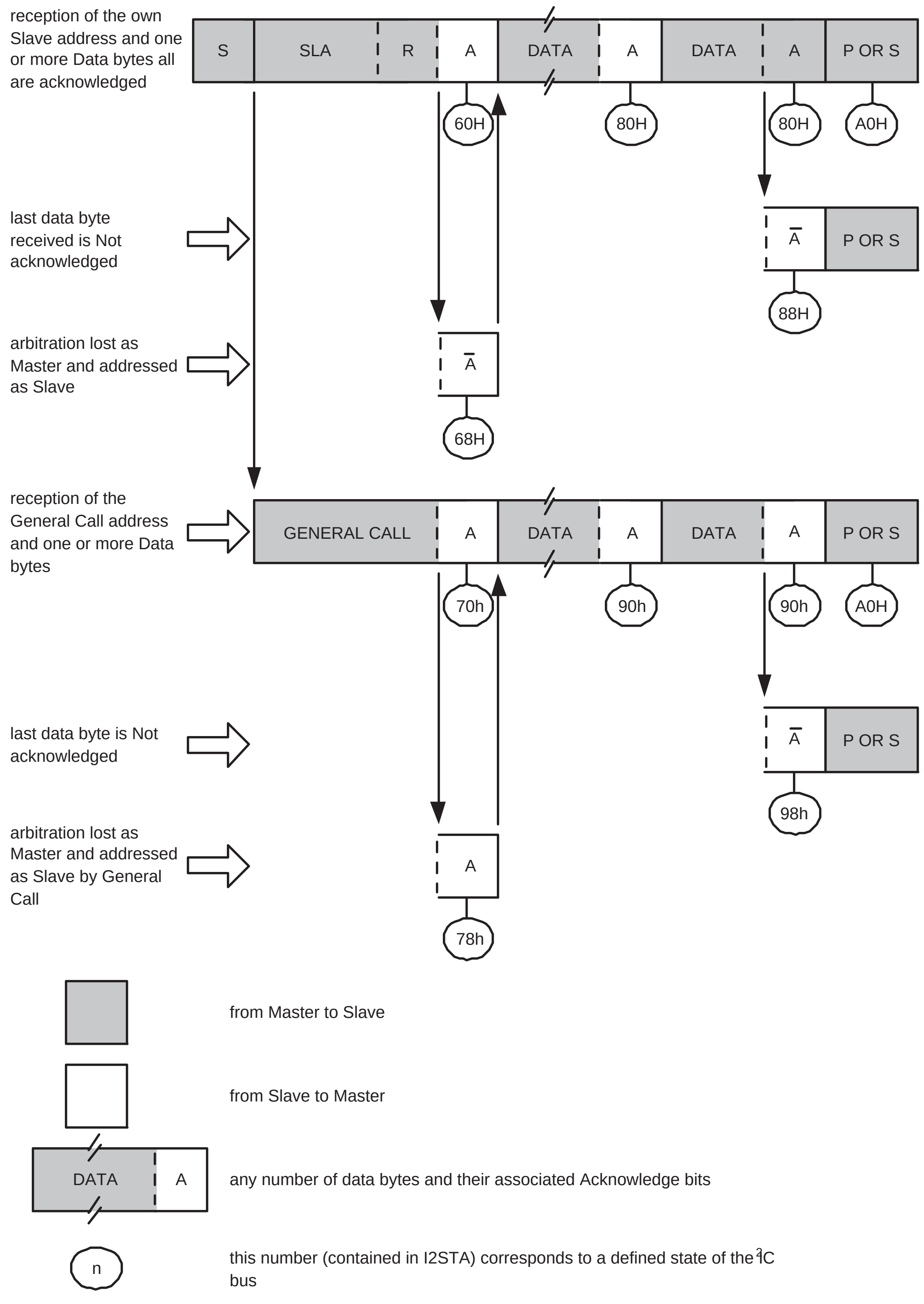

reception of the own Slave address and one or more Data bytes all are acknowledged
S
SLA
R
A
DATA
A
DATA
A
P OR S
60H
80H
80H
A0H
last data byte received is Not acknowledged
Ā
P OR S
88H
arbitration lost as Master and addressed as Slave
Ā
68H
reception of the General Call address and one or more Data bytes
GENERAL CALL
A
DATA
A
DATA
A
P OR S
70h
90h
90h
A0H
last data byte is Not acknowledged
Ā
P OR S
98h
arbitration lost as Master and addressed as Slave by General Call
A
78h
from Master to Slave
from Slave to Master
DATA
A
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 44. Format and States in the Slave Receiver mode

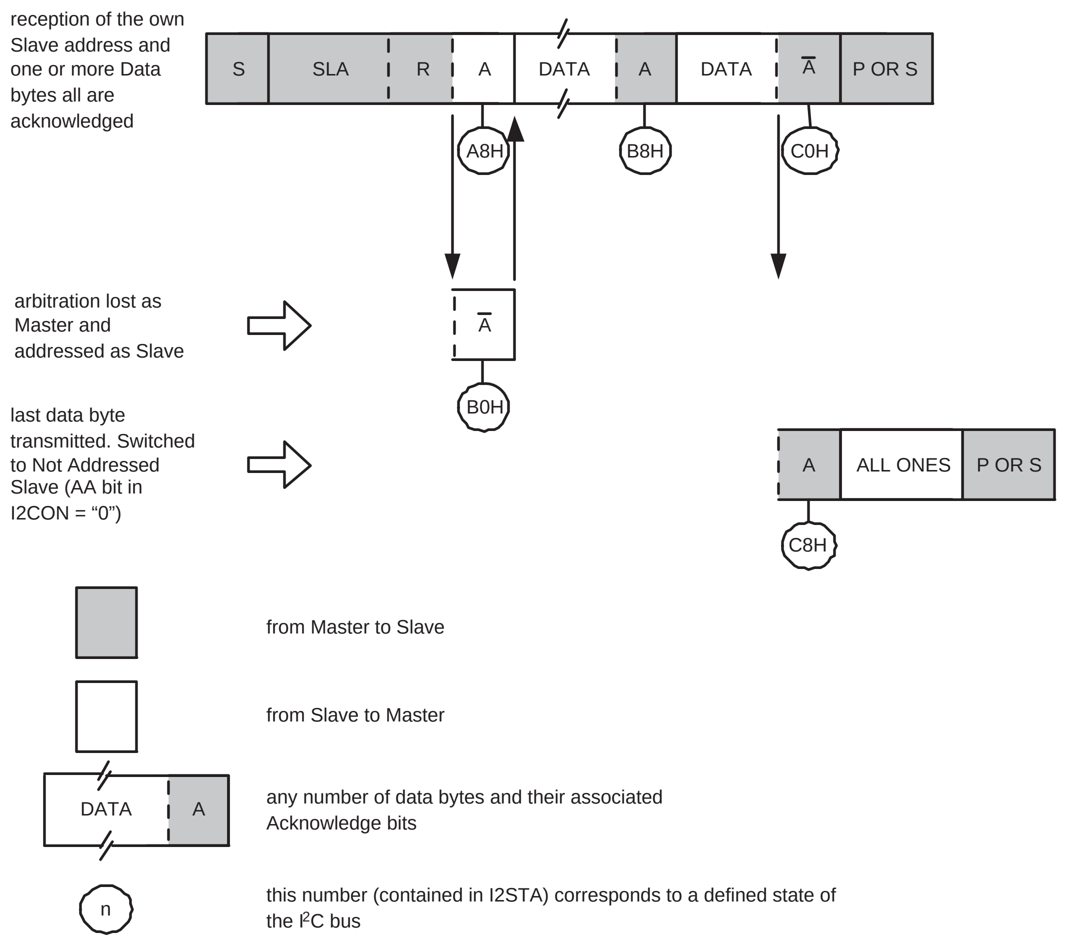

reception of the own Slave address and one or more Data bytes all are acknowledged
S
SLA
R
A
DATA
A
DATA
$\overline{A}$
P OR S
A8H
B8H
C0H
arbitration lost as Master and addressed as Slave
$\overline{A}$
B0H
last data byte transmitted. Switched to Not Addressed Slave (AA bit in I2CON = "0")
A
ALL ONES
P OR S
C8H
from Master to Slave
from Slave to Master
DATA
A
any number of data bytes and their associated Acknowledge bits
n
this number (contained in I2STA) corresponds to a defined state of the I²C bus

Fig 45. Format and States in the Slave Transmitter mode

12.9.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 45). Data transfer is initialized as in the slave receiver mode. When I2ADR and I2CON have been initialized, the I^{2}C block waits until it is addressed by its own slave address followed by the data direction bit which must be "1" (R) for the I^{2}C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 189. The slave transmitter mode may also be entered if arbitration is lost while the I^{2}C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I^{2}C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^{2}C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I^{2}C block does not respond to its own slave address or a general call address. However, the I^{2}C-bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^{2}C block from the I^{2}C-bus.

Table 186. Master Transmitter mode

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W Clear STA	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R Clear STA	X	0	0	X	SLA+W will be transmitted; the I^{2}C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in I2DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No I2DAT action or	0	0	0	X	I^{2}C-bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

Table 187. Master Receiver mode

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I^{2}C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No I2DAT action or	0	0	0	X	I^{2}C-bus will be released; the I^{2}C block will enter a slave mode.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No I2DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No I2DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No I2DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 188. Slave Receiver mode

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General call address (0x00) has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General call address has been received, ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLV address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Table 188. Slave Receiver mode

Status Code (I2CSTAT)	Status of the I^{2}C-bus and hardware	Application software response					Next action taken by I^{2}C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

Table 189. Slave Transmitter mode

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

12.9.5 Miscellaneous States

There are two I2STAT codes that do not correspond to a defined I^{2}C hardware state (see Table 190). These are discussed below.

12.9.6 I2STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I^{2}C block is not involved in a serial transfer.

12.9.7 I2STAT = 0x00

This status code indicates that a bus error has occurred during an I^{2}C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I^{2}C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I^{2}C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in I2CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 190. Miscellaneous States

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

12.9.8 Some special cases

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

12.9.9 Simultaneous repeated START conditions from two masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 46). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I^2C hardware detects a repeated START condition on the I^2C-bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, the I^2C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

12.9.10 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 40). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 42 and Figure 43).

If the STA flag in I2CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 0x08) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

12.9.11 Forced access to the I^2C-bus

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

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

12.9.12 I^{2}C-bus obstructed by a low level on SCL or SDA

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

If the SDA line is obstructed by another device on the bus (e.g., a slave device out of bit synchronization), the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 48). The I^{2}C hardware transmits additional clock pulses when the STA flag is set, but no START condition can be generated because the SDA line is pulled LOW while the I^{2}C-bus is considered free. The I^{2}C hardware attempts to generate a START condition after every two additional clock pulses on the SCL line. When the SDA line is eventually released, a normal START condition is transmitted, state 0x08 is entered, and the serial transfer continues.

If a forced bus access occurs or a repeated START condition is transmitted while SDA is obstructed (pulled LOW), the I^{2}C hardware performs the same action as described above. In each case, state 0x08 is entered after a successful START condition is transmitted and normal serial transfer continues. Note that the CPU is not involved in solving these bus hang-up problems.

12.9.13 Bus error

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

The I^{2}C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^{2}C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 190.

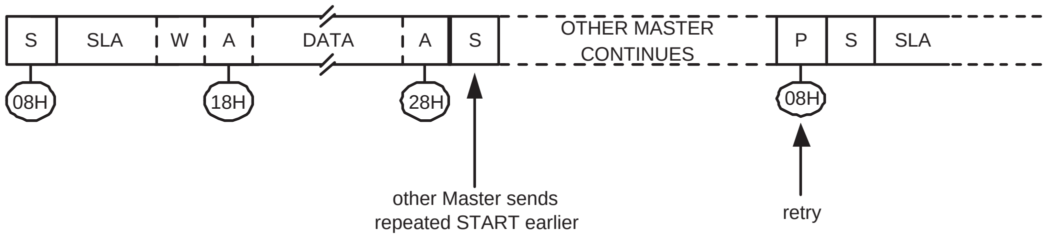


Fig 46. Simultaneous repeated START conditions from two masters

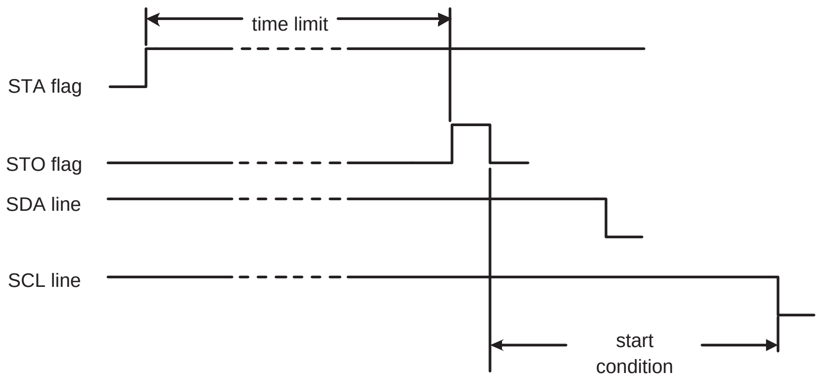


Fig 47. Forced access to a busy I^{2}C-bus

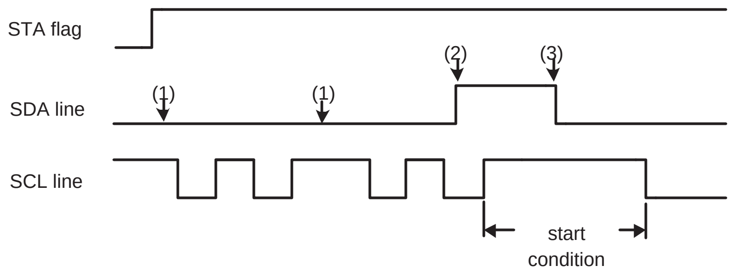


Fig 48. Recovering from a bus obstruction caused by a low level on SDA

12.9.14 I^{2}C State service routines

This section provides examples of operations that must be performed by various I^{2}C state service routines. This includes:

- Initialization of the I^{2}C block after a Reset.
- I^{2}C Interrupt Service
- The 26 state service routines providing support for all four I^{2}C operating modes.

12.9.15 Initialization

In the initialization example, the I^{2}C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- I2ADR is loaded with the part’s own slave address and the general call bit (GC)
- The I^{2}C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by loading CR0 and CR1 in I2CON. The master routines must be started in the main program.

The I^{2}C hardware now begins checking the I^{2}C-bus for its own slave address and general call. If the general call or the own slave address is detected, an interrupt is requested and I2STAT is loaded with the appropriate state information.

12.9.16 I^{2}C interrupt service

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

12.9.17 The State service routines

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

12.9.18 Adapting State services to an application

The state service examples show the typical actions that must be performed in response to the 26 I^{2}C state codes. If one or more of the four I^{2}C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of time-out during I^{2}C operations, in order to trap an inoperative bus or a lost service routine.

12.10 Software example

12.10.1 Initialization routine

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

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

12.10.2 Start Master Transmit function

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

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

12.10.3 Start Master Receive function

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

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

12.10.4 I^{2}C interrupt routine

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

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

12.10.5 Non mode specific States

12.10.6 State: 0x00

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

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.7 Master States

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

12.10.8 State: 0x08

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

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

12.10.9 State: 0x10

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

1. Write Slave Address with R/W bit to I2DAT.

2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

12.10.10 Master Transmitter States

12.10.11 State: 0x18

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

1. Load I2DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

12.10.12 State: 0x20

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

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.13 State: 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a Stop condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit
5. Load I2DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to I2CONSET to set the AA bit.
7. Write 0x08 to I2CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

12.10.14 State: 0x30

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

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.15 State: 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new Start condition will be transmitted when the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.16 Master Receive States

12.10.17 State: 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be

received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.18 State: 0x48

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

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.19 State: 0x50

Data has been received, ACK has been returned. Data will be read from I2DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from I2DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer

8. Exit

12.10.20 State: 0x58

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

1. Read data byte from I2DAT into Master Receive buffer.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit

12.10.21 Slave Receiver States

12.10.22 State: 0x60

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

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

12.10.23 State: 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

12.10.24 State: 0x70

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

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

12.10.25 State: 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

12.10.26 State: 0x80

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

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

12.10.27 State: 0x88

Previously addressed with own Slave Address. Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.28 State: 0x90

Previously addressed with general call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
3. Exit

12.10.29 State: 0x98

Previously addressed with general call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.30 State: 0xA0

A Stop condition or repeated Start has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

12.10.31 Slave Transmitter States

12.10.32 State: 0xA8

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

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

12.10.33 State: 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to I2CONSET to set the STA and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

12.10.34 State: 0xB8

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

1. Load I2DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

12.10.35 State: 0xC0

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

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit.

12.10.36 State: 0xC8

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

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

UM10114

Chapter 13: LPC21xx/22xx SPI

Rev. 4 — 2 May 2012 User manual

13.1 How to read this chapter

All LPC21xx and all LPC22xx have by default two SPI interfaces SPI0 and SPI1.

Remark: For enhanced parts only, the SPI1 interface can be selected as an SSP interface using the same pins as SPI1 (see Section 14.1).

Table 191. LPC21xx/22xx SPI configurations

Part	SPI data transfer width (see Table 195)	SSEL pin usable as GPIO (see Table 193)	SSP interface selectable for SPI1
no suffix and /00 parts			
LPC2109	8 bit, fixed	no	no
LPC2119	8 bit, fixed	no	no
LPC2129	8 bit, fixed	no	no
LPC2114	8 bit, fixed	no	no
LPC2124	8 bit, fixed	no	no
LPC2194	8 bit, fixed	no	no
LPC2210	8 bit, fixed	no	no
LPC2220	8 bit, fixed	no	no
LPC2212	8 bit, fixed	no	no
LPC2214	8 bit, fixed	no	no
LPC2290	8 bit, fixed	no	no
LPC2292	8 bit, fixed	no	no
LPC2294	8 bit, fixed	no	no
/01 parts			
LPC2109	8 to 16 bit	yes	yes
LPC2119	8 to 16 bit	yes	yes
LPC2129	8 to 16 bit	yes	yes
LPC2114	8 to 16 bit	yes	yes
LPC2124	8 to 16 bit	yes	yes
LPC2194	8 to 16 bit	yes	yes
LPC2210	8 bit, fixed	no	yes
LPC2212	8 to 16 bit	yes	yes
LPC2214	8 to 16 bit	yes	yes
LPC2290	8 bit, fixed	yes	yes
LPC2292	8 to 16 bit	yes	yes
LPC2294	8 to 16 bit	yes	yes

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

13.2 Features

- Two complete and independent SPI controllers
- Compliant with Serial Peripheral Interface (SPI) specification
- Synchronous, serial, and full duplex communication
- Combined SPI master and slave
- Maximum data bit rate of one eighth of the input clock rate
- 8 bit only or 8 to 16 bit per transfer

13.3 Description

13.3.1 SPI overview

SPI 0 and SPI1 are full duplex serial interfaces. They can handle multiple masters and slaves being connected to a given bus. Only a single master and a single slave can communicate on the interface during a given data transfer. During a data transfer the master always sends a byte of data to the slave, and the slave always sends a byte of data to the master.

13.3.2 SPI data transfers

Figure 49 is a timing diagram that illustrates the four different data transfer formats that are available with the SPI. This timing diagram illustrates a single 8 bit data transfer. The first thing you should notice in this timing diagram is that it is divided into three horizontal parts. The first part describes the SCK and SSEL signals. The second part describes the MOSI and MISO signals when the CPHA variable is 0. The third part describes the MOSI and MISO signals when the CPHA variable is 1.

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

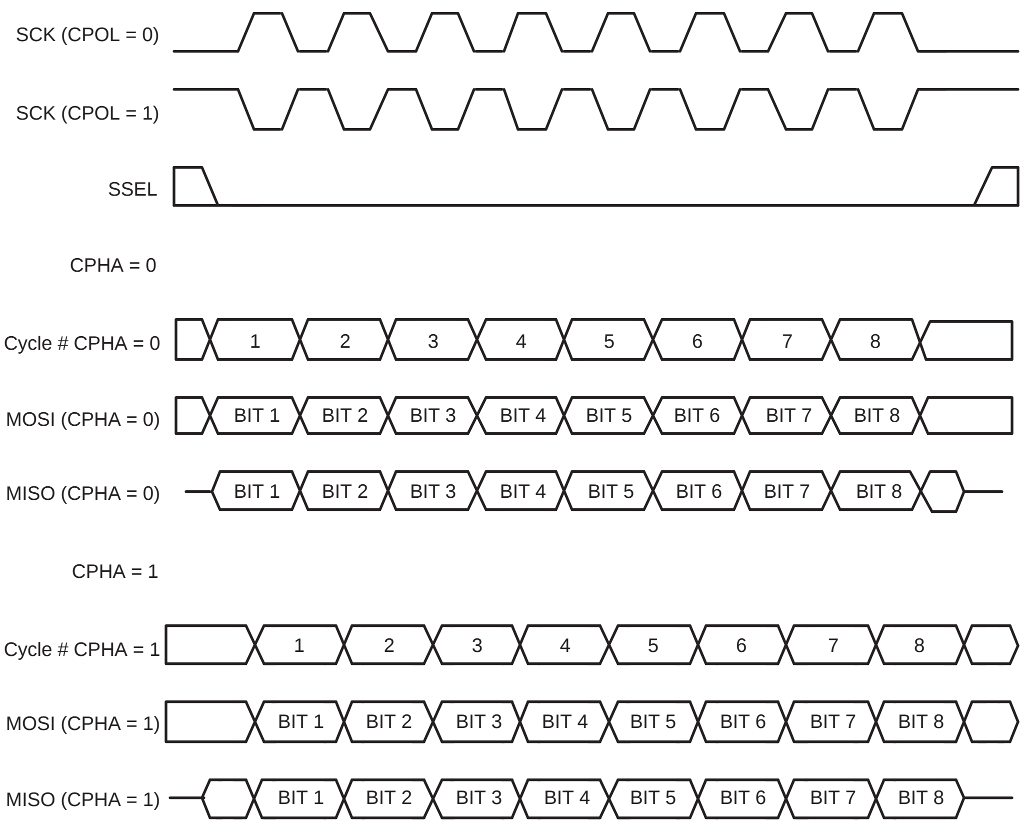

SCK (CPOL = 0)
SCK (CPOL = 1)
SSEL
CPHA = 0
Cycle # CPHA = 0
1 2 3 4 5 6 7 8
MOSI (CPHA = 0)
BIT 1 BIT 2 BIT 3 BIT 4 BIT 5 BIT 6 BIT 7 BIT 8
MISO (CPHA = 0)
BIT 1 BIT 2 BIT 3 BIT 4 BIT 5 BIT 6 BIT 7 BIT 8
CPHA = 1
Cycle # CPHA = 1
1 2 3 4 5 6 7 8
MOSI (CPHA = 1)
BIT 1 BIT 2 BIT 3 BIT 4 BIT 5 BIT 6 BIT 7 BIT 8
MISO (CPHA = 1)
BIT 1 BIT 2 BIT 3 BIT 4 BIT 5 BIT 6 BIT 7 BIT 8

Fig 49. SPI data transfer format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in Table 192. 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 192. 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.

13.3.3 SPI peripheral details

13.3.3.1 General information

There are four registers that control the SPI peripheral. They are described in detail in Section 13.5 "Register description" on page 213.

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.

13.3.3.2 Master operation

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

1. Set the SPI clock counter register to the desired clock rate.
2. Set the SPI control register to the desired settings.
3. Write the data to transmitted to the SPI data register. This write starts the SPI data transfer.
4. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
5. Read the SPI status register.

6. Read the received data from the SPI data register (optional).
7. Go to step 3 if more data is required to transmit.

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

13.3.3.3 Slave operation

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

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

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

13.3.3.4 Exception conditions

13.3.3.4.1 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.

13.3.3.4.2 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.

13.3.3.4.3 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.

13.3.3.4.4 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.

13.4 Pin description

Table 193. SPI pin description

Pin name	Type	Pin description
SCK0/ SCK1	Input/ Output	**Serial Clock.** The SPI is a clock signal used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL0/ SSEL1	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. **Remark: Flashless LPC22xx and all legacy parts (/00, /01, and no suffix) configured to operate as a SPI master MUST select SSEL functionality on an appropriate pin and have HIGH level on this pin in order to act as a master.** For all other LPC21xx and LPC22xx parts, the SSEL pin can be used for a different function when the SPI interface is only used in Master mode. For example, the pin hosting the SSEL function can be configured as an output digital GPIO pin and can be used to select one of the SPI slaves.
MISO0/ MISO1	Input/ Output	**Master In Slave Out.** The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. When a device is a slave, serial data is output on this signal. When a device is a master, serial data is input on this signal. When a slave device is not selected, the slave drives the signal high impedance.
MOSI0/ MOSI1	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.

13.5 Register description

The SPI contains 5 registers as shown in Table 194. All registers are byte, half word and word accessible.

Table 194. SPI register map

Name	Description	Access	Reset value[1]	SPI0 Address & name	SPI1 Address & name
SPCR	SPI Control Register. This register controls the operation of the SPI.	R/W	0x0000	0xE002 0000 S0SPCR	0xE003 0000 S1SPCR
SPSR	SPI Status Register. This register shows the status of the SPI.	RO	0x00	0xE002 0004 S0SPSR	0xE003 0004 S1SPSR
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.	R/W	0x0000	0xE002 0008 S0SPDR	0xE003 0008 S1SPDR
SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK.	R/W	0x00	0xE002 000C S0SPCCR	0xE003 000C S1SPCCR
SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	R/W	0x00	0xE002 001C S0SPINT	0xE003 001C S1SPINT

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

13.5.1 SPI Control Register (S0SPCR - 0xE002 0000 and S1SPCR - 0xE003 0000)

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

Table 195. SPI Control Register (S0SPCR - address 0xE002 0000 and S1SPCR - address 0xE003 0000) bit description

Bit	Symbol	Value	Description	Reset value
1:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	BitEnable[1]	0	The SPI controller sends and receives 8 bits of data per transfer.	0
		1	The SPI controller sends and receives the number of bits selected by bits 11:8.	
3	CPHA	0	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. Data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	0
		1	Data is sampled on the second clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active.	
4	CPOL	0	Clock polarity control. SCK is active high.	0
		1	SCK is active low.	

Table 195. SPI Control Register (S0SPCR - address 0xE002 0000 and S1SPCR - address 0xE003 0000) bit description

Bit	Symbol	Value	Description	Reset value
5	MSTR	0	Master mode select. The SPI operates in Slave mode.	0
		1	The SPI operates in Master mode.	
6	LSBF	0	LSB First controls which direction each byte is shifted when transferred. SPI data is transferred MSB (bit 7) first.	0
		1	SPI data is transferred LSB (bit 0) first.	
7	SPIE	0	Serial peripheral interrupt enable. SPI interrupts are inhibited.	0
		1	A hardware interrupt is generated each time the SPIF or MODF bits are activated.	
11:8	BITS[1]		When bit 2 of this register is 1, this field controls the number of bits per transfer:	0000
		1000	8 bits per transfer	
		1001	9 bits per transfer	
		1010	10 bits per transfer	
		1011	11 bits per transfer	
		1100	12 bits per transfer	
		1101	13 bits per transfer	
		1110	14 bits per transfer	
		1111	15 bits per transfer	
		0000	16 bits per transfer	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] See Table 191 for data transfer width allowed.

13.5.2 SPI Status Register (S0SPSR - 0xE002 0004 and S1SPSR - 0xE003 0004)

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

Table 196. SPI Status Register (S0SPSR - address 0xE002 0004 and S1SPSR - address 0xE003 0004) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort. When 1, this bit indicates that a slave abort has occurred. This bit is cleared by reading this register.	0
4	MODF	Mode fault. when 1, this bit indicates that a Mode fault error has occurred. This bit is cleared by reading this register, then writing the SPI control register.	0

Table 196. SPI Status Register (S0SPSR - address 0xE002 0004 and S1SPSR - address 0xE003 0004) bit description

Bit	Symbol	Description	Reset value
5	ROVR	Read overrun. When 1, this bit indicates that a read overrun has occurred. This bit is cleared by reading this register.	0
6	WCOL	Write collision. When 1, this bit indicates that a write collision has occurred. This bit is cleared by reading this register, then accessing the SPI data register.	0
7	SPIF	SPI transfer complete flag. When 1, this bit indicates when a SPI data transfer is complete. When a master, this bit is set at the end of the last cycle of the transfer. When a slave, this bit is set on the last data sampling edge of the SCK. This bit is cleared by first reading this register, then accessing the SPI data register. **Note:** This is not the SPI interrupt flag. This flag is found in the SPINT register.	0

13.5.3 SPI Data Register (S0SPDR - 0xE002 0008, S1SPDR - 0xE003 0008)

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

Table 197. SPI Data Register (S0SPDR - address 0xE002 0008, S1SPDR - address 0xE003 0008) bit description

Bit	Symbol	Description	Reset value
15:0	Data	SPI Bi-directional data port.	0

13.5.4 SPI Clock Counter Register (S0SPCCR - 0xE002 000C and S1SPCCR - 0xE003 000C)

This register controls the frequency of a master's SCK. The register indicates the number of SPI peripheral clock cycles that make up an SPI clock.

In Master mode, this register must be an even number greater than or equal to 8. Violations of this can result in unpredictable behavior. The SPI SCK rate may be calculated as: PCLK / SnSPCCR value. The PCLK rate is CCLK /APB divider rate as determined by the APBDIV register contents (see Table 76).

In Slave mode, the SPI clock rate provided by the master must not exceed 1/8 of the peripheral clock. The content of the S0SPCCR register is not relevant.

Table 198. SPI Clock Counter Register (S0SPCCR - address 0xE002 000C and S1SPCCR - address 0xE003 000C) bit description

Bit	Symbol	Description	Reset value
7:0	Counter	SPI Clock counter setting.	0x00

13.5.5 SPI Interrupt Register (S0SPINT - 0xE002 001C and S1SPINT - 0xE003 001C)

This register contains the interrupt flag for the SPI interface.

Table 199. SPI Interrupt Register (S0SPINT - address 0xE002 001C and S1SPINT - address 0xE003 001C) bit description

Bit	Symbol	Description	Reset value
0	SPI Interrupt	SPI interrupt flag. Set by the SPI interface to generate an interrupt. Cleared by writing a 1 to this bit. **Note:** This bit will be set once when SPIE = 1 and at least one of SPIF and MODF bits changes from 0 to 1. However, only when the SPI Interrupt bit is set and SPI Interrupt is enabled in the VIC, SPI based interrupt can be processed by interrupt handling software.	0
7:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.6 Architecture

The block diagram of the SPI solution implemented in SPI0 and SPI1 interface is shown in the Figure 50.

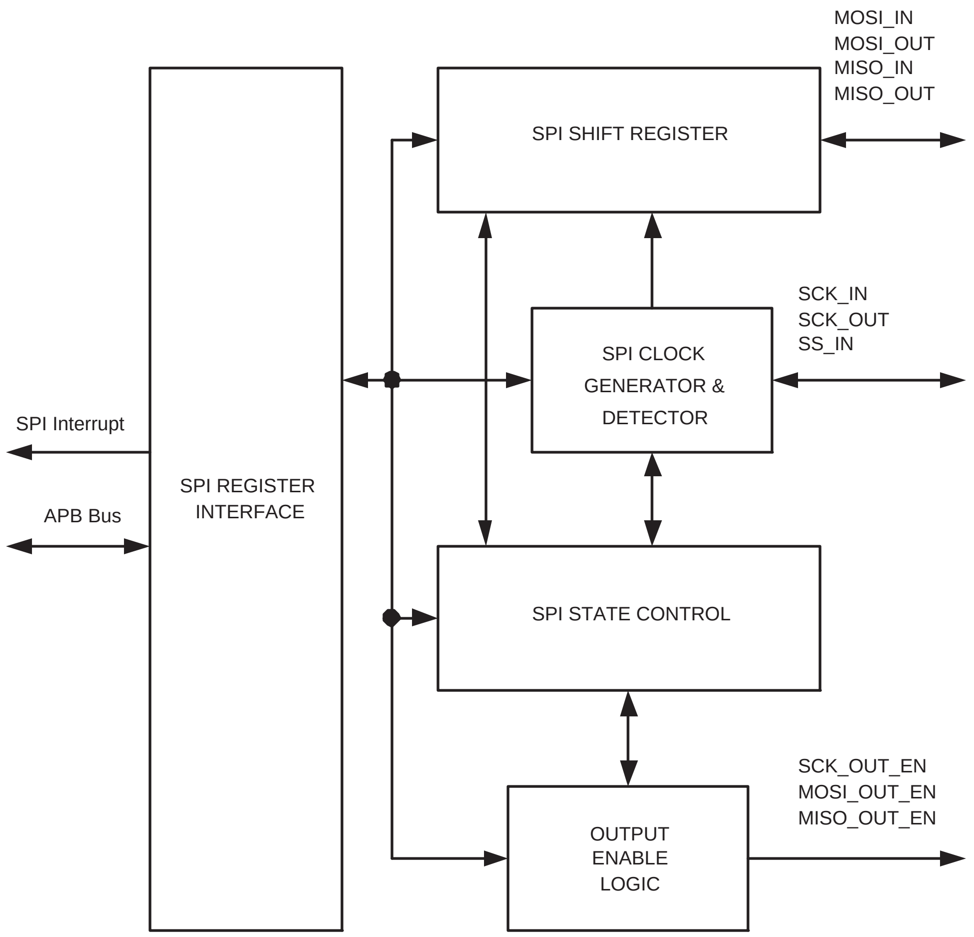


Fig 50. SPI block diagram

UM10114

Chapter 14: LPC21xx/22xx SSP interface

Rev. 4 — 2 May 2012 **User manual**

14.1 How to read this chapter

The SSP interface is available on the following parts:

- LPC2109/01, LPC2119/01, LPC2129/01
- LPC2114/01, LPC2124/01
- LPC2194/01
- LPC2210/01, LPC2220
- LPC2212/01, LPC2214/01
- LPC2292/01, LPC2294/01

The SSP interface shares its pins with the SPI1 interface. To select the SSP peripheral, select the PCSSP bit in the PCONP register (Section 6.10.3). Note that the default interface on Reset is the SPI1 interface.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

14.2 Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses
- Synchronous serial communication
- Master or slave operation
- 8-frame FIFOs for both transmit and receive
- 4 to 16 bit frame

14.3 Description

The SSP is a Synchronous Serial Port (SSP) controller capable of operation on an SPI, 4-wire SSI, or Microwire bus. It can interact with multiple masters and slaves on the bus. Only a single master and a single slave can communicate on the bus during a given data transfer. Data transfers are in principle full duplex, with frames of 4 to 16 bits of data flowing from the master to the slave and from the slave to the master. In practice it is often the case that only one of these data flows carries meaningful data.

While the SSP and SPI1 peripherals share the same physical pins, it is not possible to have both of these two peripherals active at the same time. Bit 10 (PSPI1) and bit 21 (PSSP) residing in the Section 6.10.3 control the activity of the SPI1 and SSP module respectively. The corresponding peripheral is enabled when its control bit is 1, and it is disabled when the control bit is 0. After power-on reset, SPI1 is enabled, maintaining the backward compatibility with other NXP LPC2000 microcontrollers. Any attempt to write 1 to PSPI1 and PSSP bits at the same time will result in PSPI = 1 and PSSP = 0.

To switch on the fly from SPI1 to SSP and back, first disable the active peripheral's interrupts, both in the peripheral's and VIC's registers. Next, clear all pending interrupt flags (if any set). Only then, the currently enabled peripheral can be turned off in the PCONP register. After this, the other serial interface can be enabled.

It is important to disable the currently used peripheral by clearing its bit in the PCONP register only at the very end of the peripheral's shut-down procedure. Otherwise, having 0 in a bit in PCONP will disable all clocks from coming into the peripheral controlled by that bit. Then, reading from the peripheral's registers will not yield valid data and write and/or modify access will be banned, i.e. no content can be changed. Consequently, if any of the interrupt triggering flags are left active in the peripheral's registers when the peripheral is disabled via the PCONP, the invoked ISR may not be able to successfully service pending interrupt, and the same interrupt may keep overloading the microcontroller even though its peripheral is disabled.

Table 200. SSP pin descriptions

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

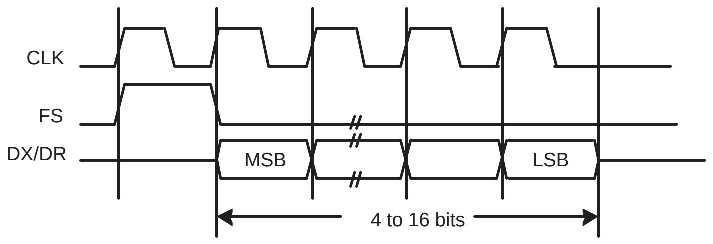
MISO1	I/O	MISO	DR(M)
			DX(S)
				SI(M)
				SO(S)
					Master In Slave Out. The MISO signal transfers serial data from the slave to the master. When the SSP is a slave, serial data is output on this signal. When the SSP is a master, it clocks in serial data from this signal. When the SSP is a slave and is not selected by SSEL, it does not drive this signal (leaves it in high impedance state).

MOSI1	I/O	MOSI	DX(M)
			DR(S)
				SO(M)
				SI(S)
					Master Out Slave In. The MOSI signal transfers serial data from the master to the slave. When the SSP is a master, it outputs serial data on this signal. When the SSP is a slave, it clocks in serial data from this signal.

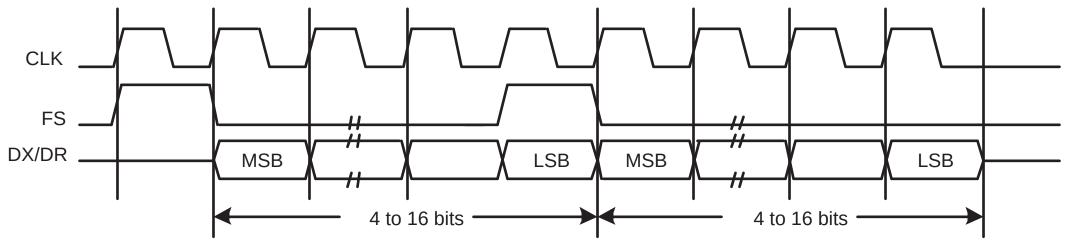
14.4 Bus description

14.4.1 Texas Instruments synchronous serial frame format

Figure 51 shows the 4-wire Texas Instruments synchronous serial frame format supported by the SSP module.



a. Single frame transfer



b. Continuous/back-to-back frames transfer

Fig 51. Texas Instruments synchronous serial frame format: a) single frame transfer and b) continuous/back-to-back two frames.

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

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

14.4.2 SPI frame format

The SPI interface is a four-wire interface where the SSEL signal behaves as a slave select. The main feature of the SPI format is that the inactive state and phase of the SCK signal are programmable through the CPOL and CPHA bits within the SSPCR0 control register.

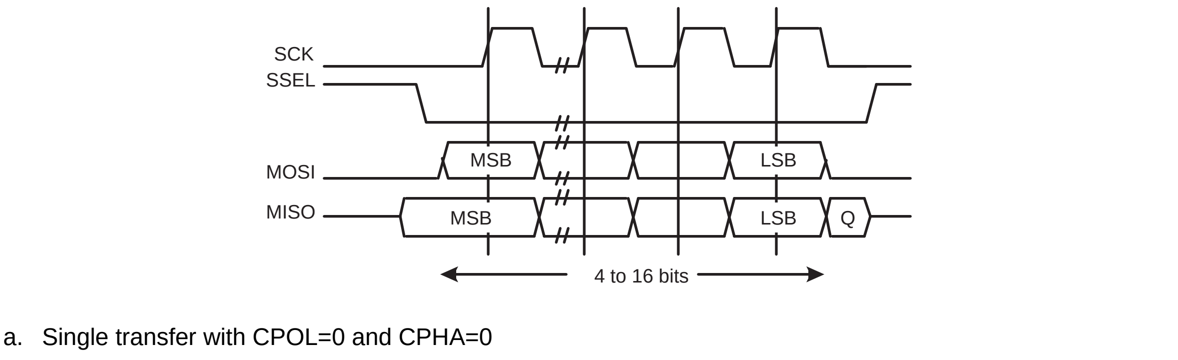
14.4.2.1 Clock Polarity (CPOL) and Phase (CPHA) Control

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

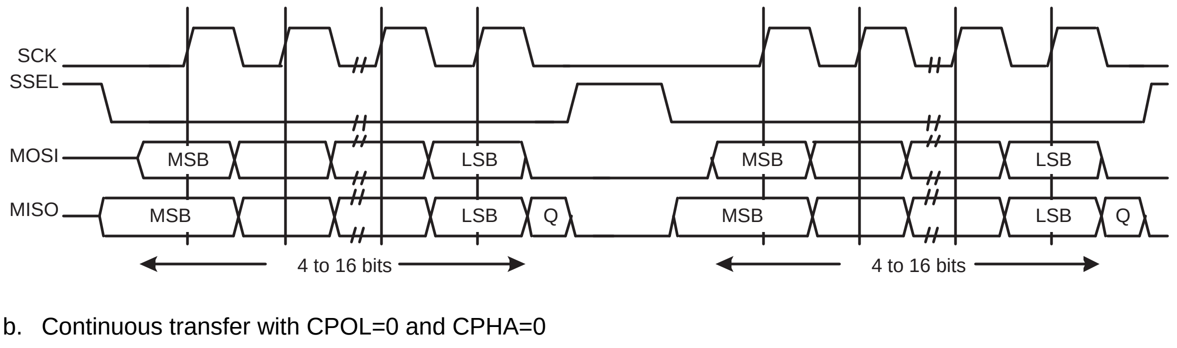
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is LOW, data is captured on the first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

14.4.2.2 SPI Format with CPOL = 0,CPHA = 0

Single and continuous transmission signal sequences for SPI format with CPOL = 0, CPHA = 0 are shown in Figure 52.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 52. Motorola SPI frame format with CPOL=0 and CPHA=0 (a) single transfer and b) continuous transfer)

In this configuration, during idle periods:

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

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

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

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

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

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

14.4.2.3 SPI format with CPOL = 0,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 53, which covers both single and continuous transfers.

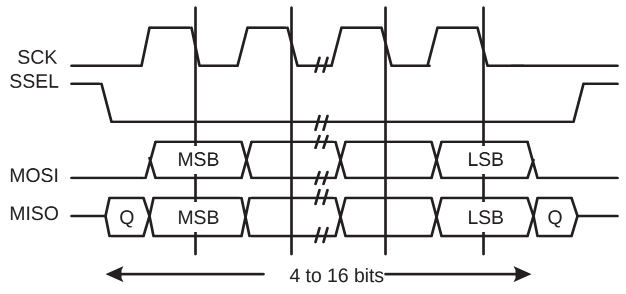


Fig 53. SPI frame format with CPOL=0 and CPHA=1

In this configuration, during idle periods:

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

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI pin is enabled. After a further one half SCK period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SCK is enabled with a rising edge transition.

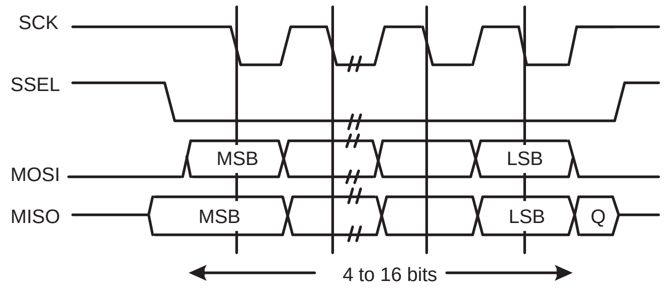
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

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

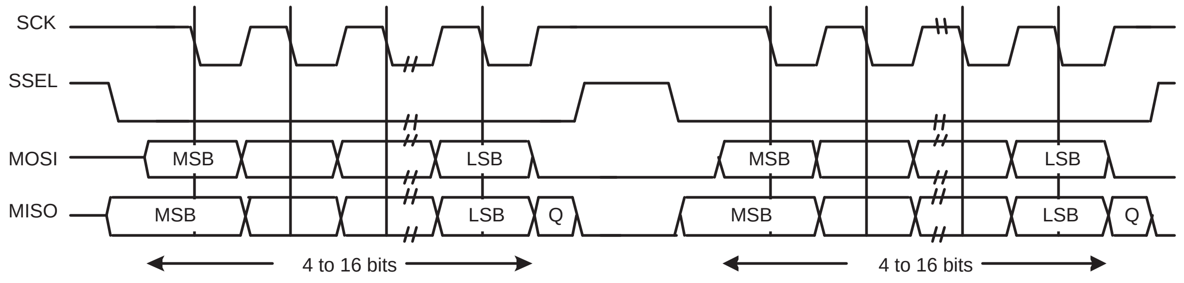
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

14.4.2.4 SPI format with CPOL = 1,CPHA = 0

Single and continuous transmission signal sequences for SPI format with CPOL=1, CPHA=0 are shown in Figure 54.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 54. SPI frame format with CPOL = 1 and CPHA = 0 (a) single and b) continuous transfer)

In this configuration, during idle periods:

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

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

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

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

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

14.4.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 55, which covers both single and continuous transfers.

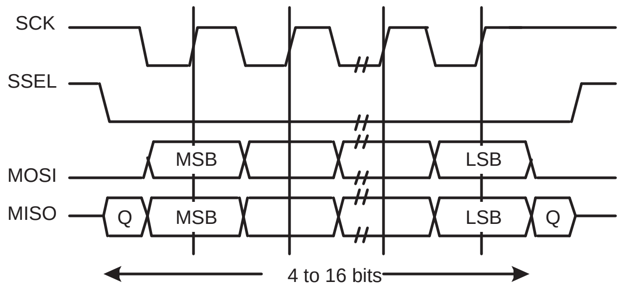


Fig 55. SPI frame format with CPOL = 1 and CPHA = 1

In this configuration, during idle periods:

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

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI is enabled. After a further one half SCK period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SCK is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SCK signal.

After all bits have been transferred, in the case of a single word transmission, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured. For continuous back-to-back transmissions, the SSEL pins remains in its active LOW state, until the final bit of the last word has been captured, and then returns to its idle state as described above. In general, for continuous back-to-back transfers the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

14.4.3 Semiconductor Microwire frame format

Figure 56 shows the Microwire frame format for a single frame. Figure 44 shows the same format when back-to-back frames are transmitted.

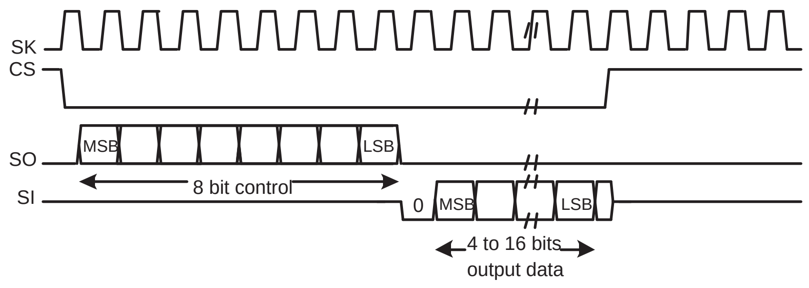


Fig 56. Microwire frame format (single transfer)

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmission, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- The SK signal is forced LOW
- CS is forced HIGH
- The transmit data line SO is arbitrarily forced LOW

A transmission is triggered by writing a control byte to the transmit FIFO.The falling edge of CS causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SO pin. CS remains LOW for the duration of the frame transmission. The SI pin remains tri-stated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SK. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSP. Each bit is driven onto SI line on the falling edge of SK. The SSP in turn latches each bit on the rising edge of SK. At the end of the frame, for single transfers, the CS signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tri-state the receive line either on the falling edge of SK after the LSB has been latched by the receive shiftier, or when the CS pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the CS line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SK, after the LSB of the frame has been latched into the SSP.

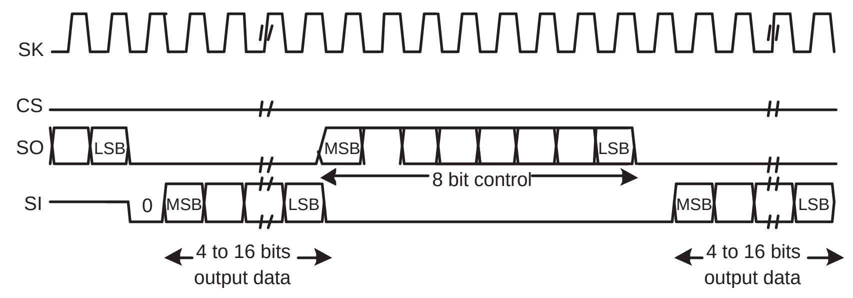


Fig 57. Microwire frame format (continuous transfers)

14.4.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode

In the Microwire mode, the SSP slave samples the first bit of receive data on the rising edge of SK after CS has gone LOW. Masters that drive a free-running SK must ensure that the CS signal has sufficient setup and hold margins with respect to the rising edge of SK.

Figure 58 illustrates these setup and hold time requirements. With respect to the SK rising edge on which the first bit of receive data is to be sampled by the SSP slave, CS must have a setup of at least two times the period of SK on which the SSP operates. With respect to the SK rising edge previous to this edge, CS must have a hold of at least one SK period.

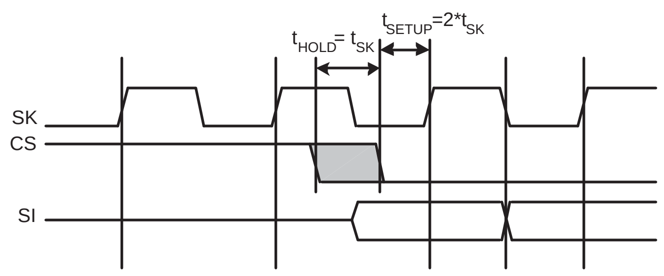


Fig 58. Microwire setup and hold details

14.5 Register description

The SSP contains 9 registers as shown in Table 201. All registers are byte, half word and word accessible.

Table 201. SSP Registers

Name	Description	Access	Reset value[1]	Address
SSPCR0	Control Register 0. Selects the serial clock rate, bus type, and data size.	R/W	0x0000	0xE005 C000
SSPCR1	Control Register 1. Selects master/slave and other modes.	R/W	0x00	0xE005 C004
SSPDR	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	R/W	0x0000	0xE005 C008
SSPSR	Status Register	RO	0x03	0xE005 C00C
SSPCPSR	Clock Prescale Register	R/W	0x00	0xE005 C010
SSPIMSC	Interrupt Mask Set and Clear Register	R/W	0x00	0xE005 C014
SSPRIS	Raw Interrupt Status Register	R/W	0x08	0xE005 C018
SSPMIS	Masked Interrupt Status Register	RO	0x00	0xE005 C01C
SSPICR	SSPICR Interrupt Clear Register	WO	NA	0xE005 C020

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

14.5.1 SSP Control Register 0 (SSPCR0 - 0xE005 C000)

This register controls the basic operation of the SSP controller.

Table 202: SSP Control Register 0 (SSPCR0 - address 0xE005 C000) bit description

Bit	Symbol	Value	Description	Reset value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0011	4 bit transfer	
		0100	5 bit transfer	
		0101	6 bit transfer	
		0110	7 bit transfer	
		0111	8 bit transfer	
		1000	9 bit transfer	
		1001	10 bit transfer	
		1010	11 bit transfer	
		1011	12 bit transfer	
		1100	13 bit transfer	
		1101	14 bit transfer	
		1110	15 bit transfer	
		1111	16 bit transfer	
5:4	FRF		Frame Format.	00
		00	SPI	
		01	SSI	
		10	Microwire	
		11	This combination is not supported and should not be used.	

Table 202: SSP Control Register 0 (SSPCR0 - address 0xE005 C000) bit description

Bit	Symbol	Value	Description	Reset value
6	CPOL	0	Clock Out Polarity. This bit is only used in SPI mode. SSP controller maintains the bus clock low between frames.	0
		1	SSP controller maintains the bus clock high between frames.	
7	CPHA	0	Clock Out Phase. This bit is only used in SPI mode. SSP controller captures serial data on the first clock transition of the frame, that is, the transition **away from** the inter-frame state of the clock line.	0
		1	SSP controller captures serial data on the second clock transition of the frame, that is, the transition **back to** the inter-frame state of the clock line.	
15:8	SCR		Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVR is the prescale divider, and the VPB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR * [SCR+1]).	0x00

14.5.2 SSP Control Register 1 (SSPCR1 - 0xE005 C004)

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

Table 203: SSP Control Register 1 (SSPCR1 - address 0xE005 C004) bit description

Bit	Symbol	Value	Description	Reset Value
0	LBM	0	Loop Back Mode. During normal operation.	0
		1	Serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively).	
1	SSE	0	SSP Enable. The SSP controller is disabled.	0
		1	The SSP controller will interact with other devices on the serial bus. Software should write the appropriate control information to the other SSP registers and interrupt controller registers, before setting this bit.	
2	MS	0	Master/Slave Mode.This bit can only be written when the SSE bit is 0. The SSP controller acts as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line.	0
		1	The SSP controller acts as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines.	
3	SOD		Slave Output Disable. This bit is relevant only in slave mode (MS = 1). If it is 1, this blocks this SSP controller from driving the transmit data line (MISO).	0
7:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.5.3 SSP Data Register (SSPDR - 0xE005 C008)

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

Table 204: SSP Data Register (SSPDR - address 0xE005 C008) bit description

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

14.5.4 SSP Status Register (SSPSR - 0xE005 C00C)

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

Table 205: SSP Status Register (SSPSR - address 0xE005 C00C) bit description

Bit	Symbol	Description	Reset value
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
4	BSY	Busy. This bit is 0 if the SSP controller is idle, or 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.5.5 SSP Clock Prescale Register (SSPCPSR - 0xE005 C010)

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

Table 206: SSP Clock Prescale Register (SSPCPSR - address 0xE005 C010) bit description

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

Important: the SSPCPSR value must be properly initialized or the SSP controller will not be able to transmit data correctly.

In Slave mode, the SSP clock rate provided by the master must not exceed 1/12 of the peripheral clock. The content of the SSPCPSR register is not relevant.

In master mode, $CPSDVSR_{min}$ = 2 or larger (even numbers only).

14.5.6 SSP Interrupt Mask Set/Clear Register (SSPIMSC - 0xE005 C014)

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

Table 207: SSP Interrupt Mask Set/Clear Register (SSPIMSC - address 0xE005 CF014) bit description

Bit	Symbol	Description	Reset value
0	RORIM	Software should set this bit to enable interrupt when a Receive Overrun occurs, that is, when the Rx FIFO is full and another frame is completely received. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTIM	Software should set this bit to enable interrupt when a Receive Timeout condition occurs. A Receive Timeout occurs when the Rx FIFO is not empty, and no new data has been received, nor has data been read from the FIFO, for 32 bit times.	0
2	RXIM	Software should set this bit to enable interrupt when the Rx FIFO is at least half full.	0
3	TXIM	Software should set this bit to enable interrupt when the Tx FIFO is at least half empty.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.5.7 SSP Raw Interrupt Status Register (SSPRIS - 0xE005 C018)

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

Table 208: SSP Raw Interrupt Status Register (SSPRIS - address 0xE005 C018) bit description

Bit	Symbol	Description	Reset value
0	RORRIS	This bit is 1 if another frame was completely received while the RxFIFO was full. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTRIS	This bit is 1 if when there is a Receive Timeout condition. **Note:** A Receive Timeout can be negated if further data is received.	0
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.5.8 SSP Masked Interrupt Register (SSPMIS - 0xE005 C01C)

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

Table 209: SSP Masked Interrupt Status Register (SSPMIS -address 0xE005 C01C) bit description

Bit	Symbol	Description	Reset value
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0
1	RTMIS	This bit is 1 when there is a Receive Timeout condition and this interrupt is enabled. **Note:** A Receive Timeout can be negated if further data is received.	0
2	RXMIS	This bit is 1 if the Rx FIFO is at least half full, and this interrupt is enabled.	0
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, and this interrupt is enabled.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.5.9 SSP Interrupt Clear Register (SSPICR - 0xE005 C020)

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

Table 210: SSP interrupt Clear Register (SSPICR - address 0xE005 C020) bit description

Bit	Symbol	Description	Reset value
0	RORIC	Writing a 1 to this bit clears the “frame was received when RxFIFO was full” interrupt.	Undefined
1	RTIC	Writing a 1 to this bit clears the Receive Timeout interrupt.	Undefined
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UM10114

Chapter 15: LPC21xx/22xx Timer 0/1

Rev. 4 — 2 May 2012 User manual

15.1 How to read this chapter

Remark: External event counting on the capture inputs can be selected for LPC21xx/01, LPC22xx/01, and LPC2220 parts only. External event counting uses the TnCTCTR registers. All other features of the counter/timer block are identical for all LPC21xx and LPC22xx parts.

Table 211. LPC21xx/22xx part-specific registers for external event counting

Part	T0CTCTR/T1CTCR registers	Part	T0CTCTR/T1CTCR registers
no suffix and /00 parts		**/01 parts**	
LPC2109	n/a	LPC2109	Section 15.6.3
LPC2119	n/a	LPC2119	Section 15.6.3
LPC2129	n/a	LPC2129	Section 15.6.3
LPC2114	n/a	LPC2114	Section 15.6.3
LPC2124	n/a	LPC2124	Section 15.6.3
LPC2194	n/a	LPC2194	Section 15.6.3
LPC2210	n/a	LPC2210	Section 15.6.3
LPC2220	Section 15.6.3	LPC2212	Section 15.6.3
LPC2212	n/a	LPC2214	Section 15.6.3
LPC2214	n/a	LPC2290	Section 15.6.3
LPC2290	n/a	LPC2292	Section 15.6.3
LPC2292	n/a	LPC2294	Section 15.6.3
LPC2294	n/a		

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

15.2 Features

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Counter or Timer operation
- External Event Counting capabilities.
- Up to four 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to four external outputs corresponding to match registers, with the following capabilities:
  - Set low on match.

  – Set high on match.
  – Toggle on match.
  – Do nothing on match.

15.3 Applications

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.
- External Event/Clock counter.

15.4 Description

The Timer/Counter is designed to count cycles of the peripheral clock (PCLK) or an externally-supplied clock, and can optionally generate interrupts or perform other actions at specified timer values, based on four match registers. It also includes four capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

15.5 Pin description

Table 212 gives a brief summary of each of the Timer/Counter related pins.

Table 212. Timer/Counter pin description

Pin	Type	Description
CAP0.3..0 CAP1.3..0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. When more than one pin is selected for a Capture input on a single TIMER0/1 channel, the pin with the lowest Port number is used. If for example pins 30 (P0.6) and 46 (P0.16) are selected for CAP0.2, only pin 30 will be used by TIMER0 to perform CAP0.2 function. Here is the list of all CAPTURE signals, together with pins on where they can be selected: • CAP0.0 (3 pins): P0.2, P0.22 and P0.30 • CAP0.1 (2 pins): P0.4 and P0.27 • CAP0.2 (3 pin): P0.6, P0.16 and P0.28 • CAP0.3 (1 pin): P0.29 • CAP1.0 (1 pin): P0.10 • CAP1.1 (1 pin): P0.11 • CAP1.2 (2 pins): P0.17 and P0.19 • CAP1.3 (2 pins): P0.18 and P0.21 Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 15.6.3 “Count Control Register (CTCR, TIMER0: T0CTCR - 0xE000 4070 and TIMER1: T1CTCR - 0xE000 8070)” on page 237.
MAT0.3..0 MAT1.3..0	Output	External Match Output 0/1- When a match register 0/1 (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel. It is also possible for example, to have 2 pins selected at the same time so that they provide MAT1.3 function in parallel. Here is the list of all MATCH signals, together with pins on where they can be selected: • MAT0.0 (2 pins): P0.3 and P0.22 • MAT0.1 (2 pins): P0.5 and P0.27 • MAT0.2 (2 pin): P0.16 and P0.28 • MAT0.3 (1 pin): P0.29 • MAT1.0 (1 pin): P0.12 • MAT1.1 (1 pin): P0.13 • MAT1.2 (2 pins): P0.17 and P0.19 • MAT1.3 (2 pins): P0.18 and P0.20

15.6 Register description

Each Timer/Counter contains the registers shown in Table 213. More detailed descriptions follow.

Table 213. TIMER/COUNTER0 and TIMER/COUNTER1 register map

Generic Name	Description	Access	Reset value[1]	TIMER/ COUNTER0 Address & Name	TIMER/ COUNTER1 Address & Name
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	0xE000 4000 T0IR	0xE000 8000 T1IR
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	0xE000 4004 T0TCR	0xE000 8004 T1TCR
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	0xE000 4008 T0TC	0xE000 8008 T1TC
PR	Prescale Register. The Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	0xE000 400C T0PR	0xE000 800C T1PR
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	0xE000 4010 T0PC	0xE000 8010 T1PC
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	0xE0004014 T0MCR	0xE000 8014 T1MCR
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	0xE000 4018 T0MR0	0xE000 8018 T1MR0
MR1	Match Register 1. See MR0 description.	R/W	0	0xE000 401C T0MR1	0xE000 801C T1MR1
MR2	Match Register 2. See MR0 description.	R/W	0	0xE000 4020 T0MR2	0xE000 8020 T1MR2
MR3	Match Register 3. See MR0 description.	R/W	0	0xE000 4024 T0MR3	0xE000 8024 T1MR3
CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0	0xE000 4028 T0CCR	0xE000 8028 T1CCR
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0(CAP0.0 or CAP1.0 respectively) input.	RO	0	0xE000 402C T0CR0	0xE000 802C T1CR0
CR1	Capture Register 1. See CR0 description.	RO	0	0xE000 4030 T0CR1	0xE000 8030 T1CR1
CR2	Capture Register 2. See CR0 description.	RO	0	0xE000 4034 T0CR2	0xE000 8034 T1CR2

Table 213. TIMER/COUNTER0 and TIMER/COUNTER1 register map

Generic Name	Description	Access	Reset value[1]	TIMER/ COUNTER0 Address & Name	TIMER/ COUNTER1 Address & Name
CR3	Capture Register 3. See CR0 description.	RO	0	0xE000 4038 T0CR3	0xE000 8038 T1CR3
EMR	External Match Register. The EMR controls the external match pins MATn.0-3 (MAT0.0-3 and MAT1.0-3 respectively).	R/W	0	0xE000 403C T0EMR	0xE000 803C T1EMR
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	0xE000 4070 T0CTCR	0xE000 8070 T1CTCR

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

15.6.1 Interrupt Register (IR, TIMER0: T0IR - 0xE000 4000 and TIMER1: T1IR - 0xE000 8000)

The Interrupt Register consists of four bits for the match interrupts and four bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 214: Interrupt Register (IR, TIMER0: T0IR - address 0xE000 4000 and TIMER1: T1IR - address 0xE000 8000) bit description

Bit	Symbol	Description	Reset value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
6	CR2 Interrupt	Interrupt flag for capture channel 2 event.	0
7	CR3 Interrupt	Interrupt flag for capture channel 3 event.	0

15.6.2 Timer Control Register (TCR, TIMER0: T0TCR - 0xE000 4004 and TIMER1: T1TCR - 0xE000 8004)

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

Table 215: Timer Control Register (TCR, TIMER0: T0TCR - address 0xE000 4004 and TIMER1: T1TCR - address 0xE000 8004) bit description

Bit	Symbol	Description	Reset value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.6.3 Count Control Register (CTCR, TIMER0: T0CTCR - 0xE000 4070 and TIMER1: T1CTCR - 0xE000 8070)

Remark: This register is available for LPC21xx/01, LPC22xx/01, and LPC2220 only.

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edges for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event corresponds to the one selected by bits 1:0 in the CTCR register, the Timer Counter register will be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input can not exceed one half of the PCLK clock. Consequently, duration of the high/low levels on the same CAP input in this case can not be shorter than 1/PCLK.

Table 216: Count Control Register (CTCR, TIMER0: T0CTCR - address 0xE000 4070 and TIMER1: T1CTCR - address 0xE000 8070) bit description

Bit	Symbol	Value	Description	Reset value
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment Timer’s Prescale Counter (PC), or clear PC and increment Timer Counter (TC).	00
		00	Timer Mode: every rising PCLK edge	
		01	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	

Table 216: Count Control Register (CTCR, TIMER0: T0CTCR - address 0xE000 4070 and TIMER1: T1CTCR - address 0xE000 8070) bit description

Bit	Symbol	Value	Description	Reset value
3:2	Count Input Select		When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking:	00
		00	CAPn.0 (CAP0.0 for TIMER0 and CAP1.0 for TIMER1)	
		01	CAPn.1 (CAP0.1 for TIMER0 and CAP1.1 for TIMER1)	
		10	CAPn.2 (CAP0.2 for TIMER0 and CAP1.2 for TIMER1)	
		11	CAPn.3 (CAP0.3 for TIMER0 and CAP1.3 for TIMER1)	
			Note: If Counter mode is selected for a particular CAPn input in the TnCTCR, the 3 bits for that input in the Capture Control Register (TnCCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	
7:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.6.4 Timer Counter (TC, TIMER0: T0TC - 0xE000 4008 and TIMER1: T1TC - 0xE000 8008)

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

15.6.5 Prescale Register (PR, TIMER0: T0PR - 0xE000 400C and TIMER1: T1PR - 0xE000 800C)

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

15.6.6 Prescale Counter Register (PC, TIMER0: T0PC - 0xE000 4010 and TIMER1: T1PC - 0xE000 8010)

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1, etc.

15.6.7 Match Registers (MR0 - MR3)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

15.6.8 Match Control Register (MCR, TIMER0: T0MCR - 0xE000 4014 and TIMER1: T1MCR - 0xE000 8014)

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 217.

Table 217: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		0	Feature disabled.	
6	MR2I	1	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		0	This interrupt is disabled	
7	MR2R	1	Reset on MR2: the TC will be reset if MR2 matches it.	0
		0	Feature disabled.	
8	MR2S	1	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		0	Feature disabled.	
9	MR3I	1	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		0	This interrupt is disabled	
10	MR3R	1	Reset on MR3: the TC will be reset if MR3 matches it.	0
		0	Feature disabled.	
11	MR3S	1	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		0	Feature disabled.	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.6.9 Capture Registers (CR0 - CR3)

Each Capture register is associated with a device pin and may be loaded with the Timer Counter value when a specified event occurs on that pin. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

15.6.10 Capture Control Register (CCR, TIMER0: T0CCR - 0xE000 4028 and TIMER1: T1CCR - 0xE000 8028)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, n represents the Timer number, 0 or 1.

Table 218: Capture Control Register (CCR, TIMER0: T0CCR - address 0xE000 4028 and TIMER1: T1CCR - address 0xE000 8028) bit description

Bit	Symbol	Value	Description	Reset value
0	CAP0RE	1	Capture on CAPn.0 rising edge: a sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
1	CAP0FE	1	Capture on CAPn.0 falling edge: a sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
2	CAP0I	1	Interrupt on CAPn.0 event: a CR0 load due to a CAPn.0 event will generate an interrupt.	0
		0	This feature is disabled.	
3	CAP1RE	1	Capture on CAPn.1 rising edge: a sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
4	CAP1FE	1	Capture on CAPn.1 falling edge: a sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
5	CAP1I	1	Interrupt on CAPn.1 event: a CR1 load due to a CAPn.1 event will generate an interrupt.	0
		0	This feature is disabled.	
6	CAP2RE	1	Capture on CAPn.2 rising edge: A sequence of 0 then 1 on CAPn.2 will cause CR2 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
7	CAP2FE	1	Capture on CAPn.2 falling edge: a sequence of 1 then 0 on CAPn.2 will cause CR2 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
8	CAP2I	1	Interrupt on CAPn.2 event: a CR2 load due to a CAPn.2 event will generate an interrupt.	0
		0	This feature is disabled.	

Table 218: Capture Control Register (CCR, TIMER0: T0CCR - address 0xE000 4028 and TIMER1: T1CCR - address 0xE000 8028) bit description

Bit	Symbol	Value	Description	Reset value
9	CAP3RE	1	Capture on CAPn.3 rising edge: a sequence of 0 then 1 on CAPn.3 will cause CR3 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
10	CAP3FE	1	Capture on CAPn.3 falling edge: a sequence of 1 then 0 on CAPn.3 will cause CR3 to be loaded with the contents of TC	0
		0	This feature is disabled.	
11	CAP3I	1	Interrupt on CAPn.3 event: a CR3 load due to a CAPn.3 event will generate an interrupt.	0
		0	This feature is disabled.	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.6.11 External Match Register (EMR, TIMER0: T0EMR - 0xE000 403C; and TIMER1: T1EMR - 0xE000 803C)

The External Match Register provides both control and status of the external match pins MAT(0-3). Bits EM3:0 can be written only when Timer is disabled (bit 1 in Timer Control Register is 0). Only under this condition an initial output level on MAT pins can be set. Once the Timer is enabled, EM3:0 can be changed only by Timer's activities specified by the EMC bits.

Table 219: External Match Register (EMR, TIMER0: T0EMR - address 0xE000 403C and TIMER1: T1EMR - address0xE000 803C) bit description

Bit	Symbol	Description	Reset value
0	EM0	External Match 0. This bit reflects the state of output MAT0.0/MAT1.0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[5:4] control the functionality of this output.	0
1	EM1	External Match 1. This bit reflects the state of output MAT0.1/MAT1.1, whether or not this output is connected to its pin. When a match occurs between the TC and MR1, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[7:6] control the functionality of this output.	0
2	EM2	External Match 2. This bit reflects the state of output MAT0.2/MAT1.2, whether or not this output is connected to its pin. When a match occurs between the TC and MR2, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[9:8] control the functionality of this output.	0
3	EM3	External Match 3. This bit reflects the state of output MAT0.3/MAT1.3, whether or not this output is connected to its pin. When a match occurs between the TC and MR3, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[11:10] control the functionality of this output.	0
5:4	EMC0	External Match Control 0. Determines the functionality of External Match 0. Table 220 shows the encoding of these bits.	00
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 220 shows the encoding of these bits.	00

Table 219: External Match Register (EMR, TIMER0: T0EMR - address 0xE000 403C and TIMER1: T1EMR - address0xE000 803C) bit description

Bit	Symbol	Description	Reset value
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 220 shows the encoding of these bits.	00
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 220 shows the encoding of these bits.	00
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 220. External match control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

15.7 Example timer operation

Figure 59 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 60 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

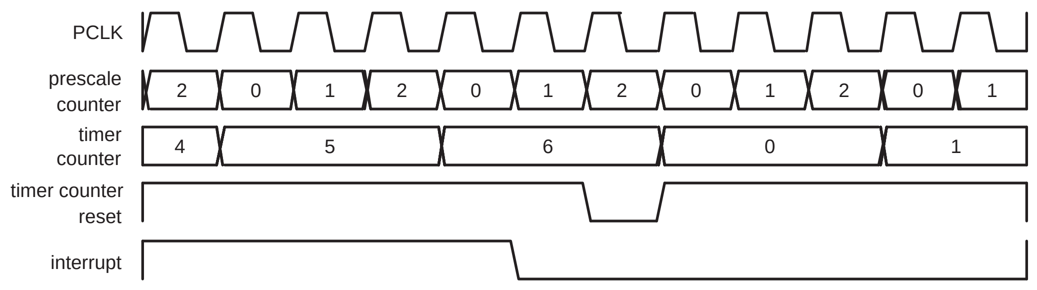


Fig 59. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

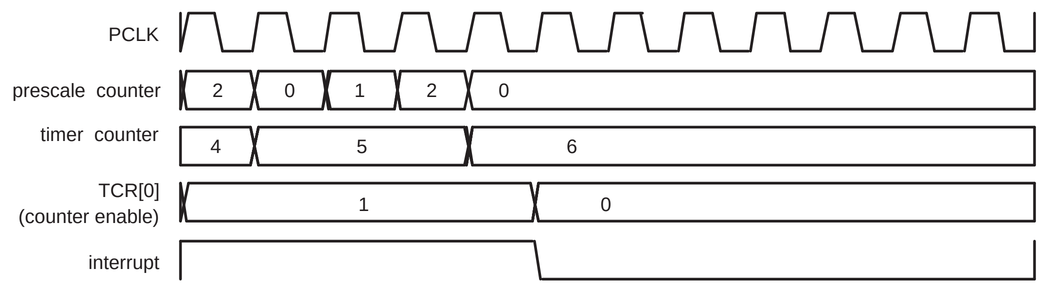


Fig 60. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

15.8 Architecture

The block diagram for TIMER/COUNTER0 and TIMER/COUNTER1 is shown in Figure 61.

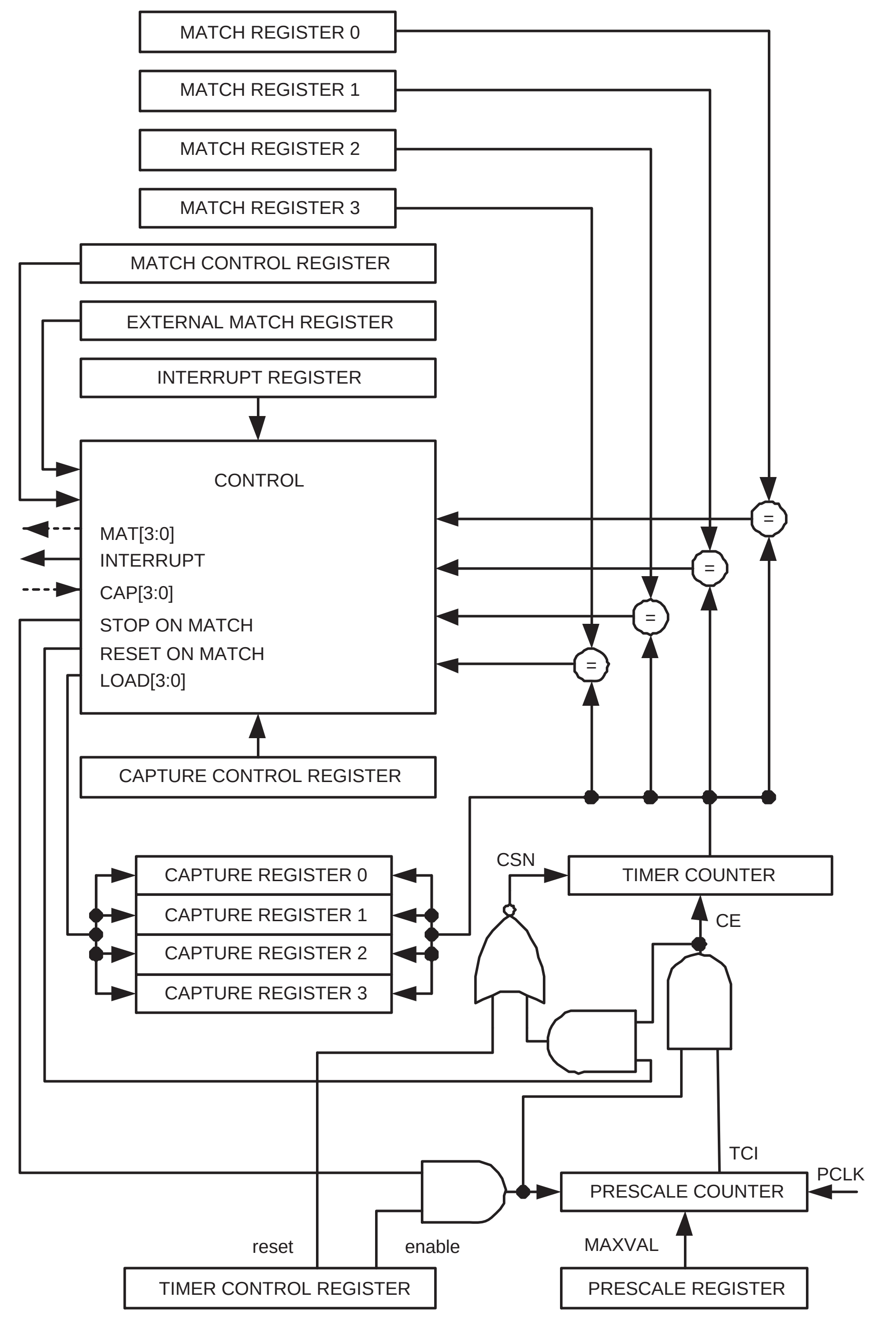


Fig 61. Timer block diagram

16.1 How to read this chapter

The PWM controller is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

16.2 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.

16.3 Description

The PWM is based on the standard Timer block and inherits all of its features, although only the PWM function is pinned out on the LPC21xx/LPC22xx. The Timer is designed to count cycles of the peripheral clock (PCLK) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. It also includes four capture inputs to save the timer value when an input signal transitions, and optionally generate an interrupt when those events occur. The PWM function is in addition to these features, and is based on match register events.

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

Two match registers can be used to provide a single edge controlled PWM output. One match register (PWMMR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position. Additional single edge controlled PWM outputs require only one match register each, since the repetition rate is the same for all PWM outputs. Multiple single edge controlled PWM outputs will all have a rising edge at the beginning of each PWM cycle, when an PWMMR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the PWMMR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions. Additional double edge controlled PWM outputs require only two match registers each, since the repetition rate is the same for all PWM outputs.

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

Figure 62 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.

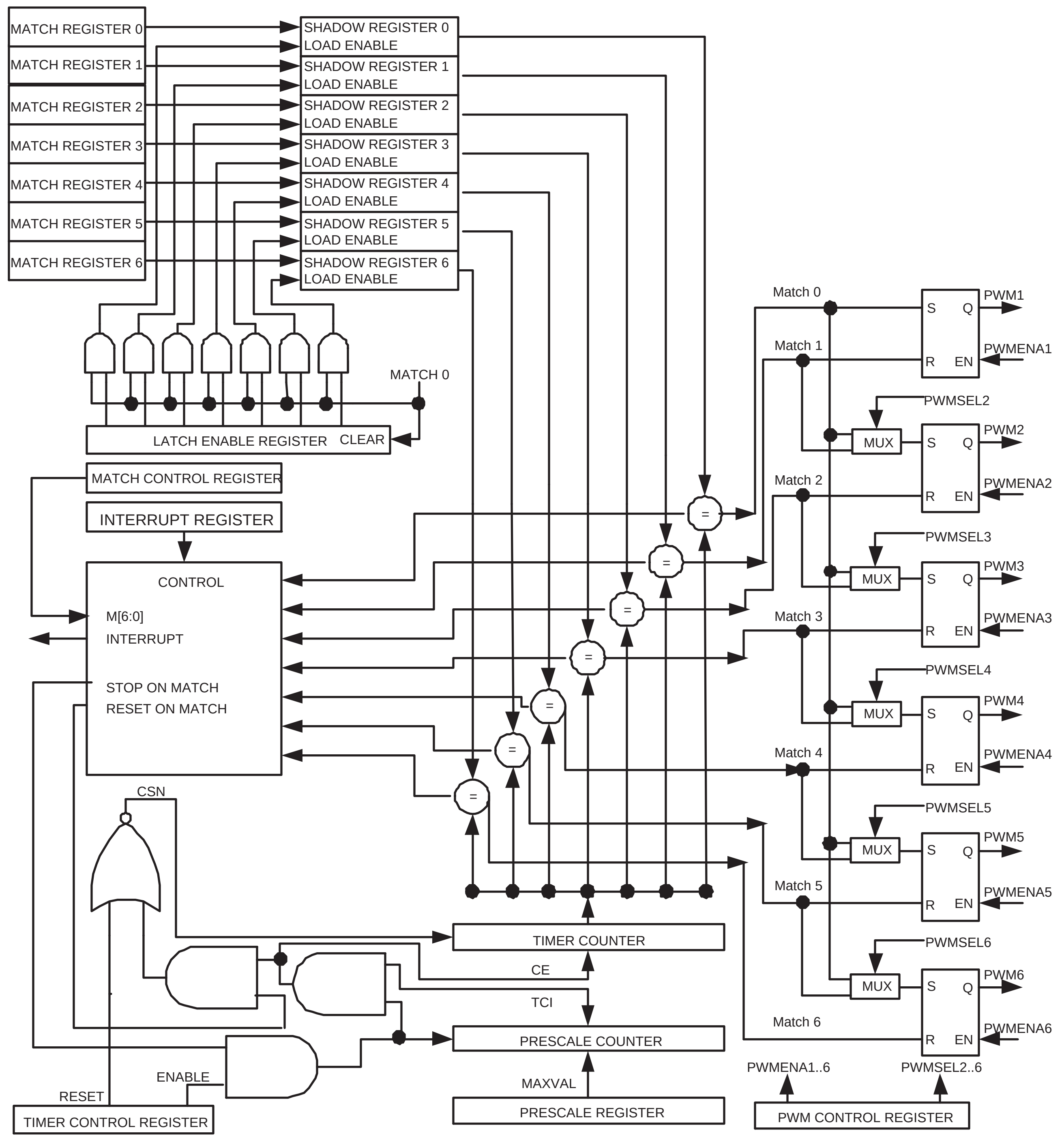
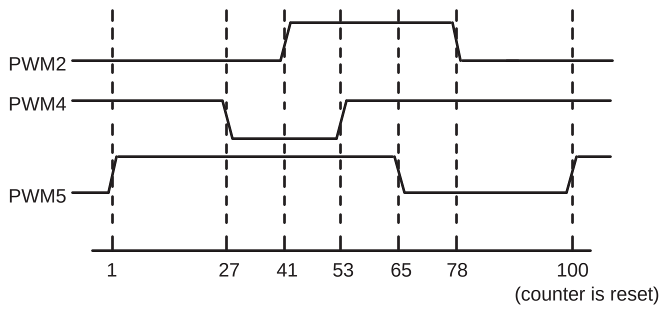


Fig 62. PWM block diagram.

A sample of how PWM values relate to waveform outputs is shown in Figure 63. PWM output logic is shown in Figure 62 that allows selection of either single or double edge controlled PWM outputs via the multiplexers controlled by the PWMSELn bits. The match register selections for various PWM outputs is shown in Table 221. This implementation supports up to N-1 single edge PWM outputs or (N-1)/2 double edge PWM outputs, where N is the number of match registers that are implemented. PWM types can be mixed if desired.



The waveforms below show a single PWM cycle and demonstrate PWM outputs under the following conditions:

The timer is configured for PWM mode (counter resets to one).

Match 0 is configured to reset the timer/counter when a match event occurs.

All PWM related Match registers are configured for toggle on match.

Control bits PWMSEL2 and PWMSEL4 are set.

The Match register values are as follows:

MR0 = 100 (PWM rate)

MR1 = 41, MR2 = 78 (PWM2 output)

MR3 = 53, MR$ = 27 (PWM4 output)

MR5 = 65 (PWM5 output)

Fig 63. Sample PWM waveforms

Table 221. Set and reset inputs for PWM Flip-Flops

PWM Channel	Single edge PWM (PWMSELn = 0)		Double edge PWM (PWMSELn = 1)	
	Set by	Reset by	Set by	Reset by
1	Match 0	Match 1	Match 0[1]	Match 1[1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2[2]	Match 3[2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4[2]	Match 5[2]
6	Match 0	Match 6	Match 5	Match 6

[1] Identical to single edge mode in this case since Match 0 is the neighboring match register. Essentially, PWM1 cannot be a double edged output.

[2] It is generally not advantageous to use PWM channels 3 and 5 for double edge PWM outputs because it would reduce the number of double edge PWM outputs that are possible. Using PWM 2, PWM4, and PWM6 for double edge PWM outputs provides the most pairings.

16.3.1 Rules for Single Edge Controlled PWM Outputs

1. All single edge controlled PWM outputs go high at the beginning of a PWM cycle unless their match value is equal to 0.

2. Each PWM output will go low when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM rate), the PWM output remains continuously high.

16.3.2 Rules for Double Edge Controlled PWM Outputs

Five rules are used to determine the next value of a PWM output when a new cycle is about to begin:

1. The match values for the **next** PWM cycle are used at the end of a PWM cycle (a time point which is coincident with the beginning of the next PWM cycle), except as noted in rule 3.
2. A match value equal to 0 or the current PWM rate (the same as the Match channel 0 value) have the same effect, except as noted in rule 3. For example, a request for a falling edge at the beginning of the PWM cycle has the same effect as a request for a falling edge at the end of a PWM cycle.
3. When match values are changing, if one of the old match values is equal to the PWM rate, it is used again once if the neither of the new match values are equal to 0 or the PWM rate, and there was no old match value equal to 0.
4. If both a set and a clear of a PWM output are requested at the same time, clear takes precedence. This can occur when the set and clear match values are the same as in, or when the set or clear value equals 0 and the other value equals the PWM rate.
5. If a match value is out of range (i.e. greater than the PWM rate value), no match event occurs and that match channel has no effect on the output. This means that the PWM output will remain always in one state, allowing always low, always high, or no-change outputs.

16.4 Pin description

Table 222 gives a brief summary of each of PWM related pins.

Table 222. Pin summary

Pin	Type	Description
PWM1	Output	Output from PWM channel 1.
PWM2	Output	Output from PWM channel 2.
PWM3	Output	Output from PWM channel 3.
PWM4	Output	Output from PWM channel 4.
PWM5	Output	Output from PWM channel 5.
PWM6	Output	Output from PWM channel 6.

16.5 Register description

The PWM function adds new registers and registers bits as shown in Table 223 below.

Table 223. Pulse Width Modulator Register Map

Name	Description	Access	Reset value[1]	Address
PWMIR	PWM Interrupt Register. The PWMIR can be written to clear interrupts. The PWMIR can be read to identify which of the possible interrupt sources are pending.	R/W	0	0xE001 4000
PWMTCR	PWM Timer Control Register. The PWMTCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the PWMTCR.	R/W	0	0xE001 4004
PWMTC	PWM Timer Counter. The 32-bit TC is incremented every PWMPR+1 cycles of PCLK. The PWMTC is controlled through the PWMTCR.	R/W	0	0xE001 4008
PWMPR	PWM Prescale Register. The PWMTC is incremented every PWMPR+1 cycles of PCLK.	R/W	0	0xE001 400C
PWMPC	PWM Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PWMPR is reached, the PWMTC is incremented. The PWMTC is observable and controllable through the bus interface.	R/W	0	0xE001 4010
PWMMCR	PWM Match Control Register. The PWMMCR is used to control if an interrupt is generated and if the PWMTC is reset when a Match occurs.	R/W	0	0xE001 4014
PWMMR0	PWM Match Register 0. PWMMR0 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR0 and the PWMTC sets all PWM outputs that are in single-edge mode, and sets PWM1 if it is in double-edge mode.	R/W	0	0xE001 4018
PWMMR1	PWM Match Register 1. PWMMR1 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR1 and the PWMTC clears PWM1 in either single-edge mode or double-edge mode, and sets PWM2 if it is in double-edge mode.	R/W	0	0xE001 401C
PWMMR2	PWM Match Register 2. PWMMR2 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR2 and the PWMTC clears PWM2 in either single-edge mode or double-edge mode, and sets PWM3 if it is in double-edge mode.	R/W	0	0xE001 4020

Table 223. Pulse Width Modulator Register Map

Name	Description	Access	Reset value[1]	Address
PWMMR3	PWM Match Register 3. PWMMR3 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR3 and the PWMTC clears PWM3 in either single-edge mode or double-edge mode, and sets PWM4 if it is in double-edge mode.	R/W	0	0xE001 4024
PWMMR4	PWM Match Register 4. PWMMR4 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR4 and the PWMTC clears PWM4 in either single-edge mode or double-edge mode, and sets PWM5 if it is in double-edge mode.	R/W	0	0xE001 4040
PWMMR5	PWM Match Register 5. PWMMR5 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR5 and the PWMTC clears PWM5 in either single-edge mode or double-edge mode, and sets PWM6 if it is in double-edge mode.	R/W	0	0xE001 4044
PWMMR6	PWM Match Register 6. PWMMR6 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between PWMMR6 and the PWMTC clears PWM6 in either single-edge mode or double-edge mode.	R/W	0	0xE001 4048
PWMPCR	PWM Control Register. Enables PWM outputs and selects PWM channel types as either single-edge or double-edge controlled.	R/W	0	0xE001 404C
PWMLER	PWM Latch Enable Register. Enables use of new PWM match values.	R/W	0	0xE001 4050

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

16.5.1 PWM Interrupt Register (PWMIR - 0xE001 4000)

The PWM Interrupt Register consists bits described in (Table 224). If an interrupt is generated then the corresponding bit in the PWMIR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 224: PWM Interrupt Register (PWMIR - address 0xE001 4000) bit description

Bit	Symbol	Description	Reset value
0	PWMMR0 Interrupt	Interrupt flag for PWM match channel 0.	0
1	PWMMR1 Interrupt	Interrupt flag for PWM match channel 1.	0
2	PWMMR2 Interrupt	Interrupt flag for PWM match channel 2.	0

Table 224: PWM Interrupt Register (PWMIR - address 0xE001 4000) bit description

Bit	Symbol	Description	Reset value
3	PWMMR3 Interrupt	Interrupt flag for PWM match channel 3.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0000
8	PWMMR4 Interrupt	Interrupt flag for PWM match channel 4.	0
9	PWMMR5 Interrupt	Interrupt flag for PWM match channel 5.	0
10	PWMMR6 Interrupt	Interrupt flag for PWM match channel 6.	0
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.5.2 PWM Timer Control Register (PWMTCR - 0xE001 4004)

The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter. The function of each of the bits is shown in Table 225.

Table 225: PWM Timer Control Register (PWMTCR - address 0xE001 4004) bit description

Bit	Symbol	Value	Description	Reset Value
0	Counter Enable	1	The PWM Timer Counter and PWM Prescale Counter are enabled for counting.	0
		0	The counters are disabled.	
1	Counter Reset	1	The PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until this bit is returned to zero.	0
		0	Clear reset.	
2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PWM Enable	1	PWM mode is enabled (counter resets to 1). PWM mode causes the shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match Register 0 - MR0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	0
		0	Timer mode is enabled (counter resets to 0).	
7:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.5.3 PWM Timer Counter (PWMTC - 0xE001 4008)

The 32-bit PWM Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the PWMTC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

16.5.4 PWM Prescale Register (PWMPR - 0xE001 400C)

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

16.5.5 PWM Prescale Counter Register (PWMPC - 0xE001 4010)

The 32-bit PWM Prescale Counter controls division of PCLK by some constant value before it is applied to the PWM Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The PWM Prescale Counter is incremented on every PCLK. When it reaches the value stored in the PWM Prescale Register, the PWM Timer Counter is incremented and the PWM Prescale Counter is reset on the next PCLK. This causes the PWM TC to increment on every PCLK when PWMPR = 0, every 2 PCLKs when PWMPR = 1, etc.

16.5.6 PWM Match Registers (PWMMR0 - PWMMR6)

The 32-bit PWM Match register values are continuously compared to the PWM Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the PWM Timer Counter, or stop the timer. Actions are controlled by the settings in the PWMMCR register.

16.5.7 PWM Match Control Register (PWMMCR - 0xE001 4014)

The PWM Match Control Register is used to control what operations are performed when one of the PWM Match Registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 226.

Table 226: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
0	PWMMR0I	1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
1	PWMMR0R	1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.	0
		0	This feature is disabled.	
2	PWMMR0S	1	Stop on PWMMR0: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR0 matches the PWMTC.	0
		0	This feature is disabled	

Table 226: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
3	PWMMR1I	1	Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
4	PWMMR1R	1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.	0
		0	This feature is disabled.	
5	PWMMR1S	1	Stop on PWMMR1: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR1 matches the PWMTC.	0
		0	This feature is disabled.	
6	PWMMR2I	1	Interrupt on PWMMR2: an interrupt is generated when PWMMR2 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
7	PWMMR2R	1	Reset on PWMMR2: the PWMTC will be reset if PWMMR2 matches it.	0
		0	This feature is disabled.	
8	PWMMR2S	1	Stop on PWMMR2: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR2 matches the PWMTC.	0
		0	This feature is disabled	
9	PWMMR3I	1	Interrupt on PWMMR3: an interrupt is generated when PWMMR3 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
10	PWMMR3R	1	Reset on PWMMR3: the PWMTC will be reset if PWMMR3 matches it.	0
		0	This feature is disabled	
11	PWMMR3S	1	Stop on PWMMR3: The PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR3 matches the PWMTC.	0
		0	This feature is disabled	
12	PWMMR4I	1	Interrupt on PWMMR4: An interrupt is generated when PWMMR4 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
13	PWMMR4R	1	Reset on PWMMR4: the PWMTC will be reset if PWMMR4 matches it.	0
		0	This feature is disabled.	
14	PWMMR4S	1	Stop on PWMMR4: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR4 matches the PWMTC.	0
		0	This feature is disabled	
15	PWMMR5I	1	Interrupt on PWMMR5: An interrupt is generated when PWMMR5 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	

Table 226: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
16	PWMMR5R	1	Reset on PWMMR5: the PWMTC will be reset if PWMMR5 matches it.	0
		0	This feature is disabled.	
17	PWMMR5S	1	Stop on PWMMR5: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR5 matches the PWMTC.	0
		0	This feature is disabled	
18	PWMMR6I	1	Interrupt on PWMMR6: an interrupt is generated when PWMMR6 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
19	PWMMR6R	1	Reset on PWMMR6: the PWMTC will be reset if PWMMR6 matches it.	0
		0	This feature is disabled.	
20	PWMMR6S	1	Stop on PWMMR6: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR6 matches the PWMTC.	0
		0	This feature is disabled	
31:21	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.5.8 PWM Control Register (PWMPCR - 0xE001 404C)

The PWM Control Register is used to enable and select the type of each PWM channel. The function of each of the bits are shown in Table 227.

Table 227: PWM Control Register (PWMPCR - address 0xE001 404C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	PWMSEL2	1	Selects double edge controlled mode for the PWM2 output.	0
		0	Selects single edge controlled mode for PWM2.	
3	PWMSEL3	1	Selects double edge controlled mode for the PWM3 output.	0
		0	Selects single edge controlled mode for PWM3.	
4	PWMSEL4	1	Selects double edge controlled mode for the PWM4 output.	0
		0	Selects single edge controlled mode for PWM4.	
5	PWMSEL5	1	Selects double edge controlled mode for the PWM5 output.	0
		0	Selects single edge controlled mode for PWM5.	
6	PWMSEL6	1	Selects double edge controlled mode for the PWM6 output.	0
		0	Selects single edge controlled mode for PWM6.	
8:7	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9	PWMENA1	1	The PWM1 output enabled.	0
		0	The PWM1 output disabled.	

Table 227: PWM Control Register (PWMPCR - address 0xE001 404C) bit description

Bit	Symbol	Value	Description	Reset Value
10	PWMENA2	1	The PWM2 output enabled.	0
		0	The PWM2 output disabled.	
11	PWMENA3	1	The PWM3 output enabled.	0
		0	The PWM3 output disabled.	
12	PWMENA4	1	The PWM4 output enabled.	0
		0	The PWM4 output disabled.	
13	PWMENA5	1	The PWM5 output enabled.	0
		0	The PWM5 output disabled.	
14	PWMENA6	1	The PWM6 output enabled.	0
		0	The PWM6 output disabled.	
15	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

16.5.9 PWM Latch Enable Register (PWMLER - 0xE001 4050)

The PWM Latch Enable Register is used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to the location of a PWM Match register while the Timer is in PWM mode, the value is held in a shadow register. When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the actual Match registers if the corresponding bit in the Latch Enable Register has been set. At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared. Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.

For example, if PWM2 is configured for double edge operation and is currently running, a typical sequence of events for changing the timing would be:

- Write a new value to the PWM Match1 register.
- Write a new value to the PWM Match2 register.
- Write to the PWMLER, setting bits 1 and 2 at the same time.
- The altered values will become effective at the next reset of the timer (when a PWM Match 0 event occurs).

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

The function of each of the bits in the PWMLER is shown in Table 228.

Table 228: PWM Latch Enable Register (PWMLER - address 0xE001 4050) bit description

Bit	Symbol	Description	Reset value
0	Enable PWM Match 0 Latch	Writing a one to this bit allows the last value written to the PWM Match 0 register to be become effective when the timer is next reset by a PWM Match event. Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
1	Enable PWM Match 1 Latch	Writing a one to this bit allows the last value written to the PWM Match 1 register to be become effective when the timer is next reset by a PWM Match event. Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
2	Enable PWM Match 2 Latch	Writing a one to this bit allows the last value written to the PWM Match 2 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
3	Enable PWM Match 3 Latch	Writing a one to this bit allows the last value written to the PWM Match 3 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
4	Enable PWM Match 4 Latch	Writing a one to this bit allows the last value written to the PWM Match 4 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
5	Enable PWM Match 5 Latch	Writing a one to this bit allows the last value written to the PWM Match 5 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
6	Enable PWM Match 6 Latch	Writing a one to this bit allows the last value written to the PWM Match 6 register to be become effective when the timer is next reset by a PWM Match event. See Section 16.5.7 “PWM Match Control Register (PWMMCR - 0xE001 4014)”.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UM10114

Chapter 17: LPC21xx/22xx WatchDog Timer (WDT)

Rev. 4 — 2 May 2012 User manual

17.1 How to read this chapter

The WDT is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

17.2 Features

- Internally resets chip if not periodically reloaded.
- Debug mode.
- Enabled by software but requires a hardware reset or a watchdog reset/interrupt to be disabled.
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled.
- Flag to indicate Watchdog reset.
- Programmable 32-bit timer with internal pre-scaler.
- Selectable time period from (T_{PCLK} x 256 x 4) to (T_{PCLK} x 2^{32} x 4) in multiples of T_{PCLK} x 4.

17.3 Applications

The purpose of the watchdog is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state. When enabled, the watchdog will generate a system reset if the user program fails to feed (or reload) the watchdog within a predetermined amount of time.

For interaction of the on-chip watchdog and other peripherals, especially the reset and boot-up procedures, please read Section 6.11 of this document.

17.4 Description

The watchdog consists of a divide by 4 fixed pre-scaler and a 32-bit counter. The clock is fed to the timer via a pre-scaler. The timer decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum watchdog interval is (T_{PCLK} x 256 x 4) and the maximum watchdog interval is (T_{PCLK} x 2^{32} x 4) in multiples of (T_{PCLK} x 4). The watchdog should be used in the following manner:

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

When the Watchdog counter underflows, the program counter will start from 0x0000 0000 as in the case of external reset. The Watchdog Time-Out Flag (WDTOF) can be examined to determine if the watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

17.5 Register description

The watchdog contains 4 registers as shown in Table 229 below.

Table 229. Watchdog register map

Name	Description	Access	Reset value[1]	Address
WDMOD	Watchdog Mode register. This register contains the basic mode and status of the Watchdog Timer.	R/W	0	0xE000 0000
WDTC	Watchdog Timer Constant register. This register determines the time-out value.	R/W	0xFF	0xE000 0004
WDFEED	Watchdog Feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer to its preset value.	WO	NA	0xE000 0008
WDTV	Watchdog Timer Value register. This register reads out the current value of the Watchdog timer.	RO	0xFF	0xE000 000C

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

17.5.1 Watchdog Mode register (WDMOD - 0xE000 0000)

The WDMOD register controls the operation of the watchdog as per the combination of WDEN and RESET bits.

Table 230. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the watchdog running.
1	0	Watchdog Interrupt Mode: debug with the Watchdog interrupt but no WDRESET enabled. When this mode is selected, a watchdog counter underflow will set the WDINT flag and the watchdog interrupt request will be generated.
1	1	Watchdog Reset Mode: operate with the watchdog interrupt and WDRESET enabled. When this mode is selected, a watchdog counter underflow will reset the microcontroller. While the watchdog interrupt is also enabled in this case (WDEN = 1) it will not be recognized since the watchdog reset will clear the WDINT flag.

Once the **WDEN** and/or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a watchdog timer underflow.

WDTOF The Watchdog Time-Out Flag is set when the watchdog times out. This flag is cleared by software.

WDINT The Watchdog Interrupt Flag is set when the watchdog times out. This flag is cleared when any reset occurs. Once the watchdog interrupt is serviced, it can be disabled in the VIC or the watchdog interrupt request will be generated indefinitely.

Table 231: Watchdog Mode register (WDMOD - address 0xE000 0000) bit description

Bit	Symbol	Description	Reset value
0	WDEN	WDEN Watchdog interrupt Enable bit (Set Only).	0
1	WDRESET	WDRESET Watchdog Reset Enable bit (Set Only).	0
2	WDTOF	WDTOF Watchdog Time-Out Flag.	0 (Only after external reset)
3	WDINT	WDINT Watchdog interrupt Flag (Read Only).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

17.5.2 Watchdog Timer Constant register (WDTC - 0xE000 0004)

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the watchdog timer. It's a 32-bit register with 8 LSB set to 1 on reset. Writing values below 0xFF will cause 0xFF to be loaded to the WDTC. Thus the minimum time-out interval is $T_{PCLK} \times 256 \times 4$.

Table 232: Watchdog Timer Constant register (WDTC - address 0xE000 0004) bit description

Bit	Symbol	Description	Reset value
31:0	Count	Watchdog time-out interval.	0x0000 00FF

17.5.3 Watchdog Feed register (WDFEED - 0xE000 0008)

Writing 0xAA followed by 0x55 to this register will reload the watchdog timer to the WDTC value. This operation will also start the watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the watchdog. A valid feed sequence must first be completed before the Watchdog is capable of generating an interrupt/reset. Until then, the watchdog will ignore feed errors. Once 0xAA is written to the WDFEED register the next operation in the Watchdog register space must be a **WRITE** (0x55) to the WDFFED register otherwise the watchdog is triggered. The interrupt/reset will be generated during the second PCLK following an incorrect access to a watchdog timer register during a feed sequence.

Remark: Interrupts must be disabled during the feed sequence. An abort condition will occur if an interrupt happens during the feed sequence.

Table 233: Watchdog Feed register (WDFEED - address 0xE000 0008) bit description

Bit	Symbol	Description	Reset value
7:0	Feed	Feed value should be 0xAA followed by 0x55.	NA

17.5.4 Watchdog Timer Value register (WDTV - 0xE000 000C)

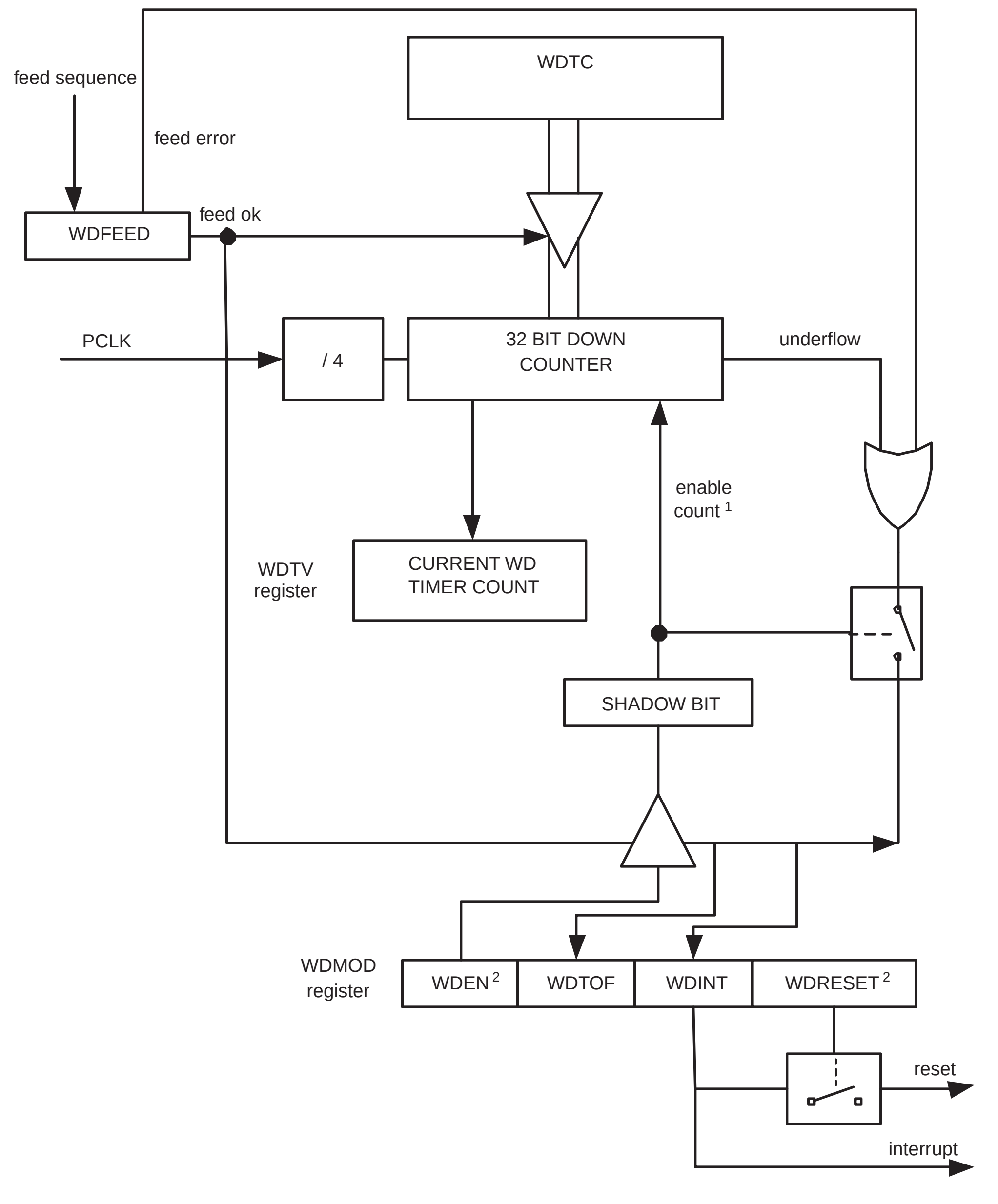
The WDTV register is used to read the current value of watchdog timer.

Table 234: Watchdog Timer Value register (WDTV - address 0xE000 000C) bit description

Bit	Symbol	Description	Reset value
31:0	Count	Counter timer value.	0x0000 00FF

17.6 Block diagram

The block diagram of the Watchdog is shown below in the Figure 64.



(1) Counter is enabled only when the WDEN bit is set and a valid feed sequence is done.

(2) WDEN and WDRESET are sticky bits. Once set they can’t be cleared until the watchdog underflows or an external reset occurs.

Fig 64. Watchdog block diagram

UM10114

Chapter 18: LPC21xx/22xx Real-Time Clock (RTC)

Rev. 4 — 2 May 2012 **User manual**

18.1 How to read this chapter

The RTC is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

18.2 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.

18.3 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 in power down mode, making it suitable for battery powered systems where the CPU is not running continuously (sleep mode).

18.4 Architecture

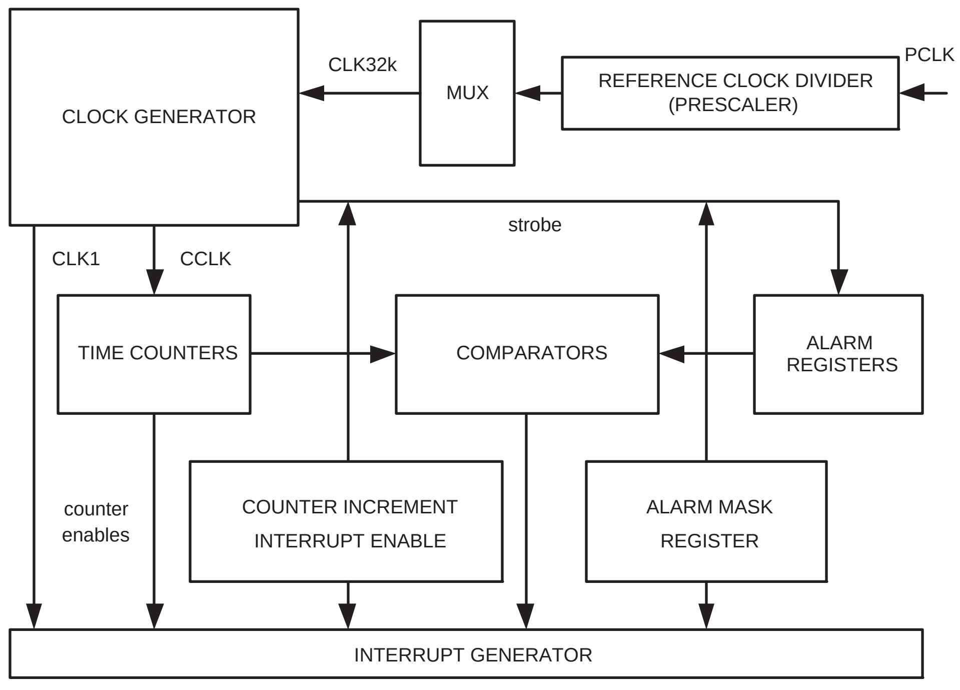


Fig 65. RTC block diagram

18.5 Register description

The RTC includes a number of registers. The address space is split into four sections by functionality. The first eight addresses are the Miscellaneous Register Group (Section 18.5.2). The second set of eight locations are the Time Counter Group (Section 18.5.12). The third set of eight locations contain the Alarm Register Group (Section 18.5.14). The remaining registers control the Reference Clock Divider.

The Real Time Clock includes the register shown in Table 235. Detailed descriptions of the registers follow.

Table 235. Real Time Clock (RTC) register map

Name	Size	Description	Access	Reset value[1]	Address
ILR	2	Interrupt Location Register	R/W	*	0xE002 4000
CTC	15	Clock Tick Counter	RO	*	0xE002 4004
CCR	4	Clock Control Register	R/W	*	0xE002 4008
CIIR	8	Counter Increment Interrupt Register	R/W	*	0xE002 400C
AMR	8	Alarm Mask Register	R/W	*	0xE002 4010
CTIME0	32	Consolidated Time Register 0	RO	*	0xE002 4014
CTIME1	32	Consolidated Time Register 1	RO	*	0xE002 4018
CTIME2	32	Consolidated Time Register 2	RO	*	0xE002 401C

Table 235. Real Time Clock (RTC) register map

Name	Size	Description	Access	Reset value[1]	Address
SEC	6	Seconds Counter	R/W	*	0xE002 4020
MIN	6	Minutes Register	R/W	*	0xE002 4024
HOUR	5	Hours Register	R/W	*	0xE002 4028
DOM	5	Day of Month Register	R/W	*	0xE002 402C
DOW	3	Day of Week Register	R/W	*	0xE002 4030
DOY	9	Day of Year Register	R/W	*	0xE002 4034
MONTH	4	Months Register	R/W	*	0xE002 4038
YEAR	12	Years Register	R/W	*	0xE002 403C
ALSEC	6	Alarm value for Seconds	R/W	*	0xE002 4060
ALMIN	6	Alarm value for Minutes	R/W	*	0xE002 4064
ALHOUR	5	Alarm value for Seconds	R/W	*	0xE002 4068
ALDOM	5	Alarm value for Day of Month	R/W	*	0xE002 406C
ALDOW	3	Alarm value for Day of Week	R/W	*	0xE002 4070
ALDOY	9	Alarm value for Day of Year	R/W	*	0xE002 4074
ALMON	4	Alarm value for Months	R/W	*	0xE002 4078
ALYEAR	12	Alarm value for Year	R/W	*	0xE002 407C
PREINT	13	Prescaler value, integer portion	R/W	0	0xE002 4080
PREFRAC	15	Prescaler value, fractional portion	R/W	0	0xE002 4084

[1] Registers in the RTC other than those that are part of the Prescaler are not affected by chip Reset. These registers must be initialized by software if the RTC is enabled. Reset value reflects the data stored in used bits only. It does not include reserved bits content.

18.5.1 RTC interrupts

Interrupt generation is controlled through the Interrupt Location Register (ILR), Counter Increment Interrupt Register (CIIR), the alarm registers, and the Alarm Mask Register (AMR). Interrupts are generated only by the transition into the interrupt state. The ILR separately enables CIIR and AMR interrupts. Each bit in CIIR corresponds to one of the time counters. If CIIR is enabled for a particular counter, then every time the counter is incremented an interrupt is generated. The alarm registers allow the user to specify a date and time for an interrupt to be generated. The AMR provides a mechanism to mask alarm compares. If all non-masked alarm registers match the value in their corresponding time counter, then an interrupt is generated.

18.5.2 Miscellaneous register group

Table 236 summarizes the registers located from 0 to 7 of A[6:2]. More detailed descriptions follow.

Table 236. Miscellaneous registers

Name	Size	Description	Access	Address
ILR	2	Interrupt Location. Reading this location indicates the source of an interrupt. Writing a one to the appropriate bit at this location clears the associated interrupt.	R/W	0xE002 4000
CTC	15	Clock Tick Counter. Value from the clock divider.	RO	0xE002 4004
CCR	4	Clock Control Register. Controls the function of the clock divider.	R/W	0xE002 4008
CIIR	8	Counter Increment Interrupt. Selects which counters will generate an interrupt when they are incremented.	R/W	0xE002 400C
AMR	8	Alarm Mask Register. Controls which of the alarm registers are masked.	R/W	0xE002 4010
CTIME0	32	Consolidated Time Register 0	RO	0xE002 4014
CTIME1	32	Consolidated Time Register 1	RO	0xE002 4018
CTIME2	32	Consolidated Time Register 2	RO	0xE002 401C

18.5.3 Interrupt Location Register (ILR - 0xE002 4000)

The Interrupt Location Register is a 2-bit register that specifies which blocks are generating an interrupt (see Table 237). 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 237: Interrupt Location Register (ILR - address 0xE002 4000) bit description

Bit	Symbol	Description	Reset value
0	RTCCIF	When one, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.	NA
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.	NA
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.5.4 Clock Tick Counter Register (CTCR - 0xE002 4004)

The Clock Tick Counter is read only. It can be reset to zero through the Clock Control Register (CCR). The CTC consists of the bits of the clock divider counter.

Table 238: Clock Tick Counter Register (CTCR - address 0xE002 4004) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
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 Section 18.7 “Reference clock divider (prescaler)” on page 270 for details.	NA

18.5.5 Clock Control Register (CCR - 0xE002 4008)

The clock register is a 5-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 239.

Table 239: Clock Control Register (CCR - address 0xE002 4008) bit description

Bit	Symbol	Description	Reset value
0	CLKEN	Clock Enable. When this bit is a one the time counters are enabled. When it is a zero, they are disabled so that they may be initialized.	NA
1	CTCRST	CTC Reset. When one, the elements in the Clock Tick Counter are reset. The elements remain reset until CCR[1] is changed to zero.	NA
3:2	CTTEST	Test Enable. These bits should always be zero during normal operation.	NA
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.5.6 Counter Increment Interrupt Register (CIIR - 0xE002 400C)

The Counter Increment Interrupt Register (CIIR) gives the ability to generate an interrupt every time a counter is incremented. This interrupt remains valid until cleared by writing a one to bit zero of the Interrupt Location Register (ILR[0]).

Table 240: Counter Increment Interrupt Register (CIIR - address 0xE002 400C) bit description

Bit	Symbol	Description	Reset value
0	IMSEC	When 1, an increment of the Second value generates an interrupt.	NA
1	IMMIN	When 1, an increment of the Minute value generates an interrupt.	NA
2	IMHOUR	When 1, an increment of the Hour value generates an interrupt.	NA
3	IMDOM	When 1, an increment of the Day of Month value generates an interrupt.	NA
4	IMDOW	When 1, an increment of the Day of Week value generates an interrupt.	NA
5	IMDOY	When 1, an increment of the Day of Year value generates an interrupt.	NA
6	IMMON	When 1, an increment of the Month value generates an interrupt.	NA
7	IMYEAR	When 1, an increment of the Year value generates an interrupt.	NA

18.5.7 Alarm Mask Register (AMR - 0xE002 4010)

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 241 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 241: Alarm Mask Register (AMR - address 0xE002 4010) bit description

Bit	Symbol	Description	Reset value
0	AMRSEC	When 1, the Second value is not compared for the alarm.	NA
1	AMRMIN	When 1, the Minutes value is not compared for the alarm.	NA
2	AMRHOUR	When 1, the Hour value is not compared for the alarm.	NA
3	AMRDOM	When 1, the Day of Month value is not compared for the alarm.	NA
4	AMRDOW	When 1, the Day of Week value is not compared for the alarm.	NA
5	AMRDOY	When 1, the Day of Year value is not compared for the alarm.	NA
6	AMRMON	When 1, the Month value is not compared for the alarm.	NA
7	AMRYEAR	When 1, the Year value is not compared for the alarm.	NA

18.5.8 Consolidated time registers

The values of the Time Counters can optionally be read in a consolidated format which allows the programmer to read all time counters with only three read operations. The various registers are packed into 32-bit values as shown in Table 242, Table 243, and Table 244. 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.

18.5.9 Consolidated Time register 0 (CTIME0 - 0xE002 4014)

The Consolidated Time Register 0 contains the low order time values: Seconds, Minutes, Hours, and Day of Week.

Table 242: Consolidated Time register 0 (CTIME0 - address 0xE002 4014) bit description

Bit	Symbol	Description	Reset value
5:0	Seconds	Seconds value in the range of 0 to 59	NA
7:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
13:8	Minutes	Minutes value in the range of 0 to 59	NA
15:14	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
20:16	Hours	Hours value in the range of 0 to 23	NA
23:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	Day Of Week	Day of week value in the range of 0 to 6	NA
31:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.5.10 Consolidated Time register 1 (CTIME1 - 0xE002 4018)

The Consolidate Time register 1 contains the Day of Month, Month, and Year values.

Table 243: Consolidated Time register 1 (CTIME1 - address 0xE002 4018) bit description

Bit	Symbol	Description	Reset value
4:0	Day of Month	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).	NA
7:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:8	Month	Month value in the range of 1 to 12.	NA
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
27:16	Year	Year value in the range of 0 to 4095.	NA
31:28	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.5.11 Consolidated Time register 2 (CTIME2 - 0xE002 401C)

The Consolidate Time register 2 contains just the Day of Year value.

Table 244: Consolidated Time register 2 (CTIME2 - address 0xE002 401C) bit description

Bit	Symbol	Description	Reset value
11:0	Day of Year	Day of year value in the range of 1 to 365 (366 for leap years).	NA
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.5.12 Time counter group

The time value consists of the eight counters shown in Table 245 and Table 246. These counters can be read or written at the locations shown in Table 246.

Table 245. Time counter relationships and values

Counter	Size	Enabled by	Minimum value	Maximum value
Second	6	Clk1 (see Figure 65)	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 246. Time counter registers

Name	Size	Description	Access	Address
SEC	6	Seconds value in the range of 0 to 59	R/W	0xE002 4020
MIN	6	Minutes value in the range of 0 to 59	R/W	0xE002 4024
HOUR	5	Hours value in the range of 0 to 23	R/W	0xE002 4028

Table 246. Time counter registers

Name	Size	Description	Access	Address
DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).[1]	R/W	0xE002 402C
DOW	3	Day of week value in the range of 0 to 6[1]	R/W	0xE002 4030
DOY	9	Day of year value in the range of 1 to 365 (366 for leap years)[1]	R/W	0xE002 4034
MONTH	4	Month value in the range of 1 to 12	R/W	0xE002 4038
YEAR	12	Year value in the range of 0 to 4095	R/W	0xE002 403C

[1] These values are simply incremented at the appropriate intervals and reset at the defined overflow point. They are not calculated and must be correctly initialized in order to be meaningful.

18.5.13 Leap year calculation

The RTC does a simple bit comparison to see if the two lowest order bits of the year counter are zero. If true, then the RTC considers that year a leap year. The RTC considers all years evenly divisible by 4 as leap years. This algorithm is accurate from the year 1901 through the year 2099, but fails for the year 2100, which is not a leap year. The only effect of leap year on the RTC is to alter the length of the month of February for the month, day of month, and year counters.

18.5.14 Alarm register group

The alarm registers are shown in Table 247. The values in these registers are compared with the time counters. If all the unmasked (See Section 18.5.7 "Alarm Mask Register (AMR - 0xE002 4010)" on page 266) 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 247. Alarm registers

Name	Size	Description	Access	Address
ALSEC	6	Alarm value for Seconds	R/W	0xE002 4060
ALMIN	6	Alarm value for Minutes	R/W	0xE002 4064
ALHOUR	5	Alarm value for Hours	R/W	0xE002 4068
ALDOM	5	Alarm value for Day of Month	R/W	0xE002 406C
ALDOW	3	Alarm value for Day of Week	R/W	0xE002 4070
ALDOY	9	Alarm value for Day of Year	R/W	0xE002 4074
ALMON	4	Alarm value for Months	R/W	0xE002 4078
ALYEAR	12	Alarm value for Years	R/W	0xE002 407C

18.6 RTC usage notes

Since the RTC operates from the APB 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 LPC21xx/LPC22xx 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 time base during system operation (by reconfiguring the PLL, the APB timer, or the RTC prescaler) will result in some form of accumulated time error.

18.7 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 LPC21xx/LPC22xx, a 13-bit integer counter is required. This can be calculated as 160 MHz divided by 32,768 minus 1 = 4881 with a remainder of 26,624. Thirteen bits are needed to hold the value 4881, but actually supports frequencies up to 268.4 MHz (32,768 × 8192).
2. The remainder value could be as large as 32,767, which requires 15 bits.

Table 248. Reference clock divider registers

Name	Size	Description	Access	Address
PREINT	13	Prescale Value, integer portion	R/W	0xE002 4080
PREFRAC	15	Prescale Value, fractional portion	R/W	0xE002 4084

18.7.1 Prescaler Integer register (PREINT - 0xE002 4080)

This is the integer portion of the prescale value, calculated as:

PREINT = int (PCLK / 32768) – 1. The value of PREINT must be greater than or equal to 1.

Table 249: Prescaler Integer register (PREINT - address 0xE002 4080) bit description

Bit	Symbol	Description	Reset value
12:0	Prescaler Integer	Contains the integer portion of the RTC prescaler value.	0
15:13	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.7.2 Prescaler Fraction register (PREFRAC - 0xE002 4084)

This is the fractional portion of the prescale value, and may be calculated as:

PREFRAC = PCLK – ((PREINT + 1) × 32768).

Table 250: Prescaler Integer register (PREFRAC - address 0xE002 4084) bit description

Bit	Symbol	Description	Reset value
14:0	Prescaler Fraction	Contains the integer portion of the RTC prescaler value.	0
15	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

18.7.3 Example of prescaler usage

In a simplistic case, the PCLK frequency is 65.537 kHz. So:

PREINT = int (PCLK / 32768) – 1 = 1 and
PREFRAC = PCLK - ([PREINT + 1] × 32768) = 1

With this prescaler setting, exactly 32,768 clocks per second will be provided to the RTC by counting 2 PCLKs 32,767 times, and 3 PCLKs once.

In a more realistic case, the PCLK frequency is 10 MHz. Then,

PREINT = int (PCLK / 32768) – 1 = 304 and
PREFRAC = PCLK – ([PREINT + 1] × 32768) = 5,760.

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

In a similar manner, any PCLK rate greater than 65.536 kHz (as long as it is an even number of cycles per second) may be turned into a 32 kHz reference clock for the RTC. The only caveat is that if PREFRAC does not contain a zero, then not all of the 32,768 per second clocks are of the same length. Some of the clocks are one PCLK longer than others. While the longer pulses are distributed as evenly as possible among the remaining pulses, this jitter could possibly be of concern in an application that wishes to observe the contents of the Clock Tick Counter (CTC) directly(Section 18.5.4 “Clock Tick Counter Register (CTCR - 0xE002 4004)” on page 265).

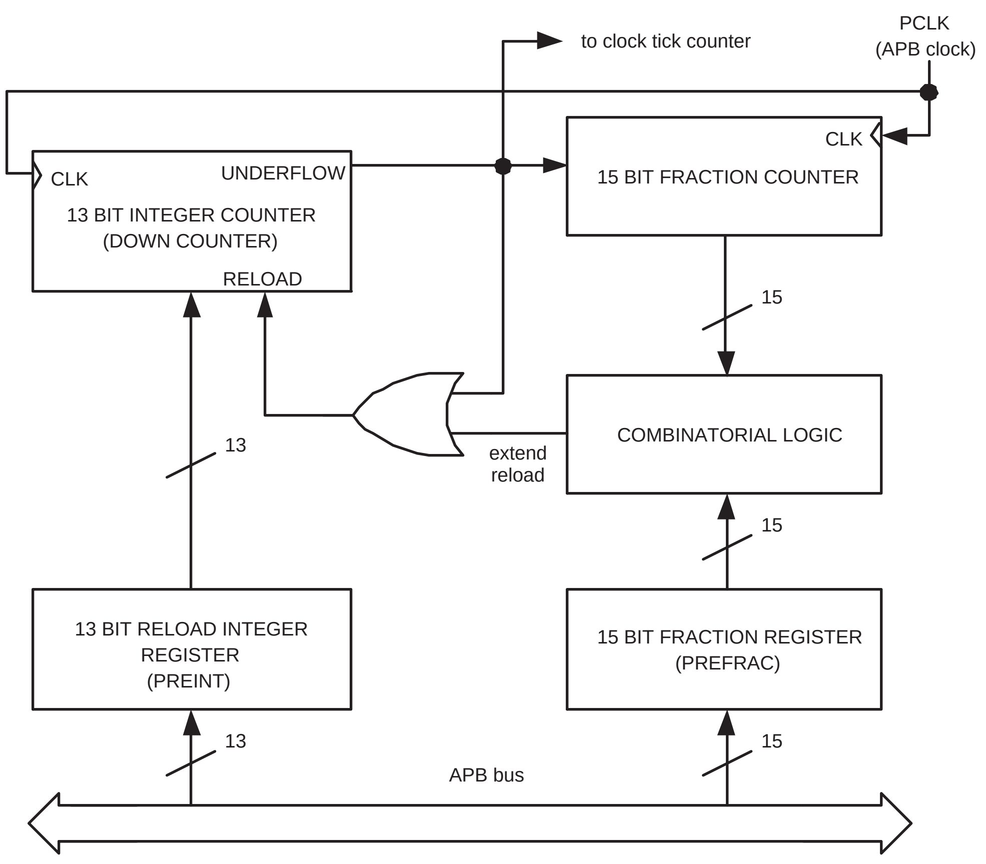


Fig 66. RTC prescaler block diagram

18.7.4 Prescaler operation

The Prescaler block labelled "Combination Logic" in Figure 66 determines when the decrement of the 13-bit PREINT counter is extended by one PCLK. In order to both insert the correct number of longer cycles, and to distribute them evenly, the combinatorial Logic associates each bit in PREFRAC with a combination in the 15-bit Fraction Counter. These associations are shown in the following Table 251.

For example, if PREFRAC bit 14 is a one (representing the fraction 1/2), then half of the cycles counted by the 13-bit counter need to be longer. When there is a 1 in the LSB of the Fraction Counter, the logic causes every alternate count (whenever the LSB of the Fraction Counter=1) to be extended by one PCLK, evenly distributing the pulse widths. Similarly, a one in PREFRAC bit 13 (representing the fraction 1/4) will cause every fourth cycle (whenever the two LSBs of the Fraction Counter=10) counted by the 13-bit counter to be longer.

Table 251. Prescaler cases where the Integer Counter reload value is incremented

Fraction Counter	PREFRAC Bit														
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`--- ---- ---- ---1`	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
`--- ---- ---- --10`	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
`--- ---- ---- -100`	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
`--- ---- ---- 1000`	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
`--- ---- ---1 0000`	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-
`--- ---- --10 0000`	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
`--- ---- -100 0000`	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-
`--- ---- 1000 0000`	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-
`--- ---1 0000 0000`	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-
`--- --10 0000 0000`	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
`--- -100 0000 0000`	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
`--- 1000 0000 0000`	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-
`--1 0000 0000 0000`	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-
`-10 0000 0000 0000`	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-
`100 0000 0000 0000`	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

UM10114

Chapter 19: LPC21xx/22xx CAN controller and acceptance filter

Rev. 4 — 2 May 2012 User manual

19.1 How to read this chapter

The following chapter only applies to parts with CAN controllers. The register descriptions are given for the full set of CAN controllers. The LPC21xx and LPC22xx have different CAN configurations depending on part number and version. Table Table 252 contains a list of all LPC21xx and LPC22xx parts with CAN interfaces, the number of CAN controllers, their register base addresses, and the pins for each part.

Remark: /01 devices contain an updated CAN controller with improved interrupt behavior in Full-CAN mode. Care should be taken when using the global CAN filter look-up table (LUT) because the numbering of CAN interfaces in the LUT is different for /01 devices (see Section 19.9):

- no suffix and /00: CAN interfaces are numbered **1** to **n** (n = 2 or 4 CAN interfaces) in the global CAN filter LUT.
- /01: CAN interfaces are numbered **0** to **n-1** in the global CAN filter LUT.
- /01: FulL-CAN mode registers available (FCANIE and FCANIC0/1).

Table 252. CAN interfaces, pins, and register base addresses

Part	CAN interfaces	Pins	CANn register base addresses used (Section 19.6)			
			CAN1 registers	CAN2 registers	CAN3 registers	CAN4 registers
no suffix and /01						
LPC2109	1	RD1; TD1	0xE004 4000	-	-	-
LPC2119	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2129	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2194	4	RD4:1; TD4:1	0xE004 4000	0xE004 8000	0xE004 C000	0xE005 0000
LPC2290	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2292	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2294	4	RD4:1; TD4:1	0xE004 4000	0xE004 8000	0xE004 C000	0xE005 0000
/01 parts						
LPC2109	1	RD1; TD1	0xE004 4000	-	-	-
LPC2119	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2129	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2194	4	RD4:1; TD4:1	0xE004 4000	0xE004 8000	0xE004 C000	0xE005 0000
LPC2290	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2292	2	RD2/1; TD2/1	0xE004 4000	0xE004 8000	-	-
LPC2294	4	RD4:1; TD4:1	0xE004 4000	0xE004 8000	0xE004 C000	0xE005 0000

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

19.2 CAN controllers

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed real-time control with a very high level of security. Its domain of application ranges from high speed networks to low cost multiplex wiring.

The CAN block is intended to support multiple CAN buses simultaneously, allowing the device to be used as a gateway, switch, or router among a number of CAN buses in industrial or automotive applications.

Each CAN Controller has a register structure similar to the NXP SJA1000 and the PeliCAN Library block, but the 8 bit registers of those devices have been combined in 32 bit words to allow simultaneous access in the ARM environment. The main operational difference is that the recognition of received Identifiers, known in CAN terminology as Acceptance Filtering, has been removed from the CAN controllers and centralized in a global Acceptance Filter. This Acceptance Filter is described after the CAN Controllers in Section 19.10 to Section 19.12.

19.3 Features

- One, two, or four CAN controllers and buses.
- Data rates to 1 Mbits/second on each bus.
- 32 bit register and RAM access.
- Compatible with *CAN specification 2.0B, ISO 11898-1*.
- Global Acceptance Filter recognizes 11 and 29 bit Rx Identifiers for all CAN buses.
- Acceptance Filter can provide FullCAN-style automatic reception for selected Standard Identifiers.

19.4 Pin description

Table 253. CAN Pin descriptions

Pin Name	Type	Description
RD4/3/2/1	Inputs	**Serial Input:** from CAN transceivers.
TD4/3/2/1	Outputs	**Serial Outputs:** to CAN transceivers.

19.5 Memory map of the CAN block

The CAN Controllers and Acceptance Filter occupy a number of APB slots, as follows:

Table 254. Memory map of the CAN block

Address Range	Used for
0xE003 8000 - 0xE003 87FF	Acceptance Filter RAM
0xE003 C000 - 0xE003 C017	Acceptance Filter Registers
0xE004 0000 - 0xE004 000B	Central CAN Registers
0xE004 4000 - 0xE004 405F	CAN Controller 1 Registers

Table 254. Memory map of the CAN block

Address Range	Used for
0xE004 8000 - 0xE004 805F	CAN Controller 2 Registers
0xE004 C000 - 0xE004 C05F	CAN Controller 3 Registers
0xE005 0000 - 0xE005 005F	CAN Controller 4 Registers

19.6 CAN controller registers

The CAN block implements the registers shown in Table 255 and Table 256. More detailed descriptions follow.

Table 255. CAN acceptance filter and central CAN registers

Name	Description	Access	Reset Value	Address
AFMR	Acceptance Filter Register	R/W	1	0xE003 C000
SFF_sa	Standard Frame Individual Start Address Register	R/W	0	0xE003 C004
SFF_GRP_sa	Standard Frame Group Start Address Register	R/W	0	0xE003 C008
EFF_sa	Extended Frame Start Address Register	R/W	0	0xE003 C00C
EFF_GRP_sa	Extended Frame Group Start Address Register	R/W	0	0xE003 C010
ENDofTable	End of AF Tables register	R/W	0	0xE003 C014
LUTerrAd	LUT Error Address register	RO	0	0xE003 C018
LUTerr	LUT Error Register	RO	0	0xE003 C01C
FCANIE	FullCAN interrupt enable register	R/W	0	0xE003 C020
FCANIC0	FullCAN interrupt and capture register 0	R/W	0	0xE003 C024
FCANIC1	FullCAN interrupt and capture register 1	R/W	0	0xE003 C028
CANTxSR	CAN Central Transmit Status Register	RO	0x003F 3F00	0xE004 0000
CANRxSR	CAN Central Receive Status Register	RO	0	0xE004 0004
CANMSR	CAN Central Miscellaneous Register	RO	0	0xE004 0008

Table 256. CAN1, CAN2, CAN3, CAN4 controller register map

Generic Register Name	Description	Access	CAN1 Address & Name	CAN2 Address & Name	CAN3 Address & Name	CAN4 Address & Name
CANMOD	Controls the operating mode of the CAN Controller.	R/W	0xE004 4000 C1MOD	0xE004 8000 C2MOD	0xE004 C000 C3MOD	0xE005 0000 C4MOD
CANCMR	Command bits that affect the state of the CAN Controller	WO	0xE004 4004 C1CMR	0xE004 8004 C2CMR	0xE004 C004 C3CMR	0xE005 0004 C4CMR
CANGSR	Global Controller Status and Error Counters	RO[1]	0xE004 4008 C1GSR	0xE004 8008 C2GSR	0xE004 C008 C3GSR	0xE005 0008 C4GSR
CANICR	Interrupt status, Arbitration Lost Capture, Error Code Capture	RO	0xE004 400C C1ICR	0xE004 800C C2ICR	0xE004 C00C C3ICR	0xE005 000C C4ICR
CANIER	Interrupt Enable	R/W	0xE004 4010 C1IER	0xE004 8010 C2IER	0xE004 C010 C3IER	0xE005 0010 C4IER

Table 256. CAN1, CAN2, CAN3, CAN4 controller register map

Generic Register Name	Description	Access	CAN1 Address & Name	CAN2 Address & Name	CAN3 Address & Name	CAN4 Address & Name
CANBTR	Bus Timing	R/W[2]	0xE004 4014 C1BTR	0xE004 8014 C2BTR	0xE004 C014 C3BTR	0xE005 0014 C4BTR
CANEWL	Error Warning Limit	R/W[2]	0xE004 4018 C1EWL	0xE004 8018 C2EWL	0xE004 C018 C3EWL	0xE005 0018 C4EWL
CANSR	Status Register	RO	0xE004 401C C1SR	0xE004 801C C2SR	0xE004 C01C C3SR	0xE005 001C C4SR
CANRFS	Receive frame status	R/W[2]	0xE004 4020 C1RFS	0xE004 8020 C2RFS	0xE004 C020 C3RFS	0xE005 0020 C4RFS
CANRID	Received Identifier	R/W[2]	0xE004 4024 C1RID	0xE004 8024 C2RID	0xE004 C024 C3RID	0xE005 0024 C4RID
CANRDA	Received data bytes 1-4	R/W[2]	0xE004 4028 C1RDA	0xE004 8028 C2RDA	0xE004 C028 C3RDA	0xE005 0028 C4RDA
CANRDB	Received data bytes 5-8	R/W[2]	0xE004 402C C1RDB	0xE004 802C C2RDB	0xE004 C02C C3RDB	0xE005 002C C4RDB
CANTFI1	Transmit frame info (1)	R/W	0xE004 4030 C1TFI1	0xE004 8030 C2TFI1	0xE004 C030 C3TFI1	0xE005 0030 C4TFI1
CANTID1	Transmit Identifier (1)	R/W	0xE004 4034 C1TID1	0xE004 8034 C2TID1	0xE004 C034 C3TID1	0xE005 0034 C4TID1
CANTDA1	Transmit data bytes 1-4 (1)	R/W	0xE004 4038 C1TDA1	0xE004 8038 C2TDA1	0xE004 C038 C3TDA1	0xE005 0038 C4TDA1
CANTDB1	Transmit data bytes 5-8 (1)	R/W	0xE004 403C C1TDB1	0xE004 803C C2TDB1	0xE004 C03C C3TDB1	0xE005 003C C4TDB1
CANTFI2	Transmit frame info (2)	R/W	0xE004 4040 C1TFI2	0xE004 8040 C2TFI2	0xE004 C040 C3TFI2	0xE005 0040 C4TFI2
CANTID2	Transmit Identifier (2)	R/W	0xE004 4044 C1TID2	0xE004 8044 C2TID2	0xE004 C044 C3TID2	0xE005 0044 C4TID2
CANTDA2	Transmit data bytes 1-4 (2)	R/W	0xE004 4048 C1TDA2	0xE004 8048 C2TDA2	0xE004 C048 C3TDA2	0xE005 0048 C4TDA2
CANTDB2	Transmit data bytes 5-8 (2)	R/W	0xE004 404C C1TDB2	0xE004 804C C2TDB2	0xE004 C04C C3TDB2	0xE005 004C C4TDB2
CANTFI3	Transmit frame info (3)	R/W	0xE004 4050 C1TFI3	0xE004 8050 C2TFI3	0xE004 C050 C3TFI3	0xE005 0050 C4TFI3
CANTID3	Transmit Identifier (3)	R/W	0xE004 4054 C1TID3	0xE004 8054 C2TID3	0xE004 C054 C3TID3	0xE005 0054 C4TID3
CANTDA3	Transmit data bytes 1-4 (3)	R/W	0xE004 4058 C1TDA3	0xE004 8058 C2TDA3	0xE004 C058 C3TDA3	0xE005 0058 C4TDA3
CANTDB3	Transmit data bytes 5-8 (3)	R/W	0xE004 405C C1TDB3	0xE004 805C C2TDB3	0xE004 C05C C3TDB3	0xE005 005C C4TDB3

[1] The error counters can only be written when RM in CANMOD is 1.

[2] These registers can only be written when RM in CANMOD is 1.

In the following register tables, the column “Reset Value” shows how a hardware reset affects each bit or field, while the column “RM Set” indicates how each bit or field is affected if software sets the RM bit, or RM is set because of a Bus-Off condition. Note that while hardware reset sets RM, in this case the setting noted in the “Reset Value” column prevails over that shown in the “RM Set” column, in the few bits where they differ. In both columns, X indicates the bit or field is unchanged.

19.6.1 Mode Register (MOD: CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000, CAN3MOD - 0x004 C000, CAN4MOD - 0x005 0000)

This register controls the basic operating mode of the CAN Controller. Bits not listed read as 0 and should be written as 0. See Table 256 for details on specific CAN channel register address.

Table 257. Mode register (MOD: CAN1MOD - address 0xE004 4000, CAN2MOD - address 0xE004 8000, CAN3MOD - address 0x004 C000, CAN4MOD - address 0x005 0000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RM	0	The CAN Controller operates, and certain registers can not be written.	1	1
		1	Reset Mode - CAN operation is disabled, and writable registers can be written.		
1	LOM	0	The CAN controller acknowledges a successfully-received message on its CAN.	0	x
		1	Listen Only Mode - the controller gives no acknowledgment on CAN, even if a message is successfully received. Messages cannot be sent, and the controller operates in “error passive” mode. This mode is intended for software bit rate detection and “hot plugging”.		
2	STM	0	A transmitted message must be acknowledged to be considered successful.	0	x
		1	Self Test Mode - the controller will consider a Tx message successful if there is no acknowledgment. Use this state in conjunction with the SRR bit in CANCMR.		
3	TPM	0	The priority of the 3 Transmit Buffers depends on their CAN IDs.	0	x
		1	The priority of the 3 Transmit Buffers depends on their Tx Priority fields.		
4	SM	0	Normal operation	0	0
		1	Sleep Mode - the CAN controller sleeps if it is not requesting an interrupt, and there is no bus activity. See the Sleep Mode description Section 19.7.2 on page 290.		
5	RPM	0	RX and TX pins are LOW for a dominant bit.	0	0
		1	Reverse Polarity Mode - RX pins are High for a dominant bit.		
6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
7	TM	0	Normal operation	0	0
		1	Test Mode. The state of the RX pin is clocked onto the TX pin.		

Note 1: The LOM and STM bits can only be written if the RM bit is 1 prior to the write operation.

19.6.2 Command Register (CMR: CAN1CMR- 0xE004 4004, CAN2CMR - 0xE004 8004, CAN3CMR - 0x004 C004, CAN4CMR - 0x005 0004)

Writing to this write-only register initiates an action. Bits not listed should be written as 0. Reading this register yields zeroes. See Table 256 for details on specific CAN channel register address.

Table 258. Command register (CMR: CAN1CMR- address 0xE004 4004, CAN2CMR - address 0xE004 8004, CAN3CMR - address 0x004 C004, CAN4CMR - address 0x005 0004) bit description

Bit	Symbol	Function	Reset Value	RM Set
0	TR	1: Transmission Request -- the message, previously written to the CANTFI, CANTID, and optionally the CANTDA and CANTDB registers, is queued for transmission.	0	0
1	AT	1: Abort Transmission -- if not already in progress, a pending Transmission Request is cancelled. If this bit and TR are set in the same write operation, frame transmission is attempted once, and no retransmission is attempted if an error is flagged nor if arbitration is lost.	0	0
2	RRB	1: Release Receive Buffer -- the information in the CANRFS, CANRID, and if applicable the CANRDA and CANRDB registers is released, and becomes eligible for replacement by the next received frame. If the next received frame is not available, writing this command clears the RBS bit in CANSR.3	0	0
3	CDO	1: Clear Data Overrun -- The Data Overrun bit in CANSR is cleared.	0	0
4	SRR	1: Self Reception Request -- the message, previously written to the CANTFS, CANTID, and optionally the CANTDA and CANTDB registers, is queued for transmission. This differs from the TR bit above in that the receiver is not disabled during the transmission, so that it receives the message if its Identifier is recognized by the Acceptance Filter.	0	0
5	STB1	1: Select Tx Buffer 1 for transmission	0	0
6	STB2	1: Select Tx Buffer 2 for transmission	0	0
7	STB3	1: Select Tx Buffer 3 for transmission	0	0

19.6.3 Global Status Register (GSR: CAN1GSR - 0xE004 0008, CAN2GSR - 0xE004 8008, CAN3GSR - 0xE004 C008, CAN4GSR 0xE005 0008)

This register is read-only, except that the Error Counters can be written when the RM bit in the CANMOD register is 1. Bits not listed read as 0 and should be written as 0. See Table 256 for details on specific CAN channel register address.

Table 259. Global Status Register (GSR: CAN1GSR - address 0xE004 0008, CAN2GSR - address 0xE004 8008, CAN3GSR - address 0xE004 C008, CAN4GSR address 0xE005 0008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS	1	Receive Buffer Status -- a received message is available in the CANRFS, CANRID, and if applicable the CANRDA and CANRDB registers. This bit is cleared by the Release Receive Buffer command in CANCMR, if no subsequent received message is available.	0	0
1	DOS	0	No data overrun has occurred since the last Clear Data Overrun command was written to CANCMR (or since Reset).	0	0
		1	Data Overrun Status -- a message was lost because the preceding message to this CAN controller was not read and released quickly enough.		
2	TBS	0	As least one previously-queued message for this CAN controller has not yet been sent, and therefore software should not write to the CANTFI, CANTID, CANTDA, nor CANTDB registers of that (those) Tx buffer(s).	1	X
		1	Transmit Buffer Status -- no transmit message is pending for this CAN controller (in any of the 3 Tx buffers), and software may write to any of the CANTFI, CANTID, CANTDA, and CANTDB registers.		
3	TCS	0	At least one requested transmission has not been successfully completed.	1	0
		1	Transmit Complete Status -- all requested transmission(s) has (have) been successfully completed.		
4	RS	1	Receive Status: the CAN controller is receiving a message.	0	0
5	TS	1	Transmit Status: The CAN controller is sending a message	0	0
6	ES	1	Error Status: one or both of the Transmit and Receive Error Counters has reached the limit set in the Error Warning Limit register.	0	0
7	BS	1	Bus Status: the CAN controller is currently prohibited from bus activity because the Transmit Error Counter reached its limiting value of 255.	0	0
15:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
23:16	RXERR	-	The current value of the Rx Error Counter.	0	X
31:24	TXERR	-	The current value of the Tx Error Counter.	0	X

19.6.4 Interrupt and Capture Register (ICR: CAN1ICR- 0xE004 400C, CAN2ICR - 0xE004 800C, CAN3ICR - 0xE004 C00C, CAN4ICR - 0xE005 000C)

Bits in this register indicate information about events on the CAN bus. This register is read-only. Bits not listed read as 0 and should be written as 0. Bits 1-9 clear when they are read. See Table 256 for details on specific CAN channel register address. Bits 16-23 are captured when a bus error occurs. At the same time, if the BEIE bit in CANIER is 1, the BEI bit in this register is set, and a CAN interrupt can occur.

Bits 24-31 are captured when CAN arbitration is lost. At the same time, if the ALIE bit in CANIER is 1, the ALI bit in this register is set, and a CAN interrupt can occur. Once either of these bytes is captured, its value will remain the same until it is read, at which time it is released to capture a new value.

The clearing of bits 1-9 and the releasing of bits 16-23 and 24-31 all occur on any read from CANICR, regardless of whether part or all of the register is read. This means that software should always read CANICR as a word, and process and deal with all bits of the register as appropriate for the application.

Table 260. Interrupt and Capture register (ICR: CR: CAN1ICR- address 0xE004 400C, CAN2ICR - 0xE004 address 800C, CAN3ICR - address 0xE004 C00C, CAN4ICR - address 0xE005 000C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RI	1	Receive Interrupt -- this bit is set whenever the RBS bit in CANSR and the RIE bit in CANIER are both 1, indicating that a received message is available.=.	0	0
1	TI1	1	Transmit Interrupt 1 -- this bit is set when the TBS1 bit in CANSR goes from 0 to 1, indicating that Transmit buffer 1 is available, and the TIE1 bit in CANIER is 1.	0	0
2	Ei	1	Error Warning Interrupt -- this bit is set on every change (set or clear) of the Error Status or Bus Status bit in CANSR, if the EIE bit in CAN is 1 at the time of the change.	0	X
3	DOI	1	Data Overrun Interrupt -- this bit is set when the DOS bit in CANSR goes from 0 to 1, if the DOIE bit in CANIE is 1.	0	0
4	WUI	1	Wake-Up Interrupt: this bit is set if the CAN controller is sleeping and bus activity is detected, if the WUIE bit in CANIE is 1.	0	0
5	EPI	1	Error Passive Interrupt -- this bit is set if the EPIE bit in CANIE is 1, and the CAN controller switches between Error Passive and Error Active mode in either direction.	0	0
6	ALI	1	Arbitration Lost Interrupt -- this bit is set if the ALIE bit in CANIE is 1, and the CAN controller loses arbitration while attempting to transmit.	0	0
7	BEI	1	Bus Error Interrupt -- this bit is set if the BEIE bit in CANIE is 1, and the CAN controller detects an error on the bus.	0	X
8	IDI	1	ID Ready Interrupt -- this bit is set if the IDIE bit in CANIE is 1, and a CAN Identifier has been received.	0	0

Table 260. Interrupt and Capture register (ICR: CR: CAN1ICR- address 0xE004 400C, CAN2ICR - 0xE004 address 800C, CAN3ICR - address 0xE004 C00C, CAN4ICR - address 0xE005 000C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
9	TI2	1	Transmit Interrupt 2 -- this bit is set when the TBS2 bit in CANSR goes from 0 to 1, indicating that Transmit buffer 2 is available, and the TIE2 bit in CANIER is 1.	0	0
10	TI3	1	Transmit Interrupt 3-- this bit is set when the TBS3 bit in CANSR goes from 0 to 1, indicating that Transmit buffer 3 is available, and the TIE3 bit in CANIER is 1.	0	0
15:11	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
20:16	ERRBIT		Error Code Capture: when the CAN controller detects a bus error, the location of the error within the frame is captured in this field. The value reflects an internal state variable, and as a result is not very linear:	0	X
		00010	ID28:21		
		00011	Start of Frame		
		00100	SRTR bit		
		00101	IDE bit		
		00110	ID20:18		
		00111	ID17:13		
		01000	CRC		
		01001	Res.Bit 0		
		01010	Data field		
		01011	DLC		
		01100	RTR bit		
		01101	Res.Bit 1		
		01110	ID4:0		
		01111	ID12:5		
		10001	Active Error flag		
		10010	Intermission		
		10011	Dominant OK bits		
		10110	Passive error flag		
		10111	Error delimiter		
		11000	CRC delimiter		
		11001	Ack slot		
		11010	End of Frame		
		11011	Ack delimiter		
		11100	Overload flag		
21	ERRDIR		When the CAN controller detects a bus error, the direction of the current bit is captured in this bit.	0	X
		0	Transmitting		
		1	Receiving		

Table 260. Interrupt and Capture register (ICR: CR: CAN1ICR- address 0xE004 400C, CAN2ICR - 0xE004 address 800C, CAN3ICR - address 0xE004 C00C, CAN4ICR - address 0xE005 000C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
23;22	ERRC		When the CAN controller detects a bus error, the type of error is captured in this field:	0	X
		00	Bit error		
		01	Form error		
		10	Stuff error		
		11	Other error		
28:24	ALCBIT	-	Each time arbitration is lost while trying to send on the CAN, the bit number within the frame is captured into this field. 0 indicates arbitration loss in the first (MS) bit of the Identifier … 31 indicates loss in the RTR bit of an extended frame. After this byte is read, the ALI bit is cleared and a new Arbitration Lost interrupt can occur.	0	X
31:29	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

19.6.5 Interrupt Enable Register (IER: CAN1IER - 0xE004 4010, CAN2IER 0xE004 8010, CAN3IER - 0xE004 C010, CAN4IER - 0xE005 0010)

This read/write register controls whether various events on the CAN controller will result in an interrupt. Bits 7:0 in this register correspond 1-to-1 with bits 7:0 in the CANICR register. See Table 256 for details on specific CAN channel register address.

Table 261. Interrupt Enable register (IER: CAN1IER - address 0xE004 4010, CAN2IER - address 0xE004 8010, CAN3IER - address 0xE004 C010, CAN4IER - address 0xE005 0010) bit description

Bit	Symbol	Function	Reset Value	RM Set
0	RIE	Receiver Interrupt Enable.	0	X
1	TIE1	Transmit Interrupt Enable (1).	0	X
2	EIE	Error Warning Interrupt Enable.	0	X
3	DOIE	Data Overrun Interrupt Enable.	0	X
4	WUIE	Wake-Up Interrupt Enable.	0	X
5	EPIE	Error Passive Interrupt Enable.	0	X
6	ALIE	Arbitration Lost Interrupt Enable.	0	X
7	BEIE	Bus Error Interrupt Enable.	0	X
8	IDIE	ID Ready Interrupt Enable.	0	X
9	TIE2	Transmit Interrupt Enable (2).	0	X
10	TIE3	Transmit Interrupt Enable (3).	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

19.6.6 Bus Timing Register (BTR: CAN1BTR - 0xE004 4014, CAN2BTR - 0xE004 8014, CAN3BTR - 0xE004 C014, CAN4BTR - 0xE005 0014)

This register controls how various CAN timings are derived from the VPB clock. It can be read at any time, but can only be written if the RM bit in CANmod is 1. See Table 256 for details on specific CAN channel register address.

Table 262. Bus Timing Register (BTR: CAN1BTR - address 0xE004 4014, CAN2BTR - address 0xE004 8014, CAN3BTR - address 0xE004 C014, CAN4BTR - address 0xE005 0014) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
9:0	BRP		Baud Rate Prescaler. The VPB clock is divided by (this value plus one) to produce the CAN clock.	0	X
13:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
15:14	SJW		The Synchronization Jump Width is (this value plus one) CAN clocks.	0	X
19:16	TESG1		The delay from the nominal Sync point to the sample point is (this value plus one) CAN clocks.	1100	X
22:20	TESG2		The delay from the sample point to the next nominal sync point is (this value plus one) CAN clocks. The nominal CAN bit time is (this value plus the value in TSEG1 plus 3) CAN clocks.	001	X
23	SAM	0	The bus is sampled once (recommended for high speed buses)	0	X
		1	The bus is sampled 3 times (recommended for low to medium speed buses)		
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

19.6.7 Error Warning Limit Register (EWL: CAN1EWL - 0xE004 4018, CAN2EWL - 0xE004 8018, CAN3EWL - 0xE004 C018, CAN4EWL - 0xE005 0018)

This register sets a limit on Tx or Rx errors at which an interrupt can occur. It can be read at any time, but can only be written if the RM bit in CANmod is 1. See Table 256 for details on specific CAN channel register address.

Table 263. Error Warning Limit register (EWL: CAN1EWL - address 0xE004 4018, CAN2EWL - address 0xE004 8018, CAN3EWL - address 0xE004 C018, CAN4EWL - address 0xE005 0018) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	EWL	During CAN operation, this value is compared to both the Tx and Rx Error Counters. If either of these counter matches this value, the Error Status (ES) bit in CANSR is set.	96_{10} = 0x60	X

19.6.8 Status Register (SR - CAN1SR 0xE004 401C, CAN2SR - 0xE004 801C, CAN3SR - 0xE004 C01C, CAN4SR - 0xE005 001C)

This register contains three status bytes, in which the bits not related to transmission are identical to the corresponding bits in the Global Status Register, while those relating to transmission reflect the status of each of the 3 Tx Buffers. See Table 256 for details on specific CAN channel register address.

Table 264. Status Register (SR - CAN1SR 0xE004 401C, CAN2SR - 0xE004 801C, CAN3SR - 0xE004 C01C, CAN4SR - 0xE005 001C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0, 8, 16	RBS		These bits are identical to the RSB bit in the GSR.	0	0
1, 9, 17	DOS		These bits are identical to the DOS bit in the GSR.	0	0
2, 10, 18	TBS1, TBS2, TBS3	0	Software should not write to any of the CANTFI, CANTID, CANTDA, and CANTDB registers for this Tx Buffer.	1	X
		1	Software may write a message into the CANTFI, CANTID, CANTDA, and CANTDB registers for this Tx Buffer.		
3, 11, 19	TCS1, TCS2, TCS3	0	The previously requested transmission for this Tx Buffer is not complete.	1	0
		1	The previously requested transmission for this Tx Buffer has been successfully completed.		
4, 12, 20	RS		These bits are identical to the RS bit in the GSR.	0	0
5, 13, 21	TS1, TS2, TS3	1	The CAN Controller is transmitting a message from this Tx Buffer.	0	0
6, 14, 22	ES		These bits are identical to the ES bit in the GSR.	0	0
7, 15, 23	BS		These bits are identical to the BS bit in the GSR.	0	0
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

19.6.9 Receive Frame Status register (RFS - CAN1RFS - 0xE004 4020, CAN2RFS - 0xE004 8020, CAN3RFS - 0xE004 C020, CAN4RFS - 0xE005 0020)

This register defines the characteristics of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 256 for details on specific CAN channel register address.

Table 265. Receive Frame Status register (RFS - CAN1RFS - address 0xE004 4020, CAN2RFS - address 0xE004 8020, CAN3RFS - address 0xE004 C020, CAN4RFS - address 0xE005 0020) bit description

Bit	Symbol	Function	Reset Value	RM Set
9:0	ID Index	If the BP bit (below) is 0, this value is the zero-based number of the Lookup Table RAM entry at which the Acceptance Filter matched the received Identifier. Disabled entries in the Standard tables are included in this numbering, but will not be matched. See Section 19.11 “Examples of acceptance filter tables and ID index values” on page 298 for examples of ID Index values.	0	X
10	BP	If this bit is 1, the current message was received in AF Bypass mode, and the ID Index field (above) is meaningless.	0	X
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
19:16	DLC	The field contains the Data Length Code (DLC) field of the current received message. When RTR = 0, this is related to the number of data bytes available in the CANRDA and CANRDB registers as follows: 0000-0111 = 0 to 7 bytes1000-1111 = 8 bytes With RTR = 1, this value indicates the number of data bytes requested to be sent back, with the same encoding.	0	X
29:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
30	RTR	This bit contains the Remote Transmission Request bit of the current received message. 0 indicates a Data Frame, in which (if DLC is non-zero) data can be read from the CANRDA and possibly the CANRDB registers. 1 indicates a Remote frame, in which case the DLC value identifies the number of data bytes requested to be sent using the same Identifier.	0	X
31	FF	A 0 in this bit indicates that the current received message included an 11 bit Identifier, while a 1 indicates a 29 bit Identifier. This affects the contents of the CANid register described below.	0	X

19.6.10 Receive Identifier register (RID - CAN1RID - 0xE004 4024, CAN2RID - 0xE004 8024, CAN3RID - 0xE004 C024, CAN4RID - 0xE005 0024)

This register contains the Identifier field of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANmod is 1. It has two different formats depending on the FF bit in CANRFS. See Table 256 for details on specific CAN channel register address.

Table 266. Receive Identifier register when FF = 0 (RID: CAN1RID - address 0xE004 4024, CAN2RID - address 0xE004 8024, CAN3RID - address 0xE004 C024, CAN4RID - address 0xE005 0024) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11 bit Identifier field of the current received message. In CAN 2.0A, these bits are called ID10-0, while in CAN 2.0B they’re called ID29-18.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 267. Receive Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29 bit Identifier field of the current received message. In CAN 2.0B these bits are called ID29-0.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

19.6.11 Receive Data register A (RDA: CAN1RDA - 0xE004 4028, CAN2RDA - 0xE004 8028, CAN3RDA - 0xE004 C028, CAN4RDA - 0xE005 0028)

This register contains the first 1-4 Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 256 for details on specific CAN channel register address.

Table 268. Receive Data register A (RDA: CAN1RDA - address 0xE004 4028, CAN2RDA - address 0xE004 8028, CAN3RDA - address 0xE004 C028, CAN4RDA - address 0xE005 0028) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If the DLC field in CANRFS >= 0001, this contains the first Data byte of the current received message.	0	X
15:8	Data 2	If the DLC field in CANRFS >= 0010, this contains the first Data byte of the current received message.	0	X
23:16	Data 3	If the DLC field in CANRFS >= 0011, this contains the first Data byte of the current received message.	0	X
31:24	Data 4	If the DLC field in CANRFS >= 0100, this contains the first Data byte of the current received message.	0	X

19.6.12 Receive Data register B (RDB: CAN1RDB - 0xE004 402C, CAN2RDB - 0xE004 802C, CAN3RDB - 0xE004 C02C, CAN4RDB - 0xE005 002C)

This register contains the 5th through 8th Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 256 for details on specific CAN channel register address.

Table 269. Receive Data register B (RDB: CAN1RDB - address 0xE004 402C, CAN2RDB - address 0xE004 802C, CAN3RDB - address 0xE004 C02C, CAN4RDB - address 0xE005 002C) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If the DLC field in CANRFS >= 0101, this contains the first Data byte of the current received message.	0	X
15:8	Data 6	If the DLC field in CANRFS >= 0110, this contains the first Data byte of the current received message.	0	X
23:16	Data 7	If the DLC field in CANRFS >= 0111, this contains the first Data byte of the current received message.	0	X
31:24	Data 8	If the DLC field in CANRFS >= 1000, this contains the first Data byte of the current received message.	0	X

19.6.13 Transmit Frame Information register (TFI1, 2, 3 - CAN1TF1n - 0xE004 4030, 40, 50; CAN2TFIn - 0xE004 8030, 40, 50; CAN3TFIn - 0xE004 C030, 40, 50; CAN4TFIn - 0xE005 0030, 40, 50)

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the format of the next transmit message for that Tx buffer. Bits not listed read as 0 and should be written as 0. See Table 256 for details on specific CAN channel register address.

Table 270. Transmit Frame Information register (TFI1, 2, 3 - CAN1TF1n - addresses 0xE004 4030, 40, 50; CAN2TFIn - addresses 0xE004 8030, 40, 50; CAN3TFIn - addresses 0xE004 C030, 40, 50; CAN4TFIn - addresses 0xE005 0030, 40, 50) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	PRIO	If the TPM bit in the CANMOD register is 1, enabled Tx Buffers contend for the right to send their messages based on this field. The lowest binary value has priority.		
15:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
19:16	DLC	This value is sent in the DLC field of the next transmit message. In addition, if RTR = 0, this value controls the number of Data bytes sent in the next transmit message, from the CANTDA and CANTDB registers: 0000-0111 = 0-7 bytes 1xxx = 8 bytes	0	X
29:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
30	RTR	This value is sent in the RTR bit of the next transmit message. If this bit is 0, the number of data bytes called out by the DLC field are sent from the CANTDA and CANTDB registers. If it's 1, a Remote Frame is sent, containing a request for that number of bytes.	0	X
31	FF	If this bit is 0, the next transmit message will be sent with an 11 bit Identifier, while if it's 1, the message will be sent with a 29 bit Identifier.	0	X

19.6.14 Transmit Identifier register (TID1, 2, 3 - CAN1TIDn - 0xE004 4034, 44, 54; CAN2TIDn - 0xE004 8034, 44, 54; CAN3TIDn - 0xE004 C034, 44, 54; CAN4TIDn - 0xE005 0034, 44, 54)

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the Identifier field of the next transmit message. Bits not listed read as 0 and should be written as 0. The register assumes two different formats depending on the FF bit in CANTFI. See Table 256 for details on specific CAN channel register address.

Table 271. Transfer Identifier register when FF=0 (TID1, 2, 3: CAN1TIDn - addresses 0xE004 4034, 44, 54; CAN2TIDn - addresses 0xE004 8034, 44, 54; CAN3TIDn - addresses 0xE004 C034, 44, 54; CAN4TIDn - addresses 0xE005 0034, 44, 54) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11 bit Identifier to be sent in the next transmit message.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 272. Transfer Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29 bit Identifier to be sent in the next transmit message.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

19.6.15 Transmit Data register A (TDA1, 2, 3: CAN1TDAn - 0xE004 4038, 48, 58; CAN2TDAn - 0xE004 8038, 48, 58; CAN3TDAn - 0xE004 C038, 48, 58; CAN4TDAn - 0xE005 0038, 48, 58)

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the first 1-4 data bytes of the next transmit message. See Table 256 for details on specific CAN channel register address.

Table 273. Transmit Data register A (TDA1, 2, 3: CAN1TDAn - addresses 0xE004 4038, 48, 58; CAN2TDAn - addresses 0xE004 8038, 48, 58; CAN3TDAn - addresses 0xE004 C038, 48, 58; CAN4TDAn - addresses 0xE005 0038, 48, 58) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If RTR = 0 and DLC >= 0001 in the corresponding CANTFI, this byte is sent as the first Data byte of the next transmit message.	0	X
15;8	Data 2	If RTR = 0 and DLC >= 0010 in the corresponding CANTFI, this byte is sent as the 2nd Data byte of the next transmit message.	0	X
23:16	Data 3	If RTR = 0 and DLC >= 0011 in the corresponding CANTFI, this byte is sent as the 3rd Data byte of the next transmit message.	0	X
31:24	Data 4	If RTR = 0 and DLC >= 0100 in the corresponding CANTFI, this byte is sent as the 4th Data byte of the next transmit message.	0	X

19.6.16 Transmit Data Register B (TDB1, 2, 3: CAN1TDBn - 0xE004 403C, 4C, 5C; CAN2TDBn - 0xE004 803C, 4C, 5C; CAN3TDBn - 0xE004 C03C, 4C, 5C; CAN4TDBn - 0xE005 003C, 4C, 5C)

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the 5th through 8th data bytes of the next transmit message. See Table 256 for details on specific CAN channel register address.

Table 274. Transmit Data register B (TDB1, 2, 3: CAN1TDBn - addresses 0xE004 403C, 4C, 5C; CAN2TDBn - addresses 0xE004 803C, 4C, 5C; CAN3TDBn - addresses 0xE004 C03C, 4C, 5C; CAN4TDBn - addresses 0xE005 003C, 4C, 5C) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If RTR = 0 and DLC >= 0101 in the corresponding CANTFI, this byte is sent as the 5th Data byte of the next transmit message.	0	X
15;8	Data 6	If RTR = 0 and DLC >= 0110 in the corresponding CANTFI, this byte is sent as the 6th Data byte of the next transmit message.	0	X
23:16	Data 7	If RTR = 0 and DLC >= 0111 in the corresponding CANTFI, this byte is sent as the 7th Data byte of the next transmit message.	0	X
31:24	Data 8	If RTR = 0 and DLC >= 1000 in the corresponding CANTFI, this byte is sent as the 8th Data byte of the next transmit message.	0	X

19.7 CAN controller operation

19.7.1 Error handling

The CAN Controllers count and handle transmit and receive errors as specified in CAN Spec 2.0B. The Transmit and Receive Error Counters are incremented for each detected error and are decremented when operation is error-free. If the Transmit Error counter contains 255 and another error occurs, the CAN Controller is forced into a state called Bus-Off. In this state, the following register bits are set: BS in CANSR, BEI and EI in CANIR if these are enabled, and RM in CANMOD. RM resets and disables much of the CAN Controller. Also at this time the Transmit Error Counter is set to 127 and the Receive Error Counter is cleared. Software must next clear the RM bit. Thereafter the Transmit Error Counter will count down 128 occurrences of the Bus Free condition (11 consecutive recessive bits). Software can monitor this countdown by reading the Tx Error Counter. When this countdown is complete, the CAN Controller clears BS and ES in CANSR, and sets EI in CANSR if EIE in IER is 1.

The Tx and Rx error counters can be written if RM in CANMOD is 1. Writing 255 to the Tx Error Counter forces the CAN Controller to Bus-Off state. If Bus-Off (BS in CANSR) is 1, writing any value 0 through 254 to the Tx Error Counter clears Bus-Off. When software clears RM in CANMOD thereafter, only one Bus Free condition (11 consecutive recessive bits) is needed before operation resumes.

19.7.2 Sleep mode

The CAN Controller will enter Sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. Software can only set SM when RM in the CAN Mode register is 0; it can also set the WUIE bit in the CAN Interrupt Enable register to enable an interrupt on any wake-up condition.

The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1) in response to a) a dominant bit on the CAN bus, or b) software clearing SM in the CAN Mode register. A sleeping CAN Controller, that wakes up in response to bus activity, is not able to receive an initial message, until after it detects Bus_Free (11 consecutive recessive bits). If an interrupt is pending or the CAN bus is active when software sets SM, the wake-up is immediate.

19.7.3 Interrupts

Each CAN Controller produces 3 interrupt requests, Receive, Transmit, and “other status”. The Transmit interrupt is the OR of the Transmit interrupts from the three Tx Buffers. Each Receive and Transmit interrupt request from each controller is assigned its own channel in the Vectored Interrupt Controller (VIC), and can have its own interrupt service routine. The “other status” interrupts from all of the CAN controllers, and the Acceptance Filter LUTerr condition, are ORed into one VIC channel.

19.7.4 Transmit priority

If the TPM bit in the CANMOD register is 0, multiple enabled Tx Buffers contend for the right to send their messages based on the value of their CAN Identifier (TID). If TPM is 1, they contend based on the PRIO fields in bits 7:0 of their CANTFS registers. In both cases the smallest binary value has priority. If two (or three) transmit-enabled buffers have the same smallest value, the lowest-numbered buffer sends first.

The CAN controller selects among multiple enabled Tx Buffers dynamically, just before it sends each message.

19.8 Centralized CAN registers

Three read-only registers group the bits in the Status registers of the CAN controllers for common accessibility. If devices with more or fewer CAN controllers are defined, the number of bits used in the active bytes will change correspondingly. Each defined byte of the following registers contains one particular status bit from each of the CAN controllers in its LS bits.

19.8.1 Central Transmit Status Register (CANTxSR - 0xE004 0000)

Table 275. Central Transit Status Register (CANTxSR - address 0xE004 0000) bit description

Bit	Symbol	Description	Reset Value
3:0	TS4:1	1: the CAN controller CAN4:1 is sending a message (same as TS in the CANGSR). **Remark:** Bits are available if the respective CAN controller is implemented and reserved otherwise (see Table 252).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:8	TBS4:1	1: For CAN controllers CAN4:1 all 3 Tx Buffers are available to the CPU (same as TBS in CANGSR). **Remark:** Bits are available if the respective CAN controller is implemented and reserved otherwise (see Table 252).	All 1
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	TCS4:1	1: For CAN controllers CAN4:1, all requested transmissions have been completed successfully (same as TCS in CANGSR). **Remark:** Bits are available if the respective CAN controller is implemented and reserved otherwise (see Table 252).	All 1
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.8.2 Central Receive Status Register (CANRxSR - 0xE004 0004)

Table 276. Central Receive Status register (CANRxSR - address 0xE004 0004) bit description

Bit	Symbol	Description	Reset Value
3:0	RS4:1	1: the CAN controller CAN4:1 is receiving a message (same as RS in CANGSR). **Remark:** Bits are available if the respective CAN controller is implemented and reserved otherwise (see Table 252).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:8	RBS4:1	1: a received message is available in the CAN controller CAN4:1 (same as RBS in CANGSR). **Remark:** Bits are available if the respective CAN controller is implemented and reserved otherwise (see Table 252).	0
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
19:16	DOS4:1	1: a message was lost because the preceding message to this CAN controller was not read out quickly enough (same as DOS in CANGSR). **Remark:** Bits are available if the respective CAN controller CAN4:1 is implemented and reserved otherwise (see Table 252).	0
31:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.8.3 Central Miscellaneous Status Register (CANMSR - 0xE004 0008)

Table 277. Central Miscellaneous Status Register (CANMSR - address 0xE004 0008) bit description

Bit	Symbol	Description	Reset Value
3:0	ES4:1	1: For CAN controller CAN4:1, one or both of the Tx and Rx Error Counters has reached the limit set in the EWL register (same as ES in CANGSR). **Remark:** Bits are available if the respective CAN controller CAN4:1 is implemented and reserved otherwise (see Table 252).	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:8	BS4:1	1: the CAN controller CAN4:1is currently involved in bus activities (same as BS in CANGSR). **Remark:** Bits are available if the respective CAN controller CAN4:1 is implemented and reserved otherwise (see Table 252).	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.9 Global acceptance filter

This block provides lookup for received Identifiers (called Acceptance Filtering in CAN terminology) for all the CAN Controllers. It includes a 512 x 32 (2 kbyte) RAM in which software maintains one to five tables of Identifiers. This RAM can contain up to 1024 Standard Identifiers or 512 Extended Identifiers, or a mixture of both types.

If Standard (11 bit) Identifiers are used in the application, at least one of 3 tables in Acceptance Filter RAM must not be empty. If the optional “fullCAN mode” is enabled, the first table contains Standard identifiers for which reception is to be handled in this mode. The next table contains individual Standard Identifiers and the third contains ranges of Standard Identifiers, for which messages are to be received via the CAN Controllers. The tables of fullCAN and individual Standard Identifiers must be arranged in ascending numerical order, one per halfword, two per word. Since each CAN bus has its own address map, each entry also contains the number of the CAN Controller to which it applies. The numbering of CAN controllers depends on the CAN peripheral implemented:

On no-suffix and /00 devices, the CAN controllers are numbered 1 to n (n = 2 or 4) in the LUT tables . However, on /01 devices, the CAN controllers are numbered 0 to n-1 in the LUT tables (see Figure 67 to Figure 69).

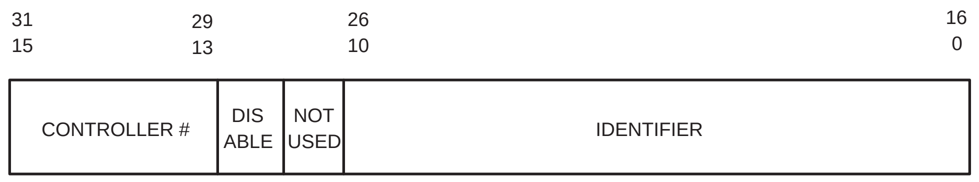


Fig 67. Entry in FullCAN and individual standard identifier tables

The table of Standard Identifier Ranges contains paired upper and lower (inclusive) bounds, one pair per word. These must also be arranged in ascending numerical order.

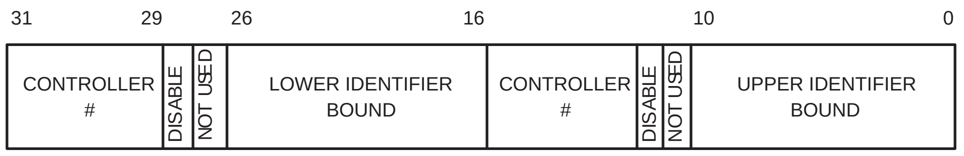


Fig 68. Entry in standard identifier range table

The disable bits in Standard entries provide a means to turn response, to particular CAN Identifiers or ranges of Identifiers, on and off dynamically. When the Acceptance Filter function is enabled, only the disable bits in Acceptance Filter RAM can be changed by software. Response to a range of Standard addresses can be enabled by writing 32 zero bits to its word in RAM, and turned off by writing 32 one bits (0xFFFF FFFF) to its word in RAM. Only the disable bits are actually changed. Disabled entries must maintain the ascending sequence of Identifiers.

If Extended (29 bit) Identifiers are used in the application, at least one of the other two tables in Acceptance Filter RAM must not be empty, one for individual Extended Identifiers and one for ranges of Extended Identifiers. The table of individual Extended Identifiers must be arranged in ascending numerical order.

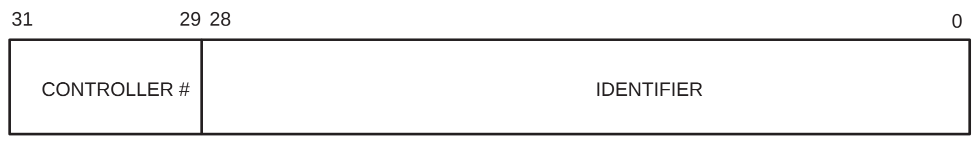


Fig 69. Entry in either extended identifier table

The table of ranges of Extended Identifiers must contain an even number of entries, of the same form as in the individual Extended Identifier table. Like the Individual Extended table, the Extended Range must be arranged in ascending numerical order. The first and second (3rd and 4th …) entries in the table are implicitly paired as an inclusive range of Extended addresses, such that any received address that falls in the inclusive range is received (accepted). Software must maintain the table to consist of such word pairs.

There is no facility to receive messages to Extended identifiers using the fullCAN method.

Five address registers point to the boundaries between the tables in Acceptance Filter RAM: fullCAN Standard addresses, Standard Individual addresses, Standard address ranges, Extended Individual addresses, and Extended address ranges. These tables must be consecutive in memory. The start of each of the latter four tables is implicitly the end of the preceding table. The end of the Extended range table is given in an End of Tables register. If the start address of a table equals the start of the next table or the End Of Tables register, that table is empty.

When the Receive side of a CAN controller has received a complete Identifier, it signals the Acceptance Filter of this fact. The Acceptance Filter responds to this signal, and reads the Controller number, the size of the Identifier, and the Identifier itself from the Controller. It then proceeds to search its RAM to determine whether the message should be received or ignored.

If fullCAN mode is enabled and the CAN controller signals that the current message contains a Standard identifier, the Acceptance Filter first searches the table of identifiers for which reception is to be done in fullCAN mode. Otherwise, or if the AF doesn’t find a match in the fullCAN table, it searches its individual Identifier table for the size of Identifier signalled by the CAN controller. If it finds an equal match, the AF signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register.

If the Acceptance Filter does not find a match in the appropriate individual Identifier table, it then searches the Identifier Range table for the size of Identifier signalled by the CAN controller. If the AF finds a match to a range in the table, it similarly signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register. If the Acceptance Filter does not find a match in either the individual or Range table for the size of Identifier received, it signals the CAN controller to discard/ignore the received message.

19.10 Acceptance filter registers

19.10.1 Acceptance Filter Mode Register (AFMR - 0xE003 C000)

Table 278. Acceptance Filter Mode Register (AFMR - address 0xE003 C000) bit description

Bit	Symbol	Value	Description	Reset Value
0	AccOff	1	if AccBP is 0, the Acceptance Filter is not operational. All Rx messages on all CAN buses are ignored. Software must set this bit before modifying the contents of any of the registers described below, and before modifying the contents of Lookup Table RAM in any way other than setting or clearing Disable bits in Standard Identifier entries.	1
1	AccBP	1	All Rx messages are accepted on enabled CAN controllers. When both this bit and AccOff are 0, the Acceptance filter operates to screen received CAN Identifiers.	0
2	eFCAN	0	Software must read all messages for all enabled IDs on all enabled CAN buses, from the receiving CAN controllers.	0
		1	The Acceptance Filter itself will take care of receiving and storing messages for selected Standard ID values on selected CAN buses. See Section 19.12 “Fullcan mode” on page 300.	
31:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

If Bits 1:0 in this register are x1 (reset), the registers can be written. If bits1:0 are 00, the acceptance filter is operational and will allow Rx interrupts if enabled. If bits1:0 are 10, all Rx messages accepted and will allow Rx interrupts if enabled.

19.10.2 Standard Frame Individual Start Address register (SFF_sa - 0xE003 C004)

Table 279. Standard Frame Individual Start Address register (SFF_sa - address 0xE003 C004) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	SFF_sa	The start address of the table of individual Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the SFF_GRP_sa register described below. For compatibility with possible future devices, please write zeroes in bits 31:11 and 1:0 of this register. If the eFCAN bit in the AFMR is 1, this value also indicates the size of the table of Standard IDs which the Acceptance Filter will search and (if found) automatically store received messages in Acceptance Filter RAM.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.3 Standard Frame Group Start Address Register (SFF_GRP_sa - 0xE003 C008)

Table 280. Standard Frame Group Start Address register (SFF_GRP_sa - address 0xE003 C008) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	SFF_GRP_sa	The start address of the table of grouped Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_sa register described below. The largest value that should be written to this register is 0x800, when only the Standard Individual table is used, and the last word (address 0x7FC) in AF Lookup Table RAM is used.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.4 Extended Frame Start Address Register (EFF_sa - 0xE003 C00C)

Table 281. Extended Frame Start Address register (EFF_sa - address 0xE003 C00C) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	EFF_sa	The start address of the table of individual Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_GRP_sa register described below. The largest value that should be written to this register is 0x800, when both Extended Tables are empty and the last word (address 0x7FC) in AF Lookup Table RAM is used.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.5 Extended Frame Group Start Address Register (EFF_GRP_sa - 0xE003 C010)

Table 282. Extended Frame Group Start Address register (EFF_GRP_sa - address 0xE003 C010) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	Eff_GRP_sa	The start address of the table of grouped Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the ENDofTable register described below. The largest value that should be written to this register is 0x800, when this table is empty and the last word (address 0x7FC) in AF Lookup Table RAM is used.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.6 End of AF Tables register (ENDofTable - 0xE003 C014)

Table 283. End of AF Tables register (ENDofTable - address 0xE003 C014) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	EndofTable	The address above the last active address in the last active AF table. If the eFCAN bit in the AFMR is 0, the largest value that should be written to this register is 0x800, which allows the last word (address 0x7FC) in AF Lookup Table RAM to be used. If the eFCAN bit in the AFMR is 1, this value marks the start of the area of Acceptance Filter RAM, into which the Acceptance Filter will automatically receive messages for selected IDs on selected CAN buses. In this case, the maximum value that should be written to this register is 0x800 minus 6 times the value in SFF_sa. This allows 12 bytes of message storage between this address and the end of Acceptance Filter RAM, for each Standard ID that is specified between the start of Acceptance Filter RAM, and the next active AF table.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.7 LUT Error Address register (LUTerrAd - 0xE003 C018)

Table 284. LUT Error Address register (LUTerrAd - address 0xE003 C018) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	LUTerrAd	It the LUT Error bit (below) is 1, this read-only field contains the address in AF Lookup Table RAM, at which the Acceptance Filter encountered an error in the content of the tables.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.8 LUT Error register (LUTerr - 0xE003 C01C)

Table 285. LUT Error register (LUTerr - address 0xE003 C01C) bit description

Bit	Symbol	Description	Reset Value
0	LUTerr	This read-only bit is set to 1 if the Acceptance Filter encounters an error in the content of the tables in AF RAM. It is cleared when software reads the LUTerrAd register. This condition is ORed with the “other CAN” interrupts from the CAN controllers, to produce the request for a VIC interrupt channel.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.9 Global FullCANInterrupt Enable register (FCANIE - 0xE003 C020)

A write access to the Global FullCAN Interrupt Enable register is only possible when the Acceptance Filter is in the off mode.

Table 286. Global FullCAN Enable register (FCANIE - address 0xE003 C020) bit description

Bit	Symbol	Description	Reset Value
0	FCANIE	Global FullCAN Interrupt Enable. When 1, this interrupt is enabled.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.10.10 FullCAN Interrupt and Capture registers (FCANIC0 - 0xE003 C024 and FCANIC1 - 0xE003 C028)

For detailed description on these two registers, see Section 19.12 "Fullcan mode".

Table 287. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0xE003 C024) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd0	FullCan Interrupt Pending bit 0.	0
...	IntPndx (0<x<31)	FullCan Interrupt Pending bit x.	0
31	IntPnd31	FullCan Interrupt Pending bit 31.	0

Table 288. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0xE003 C028) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd32	FullCan Interrupt Pending bit 32.	0
...	IntPndx (32<x<63)	FullCan Interrupt Pending bit x.	0
31	IntPnd63	FullCan Interrupt Pending bit 63.	0

19.11 Examples of acceptance filter tables and ID index values

Suppose that the five Acceptance Filter address registers contain the values shown in the third column below. In this case each table contains the decimal number of words and entries shown in the next two columns, and the ID Index field of the CANRFS register can return the decimal values shown in the right-most column, for CAN messages whose Identifiers match the entries in that table.

Table 289. Example of acceptance filter tables and ID index values

Table	Register	Value	# Words	# Entries	ID Indexes
Standard Individual	SFF_sa	0x040	8_{10}	16_{10}	$0\text{-}15_{10}$
Standard Group	SFF_GRP_sa	0x060	4_{10}	4_{10}	$16\text{-}19_{10}$
Extended Individual	EFF_sa	0x070	8_{10}	16_{10}	$20\text{-}55_{10}$
Extended Group	EFF_GRP_sa	0x100	8_{10}	16_{10}	$56\text{-}57_{10}$
	ENDofTable	0x110			

Figure 70 below is a more detailed and graphic example of the address registers, table layout, and ID Index values. It shows:

- A Standard Individual table starting at the start of Acceptance Filter RAM and containing 26 Identifiers, followed by:
- A Standard Group table containing 12 ranges of Identifiers, followed by:
- An Extended Individual table containing 3 Identifiers, followed by:
- An Extended Group table containing 2 ranges of Identifiers.

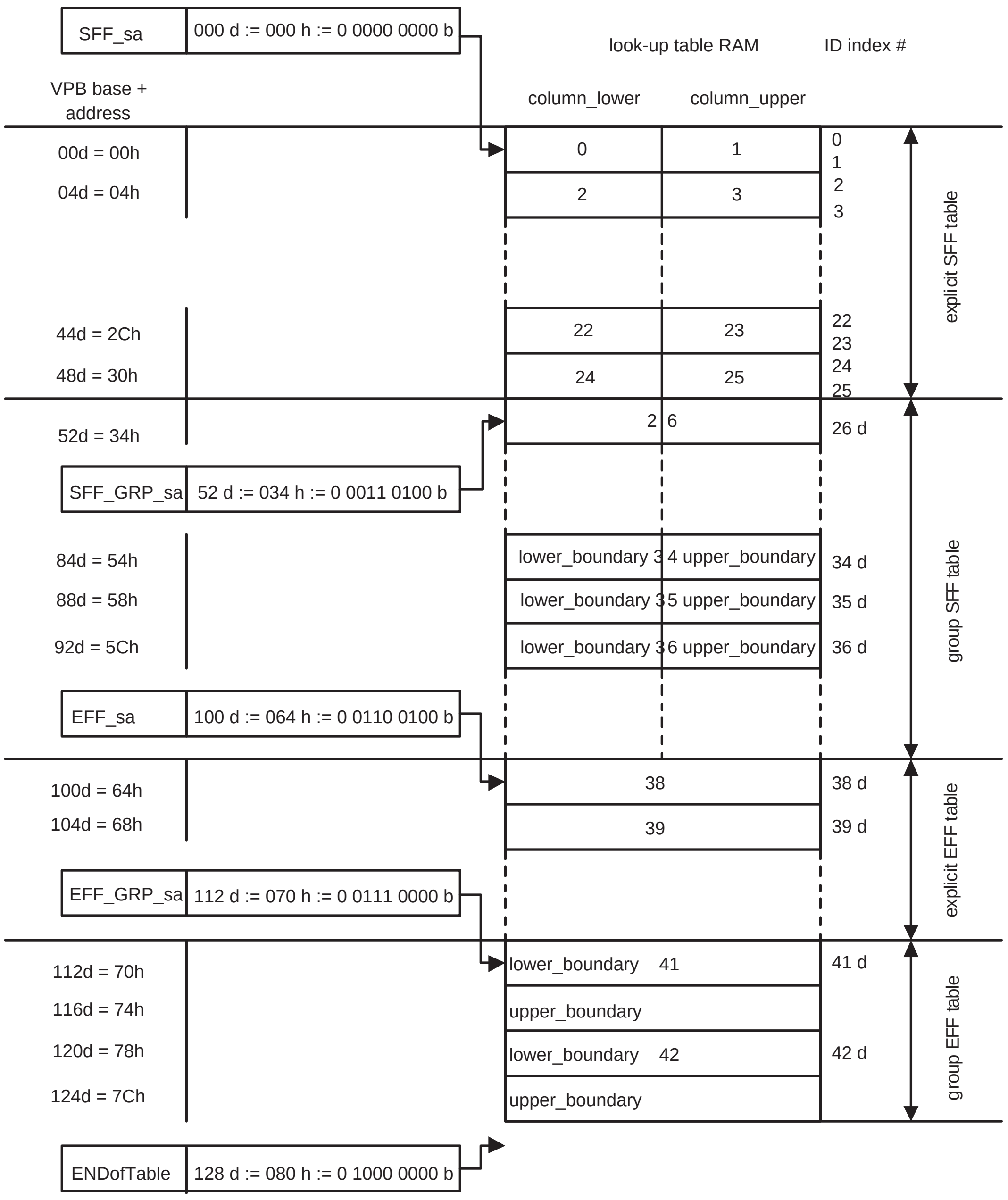


Fig 70. Detailed example of acceptance filter tables and ID index values

19.12 Fullcan mode

When fullCAN mode is enabled, the Acceptance Filter itself takes care of receiving and storing messages for selected Standard ID values on selected CAN buses, in the style of “FullCAN” controllers.

In order to set this bit and use this mode, two other conditions must be met with respect to the contents of Acceptance Filter RAM and the pointers into it:

- The Standard Frame Individual Start Address Register (SFF_sa) must be greater than or equal to the number of IDs for which automatic receive storage is to be done, times two. SFF_sa must be rounded up to a multiple of 4 if necessary.
- The EndOfTable register must be less than or equal to 0x800 minus 6 times the SFF_sa value, to allow 12 bytes of message storage for each ID for which automatic receive storage will be done.

When these conditions are met and eFCAN is set:

- The area between the start of Acceptance Filter RAM and the SFF_sa address, is used for a table of individual Standard IDs and CAN Controller/bus identification, sorted in ascending order and in the same format as in the Individual Standard ID table (see Figure 67 “Entry in FullCAN and individual standard identifier tables” on page 293). Entries can be marked as “disabled” as in the other Standard tables. If there are an odd number of “FullCAN” ID’s, at least one entry in this table must be so marked.
- The first (SFF_sa)/2 IDindex values are assigned to these automatically-stored ID’s. That is, IDindex values stored in the Rx Frame Status Register, for IDs not handled in this way, are increased by (SFF_sa)/2 compared to the values they would have when eFCAN is 0.
- When a Standard ID is received, the Acceptance Filter searches this table before the Standard Individual and Group tables.
- When a message is received for a controller and ID in this table, the Acceptance filter reads the received message out of the CAN controller and stores it in Acceptance Filter RAM, starting at (EndOfTable) + its IDindex*12.
- The format of such messages is shown in Table 290.

Table 290. Format of automatically stored Rx message

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FF	RTR	0000				SEM		0000				DLC				00000					ID										
+4	Rx Data 4								Rx Data 3								Rx Data 2								Rx Data 1							
+8	Rx Data 8								Rx Data 7								Rx Data 6								Rx Data 5							

The FF, RTR, and DLC fields are as described in Table 265. Hardware sets the SEM field to 01 when it begins to update a message, and to 11 when it finishes doing so. Software should clear SEM to 00 as part of accessing a message. Software must access the three words in a message in a particular way to ensure that they are all from the same received message. Figure 71 below shows how software should use the SEM field to ensure this.

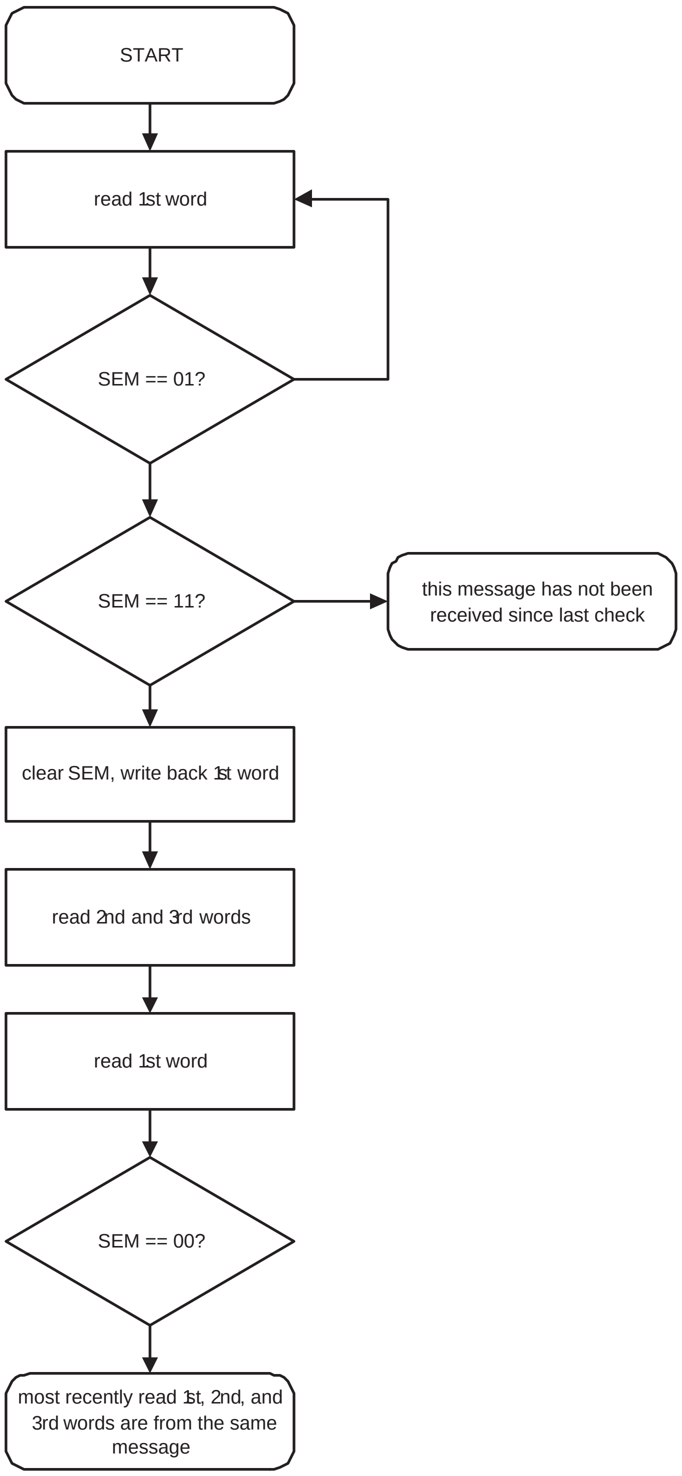

START
read 1st word
SEM == 01?
SEM == 11?
this message has not been received since last check
clear SEM, write back 1st word
read 2nd and 3rd words
read 1st word
SEM == 00?
most recently read 1st, 2nd, and 3rd words are from the same message

Fig 71. Semaphore procedure for reading an auto-stored message

UM10114

Chapter 20: LPC21xx/22xx Analog-to-Digital Converter (ADC)

Rev. 4 — 2 May 2012 User manual

20.1 How to read this chapter

Remark: The LPC21xx and LPC22xx contain different ADC features depending on part number and version. The registers and their addresses that are available in select parts only are shown in Table 291. All other register descriptions are identical for all LPC21xx/LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

Table 291. LPC21xx/22xx part-specific registers

Part	ADC channels	ADC status	Interrupt on conversion completed	Dedicated result registers
no suffix and /00 parts				
LPC2109	4	not available	not available	not available
LPC2119	4	not available	not available	not available
LPC2129	4	not available	not available	not available
LPC2114	4	not available	not available	not available
LPC2124	4	not available	not available	not available
LPC2194	4	not available	not available	not available
LPC2210	8	not available	not available	not available
LPC2220	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2212	8	not available	not available	not available
LPC2214	8	not available	not available	not available
LPC2290	8	not available	not available	not available
LPC2292	8	not available	not available	not available
LPC2294	8	not available	not available	not available
/01 parts				
LPC2109	4	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2119	4	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2129	4	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2114	4	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2124	4	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2194	4	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2210	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2212	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2214	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2290	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2292	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298
LPC2294	8	ADSTAT Table 296	ADINTEN Table 297	ADDRn Table 298

20.2 Features

- 10 bit successive approximation analog to digital converter
- Input multiplexing among 4 pins or 8 pins
- Power-down mode
- Measurement range 0 V to V_{DDA}
- 10 bit conversion time $\geq$ 2.44 μs (400,000 conversions per second)
- Burst conversion mode for single or multiple inputs
- Optional conversion on transition on input pin or Timer Match signal
- Dedicated result register to reduce interrupt overhead for every analog pin (see Table 291)

20.3 Description

Basic clocking for the ADC is provided by the APB clock. A programmable divider is included in each converter to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks.

20.4 Pin description

Table 292 gives a brief summary of each of ADC related pins.

Table 292. ADC pin description

Pin	Type	Description
AIN3:0, AIN7:4	Input	**Analog Inputs.** The ADC cell can measure the voltage on any of these input signals. Note that these analog inputs are always connected to their pins, even if the Pin function Select register assigns them to port pins (for parts with no suffix or /00 suffix only). A simple self-test of the ADC can be done by driving these pins as port outputs. **Warning:** While the ADC pins are specified as 5 V tolerant (see Section 7.2 and Section 7.3), the analog multiplexing in the ADC block is not. More than 3.3 V (V_{DDA}) should not be applied to any pin that is selected as an ADC input, otherwise the ADC reading is incorrect. If for example AIN0 and AIN1 are used as the ADC inputs and voltage on AIN0 = 4.5 V while AIN1 = 2.5 V, an excessive voltage on the AIN0 can cause an incorrect reading of the AIN1, although the AIN1 input voltage is within the right range. If the ADC is not used in an application then the pins associated with ADC inputs can be used as 5 V tolerant digital IO pins.
V_{DDA}, V_{SSA}	Power	**Analog Power and Ground.** These should be nominally the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. The V_{DDA} pin provides the voltage reference level for the A/D converter.

Remark: When the ADC is not used, the V_{DDA} pin must be connected to the power supply $V_{DD(3V3)}$, and pin V_{SSA} must be grounded. These pins should **not** be left floating.

20.5 Register description

The ADC registers are shown in Table 293.

Table 293. ADC registers

Name	Description	Access	Reset value[1]	Address
ADCR	ADC Control Register. The ADCR register must be written to select the operating mode before ADC conversion can occur.	R/W	0x0000 0001	0xE003 4000
ADGDR	ADC Global Data Register. This register contains the ADC's DONE bit and the result of the most recent ADC conversion.	R/W	NA	0xE003 4004
ADSTAT	ADC Status Register. This register contains DONE and OVERRUN flags for all of the ADC channels, as well as the ADC interrupt flag.	RO	0x0000 0000	0xE003 4030
ADINTEN	ADC Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each ADC channel to be included or excluded from contributing to the generation of an ADC interrupt.	R/W	0x0000 0100	0xE003 400C
ADDR0	ADC Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	0xE003 4010
ADDR1	ADC Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	0xE003 4014
ADDR2	ADC Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	0xE003 4018
ADDR3	ADC Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	0xE003 401C
ADDR4	ADC Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	0xE003 4020
ADDR5	ADC Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	0xE003 4024
ADDR6	ADC Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	0xE003 4028
ADDR7	ADC Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	0xE003 402C

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

20.5.1 ADC Control Register (ADCR - 0xE003 4000)

Table 294. ADC Control Register (ADCR - address 0xE003 4000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	SEL		Selects which of the ADC pins is (are) to be sampled and converted. Bit 0 selects Pin AIN0, and bit 7 selects pin AIN7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones is allowed. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The APB clock (PCLK) is divided by (this value plus one) to produce the clock for the ADC, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST	1	The AD converter does repeated conversions at the rate selected by the CLKS field, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that’s in progress when this bit is cleared will be completed. **Important:** START bits must be 000 when BURST = 1 or conversions will not start.	0
		0	Conversions are software controlled and require 11 clocks.	
19:17	CLKS		This field selects the number of clocks used for each conversion in Burst mode, and the number of bits of accuracy of the result in the RESULT bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits).	000
		000	11 clocks / 10 bits	
		001	10 clocks / 9bits	
		010	9 clocks / 8 bits	
		011	8 clocks / 7 bits	
		100	7 clocks / 6 bits	
		101	6 clocks / 5 bits	
		110	5 clocks / 4 bits	
		111	4 clocks / 3 bits	
20	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PDN	1	The ADC is operational.	0
		0	The ADC is in power-down mode.	
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 294. ADC Control Register (ADCR - address 0xE003 4000) bit description

Bit	Symbol	Value	Description	Reset value
26:24	START		When the BURST bit is 0, these bits control whether and when an ADC conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	
		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/CAP0.2 pin.	
		011	Start conversion when the edge selected by bit 27 occurs on P0.22/CAP0.0/MAT0.0 pin.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1.	
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.5.2 ADC Global Data Register (ADGDR - 0xE003 4004)

Table 295. ADC Global Data Register (ADGDR - address 0xE003 4004) bit description

Bit	Symbol	Description	Reset value
5:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin selected by the SEL field, divided by the voltage on the V_{DDA} pin (V/V_{REF}). Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on V_{SSA}, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on V_{REF}.	NA
23:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 000 identifies channel 0, 001 channel 1...).	NA
29:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	OVERUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an ADC conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

20.5.3 ADC Status Register (ADSTAT - 0xE003 4030)

The ADC Status register allows checking the status of all ADC channels simultaneously. The DONE and OVERRUN flags appearing in the ADDRn register for each ADC channel are mirrored in ADSTAT. The interrupt flag (the logical OR of all DONE flags) is also found in ADSTAT.

Table 296. ADC Status Register (ADSTAT - address 0xE003 4030) bit description

Bit	Symbol	Description	Reset value
0	DONE0	This bit mirrors the DONE status flag from the result register for ADC channel 0.	0
1	DONE1	This bit mirrors the DONE status flag from the result register for ADC channel 1.	0
2	DONE2	This bit mirrors the DONE status flag from the result register for ADC channel 2.	0
3	DONE3	This bit mirrors the DONE status flag from the result register for ADC channel 3.	0
4	DONE4	This bit mirrors the DONE status flag from the result register for ADC channel 4.	0
5	DONE5	This bit mirrors the DONE status flag from the result register for ADC channel 5.	0
6	DONE6	This bit mirrors the DONE status flag from the result register for ADC channel 6.	0
7	DONE7	This bit mirrors the DONE status flag from the result register for ADC channel 7.	0
8	OVERRUN0	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 0.	0
9	OVERRUN1	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 1.	0
10	OVERRUN2	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 2.	0
11	OVERRUN3	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 3.	0
12	OVERRUN4	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 4.	0
13	OVERRUN5	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 5.	0
14	OVERRUN6	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 6.	0
15	OVERRUN7	This bit mirrors the OVERRRUN status flag from the result register for ADC channel 7.	0
16	ADINT	This bit is the ADC interrupt flag. It is one when any of the individual ADC channel Done flags is asserted and enabled to contribute to the ADC interrupt via the ADINTEN register.	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.5.4 ADC Interrupt Enable Register (ADINTEN - 0xE003 400C)

This register allows control over which ADC channels generate an interrupt when a conversion is complete. For example, it may be desirable to use some ADC channels to monitor sensors by continuously performing conversions on them. The most recent results are read by the application program whenever they are needed. In this case, an interrupt is not desirable at the end of each conversion for some ADC channels.

Table 297. ADC Interrupt Enable Register (ADINTEN - address 0xE003 400C) bit description

Bit	Symbol	Value	Description	Reset value
0	ADINTEN0	0	Completion of a conversion on ADC channel 0 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 0 will generate an interrupt.	
1	ADINTEN1	0	Completion of a conversion on ADC channel 1 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 1 will generate an interrupt.	
2	ADINTEN2	0	Completion of a conversion on ADC channel 2 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 2 will generate an interrupt.	

Table 297. ADC Interrupt Enable Register (ADINTEN - address 0xE003 400C) bit description

Bit	Symbol	Value	Description	Reset value
3	ADINTEN3	0	Completion of a conversion on ADC channel 3 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 3 will generate an interrupt.	
4	ADINTEN4	0	Completion of a conversion on ADC channel 4 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 4 will generate an interrupt.	
5	ADINTEN5	0	Completion of a conversion on ADC channel 5 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 5 will generate an interrupt.	
6	ADINTEN6	0	Completion of a conversion on ADC channel 6 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 6 will generate an interrupt.	
7	ADINTEN7	0	Completion of a conversion on ADC channel 7 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 7 will generate an interrupt.	
8	ADGINTEN	0	Only the individual ADC channels enabled by ADINTEN7:0 will generate interrupts.	1
		1	Only the global DONE flag in ADDR is enabled to generate an interrupt.	
31:9	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

20.5.5 ADC Data Registers (ADDR0 to ADDR7- 0xE003 4010 to 0xE003 402C)

The ADC Data Register hold the result when an ADC conversion is complete, and also include the flags that indicate when a conversion has been completed and when a conversion overrun has occurred.

Table 298. ADC Data Registers (ADDR0 to ADDR7 - 0xE003 4010 to 0xE003 402C) bit description

Bit	Symbol	Description	Reset value
5:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AIN pin, divided by the voltage on the V_{REF} pin (V/V_{REF}). Zero in the field indicates that the voltage on the AIN pin was less than, equal to, or close to that on V_{SSA}, while 0x3FF indicates that the voltage on AIN was close to, equal to, or greater than that on V_{REF}.	NA
29:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits.This bit is cleared by reading this register.	
31	DONE	This bit is set to 1 when an ADC conversion completes. It is cleared when this register is read.	NA

20.6 Operation

20.6.1 Hardware-triggered conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the ADC will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

20.6.2 Interrupts

An interrupt request is asserted to the Vectored Interrupt Controller (VIC) when the DONE bit is 1. Software can use the Interrupt Enable bit for the ADC in the VIC to control whether this assertion results in an interrupt. DONE is negated when the ADDR is read.

20.6.3 Accuracy vs. digital receiver

The AIN function must be selected in corresponding Pin Select register (see Section 8.6) in order to get accurate voltage readings on the monitored pin. For a pin hosting an ADC input, it is not possible to have a have a digital function selected and yet get valid ADC readings. An inside circuit disconnects ADC hardware from the associated pin whenever a digital function is selected on that pin.

21.1 How to read this chapter

Read this chapter for LPC21xx and LPC2xx parts with on-chip flash memory.

Table 299. LPC21xx and LPC22xx flash memory options

Part	Flash size
no suffix, /00, and /01 parts	
LPC2109	64 kB
LPC2119	128 kB
LPC2129	256 kB
LPC2114	128 kB
LPC2124	256 kB
LPC2194	256 kB
LPC2210	flashless, see Section 22.1
LPC2220	flashless, see Section 22.1
LPC2212	128 kB
LPC2214	256 kB
LPC2290	flashless, see Section 22.1
LPC2292	256 kB
LPC2294	256 kB

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

21.2 Flash boot loader

The flash boot loader controls initial operation after reset and also provides the means to accomplish programming of the flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device, or programming of the Flash memory by the application program in a running system.

21.3 Features

- In-System Programming: In-System programming (ISP) means programming or reprogramming the on-chip flash memory using the boot loader software and a serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming means performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.

21.4 Applications

The flash boot loader provides both In-System and In-Application programming interfaces for programming the on-chip flash memory.

21.5 Description

The flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A LOW level after reset at the P0.14 pin is considered as an external hardware request to start the ISP command handler. Assuming that a proper signal is present on XTAL1 pin when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on whether to continue with user code or ISP handler is made. If P0.14 is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P0.14 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

Pin P0.14, which is used as hardware request for ISP, requires special attention. Since P0.14 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

21.5.1 Memory map after any reset

The boot block is 8 kB in size and resides in the top portion (starting from 0x0001 0000 for devices with 64 kB flash, from 0x0001 E000 for devices with 128 kB flash, and from 0x0003 E000 for devices with 256 kB flash) of the on-chip flash memory. After any reset the entire boot block is also mapped to the top of the on-chip memory space. i.e. the boot block is also visible in the memory region starting from the address 0x7FFF E000. The flash boot loader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in this chapter. The interrupt vectors residing in the boot block of the on-chip flash memory also become active after reset, i.e., the bottom 64 bytes of the boot block are also visible in the memory region starting from the address 0x0000 0000. The reset vector contains a jump instruction to the entry point of the flash boot loader software.

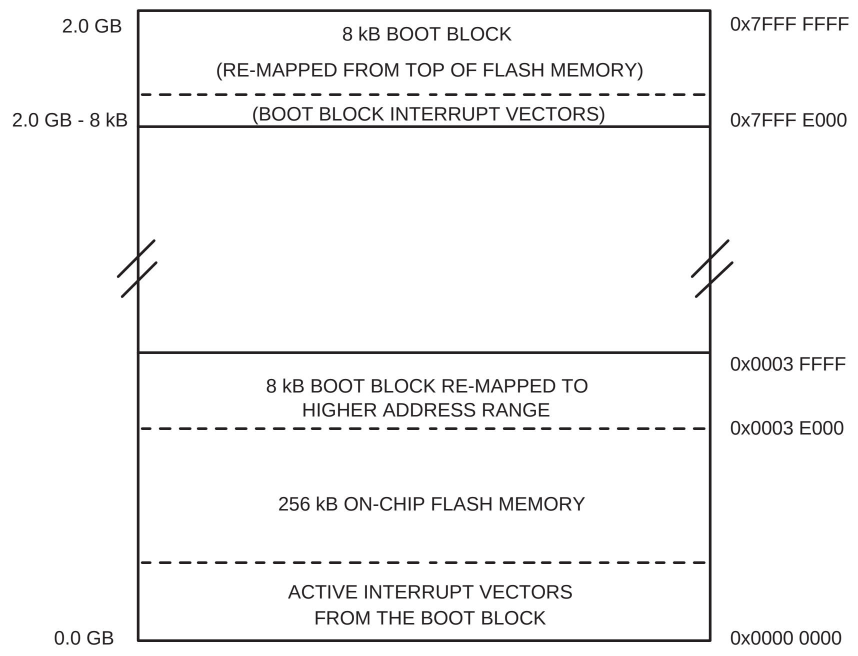

2.0 GB
8 kB BOOT BLOCK
(RE-MAPPED FROM TOP OF FLASH MEMORY)
0x7FFF FFFF
2.0 GB - 8 kB
(BOOT BLOCK INTERRUPT VECTORS)
0x7FFF E000
0x0003 FFFF
8 kB BOOT BLOCK RE-MAPPED TO HIGHER ADDRESS RANGE
0x0003 E000
256 kB ON-CHIP FLASH MEMORY
ACTIVE INTERRUPT VECTORS FROM THE BOOT BLOCK
0.0 GB
0x0000 0000

Fig 72. Map of lower memory after reset for 256 kB flash devices

21.5.2 Criterion for valid user code

The reserved ARM interrupt vector location (0x0000 0014) should contain the 2’s complement of the check-sum of the remaining interrupt vectors. This causes the checksum of all of the vectors together to be 0. The boot loader code disables the overlaying of the interrupt vectors from the boot block, then checksums the interrupt vectors in sector 0 of the flash. If the signatures match then the execution control is transferred to the user code by loading the program counter with 0x0000 0000. Hence the user flash reset vector should contain a jump instruction to the entry point of the user application code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. The host should send a ’?’ (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the Host. In response to this host should send the same string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. Host should respond by sending the crystal frequency (in kHz) at which the part is running. For example, if the part is running at 10 MHz, the response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly, the crystal frequency should be greater than or equal to 10 MHz. The on-chip PLL is not used by the boot code.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 21.9 “ISP commands” on page 320.

21.5.3 Communication protocol

All ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in UU-encoded format.

21.5.4 ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands)

21.5.5 ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Data only for Read commands)

21.5.6 ISP data format

The data stream is in UU-encode format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

21.5.7 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

21.5.8 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

21.5.9 Interrupts during ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

21.5.10 Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing the interrupt vectors from the user flash area are active. The user should either disable interrupts, or ensure that user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM, before making a flash erase/write IAP call. The IAP code does not use or disable interrupts.

21.5.11 RAM used by ISP command handler

ISP commands use on-chip RAM from 0x4000 0120 to 0x4000 01FF. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top – 32. The maximum stack usage is 256 bytes and it grows downwards.

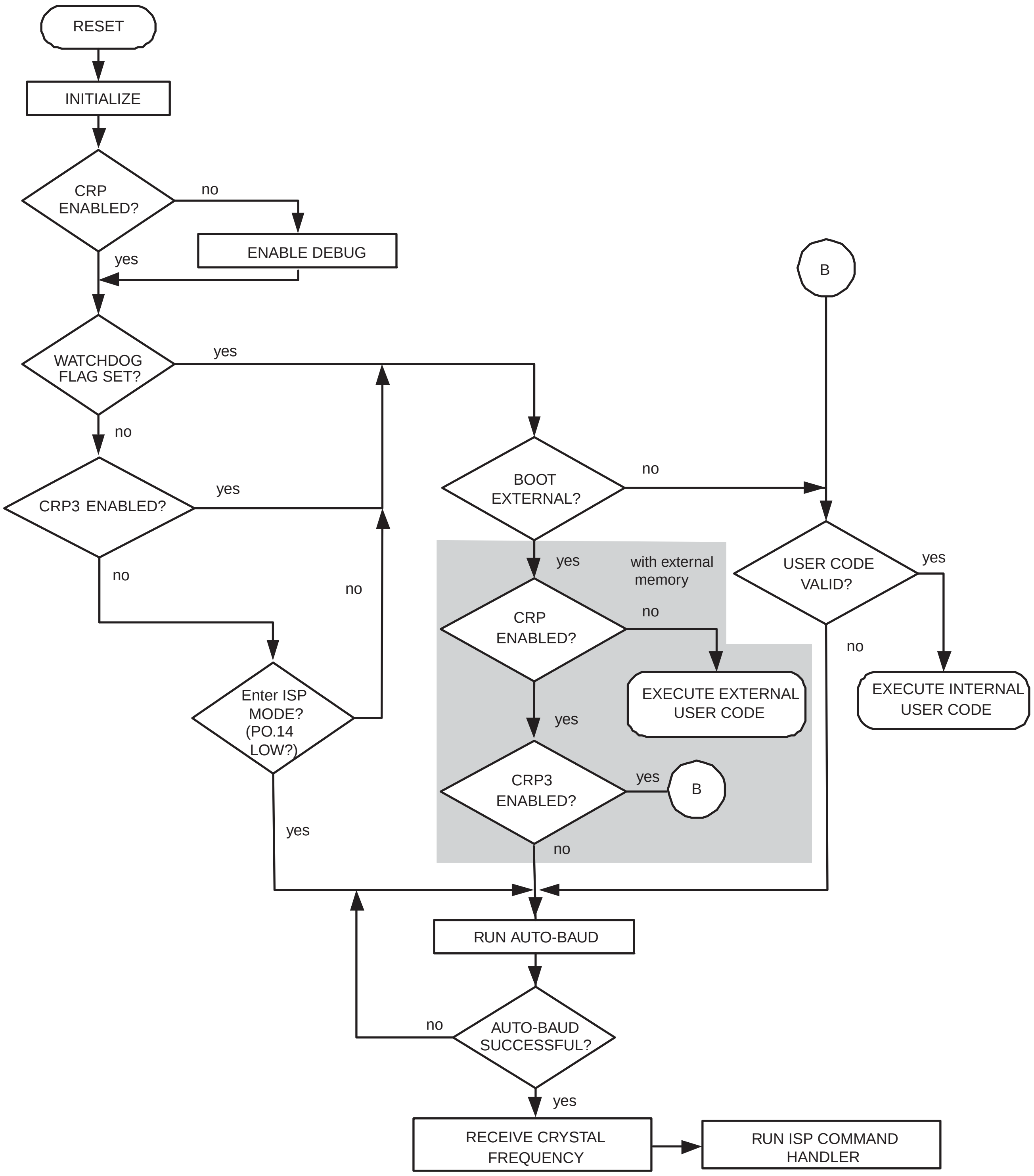
21.5.12 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes and it grows downwards.

21.5.13 RAM used by RealMonitor

The RealMonitor uses on-chip RAM from 0x4000 0040 to 0x4000 011F. he user could use this area if RealMonitor based debug is not required. The Flash boot loader does not initialize the stack for RealMonitor.

21.5.14 Boot process flowchart



The grey-shaded area is specific to the boot process for LPC22xx parts with external memory. CRP3 is available starting with boot loader versions 1.68 (see Table 304).

Fig 73. Boot process flowchart

21.6 Sector numbers

Some IAP and ISP commands operate on "sectors" and specify sector numbers. The following table indicates the correspondence between sector numbers and memory addresses for LPC21xx/LPC22xx devices containing 64 kB, 128 kB, or 256 kB of flash respectively. IAP, ISP, and RealMonitor routines are located in the boot block. The boot block is present at the top of each flash memory. Because of the boot block, only 120 kB of the 128 kB, and 248 kB of the 256 kB flash devices are available for user code. ISP and IAP commands do not allow write/erase/go operation on the boot block.

Table 300. Flash sectors

Sector Number	Sector Size [kB] 64 kB flash	Address Range	Sector Size [kB] 128 kB flash	Address Range	Sector Size [kB] 256 kB flash	Address Range
0	8	0x0000 0000 - 0x0000 1FFF	8	0x0000 0000 - 0x0000 1FFF	8	0x0000 0000 - 0x0000 1FFF
1	8	0x0000 2000 - 0x0000 3FFF	8	0x0000 2000 - 0x0000 3FFF	8	0x0000 2000 - 0x0000 3FFF
2	8	0x0000 4000 - 0x0000 5FFF	8	0x0000 4000 - 0x0000 5FFF	8	0x0000 4000 - 0x0000 5FFF
3	8	0x0000 6000 - 0x0000 7FFF	8	0x0000 6000 - 0x0000 7FFF	8	0x0000 6000 - 0x0000 7FFF
4	8	0x0000 8000 - 0x0000 9FFF	8	0x0000 8000 - 0x0000 9FFF	8	0x0000 8000 - 0x0000 9FFF
5	8	0x0000 A000 - 0x0000 BFFF	8	0x0000 A000 - 0x0000 BFFF	8	0x0000 A000 - 0x0000 BFFF
6	8	0x0000 C000 - 0x0000 DFFF	8	0x0000 C000 - 0x0000 DFFF	8	0x0000 C000 - 0x0000 DFFF
7	8	0x0000 E000 - 0x0000 FFFF	8	0x0000 E000 - 0x0000 FFFF	8	0x0000 E000 - 0x0000 FFFF
8	8	0x0001 0000 - 0x0001 1FFF	8	0x0001 0000 - 0x0001 1FFF	64	0x0001 0000 - 0x0001 FFFF
9			8	0x0001 2000 - 0x0001 3FFF	64	0x0002 0000 - 0x0002 FFFF
10 (0x0A)			8	0x0001 4000 - 0x0001 5FFF	8	0x0003 0000 - 0x0003 1FFF
11 (0x0B)			8	0x0001 6000 - 0x0001 7FFF	8	0x0003 2000 - 0x0003 3FFF
12 (0x0C)			8	0x0001 8000 - 0x0001 9FFF	8	0x0003 4000 - 0x0003 5FFF
13 (0x0D)			8	0x0001 A000 - 0x0001 BFFF	8	0x0003 6000 - 0x0003 7FFF
14 (0x0E)			8	0x0001 C000 - 0x0001 DFFF	8	0x0003 8000 - 0x0003 9FFF
15 (0x0F)			8	0x0001 E000 - 0x0001 FFFF	8	0x0003 A000 - 0x0003 BFFF
16 (0x10)			-	-	8	0x0003 C000 - 0x0003 DFFF
17 (0x11)			-	-	8	0x0003 E000 - 0x0003 FFFF

21.7 Flash content protection mechanism

The LPC21xx/LPC22xx is equipped with the Error Correction Code (ECC) capable flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user’s code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 000F are protected by the first ECC byte, Flash bytes from 0x0000 0010 to 0x0000 001F are protected by the second ECC byte, etc.

Whenever the CPU requests a read from Flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user’s Flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of user’s Flash memory is erased, corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 16 bytes (or multiples of 16), aligned as described above.

21.8 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows user to enable different levels of security in the system so that access to the on-chip Flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in Flash location at 0x0000 01FC. IAP commands are not affected by the code read protection.

Important: any CRP change becomes effective only after reset.

Table 301. Code Read Protection levels

Name	Pattern programmed in 0x000001FC	Description
CRP1	0x12345678	Access to chip via the JTAG pins is disabled. This mode allows partial Flash update using the following ISP commands and restrictions: • Write to RAM command can not access RAM below 0x40000200 • Copy RAM to Flash command can not write to Sector 0 • Erase command can erase Sector 0 only when all sectors are selected for erase • Compare command is disabled. This mode is useful when CRP is required and Flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the Flash.
CRP2	0x87654321	Access to chip via the JTAG pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to Flash • Compare. When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the JTAG pins is disabled. ISP entry by pulling P0.14 LOW is disabled if a valid user code is present in Flash sector 0. This mode effectively disables ISP override using P0.14 pin. It is up to the user’s application to provide Flash update mechanism using IAP calls if necessary. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

Table 302. Code Read Protection hardware/software interaction

CRP option	User Code Valid	P0.14 pin at reset	JTAG enabled	enter ISP mode	partial Flash update in ISP mode
No	No	X	Yes	Yes	Yes
No	Yes	High	Yes	No	NA
No	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes
CRP2	Yes	High	No	No	NA
CRP2	Yes	Low	No	Yes	No
CRP3	Yes	x	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

In case a CRP mode is enabled and access to the chip is allowed via the ISP, an unsupported or restricted ISP command will be terminated with return code CODE_READ_PROTECTION_ENABLED.

21.8.1 Boot loader options

The levels of code read protection implemented depend on the boot loader code version.The following options can be selected by the user in various revisions of the boot loader code (see Table 303).

Table 303. Code read protection options for different boot loader revisions

Option 1 (CRP1)	Option 2 (CRP2)	Option 3 (CRP 3)	Option 4 (boot loader code rev. 1.65 only)
JTAG access is blocked. Supports partial flash updates. • **ISP commands allowed**: Echo; Set Baud; Erase (except sector 0, must erase all to erase sector 0); Blank Check (fail returns value 0 at location 0); Prepare Sector; Unlock; Read Part ID; Read Boot code version; Write to RAM (addresses above 0x4000 0200); Copy RAM to Flash (except sector 0) • **ISP commands not allowed:** Write to RAM below address 0x4000 0200; Read Memory; Copy RAM to Flash (write to sector 0); Erase sector 0; Go; Compare.	JTAG access is blocked. • **ISP commands allowed**: Echo; Set Baud; Erase (all sectors only); Blank Check (fail returns value 0 at location 0); Prepare Sector; Unlock; Read Part ID; Read Boot code version. • **ISP commands not allowed**: Write to RAM; Read Memory; Copy RAM to Flash; Go; Compare.	JTAG access is blocked. No ISP commands are allowed when P0.14 is pulled LOW and a valid user program is present in flash sector 0.	JTAG access is blocked. • **ISP commands allowed**: Erase (all sectors only); Prepare Sector; Unlock. • **ISP commands not allowed**: Echo; Set Baud; Blank Check (fail returns value 0 at location 0); Write to RAM; Read Memory; Copy RAM to Flash; Go; Compare; Read Part ID; Read Boot code version.

Table 304 shows which code read protection options can be selected for any implemented boot loader revision. Note that parts with boot loader revisions ≤ 1.60 do not allow code read protection.

Table 304. Boot loader revisions

Revision	Applicable to parts	Pattern programmed @ location 0x1FC:		
		0x1234 5678	0x8765 4321	0x4321 8765
1.7	/01	option 1	option 2	option 3
1.69	/00; no suffix	option 1	option 2	option 3
1.68	/01	option 1	option 2	option 3
1.65	/00; no suffix	-	option 2	option 4
1.64 to 1.61	/00; no suffix	-	option 2	-
1.6 and lower	/00; no suffix	-	-	-

21.9 ISP commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 305. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 306
Set Baud Rate	B <Baud Rate> <stop bit>	Table 307
Echo	A <setting>	Table 309
Write to RAM	W <start address> <number of bytes>	Table 310
Read Memory	R <address> <number of bytes>	Table 311
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 312
Copy RAM to Flash	C <Flash address> <RAM address> <number of bytes>	Table 313
Go	G <address> <Mode>	Table 314
Erase sector(s)	E <start sector number> <end sector number>	Table 315
Blank check sector(s)	I <start sector number> <end sector number>	Table 316
Read Part ID	J	Table 317
Read Boot code version	K	Table 319
Compare	M <address1> <address2> <number of bytes>	Table 320

21.9.1 Unlock <unlock code>

Table 306. ISP Unlock command

Command	U
Input	Unlock code: 23130_{10}
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command is used to unlock flash Write, Erase, and Go commands.
Example	"U 23130<CR><LF>" unlocks the flash Write/Erase & Go commands.

21.9.2 Set Baud Rate <baud rate> <stop bit>

Table 307. ISP Set Baud Rate command

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 | 230400 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

Table 308. Correlation between possible ISP baudrates and external crystal frequency (in MHz)

ISP Baudrate .vs. External Crystal Frequency	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

21.9.3 Echo <setting>

Table 309. ISP Echo command

Command	A
Input	Setting: ON = 1 | OFF = 0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

21.9.4 Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The

ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 310. ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. Data should be in UU-encoded format. This command is blocked when code read protection is enabled.
Example	"W 1073742336 4<CR><LF>" writes 4 bytes of data to address 0x4000 0200.

21.9.5 Read memory <address> <no. of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 311. ISP Read memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (UU-encoded)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or Flash memory. This command is blocked when code read protection is enabled.
Example	"R 1073741824 4<CR><LF>" reads 4 bytes of data from address 0x4000 0000.

21.9.6 Prepare sector(s) for write operation <start sector number> <end sector number>

This command makes flash write/erase operation a two step process.

Table 312. ISP Prepare sector(s) for write operation command

Command	P
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot block can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the flash sector 0.

21.9.7 Copy RAM to Flash <Flash address> <RAM address> <no of bytes>

Table 313. ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination Flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot block cannot be written by this command. This command is blocked when code read protection is enabled.
Example	"C 0 1073774592 512<CR><LF>" copies 512 bytes from the RAM address 0x4000 8000 to the flash address 0.

21.9.8 Go <address> <mode>

Table 314. ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode) | A (Execute program in ARM mode).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or Flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled.
Example	"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.

21.9.9 Erase sector(s) <start sector number> <end sector number>

Table 315. ISP Erase sector command

Command	E
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip Flash memory. The boot block can not be erased using this command. This command only allows erasure of all user sectors when the code read protection is enabled.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

21.9.10 Blank check sector(s) <sector number> <end sector number>

Table 316. ISP Blank check sector command

Command	I
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR |
Description	This command is used to blank check one or more sectors of on-chip Flash memory. **Blank check on sector 0 always fails as first 64 bytes are re-mapped to flash boot block.**
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

21.9.11 Read Part Identification number

Table 317. ISP Read Part Identification number command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 318).
Description	This command is used to read the part identification number.

Table 318. LPC21xx/22xx Part identification numbers

Device	ASCII/dec coding	Hex coding
LPC2109	33685249	0x0201 FF01
LPC2119	33685266	0x0201 FF12
LPC2129	33685267	0x0201 FF13
LPC2114	16908050	0x0101 FF12
LPC2124	16908051	0x0101 FF13
LPC2194	50462483	0x0301 FF13
LPC2292	67239699	0x0401 FF13
LPC2294	84016915	0x0501 FF13
LPC2214/01	100794131	0x0601 FF13
LPC2212/01	67239698	0x0401 FF12

In addition to the part identification numbers, the user can determine the device revision by reading the register contents at address 0x0003E070. The register value is encoded as follows: 0x0 corresponds to revision '-', 0x01 corresponds to revision A, 0x02 corresponds to revision B,..., 0x1A corresponds to revision Z. This feature Is implemented starting with device revision C, so the register read will yield a value of 0x03 (for revision C) or larger.

21.9.12 Read Boot code version number

Table 319. ISP Read Boot code version number command

Command	K
Input	None
Return Code	CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>.
Description	This command is used to read the boot code version number.

21.9.13 Compare <address1> <address2> <no of bytes>

Table 320. ISP Compare command

Command	M
Input	**Address1 (DST):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Number of Bytes:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR |
Description	This command is used to compare the memory contents at two locations. **Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. First 64 bytes are re-mapped to flash boot sector**
Example	"M 8192 1073741824 4<CR><LF>" compares 4 bytes from the RAM address 0x4000 0000 to the 4 bytes from the flash address 0x2000.

21.9.14 ISP Return codes

Table 321. ISP Return codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.

Table 321. ISP Return codes Summary

Return Code	Mnemonic	Description
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.
19	CODE_READ_PROTECTION_ENABLED	Code read protection enabled.

21.10 IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. Result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case if number of results are more than number of parameters. Parameter passing is illustrated in the Figure 74. The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 2, returned by the "Blankcheck sector(s)" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 0x7FFF FFF0 location and it is thumb code.

The IAP function could be called in the following way using C.

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x7ffffff1
```

Define data structure or pointers to pass IAP command table and result table to the IAP function:

```
unsigned long command[5];
unsigned long result[3];
```

or

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x……
result= (unsigned long *) 0x……
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting function pointer:

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, result);
```

The IAP call could be simplified further by using the symbol definition file feature supported by ARM Linker in ADS (ARM Developer Suite). You could also call the IAP routine using assembly code.

The following symbol definitions can be used to link IAP routine and user application:

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
0x7fffffb0 A rm_prefetchabort_handler
0x7fffffc0 A rm_dataabort_handler
0x7fffffd0 A rm_irqhandler
0x7fffffe0 A rm_irqhandler2
0x7ffffff0 T iap_entry
```

As per the ARM specification (The ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05) up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not use this space if IAP flash programming is permitted in the application.

Table 322. IAP command summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50_{10}	Table 323
Copy RAM to Flash	51_{10}	Table 324
Erase sector(s)	52_{10}	Table 325
Blank check sector(s)	53_{10}	Table 326
Read Part ID	54_{10}	Table 327
Read Boot code version	55_{10}	Table 328
Compare	56_{10}	Table 329

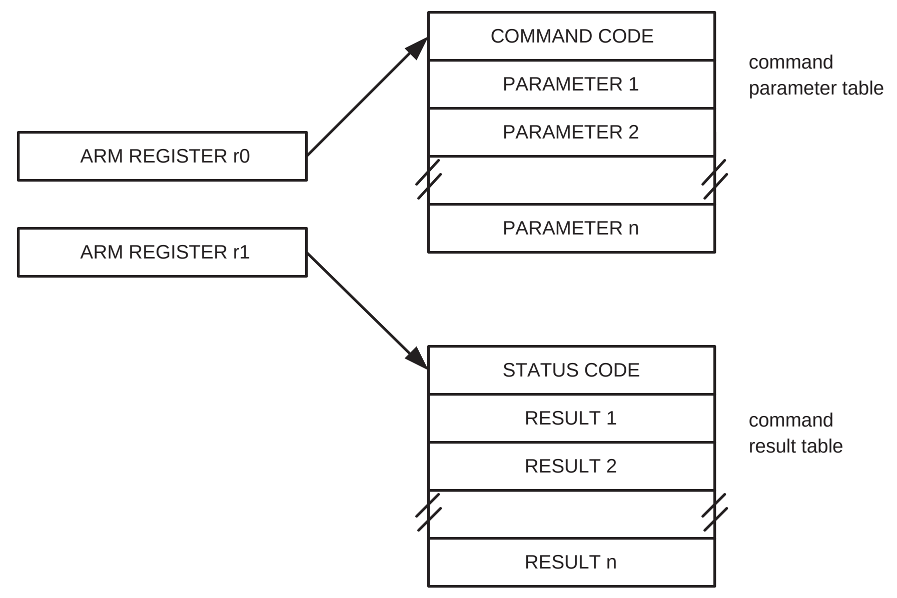


Fig 74. IAP parameter passing

21.10.1 Prepare sector(s) for write operation

This command makes flash write/erase operation a two step process.

Table 323. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Input	**Command code: 50** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).

Table 323. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.

21.10.2 Copy RAM to Flash

Table 324. IAP Copy RAM to Flash command

Command	Copy RAM to Flash
Input	**Command code: 51** **Param0(DST):** Destination Flash address where data bytes are to be written. This address should be a 256 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **Param3:** System Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY |
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command.

21.10.3 Erase sector(s)

Table 325. IAP Erase sector(s) command

Command	Erase Sector(s)
Input	**Command code: 52** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number). **Param2:** System Clock Frequency (CCLK) in kHz.
Return Code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip Flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers.

21.10.4 Blank check sector(s)

Table 326. IAP Blank check sector(s) command

Command	Blank check sector(s)
Input	**Command code: 53** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Return Code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	**Result0:** Offset of the first non blank word location if the Status Code is SECTOR_NOT_BLANK. **Result1:** Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip Flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.

21.10.5 Read Part Identification number

Table 327. IAP Read Part Identification command

Command	Read part identification number
Input	**Command code: 54** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** Part Identification Number (see Table 318 “LPC21xx/22xx Part identification numbers” on page 325 for details)
Description	This command is used to read the part identification number.

21.10.6 Read Boot code version number

Table 328. IAP Read Boot code version number command

Command	Read boot code version number
Input	**Command code: 55** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

21.10.7 Compare <address1> <address2> <no of bytes>

Table 329. IAP Compare command

Command	Compare
Input	**Command code: 56** **Param0(DST):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param1(SRC):** Starting Flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param2:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations. **The result may not be correct when the source or destination includes any of the first 64 bytes starting from address zero. The first 64 bytes can be re-mapped to RAM.**

21.10.8 IAP Status codes

Table 330. IAP Status codes Summary

Status Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.

Table 330. IAP Status codes Summary

Status Code	Mnemonic	Description
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_ FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	Flash programming hardware interface is busy.

21.11 JTAG Flash programming interface

Debug tools can write parts of the flash image to the RAM and then execute the IAP call "Copy RAM to Flash" repeatedly with proper offset.

UM10114

Chapter 22: LPC21xx/22xx On-chip serial boot loader for LPC2210/20/90

Rev. 4 — 2 May 2012 **User manual**

22.1 How to read this chapter

The on-chip serial boot loader controls the boot process for flashless LPC21xx/LPC22xx parts. Read this chapter for flashless parts

- LPC2210, LPC2210/01, LPC2220
- LPC2290, LPC2290/01

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

22.2 Description

The boot loader is designed as a tool that enables the user to load system specific application for further programming of in system available off-chip Flash and/or RAM resources. The boot loader itself does not contain any external memory programming algorithms. The boot loader implemented in flashless LPC21xx/LPC22xx supports a limited set of commands dedicated to code download and its execution from on-chip RAM only. UART0 is the sole serial channel the boot loader can use for data download. Although a fractional divider is available in the UART0, it is not used by the on-chip serial boot loader.

The serial boot loader code is executed every time the part is powered on or reset occurs. The loader executes the initial portion of the ISP command handler and pin P0.14 is sampled in software. Assuming that a proper signal is present on XTAL1 pin when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on whether to continue with user code or ISP handler is made.

If there is no request for the ISP command handler execution (P0.14 was HIGH after a reset), the external memory bank 0 configuration register will be programmed with the requested boot memory data width (8, 16 or 32 bit wide, based on BOOT pins at reset, see Section 8.6.5). The interrupt vectors will be mapped from the external memory bank 0, and code residing in the external boot memory bank 0 will be executed.

A LOW level after reset at the P0.14 pin is considered as the external hardware request to start the ISP command handler.

If P0.14 is sampled LOW and the watchdog overflow flag is not set, the part will continue with executing ISP handler code, which starts with the auto-bauding procedure.

If P0.14 is sampled LOW and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored, and external code is executed as in case when P0.14 is HIGH after reset.

Pin P0.14 that is used as hardware request for ISP requires special attention. Since P0.14 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

The boot loader flow-chart is shown in Figure 76.

22.3 Memory map after reset

The boot loader resides in an on-chip ROM sector of 8 kB in size. After any reset this entire boot sector is mapped and is also visible in the memory region starting from the address 0x7FFF E000. The serial boot loader is designed to run from this memory area and it uses parts of the on-chip RAM. The RAM usage is described later in this chapter. In addition to the above mentioned remapping, the bottom 64 bytes of the ROM boot sector are also visible in the memory region starting from the address 0x0000 0000, i.e. the interrupt vectors of the part are mapped to those from the ROM boot sector. Consequently, the reset vector contains a jump instruction to the entry point of the serial boot loader software.

However, if the ISP handler was not invoked by P0.14, the interrupt vectors residing in the boot sector of the off-chip memory (bank 0) will become active and the bottom 64 bytes of the external boot sector will become visible in the memory region starting from the address 0x0000 0000.

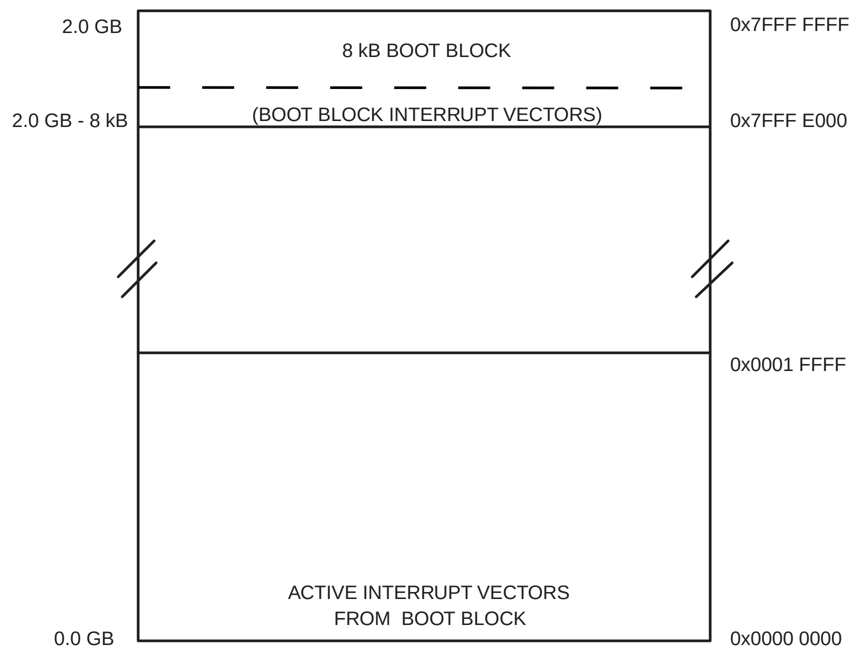


Fig 75. Map of the microcontroller’s memory after reset

If ISP handler was requested via P0.14, the auto-baud routine synchronizes with the host via serial port 0. The host should send a synchronization character(’?’) and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this the host should send the received string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. For example if the part is running at 10 MHz a valid response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly, the crystal frequency should be greater than or equal to 10 MHz. The on-chip PLL is not used by the boot code.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the "Go" command. The rest of the commands can be executed without the unlock command. The "Unlock" command is required to be executed once per ISP session. Unlock command is explained in the "ISP Commands" section.

22.4 Communication protocol

All ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in UU-encoded format.

22.5 ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

22.6 ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Data only for Read commands).

22.7 ISP data format

The data stream is in UU-encode format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format, which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes)) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

A description of UU-encode is available at wotsit.org.

22.8 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

22.9 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

22.10 Interrupts during ISP

The boot block interrupt vectors located in the ROM boot sector are active after any reset. For details on mapping interrupt vectors see Table 20.

22.11 Interrupts during IAP

IAP calls can be interrupted and an adequate interrupt service routine can be executed if interrupts are enabled. For details on how the address for interrupt service routine will be determined see Table 20. The IAP code itself does not use or disable interrupts.

22.12 RAM used by ISP command handler

ISP commands use on-chip RAM from 0x4000 0120 to 0x4000 01FF. The user could use this area, but the contents may be lost upon reset. The ROM boot loader also uses the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 256 bytes and it grows downwards.

22.13 RAM used by IAP command handler

IAP commands use top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes and it grows downwards.

22.14 RAM used by RealMonitor

The RealMonitor uses on-chip RAM from 0x4000 0040 to 0x4000 011F. The user could use this area if RealMonitor based debug is not required. The serial boot loader does not initialize the stack for the RealMonitor.

22.15 Boot process flowchart

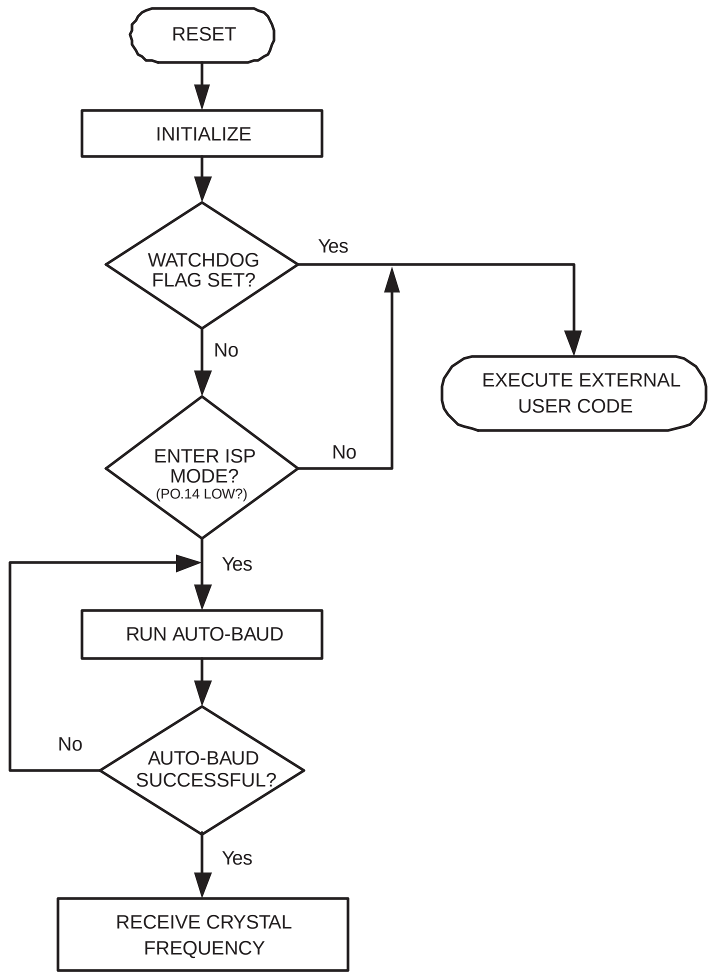


Fig 76. Boot process flowchart

22.16 ISP commands

The following commands are accepted by the ISP command handler. Detailed return codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 331. ISP Command Summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 332
Set Baud Rate	B <Baud Rate> <stop bit>	Table 333
Echo	A <setting>	Table 335
Write to RAM	W <start address> <number of bytes>	Table 336
Read Memory	R <address> <number of bytes>	Table 337
Go	G <address> <Mode>	Table 338
Read Part ID	J	Table 339
Read Boot code version	K	Table 341
Compare	M <address1> <address2> <number of bytes>	Table 342

22.16.1 Unlock <Unlock code>

Table 332. ISP Unlock command description

Command	U
Input	Unlock code: 23130
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command is used to unlock Go command.
Example	"U 23130<CR><LF>" unlocks the Go command.

22.16.2 Set Baud Rate <Baud Rate> <stop bit>

Table 333. ISP Set Baud Rate command description

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 | 230400 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

Table 334. Correlation between possible ISP baudrates and external crystal frequency (in MHz)

ISP Baudrate .vs. external crystal frequency	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456	+	+	+	+	+	+
15.3600	+					

Table 334. Correlation between possible ISP baudrates and external crystal frequency (in MHz)

ISP Baudrate .vs. external crystal frequency	9600	19200	38400	57600	115200	230400
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

22.16.3 Echo <setting>

Table 335. ISP Echo command description

Command	A
Input	Setting: ON = 1 | OFF = 0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

22.16.4 Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes)) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches then the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 336. ISP Write to RAM command description

Command	W
Input	Start Address: RAM address (on-chip only) where data bytes are to be written. This address should be a word boundary. Number of bytes: Number of bytes to be written. Count should be a multiple of 4.
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not a word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR
Description	This command is used to download data to RAM. The data should be in UU-encoded format.
Example	"W 10737442336 4<CR><LF>" writes 4 bytes of data to address 0x4000 0200.

22.16.5 Read Memory <address> <number of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UUencoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the check-sum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 337. ISP Read Memory command description

Command	R
Input	Start Address: Address (on or off-chip) where data bytes are to be read. This address should be a word boundary. Number of bytes: Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS (followed by <actual data (UU-encoded)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR
Description	This command is used to read data from on or off-chip memory.
Example	"R 1073741824 4<CR><LF>" reads 4 bytes of data from address 0x4000 0000.

22.16.6 Go <address> <Mode>

Table 338. ISP Go command description

Command	G
Input	Address: RAM address (on-chip only) from which the code execution is to be started. This address should be on a word boundary. Mode: T (Execute program in Thumb Mode) | A (Execute program in ARM Mode)
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR
Description	This command is used to execute (call) a program residing in RAM (on-chip only). It may not be possible to return to ISP command handler once this command is successfully executed. If executed code has ended with return instruction, ISP handler will resume with execution.
Example	"G 1073742336 A<CR><LF>" branches to address 0x4000 0200 in ARM Mode.

22.16.7 Read Part ID

Table 339. ISP Read Part ID command description

Command	J
Input	None

Table 339. ISP Read Part ID command description

Command	J
Return Code	CMD_SUCCESS followed by part identification number in ASCII format.
Description	This command is used to read the part identification number.
Example	"J<CR><LF>"

Table 340. LPC22xx Part identification numbers

Device	ASCII/dec coding	Hex coding
LPC2210	50462482	0x0301 FF12
LPC2210/01	50462482	0x0301 FF12
LPC2220 Rev -	50462482	0x0301 FF12
LPC2220 Rev B	50462482	0x0301 FF12
LPC2220 Rev C	50262514	0x0301 FF32
LPC2290	50462482	0x0301 FF12
LPC2290/01 Rev B	50462482	0x0301 FF12
LPC2290/01 Rev C	50262514	0x0301 FF32

22.16.8 Read Boot code version

Table 341. ISP Read Boot Code version command description

Command	K
Input	None
Return Code	CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.
Example	"K<CR><LF>"

22.16.9 Compare <address1> <address2> <number of bytes>

Table 342. ISP Compare command description

Command	M
Input	Address1(DST): Starting Address (on or off-chip) from where data bytes are to be compared. This address should be word boundary. Address2(SRC): Starting Address (on or off-chip) from where data bytes are to be compared. This address should be word boundary. Number of Bytes: Number of bytes to be compared. Count should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data is same) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not multiple of 4) ADDR_ERROR ADDR_NOT_MAPPED PARAM_ERROR

Table 342. ISP Compare command description

Command	M
Description	This command is used to compare the memory contents (on or off-chip) at two locations.
Example	"M 1073742336 1073741824 4<CR><LF>" compares 4 bytes from the on-chip RAM address 0x4000 0000 to the 4 bytes from the on-chip RAM address 0x4000 0200.
	Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. After any reset the first 64 bytes are re-mapped to on-chip ROM boot sector.

22.16.10 ISP Return Codes Summary

Table 343. ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.

22.17 IAP Commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. Result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case if number of results are more than number of parameters. Parameter passing is illustrated in Figure 77. The number of parameters and results vary according to the IAP command. The maximum number of parameters is 3, passed to the "Compare" command. The maximum number of results is 1, returned in case of every of three available IAP commands. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 0x7FFF FFF0 location and it is thumb code.

The IAP function could be called in the following way using C.

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x7ffffff1
```

Define data or pointers to pass IAP command table and result table to the IAP function

```
unsigned long command[5];
unsigned long result[2];
```

or

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x……
result= (unsigned long *) 0x……
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting function pointer

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, result);
```

The IAP call could be simplified further by using the symbol definition file feature supported by ARM Linker in ADS (ARM Developer Suite). You could also call the IAP routine using assembly code.

The following symbol definitions can be used to link IAP routine and user application.

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
0x7fffffb0 A rm_prefetchabort_handler
0x7fffffc0 A rm_dataabort_handler
0x7fffffd0 A rm_irqhandler
0x7fffffe0 A rm_irqhandler2
0x7ffffff0 T iap_entry
```

As per the ARM specification (The ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05) up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

Table 344. IAP Command Summary

ISP Command	Command code	Described in
Read Part ID	54	Table 345
Read Boot code version	55	Table 346
Compare	56	Table 347

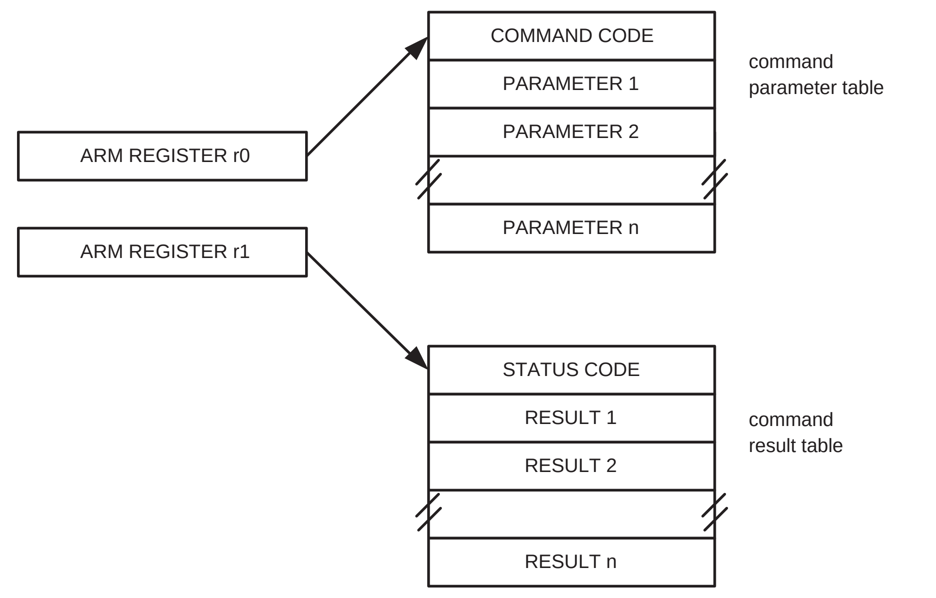


Fig 77. IAP parameter passing

22.17.1 Read Part ID

Table 345. IAP Read Part ID command description

Command	Read part ID
Input	Command Code 54 Parameters: None
Status Code	CMD_SUCCESS
Result	Result0: Part Identification Number
Description	This command is used to read the part identification number.

22.17.2 Read Boot code version

Table 346. IAP Read Boot Code version command description

Command	Read boot code version
Input	Command Code 55 Parameters: None
Status Code	CMD_SUCCESS
Result	Result0: 2 bytes of boot code version number. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

22.17.3 Compare

Table 347. IAP Compare command description

Command	Compare
Input	Command Code 56 Param0(DST): Starting address (on or off-chip) from where data bytes are to be compared. This address should be a word boundary. Param1(SRC): Starting address (on or off-chip) from where data bytes are to be compared. This address should be a word boundary. Param2: Number of bytes to be compared. Count should be in multiple of 4.
Status Code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not multiple of 4) ADDR_ERROR ADDR_NOT_MAPPED
Result	Result0: Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations.
	Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. After any reset the first 64 bytes are remapped to on-chip ROM boot sector.

22.17.4 IAP Status Codes Summary

Table 348. IAP Status Codes Summary

Status Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.

Table 348. IAP Status Codes Summary

Status Code	Mnemonic	Description
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	Flash programming hardware interface is busy.

22.18 JTAG external memory programming interface

Debug tools can write parts of the flash image to the on-chip RAM and then execute pre-loaded application dedicated to external flash programming repeatedly with proper offset.

UM10114

Chapter 23: LPC21xx/22xx Embedded ICE controller

Rev. 4 — 2 May 2012 **User manual**

23.1 How to read this chapter

The Embedded ICE controller is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

23.2 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.

23.3 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.

23.4 Description

The ARM7TDMI-S Debug Architecture uses the existing JTAG[1] 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

1. For more details refer to IEEE Standard 1149.1 - 1990 Standard Test Access Port and Boundary Scan Architecture.

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.

23.5 Pin description

Table 349. 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. **Remark:** This clock must be slower than $^{1}?_{6}$ of the CPU clock (CCLK) for the JTAG interface to operate.
TDI	Input	**Test Data In.** This is the serial data input for the shift register.
TDO	Output	**Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal.
nTRST	Input	**Test Reset.** The nTRST pin can be used to reset the test logic within the EmbeddedICE logic.
RTCK	Output	**Returned Test Clock.** Extra signal added to the JTAG port. Required for designs based on ARM7TDMI-S processor core. Multi-ICE (Development system from ARM) uses this signal to maintain synchronization with targets having slow or widely varying clock frequency. For details refer to "Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)".

23.6 Reset state of multiplexed pins

The pins above are multiplexed with P1.31-26. To have them come up as a Debug port, connect a weak bias resistor (4.7-10 kΩ depending on the external JTAG circuitry) between V_{SS} and the P1.26/RTCK pin. To have them come up as GPIO pins, do not connect a bias resistor and ensure that any external driver connected to P1.26/RTCK is either driving high, or is in high-impedance state, during Reset.

23.7 Register description

The EmbeddedICE logic contains 16 registers as shown in Table 350. 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.

Table 350. EmbeddedICE Logic registers

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

23.8 Block diagram

The block diagram of the debug environment is shown below in Figure 78.

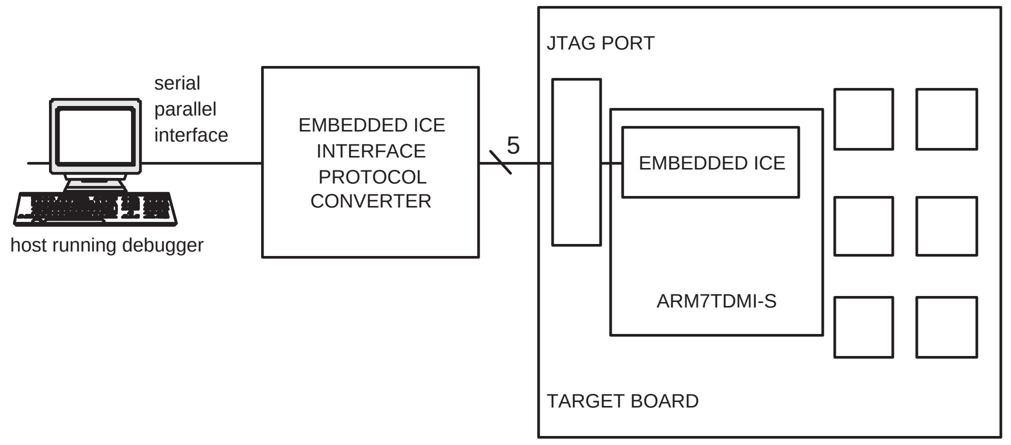


Fig 78. EmbeddedICE debug environment block diagram

UM10114

Chapter 24: LPC21xx/22xx Embedded Trace Module (ETM)

Rev. 4 — 2 May 2012 **User manual**

24.1 How to read this chapter

The ETM is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 "How to read this manual".

24.2 Features

- Closely track the instructions that the ARM core is executing.
- One external trigger input
- 10 pin interface
- All registers are programmed through JTAG interface.
- Does not consume power when trace is not being used.
- THUMB instruction set support

24.3 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.

24.4 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.

24.4.1 ETM configuration

The following standard configuration is selected for the ETM macrocell.

Table 351. 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
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)".

24.5 Pin description

Table 352. ETM Pin Description

Pin Name	Type	Description
TRACECLK	Output	**Trace Clock.** The trace clock signal provides the clock for the trace port. PIPESTAT[2:0], TRACESYNC, and TRACEPKT[3:0] signals are referenced to the rising edge of the trace clock. This clock is not generated by the ETM block. It is to be derived from the system clock. The clock should be balanced to provide sufficient hold time for the trace data signals. Half rate clocking mode is supported. Trace data signals should be shifted by a clock phase from TRACECLK. Refer to Figure 3.14 page 3.26 and figure 3.15 page 3.27 in "*ETM7 Technical Reference Manual" (ARM DDI 0158B)*, for example circuits that implements both half-rateclocking and shifting of the trace data with respect to the clock. For TRACECLK timings refer to section 5.2 on page 5-13 in "Embedded Trace Macrocell Specification" (ARM IHI 0014E).
PIPESTAT[2:0]	Output	**Pipe Line status.** The pipeline status signals provide a cycle-by-cycle indication of what is happening in the execution stage of the processor pipeline.
TRACESYNC	Output	**Trace synchronization.** The trace sync signal is used to indicate the first packet of a group of trace packets and is asserted HIGH only for the first packet of any branch address.
TRACEPKT[3:0]	Output	**Trace Packet.** The trace packet signals are used to output packaged address and data information related to the pipeline status. All packets are eight bits in length. A packet is output over two cycles. In the first cycle, Packet[3:0] is output and in the second cycle, Packet[7:4] is output.
EXTIN[0]	Input	**External Trigger Input**

24.6 Reset state of multiplexed pins

On the LPC21xx/LPC22xx, the ETM pin functions are multiplexed with P1.25-16. To have these pins come as a Trace port, connect a weak bias resistor (4.7 kΩ) between the P1.20/TRACESYNC pin and V_{SS}. To have them come up as port pins, do not connect a bias resistor to P1.20/TRACESYNC, and ensure that any external driver connected to P1.20/TRACESYNC is either driving high, or is in high-impedance state, during Reset.

24.7 Register description

The ETM contains 29 registers as shown in Table 353 below. They are described in detail in the ARM IHI 0014E document published by ARM Limited, which is available via the Internet.

Table 353. ETM Registers

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

Table 353. ETM Registers

Name	Description	Access	Register encoding
Sequencer State and Control	Holds the next state triggering events.	-	110 00xx
External Output 1 to 4	Holds the controlling events for each output.	WO	110 10xx
Reserved	-	-	110 11xx
Reserved	-	-	111 0xxx
Reserved	-	-	111 1xxx

24.8 Block diagram

The block diagram of the ETM debug environment is shown below in Figure 79.

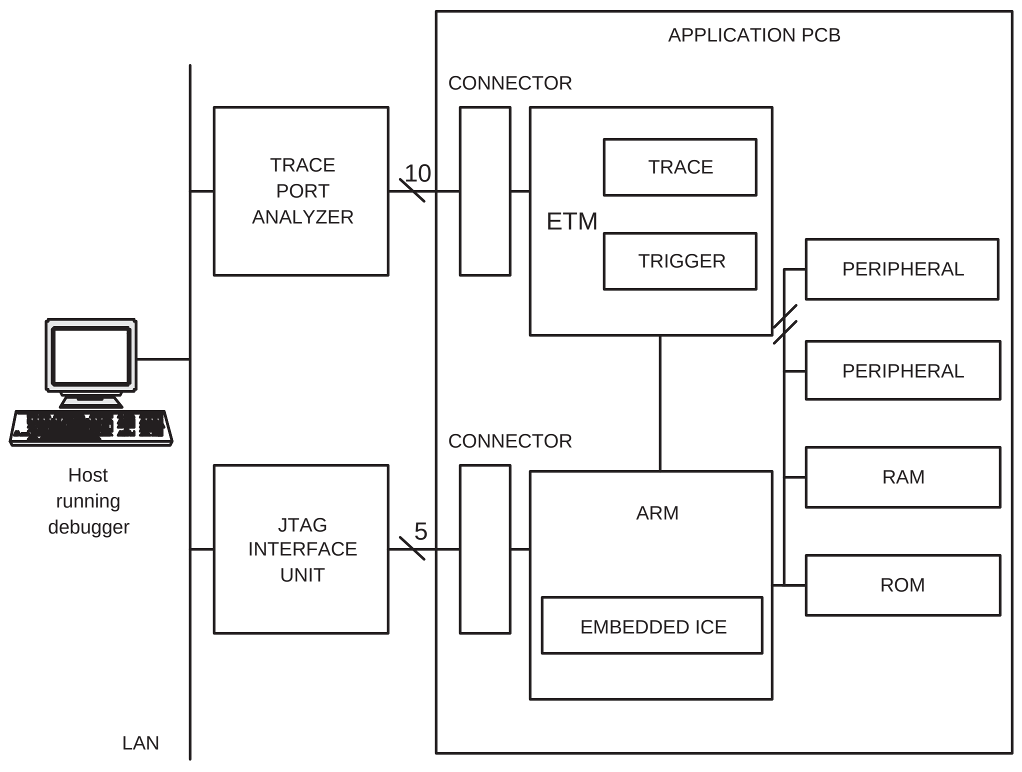


Fig 79. ETM debug environment block diagram

UM10114

Chapter 25: LPC21xx/22xx RealMonitor

Rev. 4 — 2 May 2012 User manual

25.1 How to read this chapter

The RealMonitor is identical for all LPC21xx and LPC22xx parts.

For an overview of how LPC21xx and LPC22xx parts and versions are described in this manual, see Section 1.2 “How to read this manual”.

25.2 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.

25.3 Applications

Real time debugging.

25.4 Description

RealMonitor is a lightweight debug monitor that allows interrupts to be serviced while user debug their foreground application. It communicates with the host using the DCC (Debug Communications Channel), which is present in the EmbeddedICE logic. RealMonitor provides advantages over the traditional methods for debugging applications in ARM systems. The traditional methods include:

- Angel (a target-based debug monitor)
- Multi-ICE or other JTAG unit and EmbeddedICE logic (a hardware-based debug solution).

Although both of these methods provide robust debugging environments, neither is suitable as a lightweight real-time monitor.

Angel is designed to load and debug independent applications that can run in a variety of modes, and communicate with the debug host using a variety of connections (such as a serial port or ethernet). Angel is required to save and restore full processor context, and the occurrence of interrupts can be delayed as a result. Angel, as a fully functional target-based debugger, is therefore too heavyweight to perform as a real-time monitor.

Multi-ICE is a hardware debug solution that operates using the EmbeddedICE unit that is built into most ARM processors. To perform debug tasks such as accessing memory or the processor registers, Multi-ICE must place the core into a debug state. While the processor is in this state, which can be millions of cycles, normal program execution is suspended, and interrupts cannot be serviced.

RealMonitor combines features and mechanisms from both Angel and Multi-ICE to provide the services and functions that are required. In particular, it contains both the Multi-ICE communication mechanisms (the DCC using JTAG), and Angel-like support for processor context saving and restoring. RealMonitor is pre-programmed in the on-chip ROM memory (boot sector). When enabled It allows user to observe and debug while parts of application continue to run. Refer to Section 25.5 “How To Enable RealMonitor” on page 359 for details.

25.4.1 RealMonitor Components

As shown in Figure 80, RealMonitor is split in to two functional components:

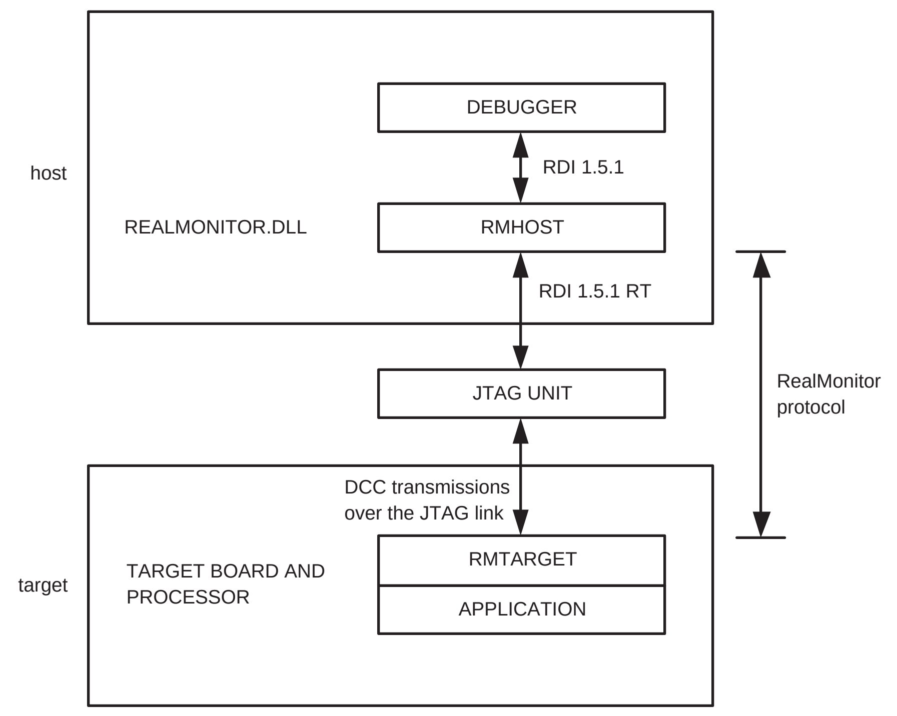


Fig 80. RealMonitor components

25.4.2 RMHost

This is located between a debugger and a JTAG unit. The RMHost controller, RealMonitor.dll, converts generic Remote Debug Interface (RDI) requests from the debugger into DCC-only RDI messages for the JTAG unit. For complete details on debugging a RealMonitor-integrated application from the host, see the ARM RMHost User Guide (ARM DUI 0137A).

25.4.3 RMTarget

This is pre-programmed in the on-chip ROM memory (boot sector), and runs on the target hardware. It uses the EmbeddedICE logic, and communicates with the host using the DCC. For more details on RMTarget functionality, see the RealMonitor Target Integration Guide (ARM DUI 0142A).

25.4.4 How RealMonitor works

In general terms, the RealMonitor operates as a state machine, as shown in Figure 81. RealMonitor switches between running and stopped states, in response to packets received by the host, or due to asynchronous events on the target. RMTarget supports the triggering of only one breakpoint, watchpoint, stop, or semihosting SWI at a time. There is no provision to allow nested events to be saved and restored. So, for example, if user application has stopped at one breakpoint, and another breakpoint occurs in an IRQ handler, RealMonitor enters a panic state. No debugging can be performed after RealMonitor enters this state.

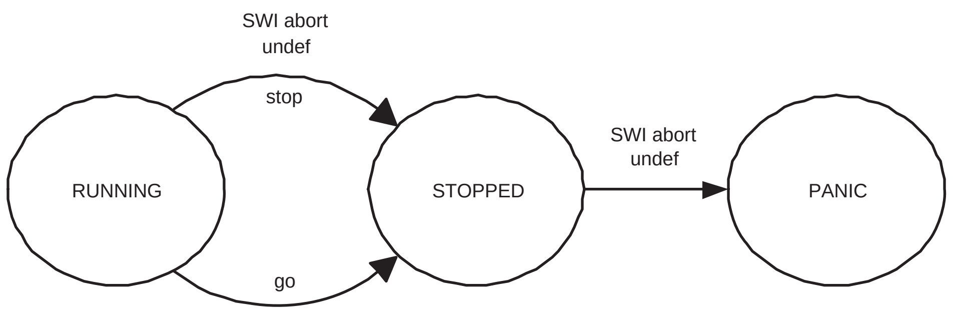


Fig 81. RealMonitor as a state machine

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 80.

The target component of RealMonitor, RMTarget, communicates with the host component, RMHost, using the Debug Communications Channel (DCC), which is a reliable link whose data is carried over the JTAG connection.

While user application is running, RMTarget typically uses IRQs generated by the DCC. This means that if user application also wants to use IRQs, it must pass any DCC-generated interrupts to RealMonitor.

To allow nonstop debugging, the EmbeddedICE-RT logic in the processor generates a Prefetch Abort exception when a breakpoint is reached, or a Data Abort exception when a watchpoint is hit. These exceptions are handled by the RealMonitor exception handlers that inform the user, by way of the debugger, of the event. This allows user application to continue running without stopping the processor. RealMonitor considers user application to consist of two parts:

- A foreground application running continuously, typically in User, System, or SVC mode
- A background application containing interrupt and exception handlers that are triggered by certain events in user system, including:
  - IRQs or FIQs
  - Data and Prefetch aborts caused by user foreground application. This indicates an error in the application being debugged. In both cases the host is notified and the user application is stopped.

  – Undef exception caused by the undefined instructions in user foreground application. This indicates an error in the application being debugged. RealMonitor stops the user application until a "Go" packet is received from the host.

When one of these exceptions occur that is not handled by user application, the following happens:

- RealMonitor enters a loop, polling the DCC. If the DCC read buffer is full, control is passed to rm_ReceiveData() (RealMonitor internal function). If the DCC write buffer is free, control is passed to rm_TransmitData() (RealMonitor internal function). If there is nothing else to do, the function returns to the caller. The ordering of the above comparisons gives reads from the DCC a higher priority than writes to the communications link.
- RealMonitor stops the foreground application. Both IRQs and FIQs continue to be serviced if they were enabled by the application at the time the foreground application was stopped.

25.5 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.

25.5.1 Adding stacks

User must ensure that stacks are set up within application for each of the processor modes used by RealMonitor. For each mode, RealMonitor requires a fixed number of words of stack space. User must therefore allow sufficient stack space for both RealMonitor and application.

RealMonitor has the following stack requirements:

Table 354. RealMonitor stack requirement

Processor mode	RealMonitor stack usage (bytes)
Undef	48
Prefetch Abort	16
Data Abort	16
IRQ	8

25.5.2 IRQ mode

A stack for this mode is always required. RealMonitor uses two words on entry to its interrupt handler. These are freed before nested interrupts are enabled.

25.5.3 Undef mode

A stack for this mode is always required. RealMonitor uses 12 words while processing an undefined instruction exception.

25.5.4 SVC mode

RealMonitor makes no use of this stack.

25.5.5 Prefetch Abort mode

RealMonitor uses four words on entry to its Prefetch abort interrupt handler.

25.5.6 Data Abort mode

RealMonitor uses four words on entry to its data abort interrupt handler.

25.5.7 User/System mode

RealMonitor makes no use of this stack.

25.5.8 FIQ mode

RealMonitor makes no use of this stack.

25.5.9 Handling exceptions

This section describes the importance of sharing exception handlers between RealMonitor and user application.

25.5.10 RealMonitor exception handling

To function properly, RealMonitor must be able to intercept certain interrupts and exceptions. Figure 82 illustrates how exceptions can be claimed by RealMonitor itself, or shared between RealMonitor and application. If user application requires the exception sharing, they must provide function (such as app_IRQDispatch ()). Depending on the nature of the exception, this handler can either:

- Pass control to the RealMonitor processing routine, such as rm_irqhandler2().
- Claim the exception for the application itself, such as app_IRQHandler ().

In a simple case where an application has no exception handlers of its own, the application can install the RealMonitor low-level exception handlers directly into the vector table of the processor. Although the IRQ handler must get the address of the Vectored Interrupt Controller. The easiest way to do this is to write a branch instruction (<address>) into the vector table, where the target of the branch is the start address of the relevant RealMonitor exception handler.

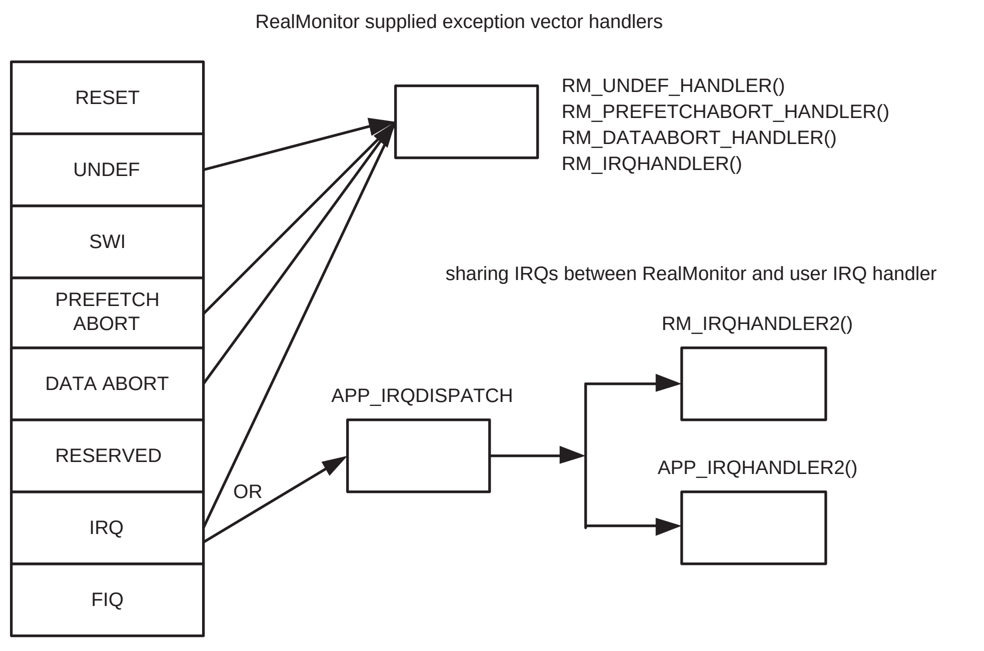


Fig 82. Exception handlers

25.5.11 RMTarget initialization

While the processor is in a privileged mode, and IRQs are disabled, user must include a line of code within the start-up sequence of application to call rm_init_entry().

25.5.12 Code Example

The following example shows how to setup stack, VIC, initialize RealMonitor and share non vectored interrupts:

```
IMPORT rm_init_entry
IMPORT rm_prefetchabort_handler
IMPORT rm_dataabort_handler
IMPORT rm_irqhandler2
IMPORT rm_undef_handler
IMPORT User_Entry ;Entry point of user application.
CODE32
ENTRY
;Define exception table. Instruct linker to place code at address 0x0000 0000

AREA exception_table, CODE

LDR pc, Reset_Address
LDR pc, Undefined_Address
LDR pc, SWI_Address
LDR pc, Prefetch_Address
LDR pc, Abort_Address
```

```
     NOP ; Insert User code valid signature here.
     LDR pc, [pc, #-0xFF0] ;Load IRQ vector from VIC
     LDR PC, FIQ_Address

Reset_Address        DCD __init           ;Reset Entry point
Undefined_Address    DCD rm_undef_handler ;Provided by RealMonitor
SWI_Address          DCD 0                ;User can put address of SWI handler here
Prefetch_Address     DCD rm_prefetchabort_handler     ;Provided by RealMonitor
Abort_Address        DCD rm_dataabort_handler         ;Provided by RealMonitor
FIQ_Address          DCD 0                ;User can put address of FIQ handler here

AREA init_code, CODE

ram_end EQU 0x4000xxxx ; Top of on-chip RAM.
__init
; /*********************************************************************
; * Set up the stack pointers for various processor modes. Stack grows
; * downwards.
; *********************************************************************/
     LDR  r2, =ram_end ;Get top of RAM
     MRS r0, CPSR ;Save current processor mode

     ; Initialize the Undef mode stack for RealMonitor use
     BIC  r1, r0, #0x1f
     ORR  r1, r1, #0x1b
     MSR  CPSR_c, r1
     ;Keep top 32 bytes for programming routines.
     ;Refer to On-chip Serial Bootloader chapter
     SUB  sp,r2,#0x1F

     ; Initialize the Abort mode stack for RealMonitor
     BIC  r1, r0, #0x1f
     ORR  r1, r1, #0x17
     MSR  CPSR_c, r1
     ;Keep 64 bytes for Undef mode stack
     SUB  sp,r2,#0x5F

     ; Initialize the IRQ mode stack for RealMonitor and User
     BIC  r1, r0, #0x1f
     ORR  r1, r1, #0x12
     MSR  CPSR_c, r1
     ;Keep 32 bytes for Abort mode stack
     SUB  sp,r2,#0x7F

     ; Return to the original mode.
     MSR  CPSR_c, r0

     ; Initialize the stack for user application
     ; Keep 256 bytes for IRQ mode stack
     SUB  sp,r2,#0x17F
```

```
; /*********************************************************************
; * Setup Vectored Interrupt controller. DCC Rx and Tx interrupts
; * generate Non Vectored IRQ request. rm_init_entry is aware
; * of the VIC and it enables the DBGCommRX and DBGCommTx interrupts.
; * Default vector address register is programmed with the address of
; * Non vectored app_irqDispatch mentioned in this example. User can setup
; * Vectored IRQs or FIQs here.
; *********************************************************************/

     VICBaseAddr          EQU 0xFFFFF000 ; VIC Base address
     VICDefVectAddrOffset  EQU 0x34

     LDR  r0, =VICBaseAddr
     LDR  r1, =app_irqDispatch
     STR  r1, [r0,#VICDefVectAddrOffset]

     BL   rm_init_entry   ;Initialize RealMonitor
     ;enable FIQ and IRQ in ARM Processor
     MRS  r1, CPSR        ; get the CPSR
     BIC  r1, r1, #0xC0   ; enable IRQs and FIQs
     MSR  CPSR_c, r1      ; update the CPSR
; /*********************************************************************
; * Get the address of the User entry point.
; *********************************************************************/
     LDR  lr, =User_Entry
     MOV  pc, lr
; /*********************************************************************
; * Non vectored irq handler (app_irqDispatch)
; *********************************************************************/

AREA app_irqDispatch, CODE
VICVectAddrOffset EQU 0x30
app_irqDispatch

     ;enable interrupt nesting
     STMFD sp!, {r12,r14}
     MRS  r12, spsr            ;Save SPSR in to r12
     MSR  cpsr_c,0x1F          ;Re-enable IRQ, go to system mode

;User should insert code here if non vectored Interrupt sharing is
;required. Each non vectored shared irq handler must return to
;the interrupted instruction by using the following code.
;    MSR  cpsr_c, #0x52                    ;Disable irq, move to IRQ mode
;    MSR  spsr, r12                        ;Restore SPSR from r12
;    STMFD sp!, {r0}
;    LDR  r0, =VICBaseAddr
;    STR  r1, [r0,#VICVectAddrOffset]      ;Acknowledge Non Vectored irq has finished
;    LDMFD sp!, {r12,r14,r0}               ;Restore registers
;    SUBS pc, r14, #4                      ;Return to the interrupted instruction

     ;user interrupt did not happen so call rm_irqhandler2. This handler
```

```
;is not aware of the VIC interrupt priority hardware so trick
;rm_irqhandler2 to return here

STMFD sp!, {ip,pc}
LDR   pc, rm_irqhandler2
;rm_irqhandler2 returns here
MSR   cpsr_c, #0x52                  ;Disable irq, move to IRQ mode
MSR   spsr, r12                      ;Restore SPSR from r12
STMFD sp!, {r0}
LDR   r0, =VICBaseAddr
STR   r1, [r0,#VICVectAddrOffset]    ;Acknowledge Non Vectored irq has finished
LDMFD sp!, {r12,r14,r0}              ;Restore registers
SUBS  pc, r14, #4                    ;Return to the interrupted instruction

END
```

25.6 RealMonitor Build Options

RealMonitor was built with the following options:

RM_OPT_DATALOGGING=FALSE

This option enables or disables support for any target-to-host packets sent on a non RealMonitor (third-party) channel.

RM_OPT_STOPSTART=TRUE

This option enables or disables support for all stop and start debugging features.

RM_OPT_SOFTBREAKPOINT=TRUE

This option enables or disables support for software breakpoints.

RM_OPT_HARDBREAKPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_HARDWATCHPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_SEMIHOSTING=FALSE

This option enables or disables support for SWI semi-hosting. Semi-hosting provides code running on an ARM target use of facilities on a host computer that is running an ARM debugger. Examples of such facilities include the keyboard input, screen output, and disk I/O.

RM_OPT_SAVE_FIQ_REGISTERS=TRUE

This option determines whether the FIQ-mode registers are saved into the registers block when RealMonitor stops.

RM_OPT_READBYTES=TRUE

RM_OPT_WRITEBYTES=TRUE

RM_OPT_READHALFWORDS=TRUE

RM_OPT_WRITEHALFWORDS=TRUE

RM_OPT_READWORDS=TRUE

RM_OPT_WRITEWORDS=TRUE

Enables/Disables support for 8/16/32 bit read/write.

RM_OPT_EXECUTECODE=FALSE

Enables/Disables support for executing code from "execute code" buffer. The code must be downloaded first.

RM_OPT_GETPC=TRUE

This option enables or disables support for the RealMonitor GetPC packet. Useful in code profiling when real monitor is used in interrupt mode.

RM_EXECUTECODE_SIZE=NA

"execute code" buffer size. Also refer to RM_OPT_EXECUTECODE option.

RM_OPT_GATHER_STATISTICS=FALSE

This option enables or disables the code for gathering statistics about the internal operation of RealMonitor.

RM_DEBUG=FALSE

This option enables or disables additional debugging and error-checking code in RealMonitor.

RM_OPT_BUILDIDENTIFIER=FALSE

This option determines whether a build identifier is built into the capabilities table of RMTarget. Capabilities table is stored in ROM.

RM_OPT_SDM_INFO=FALSE

SDM gives additional information about application board and processor to debug tools.

RM_OPT_MEMORYMAP=FALSE

This option determines whether a memory map of the board is built into the target and made available through the capabilities table

RM_OPT_USE_INTERRUPTS=TRUE

This option specifies whether RMTarget is built for interrupt-driven mode or polled mode.

RM_FIFOSIZE=NA

This option specifies the size, in words, of the data logging FIFO buffer.

CHAIN_VECTORS=FALSE

This option allows RMTarget to support vector chaining through µHAL (ARM HW abstraction API).

UM10114

Chapter 26: Supplementary information

Rev. 4 — 2 May 2012 User manual

26.1 Abbreviations

Table 355. Acronym list

Acronym	Description
ADC	Analog-to-Digital Converter
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
CAN	Controller Area Network
CISC	Complex Instruction Set Computer
FIFO	First In, First Out
GPIO	General Purpose Input/Output
I/O	Input/Output
JTAG	Joint Test Action Group
PLL	Phase-Locked Loop
PWM	Pulse Width Modulator
RISC	Reduced Instruction Set Computer
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSI	Synchronous Serial Interface
SSP	Synchronous Serial Port
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter

26.2 Legal information

26.2.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

26.2.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

26.2.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

<Name> — is a trademark of NXP B.V.

26.3 Tables

26.4 Figures

26.5 Contents

Chapter 9: LPC21xx/22xx General Purpose I/O (GPIO) controller

Chapter 10: LPC21xx/22xx Universal Asynchronous Receiver/Transmitter 0 (UART0)

Chapter 11: LPC21xx/22xx Universal Asynchronous Receiver/Transmitter 1 (UART1)